Optical Character Recognition for Indian Language Scripts using Support Vector Machines

A dissertation submitted in partial fulfillment
of the requirements of M.Tech.,Computer Science)
degree of Indian Statistical Institute, Kolkata
by

Srikanth Konagalla

under the supervision of

Mandar Mitra
CVPR Unit

Indian Statistical Institute
Kolkata-700 108.

17th July 2003

Indian Statistical Institute

203, Barrackpore Trunk Road,

Kolkata-700 108.

Certificate of Approval

This is to certify that this thesis titled **"Optical Character Recognition for Indian Language Scripts using Support Vector Machines"** submitted by **Srikanth Konagalla** towards partial fulfillment of requirements for the degree of M.Tech. in Computer Science at Indian Statistical Institute, Kolkata embodies the work done under my supervision.

[illegible]

Mandar Mitra
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Kolkata-700 108.

[illegible]

(External Expert)
CST Dept, B.E.College (DU)
Shibpur
Howrah.

Acknowledgments

I take pleasure in thanking Dr. Mandar Mitra for his friendly guidance throughout the dissertation period. His pleasant and encouraging words have always kept my spirits up.

Finally I take the opportunity to thank my family for their encouragement to finish this work.

Srikanth Konagalla

Abstract

The basic aim of this project is to find different methods for the improvement of an existing Bangla OCR system. This can be attempted in two ways. One is at the feature selection level; the other is the classification algorithm. We take the latter approach as the current feature is giving reasonably good results. We tried various post processing methods for pairs of characters which are often confused by the current nearest neighbour (NN) scheme. Almost all these methods gave good results compared to the NN method. We have also used Support Vector Machines (SVMs) for postprocessing, as well as for character recognition. While the performance of SVMs is promising in the postprocessing phase, its performance as a classification method is lower than the simple NN-based approach. Further study is necessary to see the usefulness of SVMs as a classfication scheme.

Contents

Chapter 1

Introduction to OCR

1.1 What is OCR?

Optical Character Recognition (OCR) is the process of automatic computer recognition of characters in optically scanned and digitized pages of text. It is one of the most popular and commercially successful image processing and pattern recognition systems. An OCR system recognizes paper based text automatically and converts it into electronic format for further processing. OCR is one of the challenging areas of Pattern Recognition with many practical applications. It can contribute to the advancement of automation processes and can improve the interface between man and machine in many applications, including office automation and data entry. The input to an OCR system consists of a scanned image of a text document and the output is a coded filewith ASCII or other character code representation. This enables compactstorage, editing, fast retrieval and other manipulations using computers.

1.2 Why is it important?

Some potential practical applications of OCR are:

1. Reading aid for the blind (OCR + speech synthesis).
2. Automatic text entry into computer, for desktop publication, library cataloging, ledgering, etc.
3. Automatic reading or sorting of postal mail, bank cheques, and other documents.
4. Document data compression from image to ASCII format.
5. Book keeping in libraries.

1.3 Brief history of OCR

The origin of character recognition can be found way back in 1870, when Carey invented the retina scanner - an image transmission system using a mosaic of photocells. Later, in 1890, Nipkow invented the sequential scanner, which was a major break-through both for modern television and reading machines. However, character recognition was initially considered as an aid to the visually handicapped and Tyurin, a Russian scientist, made some early successful attempts in 1900.

OCR technology took a major turn in the middle of the 1950s with the development of digital computers and improved scanning devices. For the first time OCR was realized as a data processing approach, with particular applications to the business world. From that perspective, David Shephard, founder of the Intelligent Machine Research Co. can be considered as a pioneer of the development of commercial OCR equipment. Currently, PC-based systems are commercially available to read printed documents of single font with very high accuracy.

1.4 Importance of Bangla OCR

Bangla OCR is important since Bangla is the second most popular script and language in the Indian sub-continent. About 200 million people of eastern India and Bangladesh use this language, making it the fourth most popular language in the world.

In Bangla, the number of characters is large and two or more characters may combine to form new character shapes called *compound* or *clustered characters*. As a result, the total number of characters to be recognized is about 300. Also due to the inflectional nature of this language, it is difficult to design a simple OCR error correction module. Thus OCR development is more difficult for Bangla than for any European language script.

1.5 Related work on Bangla OCR.

A complete Optical Character Recognition(OCR) system for printed Bangla script has been described in [1]. This system recognizes the 300 basic, modified and compound character shapes present in the script. The document image is first captured by a flat-bed scanner. It is then subject to skew correction, line segmentation, zone detection, word and character segmentation. From zonal information and shape characteristics, the basic, modified and compound characters are separated for the convenience of classification. The basic and modified characters which are about 75 in number and which occupy about 96% of the text corpus, are recognized by a structural-feature-based tree classifier. The compound characters are classified by a tree classifier followed by a template matching approach. The feature detection is simple and robust, and preprocessing steps like thinning and pruning are avoided. Character unigram statistics are used to make the tree classifier efficient. Several heuristics are also used to speed up the template matching approach. A dictionary-based error-correction scheme has been used where separate dictionaries containing morpho-syntactic information are compiled for *root word* and *suffixes*. For single-font, clear documents 95.50% word level recognition accuracy has been obtained.

A new feature vector for Bangla character recognition has been proposed by Garain and Chaudhuri [2]. For recognition of characters, feature based approaches are less reliable and template based approaches are less flexible to size and style variation of character font. It would be better to combine the positive aspects of feature based and template based approaches. This system proposes a run number based normalized template matching technique for character recognition. Run number vectors for both horizontal and vertical scans are computed. As the number of scans may vary from pattern to pattern, the vector has to be normalized and abbreviated. This normalized and abbreviated vector induces a metric distance. Moreover, this vector is invariant to scaling, insensitive to character style variation and more effective for more complex-shaped characters than simple-shaped ones.

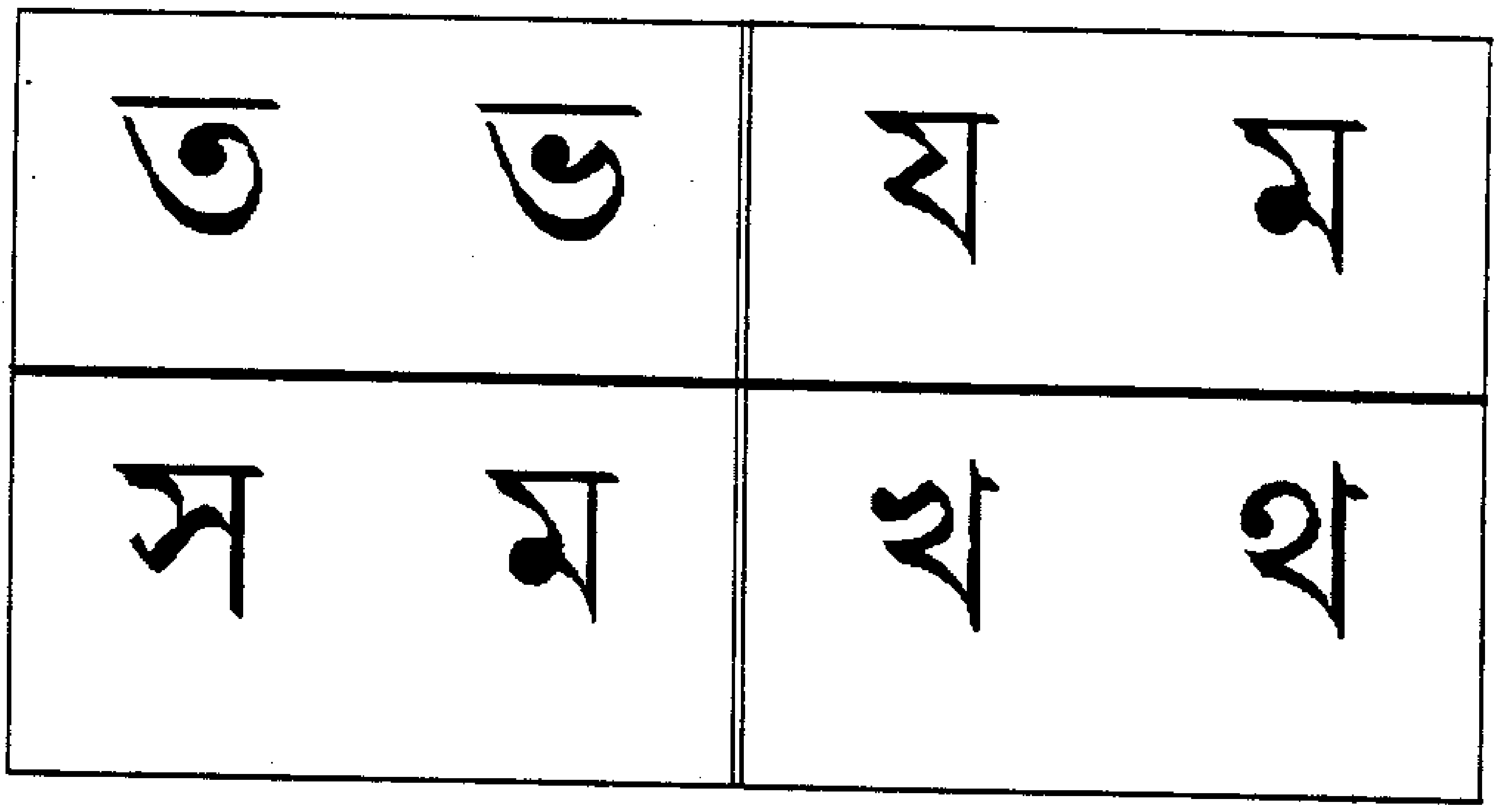

Figure 1.1: Some pairs of Bangla characters with similar shapes

1.6 Problem addressed in this thesis.

The Bangla OCR system described above is based on a nearest-neighbour approach and achieves high recognition rates for text printed on clear paper. However, several pairs of similar looking characters are often mis-recognized by this system. We propose to study the use of more sophisticated classification methods with higher discriminating capability, which may be more effective in distinguishing between such pairs of characters with similar appearances. A few examples of such pairs are given in Fig 1.1. We tried to solve the problem in 4 different ways. We test these approaches as a post-processing scheme for pairs of frequently confused characters. The rest of the thesis is organized as follows. In chapter(2) description about the current system is given. This chapter explains the feature vector used by the system and the conversion of the original vector to a fixed-dimension approximation. In chapter(3) we study variations of the L_1 distance measure used in the NN(nearest neighbor) approach. Chapter(4) describes the use of Mirror Image Learning (MIL) and its variation. Chapter(5) discusses our SVM -based approach. Chapter(6) concludes the thesis and outlines some directions for further work.

Chapter 2

Bangla OCR

2.1 Description of current system

The general scheme for the recognition of Bangla characters based on [1] and [3] can be divided into the following major steps.

1. **Image acquisition:** The document from which the characters are to be recognized is first scanned by a flatbed scanner.The resolution of a flatbed scanner varies from 100 to 900 dpi(dots per inch). The image that is scanned is a grey tone image (i.e. the pixel values range from 0 to 255). This gives a digitized image. For our purpose we have chosen TIFF format (300 dpi) for the digitized image.

2. **Preprocessing of the input document (binarization and noise removal):** For the improvement in the pictorial information for interpretation, we need to remove noise and also the image needs to be made two-tone. That is, the image should contain only two colors viz. Black and White. For this purpose a threshold value needs to be chosen. After thresholding, the image will be binary. The black pixels are represented as '1' and white pixels are represented as '0'. Thresholding removes noise partially. The binary image is much easier to process compared to a grey tone image.

3. **Skew detection and correction:**When a document page is fed in the scanner it may be skewed by a few degrees. The pages of a thick book create more problems since the scanned image may be both distorted and skewed. Casual use of scanner may also lead to skew in the document image. If the document is skewed, then the headline of each word is also skewed by the same degree. Skew angle is the angle that the headlines of the words in a document image make with the horizontal direction. To detect the headlines, the large connected components of the document are found by component labeling. Skew correction can be achieved in two steps, namely, (a) estimation of skew angle θ_s and (b) rotation of image by θ_s in the opposite direction. Skew correction is necessary for any OCR system.

4. **Segmentation of lines, words and characters:** Once the text blocks are detected, the system should automatically detect individual text lines, segment the words and separate the characters accurately. Text lines and words are detected by finding the valleys of the projection profile computed by a row wise(column wise for words) sum of gray values.

Let $S(n,m)$ be a binary image of n lines and m columns. Vertical projection profile, which is the row-wise sum of black pixels is computed to extract text lines. It is represented by a vector V of size n, and is defined by $V[i] = \sum_{j=1}^{m} S[i,j]$.

Words in a text line are identified by looking at the valleys in the horizontal profile, which is computed as the column-wise sum of black pixels . This profile is represented by the vector H of size m, which is defined by $H[i] = \sum_{j=1}^{n} S[i,j]$.

For segmenting individual characters from a word, we can delete the headline (if any) from the word. Since characters in a word are usually connected to each other through the headline, they get disconnected once the headline is erased. In actual implementation, the headline is not deleted. Rather, we run a connected component analysis for the region just below the headline.

Segmentation of shapes in the upper zone is relatively easy and achieved by a connected component analysis in the region above the headline. For each segmented character below and above the headline, we save its positional information. This is useful when we combine the shapes in the upper zone with the corresponding characters below the headline.

5. **Character recognition:** Approaches to character recognition are grouped into two categories, namely template matching and stroke feature based recognition. Both approaches have their own advantages and disadvantages. Template matching can be very accurate and efficient if the test characters are identical with the stored templates in shape and size. However, the approach can be sensitive to positional changes and is less flexible to font size and style variations. On the other hand, stroke feature based approaches are flexible to the font size and style variation but less reliable for strokes that are not correctly segmented from the characters. A hybrid technique that can combine the positive aspects of the feature-based and template based approaches is often tried in practice.

   The classifier in the present system uses horizontal and vertical crossing counts within the character image. Consider a character C enclosed by a minimum upright rectangle, called the bounding box. Let C be scanned horizontally from top to bottom and let N be the total number of scan lines. For each scan line, we count the number of black runs. A black run is a sequence of black pixels with a white pixel at either end of the sequence. For the i-th scan let $R_c[i]$ be the number of black runs. The sequence $R_c[i]$; i = 1,...,N may be considered as a vector of N integer components. We can call it the horizontal run count vector of C.

   The run count vector may be given an abbreviated notation by observing that the run-count remains unaltered over a sequence of several scan lines. Each of such sequences may be represented as a pair like (m_k, n_k), where m_k denotes the number of scan lines for which the run count is a constant n_k. Note that $\sum_k m_k = N$. To normalize the sum, we can divide the individual m_ks by N. Let $w_k = m_k/N$. Clearly, $\sum w_k = 1$. Hence, the character C can be represented by a normalized horizontal run count vector defined as

$$V_H(C) = w_k, n_k; k = 1, 2, ..., K \ where \sum_{k=1}^{K} w_k = 1 \tag{2.1}$$

Similarly, a vertical run count vector $V_v(C)$ can be computed.

In the classification phase, feature vectors for a target character are computed and matched with the stored prototypes. Matching is done by defining a distance measure explained below: Let $V_H(C)$ and $V_H(D)$ be the normalized horizontal run-count vectors of characters C and D. $V_H(C)$ and $V_H(D)$ are represented as

$$V_H(C) = w_k, n_k; k = 1, 2, ..., K \text{ where } \sum_{k=1}^{K} w_k = 1 \text{ and} \tag{2.2}$$

$$V_H(D) = w_j, n_j; k = 1, 2, ..., J \text{ where } \sum_{j=1}^{J} w_j = 1 \tag{2.3}$$

Next, define $W_k(C)=\sum_{i=1}^{k} w_i(C)$ and $W_j(C)=\sum_{i=1}^{j} w_i(C)$. Then the union of $W_k(C)$;k=1,2,...,K and $W_j(C)$;j=1,2,...,J is sorted in increasing sequence. Let this sequence of numbers be W_r; r = 1,2,...,R where $W_R = 1$. It is clear that the run-count of character C is constant over W_r - W_{r-1} for any $r = 1,2,...,R$. Similarly, the run count of character D is also constant over W_r - W_{r-1} for any $r = 1,2,...,R$.

Now, we can re-define the run-counts of C and D as $w_r n_r(C)$;r=1,2,...,R and $w_r n_r(D)$;r=1,2,...,R, respectively. The distance measure is formulated as:

$$J_H(C,D) = \sum_{r=1}^{R} w_r |n_r(C) - n_r(D)| \tag{2.4}$$

In a similar way, we can scan the characters vertically within the bounding box and find the run-count dissimilarity measure $J_V(C,D)$. The overall dissimilarity measure may be defined as

$$J(C,D) = J_H(C,D) + J_V(C,D) \tag{2.5}$$

The classifier uses two types of threshold values θ_1 and θ_2. For any input (or target) character, let D be the distance measured by (2.5). If $D < \theta_1$ the classifier returns the best matched character for which D was obtained. On the other hand, if $\theta_1 < D < \theta_2$ then classifier returns the two top characters choices, which are used during error correction phase. The classifier reports rejection for $D > \theta_2$.

2.2 Conversion of original vector to fixed-dimension approximation

The feature vector described in (2.1) is of variable dimension. We first convert it to fixed dimension, so that it can be used with the SVM approach. The original feature vector is given as $w_k, n_k;\ \ k = 1, 2, ..., K$ where $\sum_{k=1}^{K} w_k = 1$.

Now to convert it to fixed dimensionality, let us fix N dimensions. The interval from $[0, 1]$ is divided into N intervals. Therefore we have the following intervals.

$$\left[0\ ,\ \frac{1}{N}\right],\ \left[\frac{1}{N}\ ,\ \frac{2}{N}\right],\ \left[\frac{2}{N}\ ,\ \frac{3}{N}\right],\ ...,\ \left[\frac{N-1}{1}\ ,\ \frac{1}{N}\right] \tag{2.6}$$

For any feature vector w_k, n_k we will have the value n_i for the interval

$$\left[\sum_{j=1}^{i-1} w_i \, , \, \sum_{j=1}^{i} w_i \right] \qquad (2.7)$$

If a particular interval in (2.6) is completely contained within one interval in (2.7), then the value n_i is assigned to that interval. If an interval in (2.6) overlaps with 2 or more intervals in (2.7), then the value assigned to this interval is the weighted average of the n_i corresponding to the overlapping intervals of (2.7). The following example will make this clear.

Variable dimension = < 1 , 0.27 > < 2 , 0.73 >
Fixed dimension = < (0 , 0.1) , 1.00 > < (0.1 , 0.2) , 1.00 > < (0.2 , 0.3) , 1.3 >
< (0.3 , 0.4) , 2.00 > < (0.4 , 0.5) , 2.00 > ... < (0.9 , 1.0) , 2.00 >

Let C, D be two normalized run-count vectors.
Let C_f, D_f be the corresponding fixed-dimension approximations.
Then, it is easy to see that the distance measure defined in (2.4) for C, D reduces to the L_1 distance for C_f, D_f.

Chapter 3

L_1 Variations

3.1 Introduction

In this chapter we will look at 2 variations of L_1 distance. From the feature selection we can see that we are giving equal weights to all directions. Intuitively we feel that by giving more weights to important directions can increase accuracy.

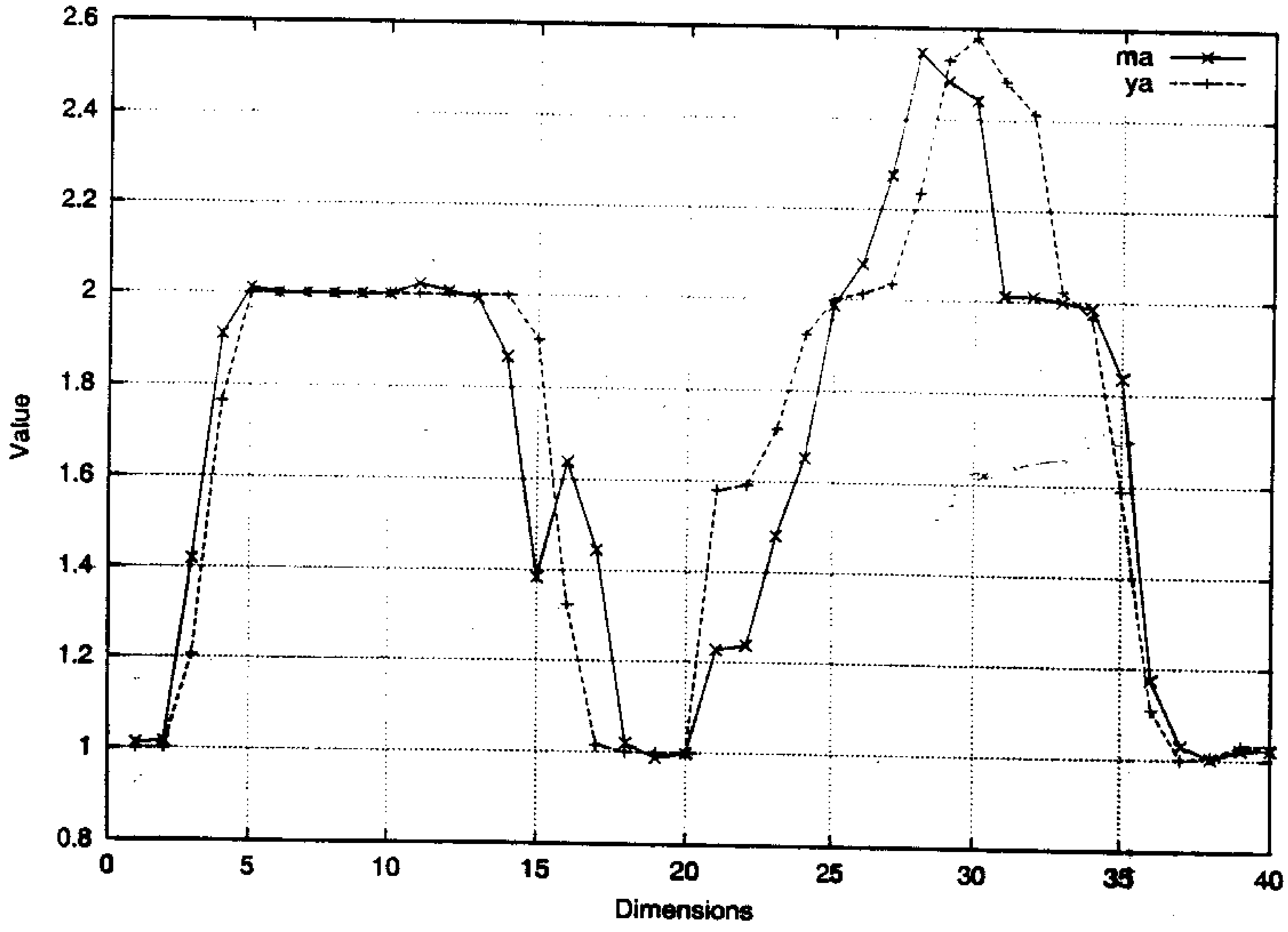

Figure 3.1: Mean Vectors of ma and ya.

Mean vector of a class is defined as *the sum of the training feature vectors of this class divided by the number of instances of this class.*

Mean Vector for class c can be defined mathematically as follows:

$$M_c = \frac{\sum_{X_c \epsilon T} X_c}{t_c}$$

where T = set of training samples. t_c = # of class c training samples.
Fig (3.1) shows the mean vectors for the pair ma. ya(similar looking characters shown in Fig(1.1)). We can see from this figure that the mean vectors for ma, ya differ more in directions 16-18, 21-24, 26-33 compared to other directions. Intuitively we feel that giving more weights to these directions (for ma, ya pair) can improve accuracy. In the next section we will see a way to give weights to directions according to their importance.

3.2 Weighted L_1

Mean vectors are some kind of representative for each class. For a two class problem the directions where these mean vectors differ more are important as compared to directions where they differ less. For each dimension we define a weight proportional to the absolute difference of the mean vectors along that dimension. Thus,

$$w_i = \frac{|m_i - m_i^{'}|}{\sum_{j=1}^{N} |m_j - m_j^{'}|}$$

where $m, m^{'}$ are mean vectors of two different classes and N is the dimension of the feature vector. The weighted L_1 distance measure is now defined as follows:

$$L_1^w(X, P) = \sum_{j=1}^{N} |X_j - P_j| w_j$$

where X is a test sample and P is a training sample. Clearly L_1^w is a metric as it satisfies all the properties of a metric.

3.3 Relative L_1

Consider 2 vectors $V_1 = (9, 99)$ & $V_2 = (10, 100)$ in 2-d Space.
9 in 10% away from 10.
99 is 1% away from 100.
But both directions contribute equally to L_1 distance between V_1, V_2. We want to note the kind of percentages shown above becuase when comparing a test sample with training samples we expect that its values in all directions are close the values in a training sample in all directions respectively. This idea mixed up with the above idea (related to weighted L_1) enabled us to modify the L_1 distance as follows:

$$L_1^r(X,P) = \sum_{j=1}^{N} \frac{|X_j - P_j||m_j - m_j'|}{P_j}$$

where
X : test sample
P : train sample
N : dimension of feature vector
m : mean vector of class I
m' : mean vector of class II.

We can show that this modified L_1 does not satisfy all properties of a metric so we can think of L_1^r as a dissimilarity measure. This measure gives better results as compared to the conventional L_1 distance.

3.4 Results

character pair	No. of training instances	No. of instances tested	NN accuracy (%)	L_1^w accuracy (%)	L_1^r accuracy (%)
M-Y	118	118	72.88	82.20	87.29
M-S	106	61	70.50	78.69	91.80
Kh - Th	55	62	40.32	56.45	58.06
Bh - T	81	61	86.88	98.36	100

Chapter 4

Mirror Image Learning

4.1 Introduction

In this chapter we will look at a new corrective learning algorithm proposed by Kimura [3]. The algorithm generates a mirror image of a pattern which belongs to one class of a pair of confusing classes and utilizes it as a learning pattern of the other class. The mirror image learning algorithm enlarges the learning sample of each class by mirror image pattern of other classes and enables us to achieve better accuracy.

4.2 Generation of Mirror Images

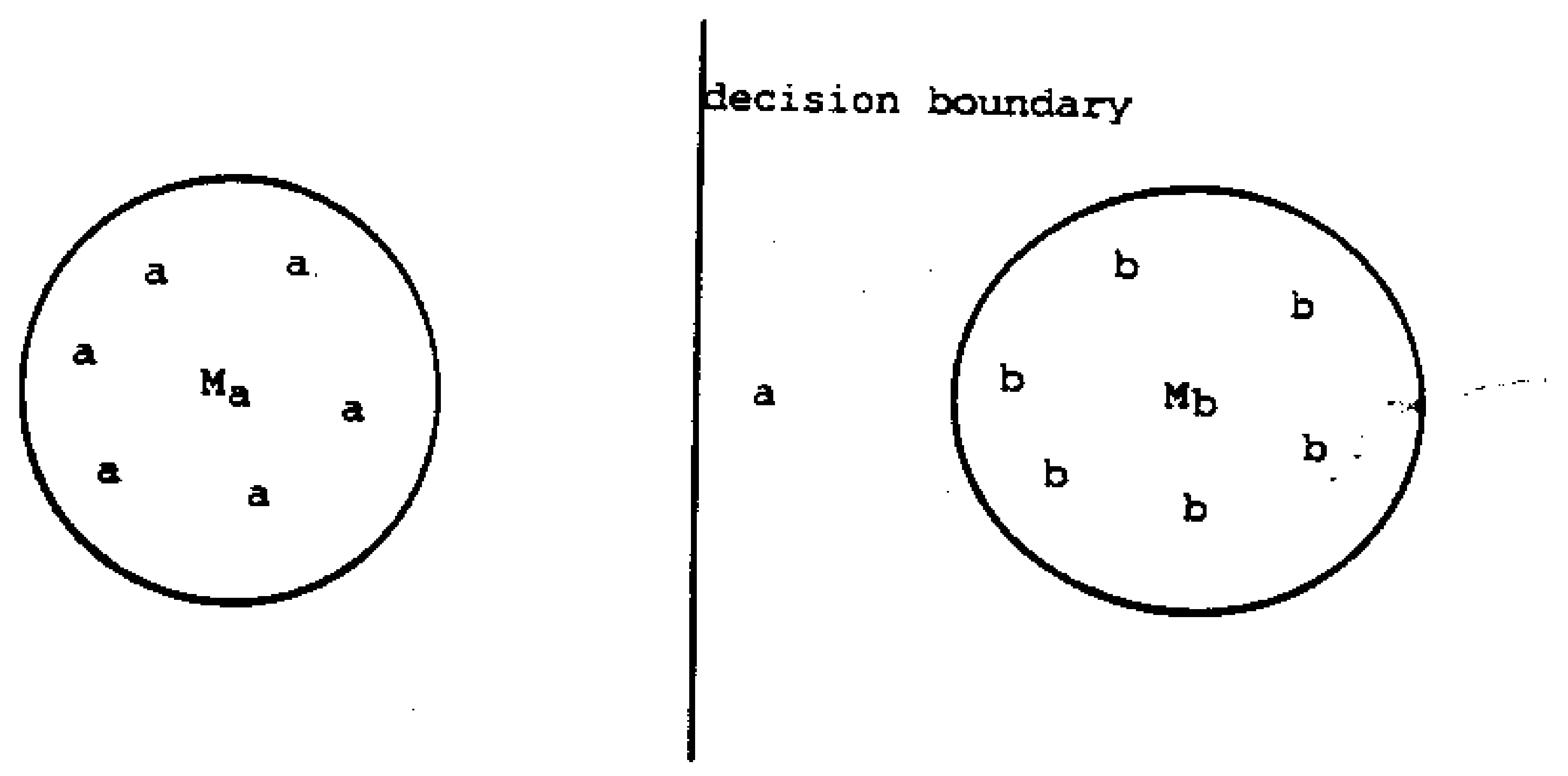

Figure 4.1: $\bar{a}$ is mirror image of a with respect to M_b

In Fig(4.1) M_a : mean vector of class 1, M_b : mean vector of class 2.
When a pattern X_1 of a class 1(2) is misclassified to a class 2(1) then the mean vector of class 2(1) is moved away from the pattern X_1 by adding its mirror image $\bar{X}_1$ (which is defined below) to the training set for class 2(1). The mirror image is formed with respect to the mean vector of class 2(1), and is formally defined as:

$$\bar{X}_1 = 2 * M_{b(a)} - X_1$$

4.3 Learning Algorithm

1.Calculate the mean vectors M_1 , M_2 from a set of original learning patterns X for classes I and II respectivley.
2.During recognition of the original learning patterns, if X from class C is misclassified then add the mirror image of X with respect to the mean vector of class C'(complement of class C) to class C'.
3.The mean vectors M_1, M_2 are re-calculated for each class using the accumulated samples.
4.Terminate if (a) there is no change in the number of misclassifications or (b)the number of the iterations exceeds a given threshold.
5.Goto step 2.

4.4 Weighted L_2

While taking L_2 distance between a training sample and a test sample it is clear that we are not giving unequal weights to different directions. So similar to weighted L_1 we can define weighted L_2 as follows:

$$L_2^w(X,Y) = \sum_{j=1}^{N} (X_j - Y_j)^2 |M_j^1 - M_j^2|$$

The results with this measure are better than the results with L_2 distance.

4.5 Results

character pair	No. of training instances	No. of instances tested	NN accuracy (%)	MIL accuracy (%)	NN with L_2 (%)	MIL with L_2 (%)
M-Y	225	225	91	94	90	98
M-S	164	164	95	96	91	98
Kh - Th	118	118	61	74	64	88
Bh - T	139	139	95	92	94	92

Here we merged train file with test files and the resultant file is given as input to MIL. After the first iteration of MIL half of the mis-classified samples are taken away from MIL for testing purpose. Test files contain these samples plus the samples which are correctly classified in the first iteration of MIL. That is the reason for having same number of samples in training as well as test files. Hardly 10 samples on an average are misclassified in the first iteration of MIL so we need more samples to test the use of MIL.

Chapter 5

Support Vector Machines

5.1 Introduction

Support Vector Machine(SVM) are learning systems that use a hypothesis space of learning functions in a high dimensional feature space, trained with a learning algorithm from optimization theory that implements a learning bias derived from statistical learning theory. Support vector machines have been widely used for isolated hand written digit recognition, object recognition, speaker recognition, face detection, text categorization, etc.

5.2 The Support Vector Algorithm

The SVM classifier is a two class classifier based on the use of discriminant functions. A discriminant function represents a surface which separates the patterns from the two classes lying on opposite sides of the surface. The SVM is essentially a separating surface which is optimal according to the criteria explained below.
Consider a two class problem where the class labels are denoted by +1 and -1. Given a set of l labelled(training) patterns, $\chi = (\mathbf{x}_i, y_i), 1 \leq i \leq l, \mathbf{x}_i \in \Re^d, y_i \in \{-1, +1\}$, the hyperplane represented by $(\mathbf{w}, b)$ where $\mathbf{w} \in \Re^d$ -represents the normal to the hyper plane and b $\in \Re$ the offset, forms a ***separating hyperplane*** or a ***linear discriminant function*** if the following separability conditions are satisfied:

$$\begin{aligned} \mathbf{w}'\mathbf{x}_i + b > 0, for\ i : y_i = +1, \\ \mathbf{w}'\mathbf{x}_i + b < 0, for\ i : y_i = -1. \end{aligned} \tag{5.1}$$

Here, $\mathbf{w}'\mathbf{x}_i$ denotes the inner product between the two vectors, and $g(\mathbf{x}) = \mathbf{w}'\mathbf{x} + b$ is the linear discriminant function.
In general the set χ may not be linearly separable. In such a case one can employ the *generalized linear discriminant function* defined by,

$$g(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}) + b \ where\ \phi : \Re^d \rightarrow \Re^{d'}, \mathbf{w} \in \Re^{d'}. \tag{5.2}$$

The original feature vector x is d-dimensional. The function ϕ represents some non-linear transformation of the original feature space and $\phi(\mathbf{x})$ is d'-dimensional. (Normally we would have

d' much larger than d.) By proper choice of function ϕ one can obtain complicated separating surfaces in the original feature space. For any choice of ϕ, the function g given by (3.2) is a linear discriminant function in $\Re^{d'}$, the range space of ϕ. However this does not necessarily mean that one can (efficiently) learn arbitrarily separating surfaces using only techniques of linear discriminant function by this trick of using ϕ. A good class of discriminant functions (say, polynomials of degree p) in the original feature space need a very high dimensional (of the order of d^p) vector, $\phi(\mathbf{x})$, and thus d' becomes much larger than d. This would mean that the resulting problem of learning a linear discriminant function in the d'- dimensional space can be very expensive both in terms of computation and memory. Another related problem is that we need to learn the d'-dimensional vector $\mathbf{w}$ and hence we would expect that we need a correspondingly larger number of training samples as well. The methodology of SVMs represents an efficient way of tackling both the issues. Here we only explain the computational issues.
Let $\mathbf{z}_i = \phi(\mathbf{x}_i)$. Thus now we have a training sample $(\mathbf{z}_i, y_i)$ to learn a separating hyperplane in $\Re^{d'}$. The separability conditions are given by 3.1 with $\mathbf{x}_i$ replaced by $\mathbf{z}_i$. Since there are only finitely many samples, given any $\mathbf{w} \in \Re^{d'}$, $b \in \Re$ that satisfy (3.1), by scaling them as needed, we can find (**w**,b), that satisfy

$$y_i[\mathbf{w}'\mathbf{z}_i + b] \geq 1, \ i = 1, ..., l. \tag{5.3}$$

Note that we have made clever use of the fact that $y_i \in +1, -1$ while writing the separability constraints as above. The (**w**,b), that satisfy (3.3) define a separating hyperplane, $\mathbf{w}'\mathbf{z} + b = 0$, such that there are no training patterns between the two parallel hyper-planes given by $\mathbf{w}'\mathbf{z} + b = +1$, *and* $\mathbf{w}'\mathbf{z} + b = -1$. The distance between these two parallel hyper-planes is $2/||\mathbf{w}||$, which is called the **margin** (of separation) of this separating hyperplane. It is intuitively clear that among all separating hyper-plane the ones with higher margin are likely to be better at generalization. The SVM is by definition, the separating hyperplane with maximum margin. Hence, the problem of obtaining the SVM can be formulated as an optimization problem of obtaining $\mathbf{w} \in \Re^{d'}$ and $b \in \Re$, to

$$\begin{aligned} Minimize &: \frac{1}{2}||\mathbf{w}||^2 && (5.4) \\ Subject\ to &: 1 - y_i(\mathbf{z}_i'\mathbf{w} + b) \leq 0 \ i = 1, ..., l. && (5.5) \end{aligned}$$

Suppose $\mathbf{w}^*$ and b^* represents the optimal solution to the above problem. Using the standard Lagrange multipliers technique, one can show that

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{z}_i, \tag{5.6}$$

where α_i^* are the optimal Lagrange Multipliers. There would be as many Lagrange multipliers as there are constraints and there is one constraint for each training pattern. From standard results in optimization theory, we must have $\alpha_i^*[1 - y_i(\mathbf{z}_i'\mathbf{w}^* + b^*)] = 0, \ \forall i$. Thus $\alpha_i^* = 0$ for all i such that the separability constraint (3.5) is strict inequality. Define a set of indices.

$$\mathcal{S} = \{i \ : \ y_i(\mathbf{z}_i'\mathbf{w}^* + b^*) - 1 = 0, \ 1 \leq i \leq l\}. \tag{5.7}$$

Now it is clear that $\alpha_i^* = 0$ if $i \notin S$. Hence we can rewrite (3.6) as

$$\mathbf{w}^* = \sum_{i \in S} \alpha_i^* y_i \mathbf{z}_i, \tag{5.8}$$

The set of patterns $\{\mathbf{z}_i : i \ s.t. \ \alpha_i^* > 0\}$ are called the support vectors.

5.3 SVM parameter variations

We use the SvmFu [5] package in our experiments. This implementation of the Support Vector classifier approach provides several parameters that can be set to an appropriate value for a given application. Even though SVMs were used to discriminate between some pairs of similar-looking characters by Singh [7], the effect of parameter variation was not comprehensively studied in his work. Table below presents the results obtained by Singh.

character pair	No. of training set	No. of instances tested	NN accuracy (%)	SVM accuracy (%)	kernel used(deg.)
M-Y	118	97	72.88	93.81	Polynomial (2)
M-S	106	61	70.50	80.32	Linear
Kh - Th	55	62	40.32	75.80	Polynomial (3)
Bh - T	81	61	86.88	98.36	Linear

We start by investigating the effect of varying the following parameter values on the accuracy of the SVM classifier:

C : The (linear) cost per unit violation of the margin. Larger values of C will tend to produce smaller margin hyperplanes with fewer and smaller violations, and will tend to require longer training times. The default is 1.
N : If this option is used, the examples in the negative class will have the specified C value, which can be different from the C value for the positive class. This is useful when there are uneven amounts of data from the two classes.
t : The tolerance to which the final solution must satisfy all of the KKT conditions. The default is 10E-4. If specified then 10E-t.
e : The smallest number (in absolute value) that the algorithm considers to be non-zero. The default is 10E-12. If specified then 10E-e.
n : For all kernels. The normalization term by which all kernel products will be divided. This is useful for numerical stability reasons. The default is 1.

The following table gives the parameters for which we got best accuracy using SVM.

character pair	C	N	e	t	n	SVM accuracy (%)	kernel used(deg.)
M-Y	.10	.01	4	12	2	94.92	Poly (3)
M-S	2	.10	3	12	3	96.72	Linear
Kh - Th	2	4	3	12	3	82.26	Linear
Bh - T	2	2	3	12	3	100	Linear

Unfortunately, we were not able to provide intuive explanations why the parameters shown above are giving better results compared to the default parameters and kernel used by Singh.

5.4 SVM in transformed space

As mentioned in Chapter 3 that it is clear that we are giving equal weights to all directions when computing the distance between 2 characters. Intuitively we feel that by giving varying weights to directions can improve accuracy. We have tried the following transformation (based on the ideas from L_1 Variations)and applied SVM in the transformed space but unfortunately this transform didn't work well.

$$\Phi : R^{40} \rightarrow R^{40}$$

defined by

$$\Phi(x_1, \ldots, x_{40}) = (x_1 w_1, \ldots, x_{40} w_{40})$$

where

$$w_i = \frac{|m_i^{(1)} - m_i^{(2)}|}{\sum_{j=1}^{40} |m_j^{(1)} - m_j^{(2)}|}$$

$m^{(1)}$ and $m^{(2)}$ are mean vectors of class I and II respectively.

5.5 SVM using Principal Components

5.5.1 What are Principal Components ?

Let X be a p-variate random vector.
Define $Y = A'X$
where A' is a $p \times q$ matrix (here $q \leq p$) such that $A'A = I_q$.
The columns of A are associated with q largest characteristic roots of Cov(X). Y_i' s are called principal components.

5.5.2 How to obain PC's ?

Let p be the feature vector dimension.
Step1: Calculate the covariance matrix Σ.
Step2: Find eigen values λ_i's and correponding eigen vectors E_i's.
Step3: Arrange eigen values in increasing order.
Without loss of generality we can assume that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$
Step4: Select first q E_i's as columns of matrix A such that $\frac{\sum_{j=1}^{q} \lambda_j}{\sum_{j=1}^{p} \lambda_j}$ lies between 0.95 and 0.99.

Unfortunately the idea of PC's did not work. One reason could be that characters belonging to two different classes have the same variation in those selected components. Variations more in one direction need not imply that in that direction the two classes differ more.

5.6 Using SVM as a multi-class classifier

From section 5.3 we can see that SVM accuracy is greater than NN accuracy for pairs of characters which are confused by NN the question that arises is whether we can extend SVMs to all characters? SVM can be extended to all classes in two ways. One-Versus-All and Pairwise Classification Schemes are the 2 ways through which SVM can be extended to all characters. In this section first we define winner-takes-all(one versus all) and pairwise classification schemes and then we will look at the results of the pairwise classification scheme for the complete set of characters in the Bangla Script.

5.6.1 One-Versus-All Classification Scheme

For a K class problem, K binary decision funcions are defined as follows:

$$f_k : R^N \rightarrow \{+1, -1\}$$

defined by

$$f_k(X) = \begin{cases} +1 & \text{for all samples } X \text{ in class } k. \\ -1 & \text{else.} \end{cases}$$

where N is the dimension of the feature vector.
Tie breaking: One usual way to break these tie situations is to neglect the signum operation in the decision functions and to use real input values to the signum operation instead.

5.6.2 Pairwise Classification Scheme

For the pairwise classification scheme, a decision function f_{kl} is defined for each pair kl of classes. Since the pairwise approach is symmetric, $f_{kl} = f_{lk}$ follows. For the ease of notation, we further define $f_{kk} = 0$.

$$f_{kl} : R^N \rightarrow \{+1, -1\}$$

defined by

$$f_{kl}(X) = \begin{cases} +1 & \text{for all samples } X \text{ in class } k. \\ -1 & \text{for all samples } X \text{ in class } l. \end{cases}$$

There exist $\binom{K}{2} = \frac{K(K-1)}{2}$ different pairwise decision functions. Using the hard decisions given by the signum function, the class decision can be calculated by summing up the pairwise decision functions: $f_k = \sum_l f_{kl}$. The class decision is given by $max_k f_k$. If there are no ties the following condition holds: $max_k f_k = (K - 1)$. This equation can be interpreted, that the winner class gets exactly $K - 1$ positive votes. One way to resolve ties is to use real input values to the signum operation instead.
According to Kressel [6] the pairwise Classification Scheme produces better decision boundaries, which are also intuitively easier to understand. Although the number of decision functions to compute increases by a factor of $\frac{(K-1)}{2}$ for a K class problem . the single decision functions are faster to adapt(fewer samples and usually fewer overlapping samples) and even sometimes faster to apply, since the number of support vectors for the single decision functions is usually smaller. Often linear decision functions are used to separate class pairs but it is possible to use different support vector approaches(different kernel functions or different degrees of the polynomial kernel) for the different class pairs, depending on the difficulty to separate the pair of classes.

5.6.3 SVM Pairwise Results

Our experiments showed that the SVM pairwise classifier takes a lot of time (105 sec for one page) and its accuracy is slightly lesser than NN. SVM pairwise classifier may be made practical by using code optimisation to decrease the time it takes for testing and using suitable parameters in the training of SVM for each pair of characters.

F.no	No. of instances	NN accuracy (%)	SVMP accuracy (%)
1	1908	94.03	93.45
2	1708	96.78	93.85
3	1502	90.13	86.71

We have used a total of 1608 samples for training.

Chapter 6

Conclusions and future work.

We mentioned earlier that this project is an attempt to improve the accuracy of an existing Bangla OCR System. L_1 variations can be used as a postprocessing method for pairs of characters which are frequently confused by NN because it is cheap and its accuracy is higher than NN accuracy. MIL may also be useful like L_1 variations as a postprocessing method but more data is needed to test this. SVMs are useful as a pairwise postprocessing method for characters where NN does not give good results but are not useful as Classification Sheme since the accuracy is comparable/marginally lower than NN accuracy and the computational cost is higher than NN. The following table gives the accuracies of all the above methods.

character pair	NN accuracy (%)	L_1^w accuracy (%)	L_1^r accuracy (%)	MIL accuracy (%)	SVM accuracy (%)
M-Y	72.88	82.20	87.29	98	94.92
M-S	70.50	78.69	91.80	98	96.72
Kh - Th	40.32	56.45	58.06	88	82.26
Bh - T	86.88	98.36	100	92	100

Future work:

(1)Have to find reasons why parameters given in table (in section 5.3) are giving good results compared to default parameter results.

(2)Try different features for the improvement of OCR System.

(3) Generalize L_1 variations from two classes to N classes.

Bibliography

[1] B.B Chaudhuri and U.Pal
A Complete Printed Bangla OCR System. *Pattern Recognition*, Vol. 31. No. 5. pp. 531-549. 1998

[2] U.Garain and B.B Chaudhuri
Compound Character recognition By Run Number Based Metric Distance. SPIE Vol. 3305, pp. 90-97, 1998.

[3] F.Kimura
Handwritten Numeral Recognition by Mirror Image Learning, Proc.6th ICDAR, IEEE Computer Society Press, pp. 338-343, 2001.

[4] T.V Ashwin and P S Sastry
A font and size-independent OCR system for printed Kannada documents using support vector machine, Sadhana, Vol.27, pp. 35-58, 2002.

[5] SvmFu : URL:http://five-percent-nation.mit.edu/SvmFu/#getting-download.

[6] B.Scholkoff, C.J.C.Burges, A.J.Smola (eds)
Advances in Kernel Methods, MIT Press, pp. 255-268, 1998.

[7] Rajiv Singh
Optical Character Recognition for Indian Language Scripts using Support Vector Machines, dissertation report, I.S.I, 2002.

[8] Bernhard Flury
Common Principal Components and Related Multivariate Models, Wiley, pp. 5-11, 1988.