# 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

**Rajiv Kumar Singh**

under the supervision of

**Mandar Mitra**
**CVPR Unit**

**Indian Statistical Institute**
**Kolkata-700 108.**

**$26^{th}$ July 2002**

# 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 **Rajiv Kumar Singh 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.

Mandar Mitra 26.7.02
Mandar Mitra
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Kolkata-700 108.

26.07.02
D.K. Basu (External Expert)
Jadavpur University
Kolkata-700 032.

# 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 classmates, friends and my family for their encouragement to finish this work.

**Rajiv Kumar Singh**

# Abstract

In this study we investigate the use of Support Vector Machines(SVM) for optical character recognition in an Indian language (Bangla). We start with a state of the art system that achieves fairly high accuracy. We identify some pairs of characters frequently confused by the current system. SVM are used to distinguish these pairs. The SVM based classifier performs significantly better than the existing classifier for these pairs of characters. Preliminary attempts to use SVMs as a general classsifier for all classes have yielded disappointing results and further investigation is necessary.

# 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 printed text and the output is a coded file with ASCII or other character code representation. This enables compact storage, editing, fast retrieval and other manipulations using computers.

## 1.2 Why is it important?

Some potential practical application of OCR are:

1. Reading aid for the blind (OCR + Speech analysis).
2. Automatic text entry into computer, for desktop publication, library cataloging, ledgering, etc.
3. Automatic reading for 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 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 Bangla OCR development is more difficult than that 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 [3]. 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 will 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

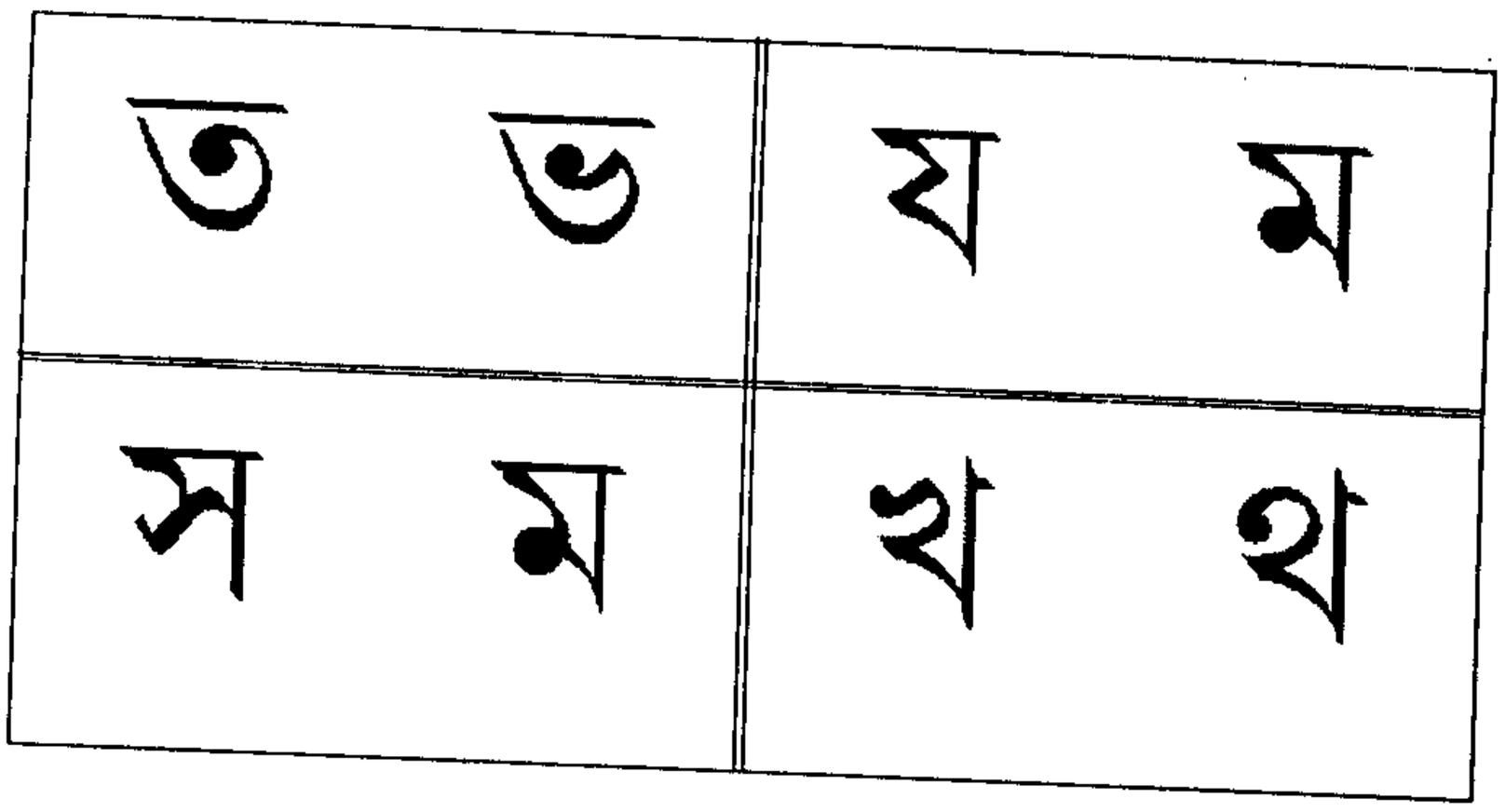


Figure 1.1: Some pairs of Bangla characters with similar shapes

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. This vector is used for representation for matching within a group of compound characters.

## 1.6 Problem addressed in this thesis.

The Bangla OCR system described above is based on a nearest-neighbour approach and achieves high recognition rate 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. Support vector machine is one such scheme. It has been found that SVM can classify correctly many characters mis-recognized by the earlier system. A few examples of such pairs are given in Fig 1.1. We test the SVM approach as a classification scheme as well 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 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) I first define the support vector machine (SVM); the method for structural risk minimization is then explained. This chapter also contains the support vector algorithm in detail. Chapter(4) describes the experiments and results. Chapter(5) concludes the paper.

# 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 $\theta_s$ and (b) rotation of image by $\theta_s$ 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, segments 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 sum of black pixels perpendicular to the y-axis, is computed to extract text lines. It is represented by a vector $V$ of size $n$, which 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 horizontal profile, which is computed as the sum of black pixels perpendicular to the x-axis. 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 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 their 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 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 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 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 weights by $N$. Let $w_k = m_k/N$. Clearly, $\sum w_k = 1$. Hence, the character $C$ is 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, vertical run count vector $V_v(C)$ is 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 $W_R = 1$. Now 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, 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. Now 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 $J_H(C,D)$. 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 $\theta_1$ and $\theta_2$. For any input (or target) character, let $D$ be the distance measured by (2.5). If $D < \theta_1$ the 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 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 $w_k, n_k; \quad k = 1, 2, ..., K$ where $\sum_{k=1}^{K} w_k = 1$.

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

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

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 >

# Chapter 3

# Basic introduction to SVM

## 3.1 Support Vector Machine

**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.

## 3.2 Structural Risk Minimization

**Structural Risk Minimization**: For the case of two-class pattern recognition, the task of *learning from examples* can be formulated in the following way: given a set of functions

$$\{f_\alpha : \alpha \in \Lambda\},\ f_\alpha : \mathbf{R}^N \to \{-1, +1\} \tag{3.1}$$

(the index set $\Lambda$ not necessarily being a subset of $R^n$ ) and a set of examples

$$(\mathbf{x}_1, y_1), ....., (\mathbf{x}_l, y_l),\ \mathbf{x}_i \in \mathbf{R}^N,\ y_i \in \{-1, +1\} \tag{3.2}$$

each one generated from an unknown probability distribution P(**x**,y), we want to find a function $f_\alpha$ which provides the smallest possible value for the risk

$$R(\alpha) = \int |f_\alpha(\mathbf{x}) - y| dP(\mathbf{x}, y). \tag{3.3}$$

The problem is that R($\alpha$) is unknown, since P(**x**,y) is unknown. Therefore an ***induction principle*** for risk minimization is necessary. The straightforward approach to minimize the *empirical risk*

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} |f_\alpha(\mathbf{x}_i) - y_i| \tag{3.4}$$

turns out not to guarantee a small actual risk (i.e. a small error on the training set does not imply a small error on a test set), if the number $l$ of training examples is limited. To make the

most out of limited amount of data, novel statistical techniques have been developed during last 25 years.

The *Structural Risk Minimization principle* is such a technique. It is based on the fact that for the above learning problem, for any $\alpha \in \Lambda$ with a probability of at least 1-$\eta$, the bound

$$R(\alpha) \leq R_{emp}(\alpha) + \Phi(\frac{h}{l}, \frac{log(\eta)}{l}) \tag{3.5}$$

holds, $\Phi$ being defined as

$$\Phi(\frac{h}{l}, \frac{log(\eta)}{l}) = \sqrt{\frac{h(log\frac{2l}{h} + 1) - log(\frac{\eta}{4})}{l}} \tag{3.6}$$

The parameter $h$ is called the VC-dimension of a set of functions. It describes the capacity of a set of functions implementable by the learning machine. For binary classification, $h$ is the maximal number of points $k$ which can be separated into two classes in all possible $2^k$ ways by using functions of the learning machine: i.e. for each possible separation there exists a function which takes the value 1 on one class and -1 on other class.

## 3.3 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:

$$\begin{aligned} \mathbf{w}'\mathbf{x}_i + b > 0, for\ i : y_i = +1, \\ \mathbf{w}'\mathbf{x}_i + b < 0, for\ i : y_i = -1. \end{aligned} \tag{3.7}$$

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 $\chi$ may not be linearly separable. In such a case one can employ the *generalized linear discriminant function* defined by,

$$g(\mathbf{x}) = \mathbf{w}'\phi(\mathbf{x}) + b\ where\ \phi : \Re^d \rightarrow \Re^{d'}, \mathbf{w} \in \Re^{d'}. \tag{3.8}$$

The original feature vector x is d-dimensional. The function $\phi$ 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 $\phi$ one can obtain complicated separating surfaces in the original feature space. For any choice of $\phi$, the function $g$ given by (3.2) is a linear discriminant function in $\Re^{d'}$, the range space of $\phi$. 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 $\phi$. 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

$$y_i[\mathbf{w}'\mathbf{z}_i + b] \geq 1, \ i = 1, ..., l. \tag{3.9}$$

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

$$\begin{aligned} Minimize \ &: \ \frac{1}{2}\|\mathbf{w}\|^2 && (3.10) \\ Subject\ to \ &: \ 1 - y_i(\mathbf{z}_i'\mathbf{w} + b) \leq 0 \ i = 1, ..., l. && (3.11) \end{aligned}$$

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

$$\mathbf{w}^* = \sum_{i=1}^{l} \alpha_i^* y_i \mathbf{z}_i, \tag{3.12}$$

where $\alpha_i^*$ 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,

$$\mathcal{S} = \{i \ : \ y_i(\mathbf{z}_i'\mathbf{w}^* + b^*) - 1 = 0, \ 1 \leq i \leq l\}. \tag{3.13}$$

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

$$\mathbf{w}^* = \sum_{i \in \mathcal{S}} \alpha_i^* y_i \mathbf{z}_i, \tag{3.14}$$

The set of patterns $\{\mathbf{z}_i : i\ s.t.\ \alpha_i^* > 0\}$ are called the support vectors. From (3.8), it is clear that the $\mathbf{w}^*$ is a linear combination of support vectors and hence the name SVM for the classifier. The support vectors are those patterns which are closest to the hyperplane and are sufficient to completely define the optimal hyperplane. Hence these patterns can be considered to be most important training examples. To learn the SVM all we need are the optimal Lagrange multipliers corresponding the problem given by (3.4) and (3.5). This can be done efficiently by solving its dual which is the optimization problem given by : Find $\alpha_i, i = 1, ..., l$, to

$$\begin{aligned} Minimize &: \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{z}_i^t \mathbf{z}_j, \\ Subject\ to &: \alpha_i \geq 0,\ i = 1, 2, ..., l, \\ &\sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \tag{3.15}$$

By solving this problem we obtain $\alpha_i^*$ and using these $\alpha_i^* s$ we get $\mathbf{w}^*$ and $b^*$ as follows.

$$\sum_{i=1}^{l} y_i \alpha_i . \mathbf{k}(\alpha_j . \mathbf{x}) + b = y_i$$

$$\Rightarrow b = y_j - \sum_{i=1}^{l} y_i \alpha_i . \mathbf{k}(\mathbf{x}_j' \mathbf{x}_i)$$

It may be noted that the dual given by (3.9) is a quadratic optimization problem of dimension $l$ (recall that $l$ is the number of training patterns) with one equality constraint and one non-negativity constraints on the variables. This is so irrespective of how complicated the function $\phi$ is. Once the SVM is obtained, the classification of new feature vector, $\mathbf{x}$, is based on the sign of(recall that $\mathbf{z} = \phi(\mathbf{x})$)

$$f(\mathbf{x}) = \phi(\mathbf{x})^t \mathbf{w}^* + b^* = \sum_{i \in S} \alpha_i^* y_i \phi(\mathbf{x}_i)^t \phi(\mathbf{x}) + b^*, \tag{3.16}$$

Where we have used (3.8). Thus, both while solving the optimization problem (given by (3.9)) and while classifying a new pattern,the only way the training pattern vectors, $\mathbf{x}_i$ come into picture are as inner products $\phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j)$. This is the only way, $\phi$ also enters into the picture.Suppose we have a function, $K : \Re^d \times \Re^d \rightarrow \Re$ such that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^t \phi(\mathbf{x}_j)$.Such a function is called a Kernel function. Now we can replace $\mathbf{z}_i^t \mathbf{z}_j$ in (3.9) by $K(\mathbf{x}_i, \mathbf{x}_j)$. Then we never need get into $\Re^{d'}$ while solving the dual. Often we can choose kernel function so that it is computationally much simpler than computing inner products in $\Re^{d'}$.Once (3.9)is solved and $\alpha_i^*$ are obtained, during classification also we never need enter $\Re^{d'}$. We can calculate the needed value of $f$ defined by (3.10) once again by using the kernel function.

Given any symmetric function $K : \Re^d \times \Re^d \rightarrow \Re$, there are some sufficient conditions, called Mercer's conditions, to ensure that there is some function $\phi$ such that $K$ gives the inner product in the transformed space. Some of the Kernel used in SVMs are listed in table 1. The polynomial kernel results in a separating surface in $\Re^d$ represented by polynomial of degree p.

With the Gaussian kernel the underlying $\phi$ is such that $\phi(\mathbf{x})$ is infinite dimensional! However, by the trick of kernel functions, you can get such arbitrarily complicated separating surfaces by solving only a quadratic optimization problem given by (3.9).

**Table 1.** Some popular kernels for SVMs.

| Type of kernel | $K(x_i, x_j)$ | Comments |
|---|---|---|
| Linear Kernel | $x_i' x_j$ | |
| Polynomial kernel | $(x_i' x_j + b)^p$ | Power p is specified *a priori* by the user |
| Gaussian kernel | $exp(-\frac{1}{2\sigma^2}\|x_i - x_j\|^2)$ | The width $\sigma^2$, common to all the kernels, is specified a *priory* |

So far, we have assumed that optimization problem specified by (3.4) - (3.5), whose dual is given by (10.9), has a solution. This is so only if in the $d'$-dimensional $\phi$-space, the(transformed) pattern vectors are linearly separable. In general this is difficult to guarantee. To overcome this we can change the optimization problem to

$$\begin{aligned} Minimize \quad &: \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i, \qquad (3.17) \\ Subject\ to \quad &: \quad 1 - y_i(\mathbf{z}_i^t\mathbf{w} + b) - \xi_i \leq 0, \ \ i = 1, ..., l. \\ &\qquad \xi_i \geq 0, i = 1, ..., l. \qquad (3.18) \end{aligned}$$

Here $\xi_i$ can be thought of as penalties for violating separability constraints. Now these are also variable over which optimization is to be performed.

# Chapter 4

# Experiments and results

While training and testing we have used the nearest neighbour (NN) and SVM algorithms. We may define a NN classification rule which assigns a pattern x of unknown classification to the class of its nearest neighbour, where we say that $s_i \in s_1, s_2, ..., s_N$ is a nearest neighbour to x if

$D(s_i, \mathbf{x}) = min\{D(s_l, \mathbf{x})\},\ l = 1, 2, ..., N$
where $D$ is any distance measure definable over the pattern space. ($L_1$ in this study)

For SVM we have used SvmFu 3 version 3.0 which is a package written for using Support Vector Machines. It is written in C++, does not require any third-party optimization engine, and is very fast. The algorithm is somewhat inspired by Platt's SVM algorithm, in that it optimizes a pair of points at each iteration. Same training set is used for both classifiers NN as well as SVM.

## 4.1 Original feature vector vs. fixed-dimension vector

When variable dimension feature vector is converted to fixed dimension feature vector so that it can be used with support vector approach, there is a difference between original feature vector and fixed dimension(40) approximation. We compared the two representations on 20 pages, with a total of almost 40,000 characters. Both representations yield identical accuracy, even though some characters are recognized correctly when the original feature vector is used, but mis-recognized when using the fixed-dimension approximation, and vice versa.

## 4.2 Using SVM for some pairs of similar characters

There are several pairs of similar looking characters frequently mis-recognized by the NN rule. Now some of the pairs have some simple topological differences (e.g. the presence of a loop in the character) which can be detected by simple testing. But for some pairs, such simple topological differences do not exist. We apply the SVM classifier to four such pairs. The results are presented in the following table.

| character pair | No. of training set | No. of instances tested | NN accuracy | SVM accuracy | kernel used(deg.) |
|---|---|---|---|---|---|
| M-Y | 118 | 97 | 83.5(%) | 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 |

## 4.3 Using SVM as a multi-class classifier

We also applied SVM for multi-class classification to all the characters. In this, we use one vs. all classifiers. SVM for multi-class classification shows 45.33% accuracy.

# Chapter 5

# Conclusions and future work.

We have successfully used Support vector Machines(SVM) for pair-wise classification.Whereas, for multi-class classification, SVM has poor performance. We have tried SVM for one-all classifiers and preliminary experiments shows that it is computationally expensive for pairwise classifiers.

**Future work:**

(1) Some more pairs have to be found out for which NN rule frequently confuses.

(2) Test the performance of SVM on those pairs.

(3) Apply SVM with pair wise classifiers for multi class classification.

(4) explore SVM parameter variations (e.g. kernel type)

# Appendix A

# Optical Character Recognition Evaluation Technique

Frequently object recognition accuracy is a key component in the performance analysis of pattern matching systems. In OCR system, OCR accuracy measure is very vital point. OCR systems are classified by there character accuracy. Curiously, a detailed review of many of OCR error occurrence counting results reveals that they are not reproducible as published and they are not strictly comparable due to large variances in the counts than would be expected by the sampling variance. Naturally, since OCR accuracy is based on ratio of number of OCR errors over the size of text searched for errors, imprecise OCR error accounting leads to similar imprecision in OCR accuracy.

**Some of the methods are :**
1) Informal OCR error accounting.
2) Non-automatic OCR error accounting.
3) Intuitive OCR error accounting.

Other methods are based on string matching algorithm such as dynamic programming using edit distance methods. Performance of OCR is relative to the method used to calculate it, as number of errors found by different methods are significantly different.

OCR(character) accuracy is typically computed as: (n-k)/n where n is the number of characters in the original text and k is the number of OCR error occurrences (the number of character misinterpretation).

**Edit distance method:**
Define e($x$,$y$), the *edit distance* between two strings $x$ and $y$, to be the minimum number of character insertions, deletions, and replacements that are required to transform $x$ into $y$. Let $x = a_1$ ..... $a_m$ and $y = b_1$ ..... $b_n$ be any string over an alphabet $\sum$ and let the possible *editing operations* on A be:

(i) *deleting* a symbol from any position, say i, to give $a_1 ... a_{i-1} a_{i+1} ... a_m$;

(ii) *inserting* a symbol $b \in \sum$ at position $i$ to give $a_1 ... a_i b a_{i+1} ... a_m$;

(iii) *changing* a symbol at position $i$ to a new symbol $b \in \sum$ to give $a_1 \ldots a_{i-1} b a_{i+1} \ldots a_m$.

Each editing step can be understood as an application of a rewriting rule $a \rightarrow b$ where $a$ and $b$,$a \neq b$, are in $\sum$ or at most one of $a$ and $b$ are empty string $e$. Rules with $b=e$ define deletions, rules with $a=e$ define insertions and rules with nonempty $a$ and $b$ define changes. Clearly, with these editing operations it is possible to convert, step-by-step, any string A into another string B.

Each editing operation $a \rightarrow b$ has a non-negative cost $\delta(a \rightarrow b)$. Given strings $A = a_1 \ldots a_m$ and $B = b_1 \ldots b_n$, we want to determine a sequence of editing operations which convert A into B so that the sum of individual costs of editing operations in the sequence is minimized. The minimum cost is denoted by D(A,B) called *edit distance* from A to B. The problem of computing D(A,B) is also known as the *string-to-string correction problem*

Computing D(A,B) becomes considerably simpler as soon as we may assume that there is always as editing sequence with cost D(A,B) converting A into B such that if an element is deleted, inserted or changed , it is not modified again. This means that all editing operations could be applied on A in one parallel step yielding B.

This requirement is easily satisfied: It suffices that the cost function $\delta$ fulfills the inequality, i.e.,

$$\delta(a \rightarrow c) \leq \delta(a \rightarrow b) + \delta(b \rightarrow c) \tag{A.1}$$

for all a,b,c such that a $\rightarrow$ c, a $\rightarrow$ b, and b $\rightarrow$ c are editing operations. We also assume that

$$\delta(a \rightarrow b) > 0 \tag{A.2}$$

for all operations a$\rightarrow$b. this is a natural requirement (since a$\neq$b) which is essential for our results. When (A,1) is true, distance D(A,B) can be determined with a well-known tabulation method as follows: For all $0 \leq i \leq m$ and $0 \leq j \leq n$, denote by $d_ij$ the edit distance $D(a_1 \ldots a_i, b_1 \ldots b_j)$ from string $a_1 \ldots a_i$ and $b_1 \ldots b_j$. Then the (m+1)x(n+1) matrix $(d_ij)$ can be computed from the recurrence

$$\begin{aligned} d_{00} &= 0 \\ d_{ij} &= min(\, d_{i-1,j-1} + replacement(a_i, b_j),\ d_{i-1,j} + 1,\ d_{i,j-1} + 1) \\ & \quad i > 0 \text{ or } j > 0. \end{aligned} \tag{A.3}$$

Where $replacement(a_i,b_j) = 0$ if $a_i = b_j$ , 1 otherwise.

Clearly, matrix $(d_ij)$ can be evaluated starting from $d_{00}$ and proceeding row-by-row or column-by-column (and assuming that all undefined values $d_{ij}$ referred to in the minimization step have default value $\infty$). This takes time and space $O(mn)$. Finally, $d_{mn}$ equals D(A,B). Moreover, the sequence of editing steps that give D(A,B) can be recovered from the matrix

$(d_{ij})$ using the standard technique applied in dynamic programming in which one follows some "minimizing path" backwards from $d_{mn}$ to $d_{00}$ and records at each stage, which of the alternative gives the minimum. So, if we have found that $d_{ij}$ is on a minimizing path and, for example $d_{ij} = d_{i-1,j} + \delta(a_i \rightarrow e)$ then $d_{i-1,j}$ is the next entry on the path and "delete $a_i$ is the editing operation.

Example : let us suppose that the string yxxz has to be converted to xyxzy, then we will have edit distance = 3 as shown in the distance matrix.

| | | x | y | x | z | y |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| y | 1 | 1 | 1 | 2 | 3 | 4 |
| x | 2 | 1 | 2 | 1 | 2 | 3 |
| x | 3 | 2 | 2 | 2 | 2 | 3 |
| z | 4 | 3 | 3 | 3 | 2 | 3 |

Fig: Matrix $(d_{i,j})$ for strings yxxz and xyxzy.

# Bibliography


[1] B.B Chaudhuri and U.Pal
**A Complete Printed Bangla OCR System..** *Pattern Recognition*, Vol. 31, N0. 5, pp. 531-549, 1998

[2] T.V Ashwin and P S Sastry
Department of Electrical Engineering, Indian Institute of Science, Bangalore 560 012, India
**A font and size-independent OCR system for printed Kannada documents using support vector machine.**

[3] U.Garain and B.B Chaudhuri
**Compound Character recognition By Run Number Based Metric Distance.**, SPIE Vol. 3305. 0277-786X/98

[4] Bernhard Scholkopf, Chris Burges, Vladimir Vapnik
**Extracting Support Data for a Given Task**

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