A NOVEL APPROACH ON A TEMPLATE-BASED ON-LINE WRITER INDEPENDENT NEPALI HANDWRITTEN CHARACTER RECOGNITION

A Thesis Presented

by

Santosh K. C.

Master of Science

Information Technology Program

Sirindhorn International Institute of Technology

Thammasat University

May 2007

A NOVEL APPROACH ON A TEMPLATE-BASED ON-LINE WRITER INDEPENDENT NEPALI HANDWRITTEN CHARACTER RECOGNITION

A Thesis Presented

By

Santosh K.C.

Submitted to

Sirindhorn International Institute of Technology

Thammasat University

In partial fulfillment of the requirement for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Approved as to style and content by the Thesis Committee:

Chair and Advisor

Dr. Cholwich Nattee

Member

Assoc. Prof. Dr. Thanaruk Theeramunkong

Member

Asst. Prof. Dr. Matthew Nelson Dailey

May 2007

Acknowledgement

The author wishes to express his deep gratitude to his Late father, Mr. Lokesh Kumar K.C., mother, Mrs. Jamuna K.C. and brother, Mr. Birendra Bahadur K.C. for their warm love, affection, and inspirations, which make possible to achieve many scholarships in his student life from the beginning of the day. He would like to put across his profound gratitude and sincere appreciation to his advisor, Dr. Cholwich Nattee for giving him a topic of research, significant advices, precious comments and constant encouragement throughout the epoch. More specifically, he is glad with his availability, patience and assistance in many aspects. The author gives great thanks to Asian Development Bank- Japan Scholarship Program (ADB-JSP) for providing huge fund in his entire study period.

The author conveys special thanks to Assoc. Prof. Dr. Thanaruk Theeramunkong (Head of Department- School of Information and Computer Technology, Sirindhorn International Institute of Technology) and Asst. Prof. Dr. Matthew N. Dailey (Asian Institute of Technology) for their remarks with possible solutions, good wishes, priceless advices and guidance in every progress presentation and seminar.

The author is grateful to Dr. Sanparith Marukatat, NECTEC, Thailand, for his valuable suggestions in many conferences like, Knowledge, Information and Creativity Support Systems (KICSS) 2006 and Pacific Rim International Conference on Artificial Intelligence (PRICAI) 2006, with the possibility of significant perfection. How can he forget to give a special thanks to the committee of the first International Conference on Knowledge, Information and Creativity Support Systems KICSS, 2006 for rewarding the Best Student Paper Award?

The author would like to convey his sincere thanks to Mr. Ithipan Methasate, Mr. Prakasith Kayasith, Ms. Chutima Pisarn and Mr. Kritsada Sriphaew, KIND LAB, Sirindhorn International Institute of Technology, Thammasat University, Thailand, for their valuable suggestions and recommendations during the entire study period.

The author grants his deepest grateful to external committee, Assoc. Prof. Dr. Boonserm Kijsirikul (Chulalongkorn University), for his immediate respond to the thesis along with a sort of appreciation to this task.

Lastly, he acknowledges to all the contributors (Nepalese natives) for giving their precious time in writing both test and training characters.

Abstract

A pencil and paper are often preferable for every one to use during the first draft preparation instead of using keyboard and other computer input interfaces like this. How much easier and faster the lives will be for those, who are non-natives to English, computer novices and feel inconvenient in using keyboard and keypad (old people), if a system (multi-lingual) having the intelligence in recognizing the natural handwriting for all possible scripts around the world exists. Under the purview of this, a novel approach on a complete template-based on-line writer independent natural handwritten character recognition system for Nepali is explored. It enjoys the advantages of 'Dynamic Time Warping' (DTW) in classifying the digitized strokes into its own class at the same time of writing.

The number of strokes, their order, directions, shapes and sizes, tilting angles and similarity in between the characters from one another are variable in cursive handwriting. Nepali is one of the examples of cursive handwriting. A number of strokes within a class of character from many users are merged based on their similarity and clustered them accordingly in order to reduce the number of prototype training samples (templates). The prototype classifier takes an advantage of single-linkage agglomerative hierarchical clustering. We construct a prototype classifier that uses the DTW algorithm to align handwritten strokes with stored stroke templates and determine their similarity. Both the pen-tip position and pen direction at every position along the pen trajectory is used as the feature of the stroke for classification. In addition to the structural feature of the stroke, spatial information has been taken into account for greater classification accuracy. To evaluate the system, we collected examples of 46 classes of different alphanumeric characters from Nepalese natives, and then performed a series of experiments. A prototype template-based classifier is developed by using both the structural properties of the strokes and their spatial relation. An inclusion of spatial relation between/among the strokes within the character in the classifier improves the performance. More than 95% classification rate is achieved from the classifier, where spatial information is included with the structural properties of the strokes while it is 5% less in the classifier, where spatial information is not in use.

Not only the recognition rate of the classifier, the recognition speed is also counted under the classifier's performance. As the classifiers use matching procedure for identification of test strokes, the speed is variably determined from one stroke to another. The number and size of strokes in the test letter and templates affect the recognition speed. In addition, degree of dimension of the feature vector sequence plays a crucial role. Hence, different recognition speeds are achieved. However, 12 seconds per character is the fastest average speed of the classifier among a series of experiments.

The methodology used in this classification engine is general and hence flexible in adding extra symbols in Nepali and can be extended to other scripts as well.

Table of Contents

Chapter	Title	Page
	Signature Page	i
	Acknowledgement	ii
	Abstract	iii
	Table of Contents	iv
	List of Figures	ix
	List of Tables	xiii
1	Introduction	1
	1.1 Handwriting Recognition	1
	1.2 The Statement of Need	2
	1.3 Goal	3
	1.4 Objectives	3
	1.5 Scope	4
	1.6 Thesis Overview	4
2	Nepalese Handwriting and Recognition Framework	5
	2.1 Introduction to Nepalese Alphanumeric Characters	5
	2.2 Handwriting Properties and Major Problems in Recognition of On-line Handwritten Nepalese Characters	6
	2.3 Recognition Technology Improvement	8
	2.4 Classification Approaches	11
	2.4.1 Structural and Statistical Classification	11
	Fisher's Linear Discriminant	13
	Logistic Regression	14
	Naive Bayes Classifier (Idiot's Bayes)	16
	Perceptron	17
	Quadratic Classifier	20
	k-Nearest Neighbor Neighbor classifier (k-NN)	20
	Neural Network (NN)	22

Bayesian Network (BN) 23
Support Vector Machine (SVM) 24
Hidden Markov Model (HMM) 27
2.5 Typical On-line Character Recognition Framework 30
2.6 Basic Tools/Techniques 30
2.6.1 Digitizer Technology 31
2.6.2 Pre-processing 32
Sampling 32
Noise Elimination 33
Normalization 34
Repetition Removal 34
2.6.3 Features 35
2.6.4 Distance Metrics 38
Euclidean 39
Euclidean Squared 39
Manhattan 39
Pearson Correlation 39
Pearson Squared 40
Chebychev 40
Spearman Rank Correlation 40
Dynamic Time Warping (DTW) 41
2.6.5 Clustering 42
K-means Clustering Algorithm 44
Fuzzy C-Means Clustering Algorithm (FCM) 45
Mixture of Gaussians Clustering Algorithm 45
Hierarchical Clustering Algorithm: Agglomerative and Divisive 46
2.6.6 Cross Validation (XV) 48

3 A Structural Approach on a Template-based Handwritten Character Recognition 51
3.1 System Design: Recognition of Nepali Stroke Number and Order Free Natural Handwritten Character 51
3.1.1 On-line Handwritten Data 51

3.1.2 Data Pre-processing 52
3.1.3 Feature Extraction 53
3.1.4 Data Reduction 54
Similarity Measure/Distance Calculation 55
Single-Linkage Agglomerative Hierarchical Clustering 55
3.1.5 Templates' Management 58
3.1.6 Classification/Recognition 59
3.2 Experimental Set Up 61
3.3 Experiment I: Character Recognition 62
3.3.1 Dataset 62
3.3.2 Results 64
3.4 Experiment II: Alphanumeric Character Recognition 64
3.4.1 Dataset 64
3.4.2 Results 65
3.5 Discussions 65
3.6 Experiment III: Inclusion of Pre-processing and its Effect in Recognition 68
3.6.1 Dataset 69
3.6.2 Results 70
3.7 Discussions 70
3.8 Experiment IV: Inclusion of Pen Position with Pen Direction as the Feature 71
3.8.1 Dataset 71
3.8.2 Results 71
3.9 Discussions 71
3.10 Experiment V: Appropriate Classifier Selection: Cross Validation (XV) 73
3.10.1 Dataset 74
3.10.2 Results: Five-fold Cross Validation 74
3.11 Discussions 75
3.12 Major Difficulties, Limiting factors, Needs and Possible Improvements 75

4 A Structural Approach on a Template-based Handwritten Character Recognition along with the Additional Use of Strokes' Spatial Information 77

4.1 System Design: Spatial Relation between/among the Strokes based Clustering - A Modern Approach to Writer Independent Character Recognition. 77
4.1.1 Spatial Relation between/among the Strokes Based Clustering 78
4.1.2 Templates' Management 81
4.1.3 Weight Determination 81
4.1.4 Classification/Recognition 83
4.2 Experiment VI 86
4.2.1 Dataset 86
4.2.2 Results 86
4.3 Discussions 87

5 Issues in Classifier's Performance 90
5.1 Difficulties, Limitations and Scope 90

6 Conclusions and Future Directions 93
6.1 Conclusions 93
6.2 Future Directions 94

Bibliography 95

Appendix A: Nepali 101
A.1 Language Overview 101
A.1.1 Learning Strategies: Review 103
A.1.2 Dictionaries 103
A.2 Characters, Numerals, words and Sentences 104

Appendix B: Glossary 105
B.1 Natural Input 105
B.2 Uni-stroke Input 105
B.3 Boxed Input 106

Appendix C: Source Code 107
C.1 Input Frame Design 107
C.2 Alignment of Two Non-linear Sequences and Averaging 113
C.3 Agglomerative Clustering 115

C.4 Feature Vector Sequence 117

Appendix D: List of Publications 119

D.1 International Journal 119

D.2 National Journal 119

D.3 International Conferences 119

List of Figures

Figure		Page
2.1	Some examples of natural handwriting of 46 classes of alphanumeric characters with their respective codes	6
2.2	Four samples of syllable	6
2.3	A sample of a complete sentence in natural handwriting	7
2.4	A demonstration of variable natural handwriting for the first consonant क	8
2.5	Two demonstrations of projection	14
2.6	Projection onto one-dimensional subspace	15
2.7	Projection onto vertical line and horizontal along with variance measurement	15
2.8	A perceptron	17
2.9	The decision surface represented by two-input perceptron	17
2.10	A sigmoid threshold unit	18
2.11	Two methods for fitting quadratic boundaries, quadratic decision boundaries (using LDA in five dimensional space; $x_1, x_2, x_1x_2, x_1^2, x_2^2$) on the left and quadratic decision boundaries (using QDA) on the right	20
2.12	Classification of query instance x_q in the sets of both positive and negative training examples by using different value of k	21
2.13	Feed-forward neural network with single hidden layer	23
2.14	Demonstration of Bayesian network with the help of chain rule for gathering knowledge of variables by using probability	24
2.15	Margin measurement for an optimal hyperplane	25
2.16	Measurement of absolute distance from an example x to boundary $g(x) = 0$	25
2.17	Use of slack variable in non-linearly separable case and its implication	26
2.18	Demonstration of non-linear mapping in case of low-dimensional space	27
2.19	A simple architecture of HMM	28
2.20	HMM topology for a character	29
2.21	A $(x_t, y_t/\theta_t)$ HMM architecture with pen direction feature (θ_t) and pen coordinate feature (x_t, y_t) at each self transition and inter-state transition respectively	29
2.22	A (x_t, y_t, θ_t) HMM architecture	29

2.23 A basic block diagram of a template-based on-line character recognition system 30

2.24 A sample of Jitter 33

2.25 Words with different orientations and skews 33

2.26 A sample of slope measurement along the pen trajectory 34

2.27 Feature design with the help of directional codes (Lee et al., 2000) (line segment approximation) 36

2.28 Feature design (Gao et al., 2000) with the help of standard strokes 36

2.29 Malik's proposed components of Urdu character 37

2.30 Sanparith's proposed elementary strokes for on-line handwriting. Elementary strokes are designed by 12 straight lines (es1 to es12), 12 convex strokes (es13 to es24) and 12 concave strokes (es25 to es36) 37

2.31 Zoning information from 8 different regions 38

2.32 Examples: (a) Euclidean, (b) City-Block and (c) Chess-Board distance measures 39

2.33 An example of warping path during the alignment of two non-linear sequences by using DTW 42

2.34 A sample of dendrogram providing an idea of agglomerative and divisive clustering algorithms 47

2.35 Types of agglomerative hierarchical clustering 48

2.36 Holdout Cross Validation 49

2.37 *K*-Fold Cross Validation 50

2.38 Leave-One-Out Cross Validation 50

3.1 A sample of complete specific stroke pre-processing 52

3.2 Feature selection 53

3.3 An example to determine the distance between two non-linear sequences using DTW technique 56

3.4 A graphical measurement of similarity/dissimilarity between the numerals pairs १ and २ 57

3.5 A graphical agglomerative hierarchical clustering demonstration with significant cluster threshold by taking one of the numeral classes २ 58

3.6 A graphical demonstration of grouping similar strokes from every character within a class क. Each encircled group in क-frame produces a representative stroke 59

3.7 Confusion pairs and their reasons of similarity 62

3.8 A rejected sample (क) due to very long descender 63

3.9 A sample of similar characters for humans to read 64
3.10 Three samples of characters having diminished descenders 66
3.11 Effect of pre-processing in correct recognition 69
3.12 Two rejected samples (भ and क) due to rewriting strokes 71
3.13 A complete comparison of four classifiers based on their accuracies 74

4.1 Four samples having multiple straight sequences 78
4.2 Threshold range $(St - Cs)$ determination for shirorekha 79
4.3 Spatial relation between two strokes of the character क 80
4.4 A real example of how spatial relation is determined by taking a character क 80
4.5 Grouping of characters of a specific class अ based on the number of strokes used 81
4.6 Clustering technique for two-stroke अ 82
4.7 Template management for a class of character अ having many groups 82
4.8 Correct recognition of confusion character म (भ) by handling spatial information of the strokes 83
4.9 Confusion between य↔च 83
4.10 Two confusions pairs: द→ढ 85
4.11 Two confusions pairs: स→ग 85
4.12 Misclassified example: न→म 86
4.13 Misclassified example: ल→ त, which are looking different in printed format 88
4.14 Confusion of झ with स even though a lot of dissimilar feature in between them in printed format (झ→स) 88

A.1 Both full and half Consonants 101
A.2 A Few samples of Conjunct Consonants 102
A.3 Vowels 102
A.4 Numerals 102
A.5 Sample of Vowels, as a modifier for consonant 102
A.6 A few samples of words formation 102
A.7 A few samples of complete sentences 102

B.1 Natural Writings 105

C.1 Input Window 107

C.2 Discrete Warping Path 115
C.3 Distance Matrix 115
C.4 Clustering Results - Method: ‘Single’ and k =4 116
C.5 A sample of a stroke 117

List of Tables

Table		Page
3.1	Experimental Results (System I)	61
3.2	Experimental Results (System II)	63
3.3	Experimental Results	65
3.4	Error Analysis (Test Data)	66
3.5	Experimental Results	68
3.6	Error Analysis (Test Data)	69
3.7	Experimental Results	72
3.8	Error Analysis (Test Data)	73
3.9	Experimental Results (Five-fold Cross Validation)	75
4.1	Experimental Results	84
4.2	Error Analysis (Test Data)	86
4.3	Class-wise Experimental Results (Test Data)	87

Chapter 1

Introduction

1.1 Handwriting Recognition

Human eye can see and read what is written/displayed either in natural handwriting or in printed format. The same work in case the machine does is called handwriting recognition, which is the common definition. Handwriting recognition has been a popular area of research since few decades under the purview of pattern recognition and image processing. Handwriting recognition can be broken down into two categories: off-line and on-line.

- *Off-line*:

  Off-line character recognition takes a raster image from a scanner (scanned images of the paper documents), digital camera or other digital input sources. The image is binarized through threshold technique based on the color pattern (color or gray scale), so that the image pixels are either 1 or 0.

- *On-line*:

  In on-line, the current information is presented to the system and recognition (of character or word) is carried out at the same time. Basically, it accepts a string of (x, y) coordinate pairs from an electronic pen touching a pressure sensitive digital tablet.

Once the image is binarized in off-line case, the rest of the techniques for classification can be identical with only two basic differences. Firstly, off-line recognition happens after the writing completes and the scanned image is pre-processed. Secondly, it has no temporal information associated with the image due to which it is not known to the classifier about the way and order of writing. So, we can say, its knowledge about the character is limited. It means, off-line data only represents the final result after writing i.e an image.

Why do the global market demand follow the on-line recognition system? On-line handwriting recognition system, by contrast, captures the temporal or dynamic information of the writing, enhances the accuracy over off-line. Another advantage is interactivity, which means recognition errors can be corrected immediately with the series of test. Yet, adaptation of any drawings of character is also an advantage over off-line. when the user faces that some characters are not recognized accurately, user can alter his way of drawing until it recognizes. It means user can adapt to the machine. Conversely, recognizers are capable of adapting users' drawing, usually by storing a possible samples from a large number of users for subsequent recognition. Thus, there is adaptation of user to machine and of machine to user. Electronic pen input is the direct method to compare with the both off-line and keyboard entry to the system having recognition intelligence. In addition, on-line recognition

improves the work-flow, the information is immediately available. However, the natural and comfortable style in writing effectively reduces difficulty at the threshold of using computers for common users. Moreover, it is recently showed that handwriting input is the most acceptable and welcomed input style.

A handwriting recognition can further be broken down into two categories of writer independent and writer dependent.

- *Writer Independent and Writer Dependent*:

  A writer independent recognition system recognizes a wide ranges of possible writing styles, while a writer dependent recognition system is trained to recognize only from specific users. Therefore, a writer dependent recognition system works on data with a smaller variability and therefore a chance of having higher reliability is achieved in contrast to writer independent recognition system. Writing one's style brings unevenness in writing units, which is the most difficult part. Variability in stroke numbers, their order, shape and size, tilting angle and similarity among characters from one another are found more often in writer independent recognition system. Broadly, there are two kinds of writing styles. They are hand printed and cursive handwriting. In cursive style, strokes are deliberately linked forming one from many to draw the character, while in hand printed style possible number of strokes are used, each stroke has significant role to complete the character. In cursive style, the important information such as intersections, loops, curves, straight lines and hooks etc. are missing. Some times, both writing styles are mixed. Natural handwriting contains all types of styles in writing from any of the users. Specifically, the writing is said to be natural as if user writes on a piece of paper.

With the introduction of portable hand held computers and computing devices such as PDAs (Personal Digital Assistant), non-keyboards and non-keypads based methods for inputting data are receiving more interest in both academic and commercial research communities, which are often writer independent. The most promising options are pen based and voice based inputs. Pen based method in inputting can be either off-line or on-line.

1.2 The Statement of Need

One can imagine how easy lives we have in this busy dynamic world in case the portable machine can understand what we write either in discrete or in natural handwriting mode. It would certainly be difficult that writing/typing addresses, memos, important information and communication as well for those who are non natives to English. In such a case, writing would be more cleared and easy to understand in their own local languages. In addition, it is helpful for those such as, computer novices, old people in using computer conveniently without the use of both keyboard and keypad. Therefore, the system having the intelligence in recognizing the natural handwriting for all possible scripts around the world is the global market demand.

Frequently, handwritten data is entered into IT solutions by human operators using keyboards. One can think of the processing of the forms, questionnaires and notes etc. not only

costlier but also time consuming. The easiest solution is the handwriting recognition system, which converts handwritten data into the format that can be used in further computing. Therefore, handwriting recognition is the connector between handwritten information and the IT world.

Despite many years of research in the field of handwriting recognition technology, IT has not reached the masses in local languages. Nepali is the one. According to the most recent official census, conducted by His Majesty's Government of Nepal (HMG) in 2001, Nepali is the mother tongue for 11 million people, which reflects the need of building a complete handwriting recognition system with maximum reliability.

A pencil and paper are often preferable for every one to use during the first draft preparation instead of using keyboard and other computer input interfaces like this. In such a case, handwriting recognition has gained the advantages. In addition, the languages having a large set of symbols, alphabets and numerals, designing a keyboard is bulky and cumbersome to use as well. For example, Nepali, Chinese and Japanese are under this category. Specifically, the basic set of symbols in Nepali consists of 31 pure consonants, 13 vowels, 16 modifiers and 10 numerals. In addition to these, consonants occur together in clusters, often called conjunct consonants. Modifiers appear on the top, at the bottom, on the left and to the right of the consonants and vowels, forming syllables. The script's rich of set of conjuncts make it complex to read and write. Altogether, more than 500 different symbols are used in Nepali. As, characters are derived from the mentioned basic sets, its input from the keyboard is cumbersome. However, new pen tablets offer the possibility of online handwriting when combined with handwriting recognition technology. The best way to solve the need is not to design the keyboard but to design the complete writer independent natural handwriting recognition system, such that one can write in one's writing style. In case the keyboard is designed, one needs to practice to use but no extra task is necessary for one to write one's style. To the best of my knowledge, the proposed recognition system will be one of the contributions for Nepali handwritten character recognition.

1.3 Goal

A novel approach on a template-based on-line writer independent Nepali natural handwritten character recognition by using both structural and spatial information is proposed.

1.4 Objectives

- To build a complete prototype classifier based on structural approach for natural handwritten characters.
- To design a stroke number and order free handwritten alphanumeric character recognition system.
- To determine the effects of pre-processing and feature selection in classification of natural handwritten characters.

- To handle spatial information about the strokes within a character for enhancing the reliability of the prototype classifiers.

1.5 Scope

The market of small hand-held Personal Computers (PCs) and pocket PCs is growing very fast in the recent years. With the advent of Personal Digital Assistant (PDA), smart phones and tablet PC's, rely on natural handwriting input, real-time handwriting has gained an immense importance. Writing short memos, ideas, editing simple texts, storing addresses, telephone numbers and so on are the today's natural input based users' demand. Not only this, many computationally intensive applications, like video editing and complicated mathematical calculations are also possible. As the electronic pen is turning to be more popular, the demand of handwriting input from intellectuals such as scholars, business managers, professors and doctors etc. is getting stronger. Within the scope of handwriting recognition, the focus of this research is on two problems, namely, the classification of natural Nepali handwritten alphabets and numerals.

Both from the commercial and academic fields have raised vital efforts and developments as well to make faster, easier, and more precise handwriting recognition system. However, the recent use of portable pen based computer has not reached the line of global demand. Recently, varieties in handwriting classifiers have very limited functions, because they were build with the assumption that people would use simply for writing short memos, storing addresses etc. This largely limits the things what the hand-held devices can do.

1.6 Thesis Overview

An overall thesis is outlined in the following chapters. The state of the art of off-line and on-line handwriting recognition during a period of renewed activity for Devanagari script is presented in chapter 2, which is based on extensive review of literature, including journal articles, conference proceedings and technical reports, major difficulties in on-line handwriting (especially in Nepali) with comprehensive ideas about the basic classification approaches (structural and statistical information), techniques and tools used in on-line handwriting character recognition. It provides reader a great chance to know about the varieties of works related to digitizer technologies, pre-processing strategies, feature selection techniques and model evaluation. Chapter 3 provides a complete proposed methodology in classifying a cursive characters along with the results from a series of experiments. Chapter 4 begins with novel idea in building a competitive classifier, which provides real solutions of some major difficulties. A constructed prototype classifier is template-based using structural properties of the on-line handwritten strokes. Some of the shortcomings, which are to be cleared are fixed with the addition of spatial information of the strokes with the introduction of newly proposed strategy in handling the spatial information of the characters. A discussion of the entire task is provided in every possible section for better understanding along with real time experimental results. Both global and local limitations of the on-line handwriting recognition system are critically analyzed in chapter 5. Chapter 6 concludes the thesis including recommendations.

Chapter 2

Nepalese Handwriting and Recognition Framework

2.1 Introduction to Nepalese Alphanumeric Characters

Around 11th century AD., Devanagari script was descended from Brahmi script. It is believed that Brahmins invented Devanagari script in order to conceal knowledge from the common person. Non-Brahmins were not able to learn due to the difficult nature of the script, which enabled the Brahmins to maintain their stranglehold over the non-Brahmin races and crushed them into a sub-human existence. Devanagari is derived from two roots: 'deva' and 'nagari'. 'deva' means 'deity' and 'nagari' means 'city'. Together, it implies both religion and urbane. Many countries such as, Nepal, India (many languages), and others used this script as an official script. Over 500 million people around the world use Devanagari script. The basic set of symbols in Nepali consists of 33 pure consonants and also half forms, 13 vowels, 16 modifiers and 10 numerals. In addition to these, consonants occur together in clusters, often called conjunct consonants. The script's rich of set of conjuncts make it complex to read and write. Modifiers appear on the top, at the bottom, on the left and to the right of the consonants and vowels, forming syllables. A syllable sometimes can be formed by using a vowel without any modifiers but not from consonant alone. More than 500 symbols are used in Nepali. Each symbol can be either from a vowel or a combination of consonant and vowel. In other words, a syllable is sometimes known by a symbol.

Nepali is written from left to right with the use of horizontal line often called 'shirorekha'. Every character requires one 'shirorekha' by which the text is suspended. However, looking into the users' natural handwriting, the shirorekha may not always be horizontal (it may be a line with either negative or positive slope) and on the top exactly (sometimes it may intersect with the text) . Not only this, it may be a small curve sometimes, which acts as a 'shirorekha' but not the text. In order to build a word, characters/symbols/syllables are joined by a top horizontal bar, which creates an imaginary line (shirorekha). The completeness of one sentence is represented by single or double vertical line at the end often called 'purnaviram'. Nepalese handwriting can be considered as an example of cursive handwriting. A set of natural handwritten characters from a group of users are shown in Fig. 2.1. In addition, a sample of writing a syllable, a word and a complete sentence are demonstrated in Fig. 2.2 and Fig. 2.3.

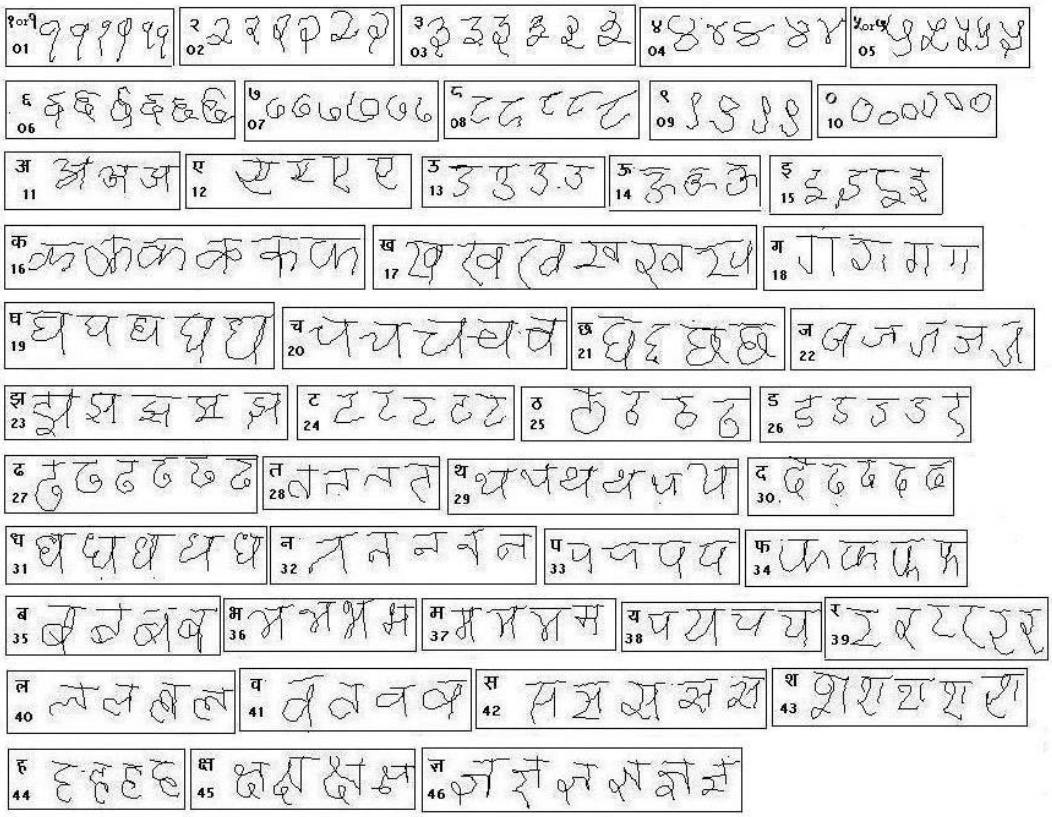

Figure 2.1: Some examples of natural handwriting of 46 classes of alphanumeric characters with their respective codes

2.2 Handwriting Properties and Major Problems in Recognition of On-line Handwritten Nepalese Characters

Handwriting recognition is notoriously a difficult task. However, it has been investigating since many years and has been put into a general framework, under pattern recognition and image processing. One of the main difficulties in online character recognition is, writing that looks identical with the printed characters (off-line graphical representation) may have a chance of misclassification because of different sequential representation (on-line). This is not only the problem of Nepali but also of other scripts as well. Main problems associated with on-line handwritten character recognitions are classified into following categories:

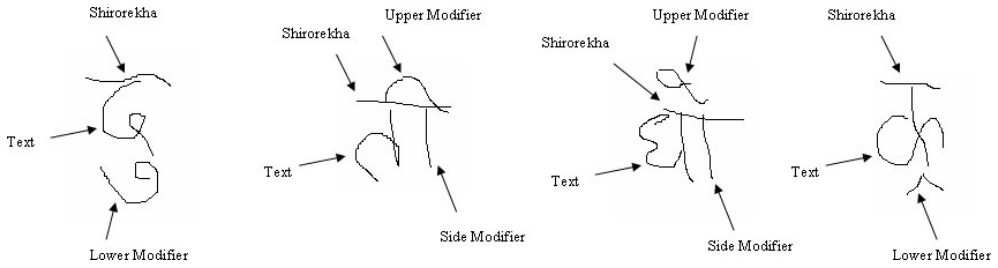

Figure 2.2 Four samples of syllable

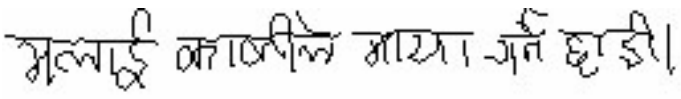

Figure 2.3 A sample of a complete sentence in natural handwriting

- *Similarity in drawings among many classes of characters*:

  Many of the characters are similar to each other in structure. In addition, users writes two or more classes of character in a similar way with the use of same number of strokes in a same direction and their order, which can be the most difficult part to classify correctly.

  - In Nepali, (क, फ), (छ, ध), (य, प), (स, झ), (ढ, द), (इ, ड), (ठ, ट), (भ, म), (ध, घ), (थ, य), (न, त), (व, त), (च, थ) and (ढ, ट) etc. are the some similar characters' pairs, which are hard to analyze their dissimilarities from one another in case of natural handwriting.

- *Writing Units*:

  The numbers of strokes, their order, shapes and sizes, directions, skew angle etc. are the considerable writing units. Writing units are always vary from one user to another. There is no guarantee that same user always writes in a same way.

  - Strokes may vary in direction in different peoples' styles.

  - The length of a string of 2D coordinates of a specific stroke is variable.

  - The number and order of strokes are variable within a specific letter. The number of strokes to complete a character varies from one to four (without modifiers) in case of Nepali(according to our dataset).

  - The size (big or small) of a specific stroke is variable.

  - The speed in writing is variable, which determines how densely the coordinates are collected and what shapes (either smooth or rough) of the strokes are.

  - Writings can be tilted by some angle with respect to the horizontal line (either on the left or to the right).

  Even when, numbers, order, shape and size, direction of strokes and speed varied from time to time writing even within the same class of character and from the same user, the writing samples are readable. Some examples are shown in Fig. 2.4. The degree of variation depends on the style and speed of writing. In Nepali, variable number of strokes are used for the same character to complete. In purely cursive mode of writing, user uses only one stroke to complete the text in a character, while two or more than two strokes are used in purely discrete mode. In addition, it is not sure that the direction(s) of stroke(s) while drawing the same character in different time by using the same number of stroke(s) is/are same from the same user. Nepalese characters are mostly influenced with the variability in writing units. This is how it is regarded as one of the cursive scripts. It is therefore difficult to classify correctly not only in cursive handwriting but also in purely discrete mode. In case of purely discrete mode, a lots of strokes can be there, in which each stroke independently may not have any information about the character.

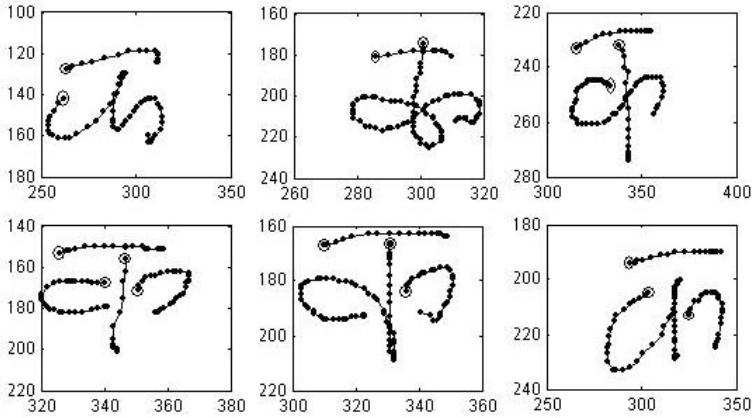

Figure 2.4 A demonstration of variable natural handwriting for the first consonant क

- *Digitizer*:

  Problem emerges into two major forms.

  - Pen's or mouse's physical position on tablet (or mouse pad, in the case of using mouse), a continuous movement as a function of time, is mapped into discrete (x, y) pixels in screen coordinates, thus there is an information loss due to digitization.

  - Noise due to physical condition of devices, for instance, dirt on the tablet surfaces and on the mouse's wheel.

- *Traditional Technologies*:

  Traditional technologies can not reach the line of the global market demand in the field of handwriting recognition technology. Most of the commercial machines are reliable only on the boxed discrete characters and character with purely discrete strokes. They are limited to natural (cursive handwriting). To the best of our knowledge, many of the existing works were concentrated on printed and off-line writings in Devanagari. From the practical experience, Optical Character Recognition (OCR) has been used in Devanagari. However, new demands are to be done in this script. The necessity of new technology along with the novel idea for efficient and faster machine in order to classify the characters and numerals is to be taken.

As a character can be written in different varieties of drawing, the fundamental properties of writing which makes communication/recognition possible is that the differences between the different character are more significant than differences between different drawings of the same character.

2.3 Recognition Technology Improvement

It is obviously difficult to scan every document character by character with the help of human eyes and brains. What do you think about repeating the same document then? There are many chances of making mistakes from the humans, then why don't you think about the machine for the same task to do. This was the motivation for all scientists in the past days. Along with this, scientists began their projects with the assumption that the machine

performed well continuously and efficiently without taking any rest. OCR (Optical Character Recognition) began as a field of research in pattern recognition, artificial intelligence and machine vision. Though academic research in the field continues, the focus on OCR has shifted to implementation of proven techniques. OCR (using optical techniques such as, mirrors and lenses) and digital character recognition (using scanners and computer algorithms) were originally considered in separate fields. Early systems required 'training' (essentially, the provision of known samples of each character) to read a specific font. Currently, though, 'intelligent' systems that can recognize most fonts with a high degree of accuracy are now common. It was started from late 1920s, G. Tauschek obtained a patent on OCR in Germany, followed by Handel who obtained a US patent on OCR in the USA in 1933. In 1935, Tauschek was also granted a US patent on his method. The machine was a mechanical device that used templates. A photodetector was placed so that when the template and the character to be recognized was lined up for an exact match, and a light was directed towards it, no light would reach the photodetector. David Shepard, in 1950, a cryptanalyst at the Armed Forces Security Agency in the United States, was asked by Frank Rowlett, who had broken the Japanese PURPLE diplomatic code, to work with Dr. Louis Tordella to recommend data automation procedures for the Agency. This included the problem of converting printed messages into machine language for computer processing. Shepard thought it was possible to build a machine, and with the help of Harvey Cook, a friend, built 'Gismo', a patent from the U.S. was issued in the end. Shepard then founded Intelligent Machines Research Corporation (IMR), which went on to deliver the world's first several OCR systems used in commercial operation. While both Gismo and the later IMR systems used image analysis, as opposed to character matching, and could accept some font variation, Gismo was limited to reasonably close vertical registration, whereas the following commercial IMR scanners analyzed characters anywhere in the scanned field, a practical necessity on real world documents. The first commercial system was installed at the Readers Digest in 1955, which, many years later, was donated by Readers Digest to the Smithsonian, where it was put on display. The second system was sold to the Standard Oil Company of California for reading credit card imprints for billing purposes, with many more systems sold to other oil companies. Other systems sold by IMR during late 1950s included a bill stub reader to the Ohio Bell Telephone Company and a page scanner to the United States Air Force for reading and transmitting by teletype typewritten messages. IBM and others were later licensed on Shepard's OCR patents. The United States Postal Service has been using OCR machines to sort mail since 1965 based on technology devised primarily by the prolific inventor Jacob Rabinow. The first use of OCR in Europe was by the British General Post Office or GPO. Surprisingly, the use of this techniques changed a lot in every aspect. In 1965, it began planning an entire banking system, the National Giro, using OCR technology, a process that revolutionized bill payment systems in the UK. Canada Post has been using OCR systems since 1971. OCR systems read the name and address of the addressee at the first mechanized sorting center, and print a routing bar code on the envelope based on the postal code. After that the letters need only be sorted at later centers by less expensive sorters which need only read the bar code. To avoid interference with the human-readable address field which can be located anywhere on the letter, special ink is used that is clearly visible under ultraviolet light. This ink looks orange in normal lighting conditions. Envelopes marked with the machine readable bar code may then be processed. During the past 25 years, the field of OCR has made significant advances in handwriting recognition (Suen et al., 2000).

Along with this linear improvement in OCR technology, users demands were leading to get

into the faster one, which were began with the introduction of PDAs. The first PDA, which provided written input was the Apple Newton, which exposed the public to the advantage of a streamlined user interface. However, the device was not a commercial success, owing to the unreliability of the software, which tried to learn a user's writing patterns. By the time of the release of the Newton OS 2.0, wherein the handwriting recognition was greatly improved, including unique features still not found in current recognition systems such as modeless error correction, the largely negative first impression had been made. Another effort was Go's tablet computer using Go's Penpoint operating system and manufactured by various hardware makers such as NCR and IBM. IBM's Thinkpad tablet computer was based on Penpoint operating system and used IBM's handwriting recognition. This recognition system was later ported to Microsoft Windows for Pen, and IBM's Pen for OS/2. None of these were commercially successful. Palm later launched a successful series of PDAs based on the Graffiti® recognition system. Graffiti improved usability by defining a set of pen strokes for each character. This narrowed the possibility for erroneous input, although memorization of the stroke patterns did increase the learning curve for the user. A modern handwriting recognition system can be seen in Microsoft's version of Windows XP operating system for Tablet PCs. A Tablet PC is a special notebook computer that is outfitted with a digitizer tablet and a stylus, and allows a user to handwrite text on the unit's screen. The operating system recognizes the handwriting and converts it into typewritten text. Notably, Microsoft's system does not attempt to learn a user's writing pattern and instead maintains an internal recognition database containing thousands of possible letter shapes. This system is distinct from the less advanced handwriting recognition system employed in its Windows Mobile OS for PDAs. In recent years, several attempts were made to produce ink pens that include digital elements, such that a person could write on paper, and have the resulting text stored digitally. The success of these products is yet to be determined. Recognition of the cursive handwriting is the hot topic in these days. The nature of some of the scripts is cursive and hence obviously the writing, which is the difficult point.

Although handwriting recognition is an input form that the public has become accustomed to, it has not achieved widespread use in either desktop computers or laptops. It is still generally accepted that keyboard input is both faster and more reliable. As of 2006, many PDAs offer handwriting input, sometimes even accepting natural cursive handwriting, but accuracy is still a problem, and some people still find even a simple on-screen keyboard more efficient. This is how the development has been placed roughly in the field of handwriting technology and reached to the streamline of users. As the trends led to re-newed efforts in directing towards more sophisticated systems, not only related to processing, feature extraction and classification stages of the methods, but also approaches like neural network (NN), mathematical morphology and hidden markov model (HMM) (Suen et al., 2000).

Despite such a huge development in handwritten character recognition, it has not reached many local languages. Nepali is the one as an instance. The contributions and the patents in case of the Devanagari script is still behind the global streamline. However, few tasks have been done, which are taken into consideration here. With the invention of OCR, Malaviya et al., 1996; Chaudhari et al., 1997 have been explored their ideas and determination in case of Devanagari OCR. As the time passed, more ideas have been explored in off-line and printed character recognition (Palit et al., 1995; Thapaliya et al., 1998; Sethi et al., 1997). Very few chances of experiencing on-line Devanagari character recognition systems until this date. Keeni et al, 1996 proposed a recognition of Devanagari characters with the help of neural

network. It reduced the number of output necessary for conventional neural network. In addition, it used an automatic coding procedure along with the heuristic method for the final classification by exploiting the structural information, resulting the intelligence of 98% for 44 classes of characters. Connell et al., 2000 proposed recognition of unconstrained on-line Devanagari characters through the combination of multiple classifiers, which focussed on both local and global features, providing an accuracy of 86.5% from 40 classes of characters (consonants and vowels). The recognition task is accomplished through the use of combination of 5 different classifiers consisting of two hidden Markov model classifiers and three nearest neighbor classifiers, in which each classifier's error characteristics determine it's level of contribution to the final classification. Hidden Markov model classifiers focussed on two different levels of features along the trace of the on-line sample point sequence and three nearest neighbor classifiers focussed on off-line features, which are extracted from the character as a whole rather than as a sequence of features corresponding to the trace of the pen. In the similar manner, Joshi et al., 2005 proposed an automatic recognition of writer dependent isolated handwritten Devanagari characters obtained by linearizing consonant conjuncts. In addition, it used structural recognition and subspace techniques, i.e. structural recognition and feature based matching are mapped to give final output. It has an intelligence of an average of 95%. Among the most up-to-date techniques, a template-based on-line handwriting recognition by using the structural properties of every stroke used in completing a character along with strokes' spatial relation is proposed in this task. A new scheme in building a prototype classifier for classifying Nepali natural alphanumeric handwritten characters, which is independent of stroke number and their order is modeled. It focusses on Dynamic Programming (DP), which provides always fixed output code as it does not depend on probabilistic properties of the character.

Both on-line and off-line classifiers are running based on their applications and importance in the field of handwriting recognition technology. Each classifier has its own scope if we look into the area of working. For instance, for scanned images, off-line classifier is used while temporal information of the sequence of images are related to on-line classification. It is possible to use off-line classifier from temporal information sequence of (x, y) 2D coordinates after plotting. Therefore, a hybrid classifier (having both on-line and off-line feature) is the global market demand for the reliability.

2.4 Classification Approaches

As handwritten recognition can be of any type (on-line or off-line, cursive or printed and writer independent or dependent), the system's performance is based on the type along with the techniques (classification). However, higher accuracy is intended either any of the classification techniques. Broadly, methods generally used in recognition system can be categorized as: structural and statistical.

2.4.1 Structural and Statistical Classification

Structural approach deals with stroke analysis (either stroke order dependent models or stroke order). The techniques such as matching, moments, characteristic points and mathematical transforms etc. are included. In this approach, single classifier and combined

classifiers are in use for higher reliability depending on the problems. Tappert, 1982 used Dynamic Programming, where matching is involved in between the time sequences of x, y coordinate data of words handwritten on an electronic tablet. Based on the results, Tappert, 1982 proved elastic matching is a promising technique for the recognition of cursive writing. Cha et al., 1999 presented the proposed algorithm, 'approximate stroke sequence matching' for character recognition and analyzed both on on-line and off-line cases. It is based on the approximation of stroke or contour sequence string matching rather than exact matching, which is useful to retrieve the similar style of handwritings with a certain degree of variations since their results are quite comparable to human subjective decision. Dynamic Time Warping algorithm is widely used from speech recognition to handwriting recognition. A new warping technique is proposed by Feng et al., 2003, for handwritten on-line signature verification, where extreme points warping is used. As the execution time is proportional to the square of the signal size (Sankoff et al., 1983), Hangai et al., 2000 defined boundary conditions in DTW matching matrix, however, the resultant computation time is relatively long. In order to recover these all, Feng et al., 2003 proposed new technique. Elastic matching has been used in Tamil handwritten character (Joshi et al, 2004) as well, from which it is confirmed that the technique has gained an immense popularity in cursive handwriting. Garain et al., 2004, used structural analysis in addition to symbol recognition in order to achieve higher accuracy for on-line handwritten mathematical expressions. They used several on-line and off-line features in the structural analysis phase to identify the spatial relationships among symbols. Interpretation of the 2-dimensional structures is the common task. The proposed system supported the entry of Devanagari (Hindi) on-line text and eventually helps to prepare scientific and technical documents in Indian languages (Garain et al., 2002).

While statistical approach deals with holistic shape information. It includes significant structural features such as skeleton and contours. It includes nearest neighbor, Bayesian, Polynomial discriminant, neural network, self organizing maps and decision tree. Nathan et al., 1995 proposed an unconstrained writer independent real time handwriting recognition by using statistical approach; Hidden Markov Model based system. A series of both fast match and detail match were taken into consideration in order to align the shape of the symbols, producing an intelligence of an average of 81% over 21,000 word vocabulary task with no grammar. Degenerate single state HMM is modeled to generate a short list of candidate hypothesis while a series of state is modeled for a character in the detail match model, where degeneracy is removed. HMM is widely used in handwriting recognition field: Okumura et al., 2005; Bellegarda et al., 1994; Starner et al., 1994; Kim et al., 1997; Connell wt al., 2000; Hasegawa et al., 2000; Nakai et al., 2001; Murakatat et al., 2004. A range of methods has been employed based on need and applications. Namboodiri et al., 2004 proposed a method to classify words and lines in an on-line handwritten document into one of the six major scripts: Arabic, Cyrillic, Devanagari, Han, Herbew, or Roman. The classification is based on 11 different spatial and temporal features extracted from the strokes of the words, giving an accuracy of an average of 95%. They used separate feature for separate script because of different writing styles from one script to another. A ranges of classifiers were used, k-NN classifier, Bayes quadratic classifier, Bayes classifier with a mixture of Gaussian densities, decision tree based classifier, neural network (NN) based classifier and support vector machine (SVM). In this approach/procedure (http://en.wikipedia.org/wiki/Statistical_classification), individual items are placed into groups based on quantitative information and training set of the labeled items. Use of statistical classification and applications are varying. Mapping of a feature space into a set of labels is one of the problems. However, this is not concrete idea

to classify as it does not provide a confidence in classification, extra post-processing block has to be added. Classification of the estimation is another problem, which can be expressed as,

$$P(class|\overrightarrow{x}) = f(\overrightarrow{x};\overrightarrow{\theta}) \tag{2.1}$$

where $\overrightarrow{x}$ is the feature vector input, f is parameterized by some parameters $\overrightarrow{\theta}$. In Bayesian, the result is integrated with every possible θ, given the training data D, instead of using single parameter vector $\overrightarrow{\theta}$.

$$P(class|\overrightarrow{x}) = f(\overrightarrow{x};\overrightarrow{\theta})P(\overrightarrow{\theta}|D)d\overrightarrow{\theta} \tag{2.2}$$

Similarly, class conditional probability $P(\overrightarrow{x}|class)$ can be determined.

Broadly, statistical classification algorithms are,

- Linear Classifier
  - Fisher's Linear Discriminant
  - Logistic Regression
  - Naive Bayes Classifier
  - Perceptron
- Quadratic Classifier
- k-Nearest Neighbor Classifier
- Boosting
- Decision Tree
- Neural Network
- Bayesian Network
- Support vector Machine
- Hidden Markov Model

Fisher's Linear Discriminant

The main idea is to project the samples into a line from different classes for classification (Veksler, 2/2/2007). Suppose, we have two classes and two dimensional samples, $x_1, x_2, \ldots x_n$, where n_1 and n_2 come from first and second class, respectively. A simple projection is shown in Fig. 2.5 (bad line to project to, is on the left while good line to project to, classes are well separated, is on the right). Scalar $v^t x_i$ is the distance of projection of x_i from the origin. The projection of sample x_i onto the direction v is given by $v^t x_i$, which is shown in Fig. 2.6. Considering the case of two classes for classifying. The separation between two classes can be measured by the use of their means. The separation can be expressed as,

$$Separation = |\widetilde{\mu_1} - \widetilde{\mu_2}| \tag{2.3}$$

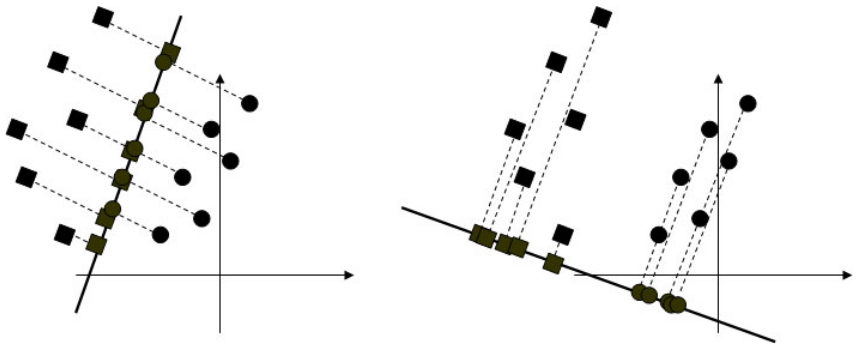

Figure 2.5 Two demonstrations of projection

where, $\widetilde{\mu_1}$ and $\widetilde{\mu_2}$ are the means of their projections from first and second class respectively. $\widetilde{\mu_1}$ can be determined as,

$$\widetilde{\mu_1} = \frac{1}{n_1} \sum_{x_i \in c1}^{n_1} v^t x_i = v^t (\frac{1}{n_1} \sum_{x_i \in c1}^{n_1} x_i) = v^t \mu_1 \tag{2.4}$$

and similarly,

$$\widetilde{\mu_2} = v^t \mu_2 \tag{2.5}$$

The question of how good is the separation, is answered by larger the $|\widetilde{\mu_1} - \widetilde{\mu_2}|$, the better is the expected separation. Moreover, projection can be either vertical or horizontal for class separability. Usually, vertical axis is better line than horizontal to project. However, $\widehat{\mu_1} - \widehat{\mu_2} > \widetilde{\mu_1} - \widetilde{\mu_2}$, which is shown in Fig. 2.7. As $\widetilde{\mu_1} - \widetilde{\mu_2}$ does not consider the variance of the classes, it needs to normalize by a factor proportional to the variance. Let $y_i = v^t x_i$, i.e. y_i's are the projected samples. Scatter for projected samples of first class and second class are,

$$\begin{aligned} \widetilde{s_1}^2 &= \sum_{y_i \in Class1} (y_i - \widetilde{\mu_1})^2 \\ \widetilde{s_2}^2 &= \sum_{y_i \in Class1} (y_i - \widetilde{\mu_2})^2 \end{aligned} \tag{2.6}$$

Hence, Fisher linear Discriminant is to project on line in the direction of v, which maximizes,

$$J(v) = \frac{\widetilde{\mu_1})^2 - \widetilde{\mu_2})^2}{\widetilde{s_1}^2 - \widetilde{s_2}^2} \tag{2.7}$$

It is guaranteed that the classes are well separated because $J(v)$ is large.

Logistic Regression

Logistic regression is a statistical regression model for Bernoulli-distributed dependent variables. It is a generalized linear model that utilizes the logit as its link function. It can be expressed as,

$$\begin{aligned} logit(p_i) = ln(\frac{p_i}{1-p_i}) = \alpha + \beta_1 x_{1,i} + \cdots + \beta_k x_{k,i} \\ i = 1, 2, \ldots, n \end{aligned} \tag{2.8}$$

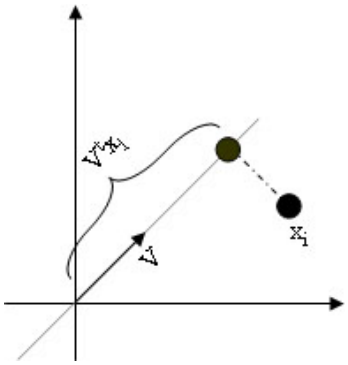

Figure 2.6 Projection onto one-dimensional subspace

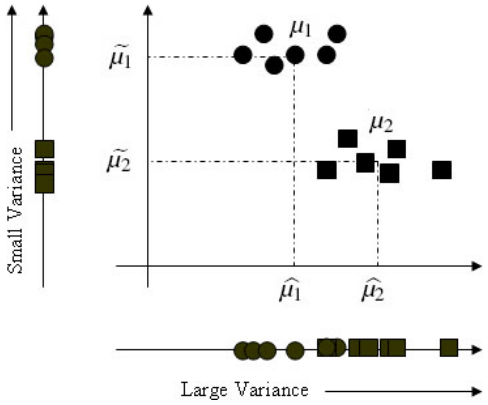

Figure 2.7 Projection onto vertical line and horizontal along with variance measurement

Instead of classifying an observation into one group or the other, logistic regression predicts the probability that an indicator variable is equal to 1. To be precise, logistic regression equation does not directly predict the probability that the indicator is equal to 1. It predicts the log odds that an observation will have an indicator equal to 1. The odds of an event is defined as the ratio of the probability that an event occurs to the probability that it fails to occur. Thus,

$$Odds(indicator = 1) = Pr(indicator = 1)/[1 - Pr(indicator = 1)]$$

or,

$$Odds(indicator = 1) = Pr(indicator = 1)/Pr(indicator = 0) \tag{2.9}$$

The log odds is just the (natural) logarithm of the odds. Probabilities are constrained within 0 and 1 with the neutral value 0.5. This makes that it is impossible to construct a linear equation in order to predict probabilities. On the other hand, odds lie in between 0 and +∞ with 1 as the neutral value for which both outcomes are equally likely. Odds are symmetric and similarly log odds too, which lies in the range of -∞ to +∞. The logarithm of the odds (the probability divided by one minus the probability) of the outcome is modeled as a linear function of the explanatory variables, X_i. This can be written equivalently as,

$$p_i = Pr(Y_i = 1|X) = \frac{e^{\alpha+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}{1+e^{\alpha+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}}$$

or,

$$p_i = Pr(Y_i = 1|X) = \frac{1}{1+e^{-\alpha+\beta_1 x_{1,i}+\cdots+\beta_k x_{k,i}}} \tag{2.10}$$

where, the parameters $\beta_1, \beta_2, \ldots, \beta_k$ are usually estimated by the help of maximum likelihood.

Naive Bayes Classifier (Idiot's Bayes)

One highly practicable Bayesian learning method is the naive Bayes classifier. A descriptive explanation of Naive Bayes Classifier is found in Machine Learning by Micthell, 1997. The naive bayes classifier applies to learning tasks where each instance x is described by a conjunction of attribute values and where the target function $f(x)$ can take on any value from some finite set V. This approach is to classify new instance is to assign the most probable target value, υ_{MAP}, given the attribute values $< a_1, a_2, \ldots, a_n >$ that describe the instance.

$$\upsilon_{MAP} = argmax_{(\upsilon_j \in V)} p(\upsilon_j | a_1, a_2, \ldots, a_n) \tag{2.11}$$

According to Bayes theorem,

$$\begin{aligned} \upsilon_{MAP} &= argmax_{(\upsilon_j \in V)} \frac{p(a_1, a_2, \ldots, a_n|\upsilon_j)p(\upsilon_j)}{p(a_1, a_2, \ldots, a_n)} \\ &= argmax_{(\upsilon_j \in V)} p(a_1, a_2, \ldots, a_n|\upsilon_j)p(\upsilon_j) \end{aligned} \tag{2.12}$$

It is easy to estimate each of the $p(\upsilon_j)$ simply by counting the frequency with which each target value υ_j occurs in the training data. But it is not the feasible way in case of large set of training examples. The problem is that the number of these terms is equal to the number of possible instances times the number of target values. The naive Bayes classifier is based on the simplifying assumption that the attribute values are conditionally independent given the target value. In the sense, the assumption is that given the target value of the instance, the probability of observing the conjunction $a_1, a_2, a_3, \ldots, a_n$ is just the product of probabilities for the individual attributes,

$$p(a_1, a_2, \ldots, a_n|\upsilon_j)p(\upsilon_j) = \prod_i p(a_i|\upsilon_j) \tag{2.13}$$

Therefore, we have,

$$\upsilon_{NB} = argmax_{(\upsilon_j \in V)} p(\upsilon_j) \prod_i p(a_i|\upsilon_j) \tag{2.14}$$

where υ_{NB} is the target value output by the naive Bayes classifier.

One interesting difference between the naive Bayes learning method and other learning methods we have considered is that there is not explicit search. through the space of possible hypothesis (the space of possible hypothesis is the space of possible values that can be assigned to the various $p(\upsilon_j)$ and $p(a_i|\upsilon_j)$ terms). Instead, the hypothesis is formed without searching, simply by counting the frequency of various data combinations within the training examples.

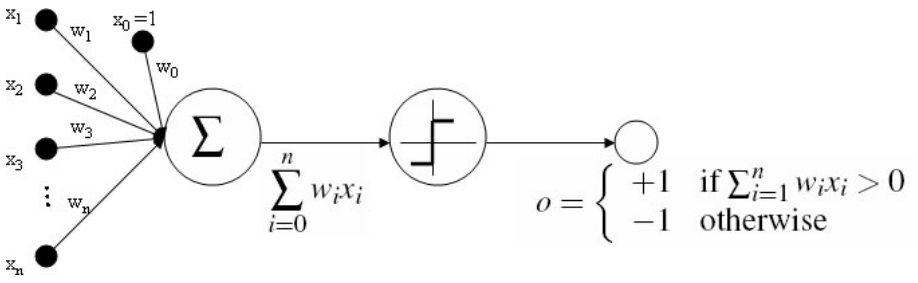

Figure 2.8 A perceptron

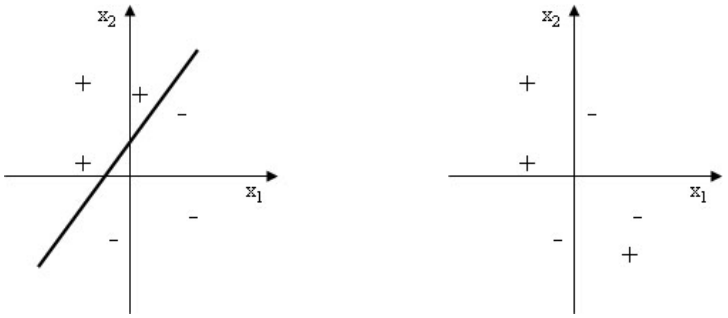

Figure 2.9 The decision surface represented by two-input perceptron

Perceptron

The Perceptron is a single layer neural network whose weights and biases could be trained to produce a correct target vector when presented with the corresponding input vector (Machine Learning: Michtel, 1997). The training technique used is called the perceptron learning rule. The perceptron generated great interest due to its ability to generalize from its training vectors and work with randomly distributed connections. Perceptrons are especially suited for simple problems in pattern classification. Usually, it learns concepts, i.e. responds with True (1) or False (0) for inputs we present to it, by repeatedly 'studying' examples presented to it.

It is sometimes know by the name binary classifier. Mathematically,

$$o(x_1,\ldots,x_n) = \sum_{i=0}^{n} w_i x_i \tag{2.15}$$

where,

$$o(x_1,\ldots,x_n) = \begin{cases} +1 & \text{if } \sum_{i=1}^{n} w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

where x is an input vector to be mapped to an output with the help of real valued weight w_i and bias w_0(consonant). $\sum_{i=0}^{n} w_i x_i = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 +, \ldots, w_n x_n$, where, $x_0 = 1$. $< w, x >$ is the dot product.

The perceptron outputs 1 for instances lying one side of the hyperplane and outputs -1 for instances lying on the other side, as illustrated in Fig. 2.8

This is the concrete theory in case of linearly separable sets of examples. Several algorithms

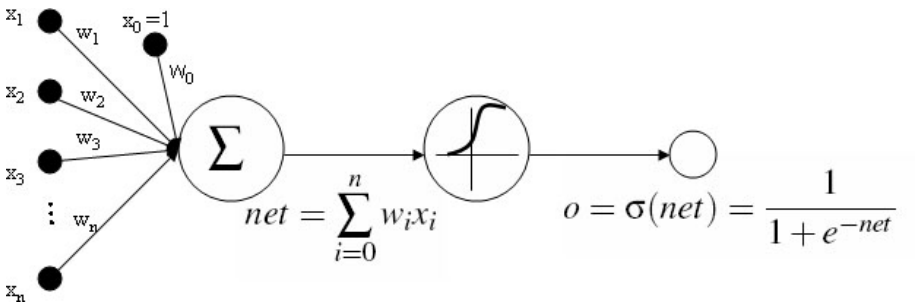

Figure 2.10 A sigmoid threshold unit

are developed in order to get the appropriate learning problem, peceptron training rule and delta rule are acceptable ones.

Weights are modified each step according to the perceptron training rule, which revises the weight associated with input x_i,

$$w_i \leftarrow w_i + \Delta w_i$$

where,

$$\Delta w_i = \eta(t-o)x_i \tag{2.16}$$

Here, t is the target output for the current training example, o is the output generated by the perceptron and η is a positive constant called the learning rate. Training example is successively classified correctly by the perceptron, whenever $(t-o)=0$ making $\Delta w_i = 0$, and then no weights are updated.

Delta rule is designed to overcome the difficulty of the perceptron training rule, in which data are not linearly separable. The key idea behind the rule is to use the gradient descent rule to search the hypothesis space of possible weight vectors to find the weight that best fit the training examples. It is sometimes called by the name unthresholded perceptron, In order to design weight, error need to be measured,

$$E(\overrightarrow{w}) = \frac{1}{2}\sum_{d\in D}(t_d - o_d)^2 \tag{2.17}$$

where, D is the training examples, t_d is the target output for training example d and $o(d)$ is the output of the linear unit for training example d. It is noticed that E is the function of $\widetilde{w}$. Not only this E is also depend on the set of training examples. The vector derivative is te=he gradient of E with respect to w,

$$\nabla E(\overrightarrow{w}) = [\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}] \tag{2.18}$$

Gradient specifies the direction of steepest increase of E, the training rule is, $\overrightarrow{w} = \overrightarrow{w} + \Delta\overrightarrow{w}$, and $\overrightarrow{w} = -\eta\nabla E(\overrightarrow{w})$. The negative sign is for decreasing the value of E. Also, it can be written as,

$$w_i \leftarrow w_i + \Delta w_i$$

where,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \tag{2.19}$$

Now, the vector of $\frac{\partial E}{\partial w_i}$ can be expressed as,

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ &= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \overrightarrow{w_d}.\overrightarrow{x_d}) \\ &= \sum_{d \in D} (t_d - o_d)(-x_{id}) \end{aligned} \tag{2.20}$$

where, x_{id} denotes the single input component x_i for training example d. Weight update rule for gradient descent is,

$$\Delta w_i = -\eta \sum_{d \in D} (t_d - o_d) x_{id} \tag{2.21}$$

Fig. 2.9 shows both linearly separable and non-linearly sets of examples, where a set of examples and a decision surface of a perceptron that classifies them correctly on the left while non-linearly separable set of examples are on the right.

Perceptron networks have several limitations. First, the output values of a perceptron can take on only one of two values (True or False). Second, perceptrons can only classify linearly separable sets of vectors. If a straight line or plane can be drawn to separate the input vectors into their correct categories, the input vectors are linearly separable and the perceptron will find the solution. If the vectors are not linearly separable learning will never reach a point where all vectors are classified properly. The most famous example of the perceptron's inability to solve problems with linearly nonseparable vectors is the boolean exclusive-or problem.

The perceptron provides discontinuous threshold, which makes it undifferentiable and hence unsuitable for gradient descent. The need is the output should be both the nonlinear function and differentiable threshold function of its inputs. Sigmoid unit is the solution of this difficulty, where the threshold output is a continuous function of its input. More precisely, the sigmoid inputs computes its output o as,

$$o = \sigma(\overrightarrow{w}.\overrightarrow{x}) \tag{2.22}$$

where,

$$\sigma(y) = \frac{1}{1+e^{-y}}$$

The sigmoid function σ is alternatively known by logistic function, which is ranging in between 0 and 1. It maps a very large input domain to a small range of outputs, thereby it is referred to as the squashing function of the unit. Its derivative now easily expressed as,

$$\frac{d\sigma(y)}{dy} = \sigma(y).(1 - \sigma(y)) \tag{2.23}$$

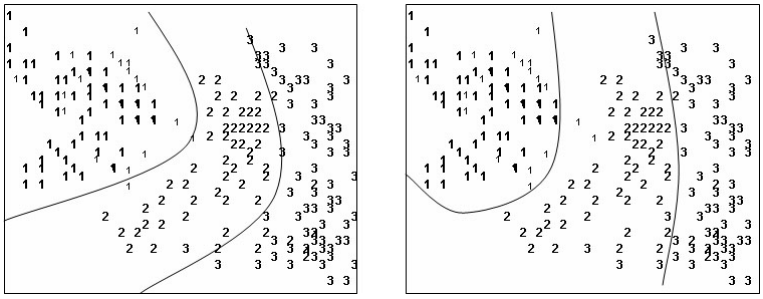

Figure 2.11: Two methods for fitting quadratic boundaries, quadratic decision boundaries (using LDA in five dimensional space; $x_1, x_2, x_1x_2, x_1^2, x_2^2$) on the left and quadratic decision boundaries (using QDA) on the right

e^{-y} is sometimes replaced by e^{-ky}, where k is a positive constant and definitely determine the the steepness of the threshold.

Quadratic Classifier

Quadratic classifier is a more general version of linear classifier. Fig. 2.11 shows the graphical classification of three sets of examples by using both Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA). For a quadratic classifier, the correct solution is assumed to be quadratic in the measurements, so y will be decided based on,

$$x^T Ax + b^T x + c$$

While Quadratic Discriminant Analysis is the most commonly used method for obtaining a classifier, other methods are also possible. One such method is to create a longer measurement vector from the old one by adding all pairwise products of individual measurements. For instance, the vector

$$[x_1,\ x_2,\ x_3]$$

would become,

$$[x_1,\ x_2,\ x_1^2,\ x_1x_2,\ x_1x_3,\ x_2^2,\ x_2x_3,\ x_3^2]$$

Finding a quadratic classifier for the original measurements would then become the same as finding a linear classifier based on the expanded measurement vector. For linear classifiers based only on dot products, these expanded measurements do not have to be actually computed, since the dot product in the higher dimensional space is simply related to that in the original space. This is an example of the so-called kernel trick, which can be applied to linear discriminant analysis, as well as the support vector machine.

k-Nearest Neighbor Neighbor classifier (k-NN)

The most basic instance-based method in classifying is the k-nearest neighbor algorithm (Machine Learning: Micthell, 1997). All the instances are assumed to the points in the

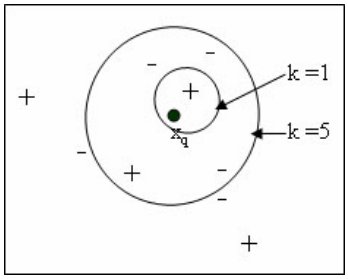

Figure 2.12: Classification of query instance x_q in the sets of both positive and negative training examples by using different value of k

n-dimensional space $\Re^n$. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. More precisely, let an arbitrary instance x be described by the feature vector,

$$< a_1(x), a_2(x), a_3(x), \ldots, a_n(x) >$$

where, $a_r(x)$ denotes the value of the r the attribute of instance x. Then the distance between two instances x_i and x_j is,

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{2.24}$$

Both real-valued and discrete-valued target function can be applied in nearest neighbor learning approach. For better understanding, discrete-valued is considered here. Let the discrete-valued target function is $f : \Re^n \rightarrow V$, where V is the finite set, $(v_1, v_2, \ldots, v_s)$. The simple algorithm goes like this.

a. For each general training example $< x, f(x) >$, add the example to the list of training examples.

b. Given a query instance x_q, to be classified, let $x_1, x_2, \ldots, x_k$ denote the k instances from training examples that are nearest to x_q.

$$f(\tilde{x_q}) \leftarrow argmax_{\upsilon \in V} \sum_{i=1}^{k} \delta(\upsilon, f(x_i)) \tag{2.25}$$

where $\delta(a,b) = 1$ if $a = b$ and 0 otherwise. As the value of $f(\tilde{x_q})$ returned as its estimate of $f(x_q)$ is just the common value of f among the k training examples nearest to x_q. If we choose $k = 1$, then 1-nearest neighbor assigns to $f(\tilde{x_q})$ the value of $f(x_i)$ where x_i is the training instance nearest to x_q. For larger value of k, the algorithm assigns the most common value among the k-nearest training examples. Fig. 2.12 illustrates the operation of the k-nearest neighbor for the case where the instances are points in 2-dimensional space and target function is the boolean valued. A query point x_q is found to be as a positive example when $k = 1$, and interestingly, it is classified as a negative example for 5-nearest neighbor algorithm.

Neural Network (NN)

Artificial neural networks (NNs) are said to be massively parallel interconnected networks of simple (usually adaptive) elements and their hierarchical organizations which are intended to interact with the objects of real world in the same way as biological nervous systems do (Dyer, 5/3/2007). The simplest interesting class of neural network is the perceptron (1-layer NN, having linear threshold unit) described above.

Multi-layer, feed forward network generalizes 1-layer network into n-layer networks. This can be done by,

- Partition units into $n+1$ layers such that layer 0 contains input units, layers 1, ..., $n-1$ are the hidden layers and layer n contains the output units.
- Each unit in layer k, $k = 0, \ldots, n-1$, is connected to all the units in layer $k+1$.
- Connectivity means bottom up connections only, with no cycles, hence the name 'feed-forward' nets.
- User defines the number of hidden layers and the number of units in each layer (input, hidden and output).
- As an example, 2-layer feed-forward neural nets (nets with 1-hidden layer) with linear threshold unit at each hidden and output layer unit, can compute functions associated with classification regions that are convex regions in the space of possible input values. 3-layer feed-forward neural nets (nets with 2-hidden layers) with an linear threshold unit at each hidden and output layer unit, can compute arbitrary functions (hence they are universal computing devices) although the complexity of the function is limited by the number of units in the network.
- Recurrent networks are multi-layer networks which in which cyclic (feedback) connections are allowed. In particular, if the ouput of a unit is connected to the other units in its layer, then this special case of cyclic connections define networks with mutual inhibition links.

Fig. 2.13 shows an example of 2-layer feed-forward neural network. Learning neural network is the important feature. Learning strategy is different from delta rule as described in perceptron learning before as hidden units do not have teacher (desired values). Back propagation learning in feed-forward neural nets is summarized below. Each training example is a pair of the form $<\vec{x},\vec{t}>$, where $\vec{x}$ is the vector of network input values and $\vec{t}$ is the vector of target network output values. η is the learning rate (e.g. 0.5) n_{in} is the number of network inputs, n_{hidden} the number of inputs in hidden layer, and n_{out} the number of output units. The input from unit i to j is denoted by $x_{j,i}$ and the weight from unit i to j is denoted by $w_{j,i}$.

Back propagation Algorithm

- Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all the networks weights to small random numbers (e.g. between -0.05 and 0.05).

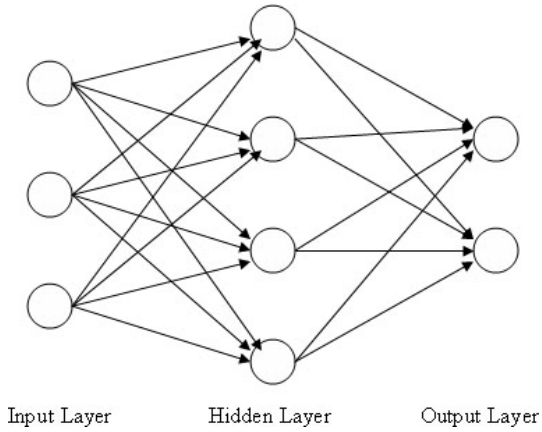

Figure 2.13 Feed-forward neural network with single hidden layer

- Until the termination condition is meet, Do

  For each $< \vec{x}, \vec{t} >$ in training_examples, Do

  Propagate the input forward through the network:

a. Input the instance $\vec{x}$ to the network and compute the output O_u of every unit u in the network.

   Propagate the errors backward through the network:

b. For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k) \tag{2.26}$$

c. For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in outputs} w_{k,h}\delta_k \tag{2.27}$$

d. Update each network weight $w_{j,i}$,

$$w_{j,i} \leftarrow w_{j,i} + \Delta w_{j,i} \tag{2.28}$$

where,

$$\Delta w_{j,i} = \eta \delta_j x_{j,i}$$

Bayesian Network (BN)

Consider a set of variables, each has a finite set of value. A set of directed arcs in between them forming a acyclic graph. Every node A has parents $B_1, B_2, \ldots, B_n$ and can be specified by,

$$p(A|B_1, \ldots, B_n)$$

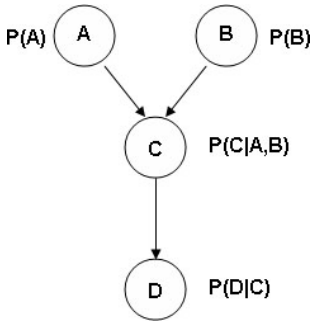

Figure 2.14: Demonstration of Bayesian network with the help of chain rule for gathering knowledge of variables by using probability

The bayesian network can be expressed with the help of chain rule. Let $V_1, \ldots, V_n$ are the variables having values $v_1, \ldots, v_n$ respectively, then

$$p(V_1 = v_1, \ldots, V_n = v_n) = \prod_{i=1}^{n} p(V_i = v_i | parents(V_i)) \tag{2.29}$$

As shown in Fig. **??**, interestingly, the element of the joint distribution is a product of terms, one for each node, expressing the probability it takes on that value given the values of the parents.

$$\begin{aligned} p(ABCD) &= p(D|ABC)p(ABC) \\ &= p(D|C)p(ABC) \\ &= p(D|C)p(C|AB)p(AB) \\ &= p(D|C)p(C|AB)p(A)p(B) \end{aligned} \tag{2.30}$$

Support Vector Machine (SVM)

The basic idea is generated some what from the Linear Discriminant and Kernel trick approaches (Veksler, 3/2/2007). Some times examples are very closed to classification hyperplane, which provides the possibility of having misclassification for new examples. The key idea behind this classification with the help of hyperplane is to maximize the distance to the closest example. An illustration of basic idea to maximize the distance is shown in Fig. 2.15, where distance to the closest positive example from an optimal hyperplane is equal to the distance to the closest negative example. The margin is now twice the absolute value of distance b of the closest example to the separating hyperplane. This will be the better generalization for linearly separable examples. Support vectors are the examples closest to the separating hyperplane. The difficulty is, we do not know which examples are the support vectors without finding the optimal hyperplane. margin can be expressed as,

$$g(x) = w^t x + w_0 \tag{2.31}$$

Absolute distance between x and the boundary $g(x) = 0$ is (Fig. 2.16),

$$\frac{|w^t x + w_0|}{||w||} \tag{2.32}$$

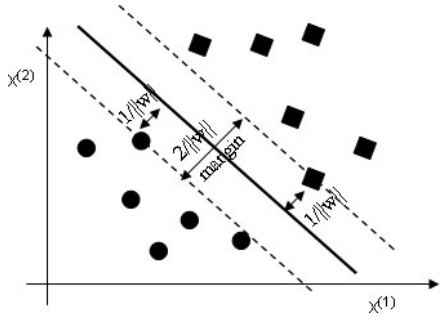

Figure 2.15 Margin measurement for an optimal hyperplane

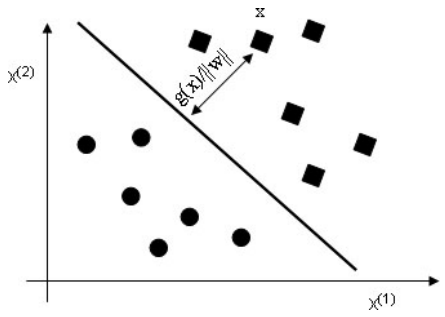

Figure 2.16 Measurement of absolute distance from an example x to boundary $g(x) = 0$

The distance is unchanged for the hyperplane $g_1(x) = \alpha g(x)$,

$$\frac{|\alpha w^t x + \alpha w_0|}{||\alpha w||} = \frac{|w^t x + w_0|}{||w||} \tag{2.33}$$

Let x_i be an example closest to the boundary. Then, $|w^t x + w_0| = 1$. This is how the largest margin hyperplane is unique. Then the distance from closest example x_i to $g(x) = 0$ is,

$$\frac{|w^t x + w_0|}{||w||} = \frac{1}{||w||} \tag{2.34}$$

Thus the margin is, $\frac{2}{||w||}$, which is shown in Fig. 2.15. There are certain constraints,

$$\begin{cases} w^t x_i + w_0 \geq 1 & \text{if } \xi_i \text{ is positive example} \\ w^t x_i + w_0 \leq -1 & \text{if } \xi_i \text{ is negative example} \end{cases}$$

Let

$$\begin{cases} z_i = 1 & \text{if } x_i \text{ is positive example} \\ z_i = -1 & \text{if } x_i \text{ is negative example} \end{cases}$$

then, the problem can be converted into, distance from closest example x_i to $g(x) = 0$ is,

$$\begin{cases} J(w) = \frac{1}{2}||w||^2 & \text{minimize} \\ z_i(w^t x_i + w_0) \geq 1 \forall i & \text{constrained to,} \end{cases}$$

$J(w)$ is a quadratic function, thus there is a single global minimum.

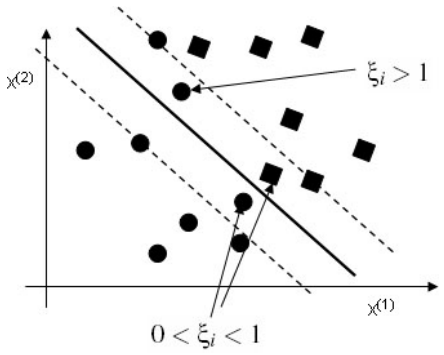

Figure 2.17 Use of slack variable in non-linearly separable case and its implication

Classification performance is still better in non-separable cases. Data should be almost linearly separable for good performance. Slack variables $\xi_1, \ldots, \xi_n$ (one for each sample) are used in case of non-separable samples. The constraints factor now changed from $z_i(w^t x_i + w_0) \geq 1 \ \forall \ i$ to $z_i(w^t x_i + w_0) \geq 1 - \xi_i \ \forall \ i$. Fig. 2.17 shows the use of slack variables and its direct implication in classifying the examples in non-separable case. Based on the value of slack variable, samples' positions are determined. It means, ξ is a measure of deviation from the ideal sample i.

- $\xi_i > 1$: sample i is on the wrong side of the separating hyperplane.
- $0 < \xi_i < 1$: sample i is on the right side of the separating hyperplane but within the region of maximum margin.
- $\xi_i < 0$: is the ideal case for sample i.

Hence, we would like to minimize,

$$J(w, \xi_1, \ldots, \xi_n) = \frac{1}{2}||w||^2 + \beta \sum_{i=1}^{n} I(\xi_i > 0) \tag{2.35}$$

where $\sum_{i=1}^{n} I(\xi_i > 0)$ is number of samples not in ideal location and

$$I(\xi_i > 0) = \begin{cases} 1 & \text{if } \xi_i > 0 \\ 0 & \text{if } \xi_i \leq 0 \end{cases}$$

which is constrained to $z_i(w^t x_i + w_0) \geq 1 - \xi_i$ and $\xi_i \geq 0 \ \forall \ i$, β is a constant. Non-linear mapping is also solved with the help of SVM by using dimension transformation criterion. Generally speaking, pattern classification problem cast in a high dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space. Fig. 2.18 gives the concrete idea behind the dimensional space transformation. In Fig. 2.18, data are projected to high dimension with the use of the function, $\varphi(s)$. The linear discriminant function $\varphi(s)$ is transformed into non-linear, $g(x) = w^t \varphi(x) + w_0$, where $\varphi(x) = (x, x^2)$.

SVM enjoys several advantages. One of the most important factor is, it avoids the curse of dimensionality problem by enforcing the largest margin permits good generalization (generalization in SVM is a function of margin, independent of the dimensionality), and computation in the higher dimensional case is performed only implicitly through the use of 'kernel' functions.

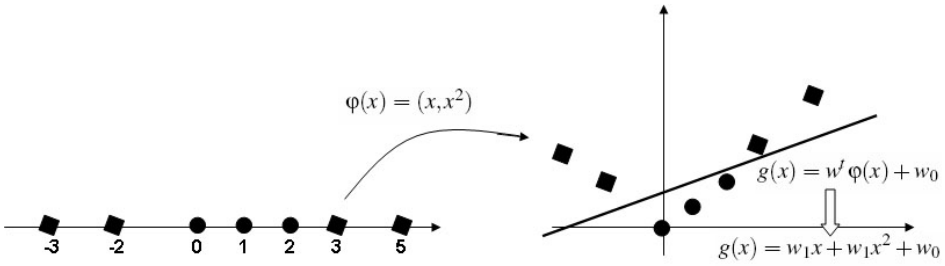

Figure 2.18 Demonstration of non-linear mapping in case of low-dimensional space

Hidden Markov Model (HMM)

A hidden Markov model (HMM) is a statistical model in which a system being modeled is assumed to be a Markov process with unknown parameters and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. A HMM can be considered as the simplest dynamic Bayesian network. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are only the parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. Hidden Markov models are especially known for their applications in temporal pattern recognition such as speech, handwriting, gesture recognition, musical score following and bio-informatics. On the other hand, A HMM process is known by a doubly stochastic process: an underlying process, which is hidden from observation process and observable process, is determined by the underlying process. A conditional state transition probability distribution following the underlying process, where a current state is hidden from observation and depended on the previous states. Observable process is determined by a conditional symbol emission probability distribution, where a current symbols depend on current state transitions. Two different events can be proposed; discrete or continuous symbol observations. Discrete symbol observation requires the input in the form of discrete symbol of feature vector, where vector quantization algorithm is in use. While, the later approach uses the variance and co-variance of the feature to estimate the probability of the occurrence of the feature vector under special feature distribution, normally Gaussian. However, the main goal of the HMM algorithm is to find the probability that the specific class how much likely to occur given a sequence of observations either discrete or continuous. The essence of the HMM is to determine the posteriori probability for a specific class of character, given an observed sequences, where the movement of one state to another state is descried by the Markov process. An architecture of a simple HMM is shown in Fig. 2.19, where it is clear that the value of the hidden variable $x(t)$ (at time t) only depends on the value of the hidden variable $x(t-1)$ (at time $t-1$). This is called the Markov property. Similarly, the value of the observed variable $y(t)$ only depends on the value of the hidden variable $x(t)$. In a simpler manner, The probability of observing a sequence $Y = y(0), y(1), \ldots, y(L-1)$ of length L is given by,

$$P(Y) = \sum_{X} P(Y|X)P(X) \tag{2.36}$$

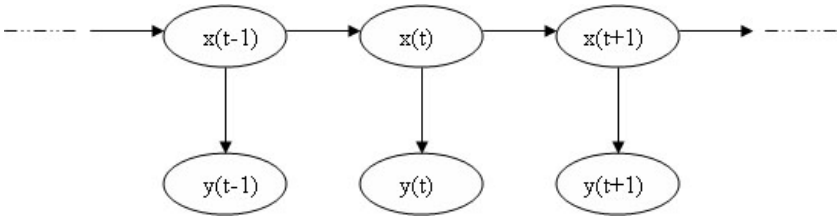

Figure 2.19 A simple architecture of HMM

where the sum runs over all possible hidden node sequences $X = x(0), x(1), ..., x(L-1)$. A brute force calculation of $P(Y)$ is intractable for realistic problems, as the number of possible hidden node sequences typically is extremely high. The calculation can however be speed up enormously using an algorithm called the forward-backward procedure. A HMM topology for a character is demonstrated in Fig. 2.20, where it contains L_i states labeled $S_1, S_2, \ldots, S_{Li}$: *Li* is the average number of frames for that character. Transitions; t_1 and t_2 result in the emission of an observation feature vector. Each character is modeled by a series of states, each of which has associated with it an output distribution corresponding to the portion of the character that it models. State transition probabilities, $p(S_i, t_j)$ and the output probability distributions, $p(f_t|S_i, t_j)$ completely specify the model. As output transition probabilities: $p(f_t|S_i, t_1) = p(f_t|S_i, t_2)$, it can be written as, $p(f_t|S_i)$. Using Gaussian distributions,

$$p(f_t|S_i) = \sum_k p(f_t|g_k) p(g_k|S_i) \tag{2.37}$$

Therefore, HMM is completely specified by the state transition probabilities and the mixture of coefficients.

A real time example in handwriting character recognition is demonstrated by many of the researchers. Among them, an on-line character recognition technique based on HMM technique is explained for better understanding (Okumura et al., 2005). It used both pen direction feature and pen co-ordinate feature separately. For instance, pen co-ordinate feature at each inter-state transition and outputs a pen direction feature at each intra-state transition. Each state specifies the starting position and the direction of a line segment by its incoming intra-state transition, respectively. A conventional HMM (θ-HMM) is graphically explored in Fig. **??** along with the possibility of designing HMM topology in Fig. 2.21. Not only in Okumura et al., 2005 model, but also a large number of tasks used the pen direction feature in conventional HMM topology (Bellegarda et al., 1994; Starner et al., 1994; Kim et al., 1997; Hasegawa et al., 2000; Nakai et al., 2001). As the number of states is equal to the number of sample points, model complexity is increased. This is one of the demerits.

The use of HMM was highlighted along with the use of spatial information (Murakatat et al., 2004) has one drawback, which is concerned with the complexity of model. In which, it was guaranteed that the accuracy of the classifier can be increased by using many models (based on HMM) for different writing styles of a specific character. But if we think about the memory management and the complexity of the model, it would not be a reliable classifier. However, in order to achieve the accuracy, many models are to be used for a particular character based on its styles in drawing.

Along with this, artificial neural network has been used widely for higher reliability of the system due to their strong discriminative power. These days, hidden markov model has been

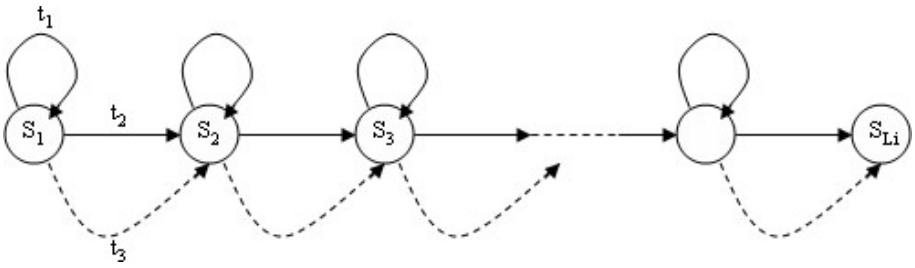

Figure 2.20 HMM topology for a character

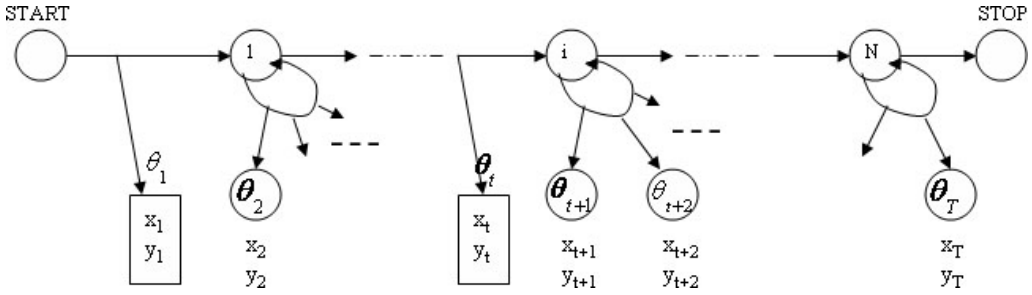

Figure 2.21: A $(x_t, y_t/\theta_t)$ HMM architecture with pen direction feature (θ_t) and pen co-ordinate feature (x_t, y_t) at each self transition and inter-state transition respectively

emerged in pattern recognition field because it provides good probabilistic representation of patterns. In homogeneous approach, multi-layer perceptrons have been used in which each classifier is trained independently with particular features, which reduces the burden in difficulties in training. In a similar manner, a hybrid classifier is proposed under heterogeneous approach. For the recognition of hand-written characters, a hybrid system is proposed by Freund et al., 2000, where neural network and structural/syntactical analysis methods are explored. Along with the existing traditional ways and approaches, a template-based approach can not be excluded. The proposed classifier for the classification of Nepali characters and numerals is based on structural features and statistical information about the strokes, which is template-based.

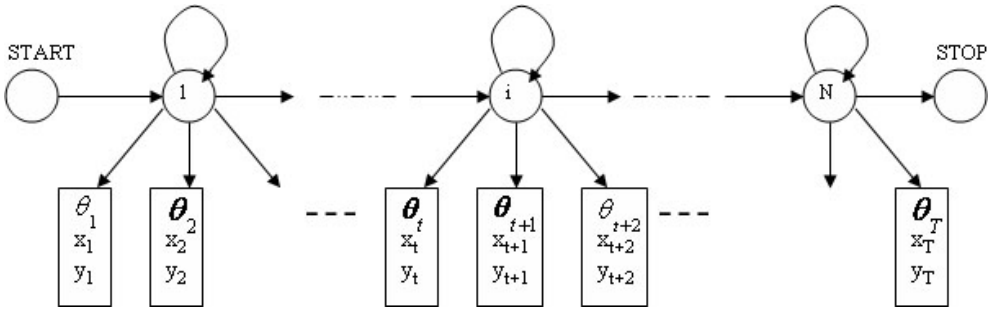

Figure 2.22 A (x_t, y_t, θ_t) HMM architecture

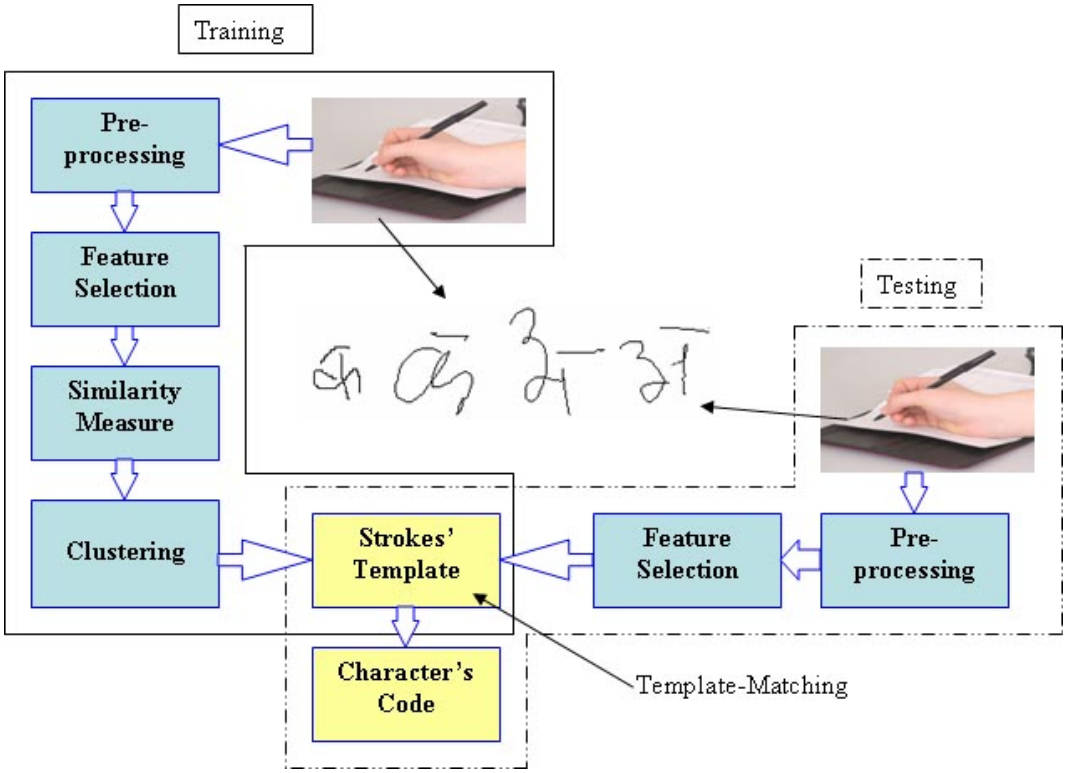

Figure 2.23: A basic block diagram of a template-based on-line character recognition system

2.5 Typical On-line Character Recognition Framework

On-line character recognition is the task of transforming a symbol represented in its spatial form of graphical marks into symbolic representation concurrently to the writing process (Bahlmann et al., 2004). The recognition system typically comprises of two stages: a training stage and a test stage. In the training stage, data are refined (pre-processing), extracted their remarkable features (feature extraction), merged similar symbols (clustering) and stored their features' representatives as training samples (templates' management). Then in the test stage, matching takes place for identifying similar features with test features in classification. A basic block diagram of a template-based on-line character recognition system is shown in Fig. 2.23. A brief explanation of every block along with the existing methodologies and techniques are presented in the following sections.

2.6 Basic Tools/Techniques

What kind of magic is required for a machine to become a competent handwriting classifier? All the components used in the classifier (varying case by case) are the requirements to fulfill the necessity of the competent classifier, which are explained below.

2.6.1 Digitizer Technology

People in various fields and generations come to handle the computers at their offices, school, homes and on the street corners etc. Thatswhy, an easier and flexible input device instead of keyboard is needed for faster processing. In such a case, computers with natural handwritten input is the possible solution, however, it is not a new in this era. A wide range of digitizer with different technologies is available in the market. In other words, use of digitizer tablets covers a wide range, based on their applications and reliability. Digital pen has been used as a human computer interface since few decades because of its flexibility in writing any kinds of texts, drawing graphics according to the users' desire. These are the cases, where the recognition block for handwritten graphics, texts and so on should be combined with the system. As there are large number of tablet digitizers, the main measuring precision is characterized by resolution, accuracy and sampling rate. A basic function of the digitizing tablet within a pen-computer is to detect the stylus on the writing surface and measure its position at its nominal sampling rate (Sandip, 2004). The sampling rate typically varies from 50-200 Hz depending on the application. Finer resolution is received with the higher sampling rate, which can accurately measure the fast strokes while reverse is the case for rough resolution. One of the interesting technologies is, the digitizing tablet is combined with the display screen, providing high level of interactivity. Users can perceive the drawing/writing on the same screen at the same time, which provides similar experience to that of drawing/writing using the conventional pen and paper. However, a special attention is needed to design a display screen along the digitizing surface.

Electromagnetic/electrostatistic tablets have x and y grids of conductors, spaced from 0.1 to 0.5 and a loop of wire in the stylus tip (Tappert et al., 1990). Either the loop or the grid is excited with an electromagnetic pulse and other detects an induced voltage or current in a sinusoidal signal. In order to determine the precise location/position, the tablet conductors are scanned to locate pair closest to the loop and interpolation is performed.

Pressure sensitive tablets have layers of conductive and resistive materials with a mechanical spacing between the layers. An electrical potential is applied across one of the resistive layers in order to set up the voltage gradients that corresponds to the position. Pressure from the stylus tip at a point results in the conductive layer picking off the voltage from the resistive layer.

Resistive technology (Scholey, 2006) offers a fast, reasonably accurate and affordable technology that recognizes the touch input from any stylus, finger, gloved hand, pen/tool. Further, due to its mainstream availability and low cost, resistive technology is the choice. It consists of a mechanical sensor mounted on the top of the display and an embedded controller. When the finger touches the surface, the surface touches the lower resistive layer and activates the signal. The control electronics alternates the voltage and hence (x, y) touch coordinates are recorded.

Another popular technology is the inductive technology (Scholey, 2006), used more than 20 years in graphics tablets and tablet PCs. A technology of single chip solutions made the tablet eight times smaller than it was before. It is comprised of printed circuit board sensor, a mixed IC controller, driver software and a stylus pen. The sensor emits an electromagnetic signal. The energy of the magnetic fields maintain the circuitry taking the energy from sensor to pen. An inductor/capacitor resonates to the frequency to determine its value. The

energy is then reflected to the sensor, where it received the analog signal. This analog signal is converted to digital signals, i.e. (x, y) coordinates.

Recently, electronic pen with the ultrasound technology is in use (Fujitsu,2005). The ultrasound is generated from the special pen and the pen position is calculated when the ultra sound propagates to a receiver that is fixed to computer screen or paper. Accurate handwriting input, precise measurements with a compact device (0.1 mm resolution) and no blind spot in the writing areas in any corners are the main challenging properties.

WACOM, summarizes both the recent properties and possibilities in electronic pens (WACOM, 2006). As electronic pen does not need battery, a range of design options in terms of the weight and size (lightness, thickness etc.). The pressure can be detected and freely be set in between the standard purpose 256 steps and maximum level of 1024 steps. Utilizing the changes in capacitance and inductance are also proposed. Magnetic field noise is one of the difficulties, WACOM's pen-abled technology reduced by the use of combination of altering the arrangements of installed parts, adjusting the frequencies of parts, so that the sensor board is not affected.

2.6.2 Pre-processing

Data, directly collected from users are often incomplete noisy and inconsistent, which are needed to be pre-processed before applying to the system in order to receive the correct classification. All the ways (techniques) to refine the data suitable for analyzing are included under the pre-processing technique. Sampling, noise elimination, discretization, integration, transformation are the basic techniques of pre-processing. Different systems use a variety of different techniques, which are varying from one script to another and depending on the goal as well. Some of the existing techniques are explained below.

Sampling

Pen-up and pen-down information is captured as an integral part of data acquisition. A string of coordinates as a function of time is recorded along the pen trajectory during the pen movement over the surface of the sensitive screen. This facilitates to track the number of strokes and their order within a character. The length of the strokes (number of coordinates in a string) varies even users write with the same speed. In order to achieve a constant number of coordinates in every string, re-sampling is necessary. One of the procedures to re-sample the sequence (Joshi et al., 2004) is explained below.

- *single stroke character*:
  The total length of the trajectory is divided by the number of intervals required after sampling.
- *multi stroke character*:
  The length of the stroke is divided by the total length of the character. In this case two steps are to be covered; one is to find the length of each stroke and next is to re-sample each stroke according to the ratio of the stroke length to the total length of the character. The length of the complete character is needed to preserve the proportion of the strokes included in a character.

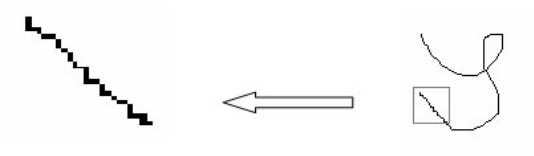

Figure 2.24 A sample of Jitter

Figure 2.25 Words with different orientations and skews

Re-sampling also helps to reduce the anomalous cases such as having a large number of samples at the same position when the user holds down the pen at a point.

Noise Elimination

A stroke inevitably contain noises, typically come from many sources. Hand fluctuation during writing and digitizing error of the input devices are the major sources. The noisy sequence in addition to the information sequence may not rigorously harm the character in off-line graphical representation, while severely affect on on-line data sequence. Varieties of different filters are used based on the kinds of noises.

Joshi et al., 2004 reduced the effect of noise with the help of 5-tap low pass Gaussian filter, where each stroke is smoothed separately.

$$\begin{aligned} x_n^{filt} &= w_n * x_n^{orig} \\ y_n^{filt} &= w_n * y_n^{orig} \end{aligned} \tag{2.38}$$

where, $w_n = \dfrac{e^{-\frac{n^2}{2\sigma^2}}}{\sum_{n=-(N-1)/2}^{(N-1)/2} e^{\frac{-n^2}{2\sigma^2}}}$ is the filter's coefficient.

Filtering can be done by the use of convolution with one dimensional Gaussian kernels (Namboodiri et al., 2004), which reduces the noise due to pen vibrations and errors in the sensing mechanism. They proposed Gaussian low pass filter for smoothing the strokes. Malik et al., 2005 proposed time domain filter by using the convolution of input sequence with a finite impulse response for smoothing a jitter appeared in the sequence (Fig. 2.24). Although many techniques/filters are used in smoothing by suppressing the noisy coordinates, it is very difficult to select the filter such that it can work equally for all strokes appeared (Sezgin, 2001).

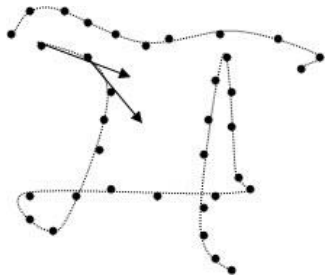

Figure 2.26 A sample of slope measurement along the pen trajectory

Normalization

Basically, recognition is better for only in case of nominal size of writing as well as standard orientation (normally horizontal) and a nominal slant (normally up-right) (Beigi et al., 1994). However, this is not happened from every writing. Therefore, the need of normalization is facing before feeding into feature extraction block. Normalization includes basic techniques like, scaling, translation and rotation etc. Writing in any slope can be changed into full upright position with the use of their base line, top line and middle line. In contrast to a word processing, changing the orientation of the character into the full up-right position is more difficult. In case of a character, determining the orientation and skew angle is the main problem especially in Nepali. In contrast to Roman script, the difficulty in determining the orientation of the Nepali word is demonstrated in Fig. 2.25. In Nepali, an additional work has to be done because the orientation and the skew of both text and horizontal line (shirorekha) are not always the same. Further, size normalization is carried out to a standard size for all characters, which is an essential case in writer independent recognition but not so crucial in writer dependent (Nathan et al., 1995).

He et al., 2005 demonstrated the role of size normalization in recognition by taking static handwritten numerals. They normalized images into different sizes and applied to the same classifier and features to observe the relationship between normalized images and recognition rates.

Repetition Removal

The digitizer is so sensitive because it detects every slight movement of the pen, even when the tip is not quite touching the digitizer but is over the plane. This will cause the co-occurrence of coordinates at a point. Further, very slow handwriting will generate repetition of coordinates at the same position, usually at the dominant points (for instance, corners). These unnecessary points are to be removed by any means for better performance of the system such as, retrieving quality image of the sequence and enhancing the speed by reducing the length etc.

Use of excessive pre-processing is undesirable because it may result in premature, limiting distortions or loss of information. In a sense, a designed filter may not appropriate for all images. It means, the techniques used in pre-screening can work for some of the characters but not for all. Sometimes, it may deteriorate the some classes of characters. Hence, the

techniques, covering wide ranges are preferred.

2.6.3 Features

If you have complete address of your friend then you can easily find him/her without an additional help of other people on the way. The case is similar in character recognition to know the stroke with the help of the feature. Here, an address refers to a feature. Therefore, the complete/sufficient feature selection from the provided input is the necessary point. Elegant feature selection can greatly decrease the workload and simplify the subsequent design process of the classifier.

Features should contain information required to distinguish between the classes, be sensitive to irrelevant variability of the input, and also be limited to permit efficient computation of discriminant functions and to limit the amount of training data required (Lippmann, 1989). It is guaranteed that quality feature selection affects the classification rate. It is easy for classifier to recognize the stroke if it has feature with sufficient distinguishing characteristics. Different systems use different varieties of feature, having different accuracies and goals as well. The feature used in one system may not fit with the other systems. Features taken from the same input data can be variably used. It is noted that the feature used in on-line handwriting is different from off-line because of the different input techniques.

The root of the on-line handwriting recognition is the real time data collection by using the sampling phenomenon. Common devices are digitizing tablets and touch pads, where the written data is digitized. Digitized data points are in the form of either two dimensional (x,y) or three dimensional (x,y,z) coordinates (z is the pressure sensitivity). Broadly, many features such as geometric features, ink related features, directional features and global features etc. are variably used from one task to another (Watt et al., 2005). On the other hand, number of strokes, length of a stroke, width-height ratio, number of intersection points, number of loops, number of hooks, point density, initial point position and direction, end point position, and direction and initial-end direction etc. are the basic features.

Toyozumi et al., 2004, proposed symbol (stroke) segmentation method for handwriting mathematical formula by using positional stroke relation and geometrical features such as shape, sizes and so forth. They clearly segmented the mathematical formula by using the knowledge of stroke neighbor relation.

Malik et al., 2005, used analytical segmentation on the input string of 2D coordinates in order to achieve the feature. The feature selected from the input string are starting and ending coordinates, slope, writing direction, size/length and hat features. Both starting and ending coordinates are used to find the size, direction and location (with respect to other) of the component. The slope is used to measure the slant line.

$$Slope = \frac{VerticalChange}{HorizontalChange} \tag{2.39}$$

Hat feature is an unique feature in Urdu, which cannot be ignored as in Roman. Along with this, characters are divided into horizontal, vertical and slant line. How did they break a character into components are shown in Fig. 2.29.

Both the pen position and pen direction features (x,y,θ) are used (Okumura et al., 2005)

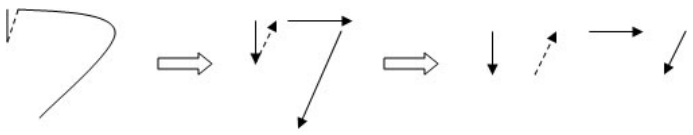

Figure 2.27: Feature design with the help of directional codes (Lee et al., 2000) (line segment approximation)

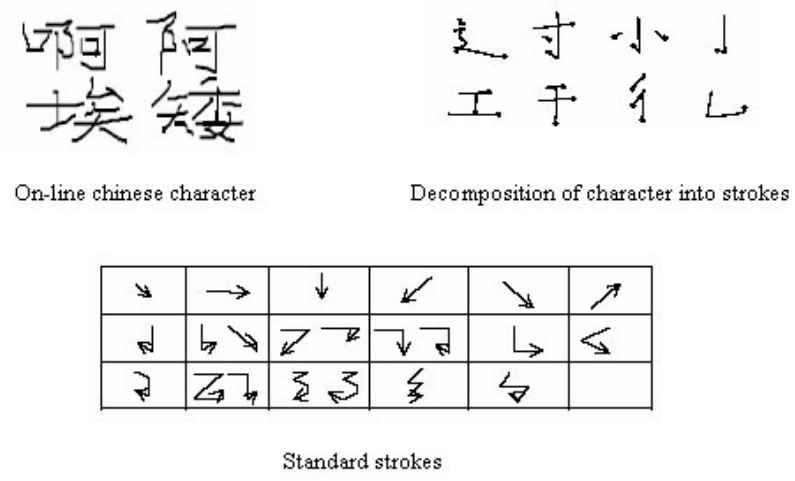

Figure 2.28 Feature design (Gao et al., 2000) with the help of standard strokes

instead of using only the string of coordinates. This is because that the pen coordinate feature is not stable. A combination of Δx, Δy, $\cos\theta$ and $\sin\theta$ (Nathan et al., 1995) comprised of a feature vector instead of taking only the pen-tip position and the tangent angle.

Verma et al., 2004, proposed a feature extraction technique for on-line handwriting, where they used directional and zoning information and combined them to make a single feature vector. Pen-down and pen-up calculation, direction of start and end point, change of writing direction, width height ratio and zone information are the parameters used in feature.

- *Pen-up*

  A stroke is a series of points from pen-down to pen-up. The number of strokes used in completing a character is equivalent to how many times the pen-down occurs. Pen-up calculation is needed as only the pen-down is not enough because the different characters may get the same number of strokes. It is used to check how well the recognized character matches to the standard one (the average in database). It is calculated as,

  $$PEN-UP = e^{|average-x|} \tag{2.40}$$

  where, x is the real strokes.

- *Zone Information*

  It is carried out by using x_{max}, y_{max}, x_{min} and y_{min} from all numbers of strokes within a character, which are used to determine the boundary of the whole character and then separated into possible regions. Each region contains distinct information. A simple zoning information is demonstrating in Fig. 2.31.

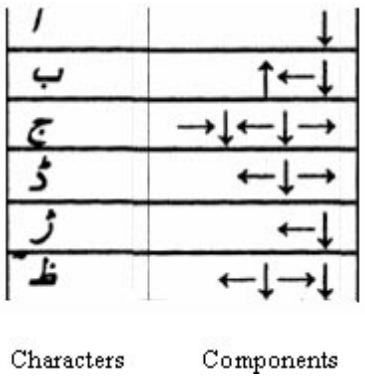

Figure 2.29 Malik's proposed components of Urdu character

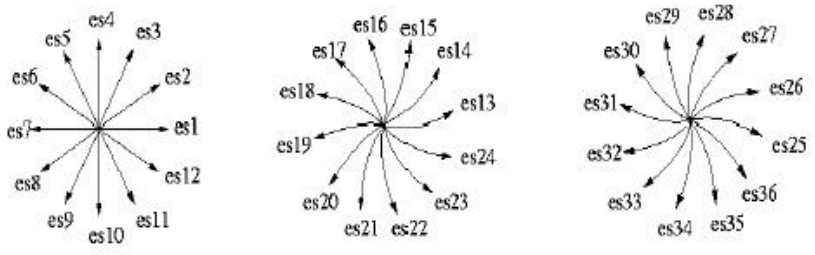

Figure 2.30: Sanparith's proposed elementary strokes for on-line handwriting. Elementary strokes are designed by 12 straight lines (es1 to es12), 12 convex strokes (es13 to es24) and 12 concave strokes (es25 to es36)

Interestingly, first and second derivatives of (x,y) coordinates, and relative change in pressure are also used as feature set (Vural et al., 2005). They also added number of neighboring points and relatives height of each point with respect to the base line. As 2D coordinates (x,y) are not translation invariant, the first and second derivatives are taken, which are calculated as,

$$dx_t = \frac{\sum_{\theta=1}^{\Theta} \theta * (x_{t+\theta} - x_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \tag{2.41}$$

where, x_t is the x coordinate at time t and Θ is the half window width. The percentage change in pressure can be determined as,

$$dp_t = \frac{p_{t+1} - p_{t-1}}{2p_t} \tag{2.42}$$

where, p_t is the pressure at time t.

Another primitive knowledge of the character is the standard strokes (structural representation). Mostly in Chinese, Japanese and Korean characters, the case is used. Since large number of symbols are used to complete a character, many standard strokes/line segments are defined as the basic components of the character. Typically, strokes as the directional arrows are of eight types. They are, $\rightarrow$ (0), $\nearrow$ (1), $\uparrow$ (2), $\nwarrow$ (3), $\leftarrow$ (4), $\swarrow$ (5), $\downarrow$ (6) and $\searrow$

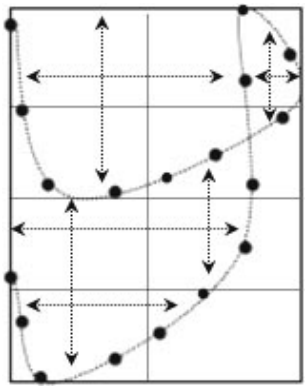

Figure 2.31 Zoning information from 8 different regions

(7), including the codes from 0-7. This can be expressed as,

$$\begin{pmatrix} \nwarrow & \uparrow & \nearrow \\ \leftarrow & \dots & \rightarrow \\ \swarrow & \downarrow & \searrow \end{pmatrix}$$

As directional arrows provide only the directional feature of the strokes/line segments, it would carry more information if the length of the standard strokes are taken into account (Cha et al., 1999). Not only in the mentioned scripts (Japanese, Chinese and Korean) but also in roman, the directional codes are used as feature (Fig 2.27: line segment approximation). In some cases, extra standard strokes (properties of arcs) are also designed Gao et al., 2000; Rowely et al., 2002. Murakatat et al., 2003 and Murakatat et al., 2004 proposed both low-level and high level elementary strokes from on-line handwriting, shown in Fig. 2.30. Also, the slope angle obtained from two consecutive coordinates along the pen movement is quantized into eight levels as in the arrow matrix above, which makes the feature invariant to noise (Joshi et al., 2004). Dominant features are also used along the sequence. Dominant points are those points in the sequence where the slope changes noticeably/drastically. This is easier for finding the change of quantized slope. Dominant points are extremely important in configuring the shape of the stroke.

Alvertos et al., 1989 used a large set of geometric information such as, existence of closed curves, number of intersections, number of free ends, horizontal and vertical symmetry, and existence of the diagonal neighbors to classify Greek character of any size. Even though a large set of geometric information about the character, this feature is reliable for only printed characters instead of natural on-line handwriting. More interestingly, on-line information is retrieved from off-line handwriting (Lallican et al., 2000), experimentally proved that the recognition rate is close to those of an on-line recognition system.

2.6.4 Distance Metrics

A quick overview of distance measurement techniques between the data points/sequences are carried out in the following subsections. It is often useful in image processing in order to calculate the distance between two pixels in an image, but this is not as easy as it seems. The presence of the pixel grid makes several so-called distance metrics possible, which often give different answers to each other for the distance between the same pair of points.

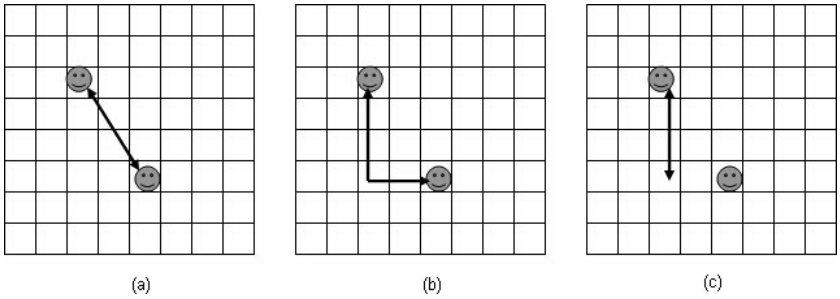

Figure 2.32: Examples: (a) Euclidean, (b) City-Block and (c) Chess-Board distance measures

Euclidean

The most familiar distance metric, which is widely applicable from all ranges of data. The Euclidean distance function measures the 'as-the-crow-flies' distance. The distance between two sequences X and Y is,

$$d_{Euclidean} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.43}$$

where, $X = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_i, \ldots, y_n]$. Euclidean distance between two data points involves computing the square root of the sum of the squares of the differences between corresponding values.

Euclidean Squared

Mathematically speaking, it uses the same equation as the Euclidean distance metric, but does not take the square root. As a result, clustering with the Euclidean Squared distance metric is faster than clustering with the regular Euclidean distance. The output of Jarvis-Patrick and K-Means clustering is not affected if Euclidean distance is replaced with Euclidean squared. However, the output of hierarchical clustering is likely to change.

Manhattan

This is also known by the name City-Block distance metric. The distance is calculated as,

$$d_{Manhattan} = \sum_{i=1}^{n} |x_i - y_i| \tag{2.44}$$

where, $X = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_i, \ldots, y_n]$.

Pearson Correlation

It is used to measure the similarity in shape between two profiles. Mathematically, Pearson correlation distance is,

$$d_{Pearson} = 1 - r \tag{2.45}$$

where, r is the dot product of the z-scores of the vector x and y. It is expressed as,

$$r = \frac{z(x).z(y)}{n} \tag{2.46}$$

The z-score of x is constructed by subtracting mean $\bar{x}$ from x and dividing by it standard deviation, i.e. $z(x) = \frac{x-\bar{x}}{\sigma}$. This is used in the z score of y.

Pearson Squared

The Pearson Squared distance measures the similarity in shape between two profiles, but also captures inverse relationship. Mathematically,

$$d_{PearsonSquared} = 1 - 2r \tag{2.47}$$

where, r is the Pearson correlation defined above.

Chebychev

The Chebychev distance between two sequences is the maximum distance between the elements in any single dimension. The distance between the sequences $X = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_i, \ldots, y_n]$) is computed using the formula,

$$d_{Chebychev} = max_i|x_i - y_i| \tag{2.48}$$

where, x_i and y_i are the values of the i-th elements in both the sequences X and Y respectively. The Chebychev distance may be appropriate if the difference between the sequences is reflected more by differences in individual dimensions rather than all the dimensions considered together. It is noted that this distance measurement is very sensitive to outlying measurements. It is also called by chessboard distance metric.

Spearman Rank Correlation

It measures the correlation between two sequences. The two sequences are ranked separately and the differences in rank are calculated at each position, i. The distance between sequences $X = [x_1, x_2, \ldots, x_i, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_i, \ldots, y_n]$) is computed by,

$$d_{SpearmanRank} = 1 - \frac{6\sum_{i=1}^{n}(rank(x_i) - rank(y_i))^2}{n(n^2-1)} \tag{2.49}$$

where, x_i and y_i are the i-th values of sequences X and Y respectively. The range of Spearman correlation is from -1 to 1. Spearman correlation can detect certain linear and non-linear correlations. However, Pearson Correlation may be more appropriate for finding linear correlations.

Dynamic Time Warping (DTW)

It is very easy to align two sequences having equal lengths but, how is it possible to align two non-linear sequences? The question is answered by the DTW itself. DTW overcomes the shortcoming of Euclidean distance measure and other common and simple techniques that are used for determining the distance between two sequences having equal length. In both speech and character recognition areas, such a case (sequences having different lengths) is occurred. Keogh et al., 1999 provided a concrete difference between the simple Euclidean distance and DTW. It greatly focussed on DTW in case of massive datasets.

A brief explanation of DTW with an example is described in the following. Consider two test T and reference R sequences, of length n and m respectively.

$$\begin{aligned} T &= t_1, t_2, \ldots, t_i, \ldots, t_n \\ R &= r_1, r_2, \ldots, r_i, \ldots, r_m \end{aligned} \tag{2.50}$$

In order to align two variable length sequences a matrix of size $n \times m$ is constructed. An element of that matrix contains the distance of two points t_i and r_j. Euclidean distance between two points can be expressed as,

$$d(i,j) = \sqrt{(t_i - r_j)^2} \tag{2.51}$$

Each matrix element (i,j) corresponds to the alignment between the points t_i and r_j. A warping path WP, now can be defined in the set of matrix elements that defines between T and R. The k-th element of W is, $w_k = (i,j)_k$. Now, we have,

$$W = w_1, w_2, \ldots, w_k, \ldots, w_K \tag{2.52}$$

with the condition that $max(m,n) \leq K < m+n-1$.

Following constraints are considered in warping path:

- *Boundary Condition*:

  $w_1 = (1,1)$ and $w_k = (n,m)$, is simply stated. This is necessary the warping path to start and finish diagonally in opposite corner cells in the matrix.

- *Continuity*:

  Given $w_k = (a,b)$ then $w_{k-1} = (a^{'}, b^{'})$, where $a - a^{'} \leq 1$ and $b - b^{'} \leq 1$. This makes the path runs diagonally not to other adjacent cells.

- *Monotonicity*:

  Given $w_k = (a,b)$ then $w_{k-1} = (a^{'}, b^{'})$, where $a - a^{'} \geq 0$ and $b - b^{'} \geq 0$. This forces the points in W to be monotonicity space time.

On the other hand, the warping path can be efficiently determined by using Dynamic Programming. It can be expressed as,

$$D(i,j) = \min[D(i-1, j-1), D(i-1, j), D(i, j-1)] + d(t_i, r_j) \tag{2.53}$$

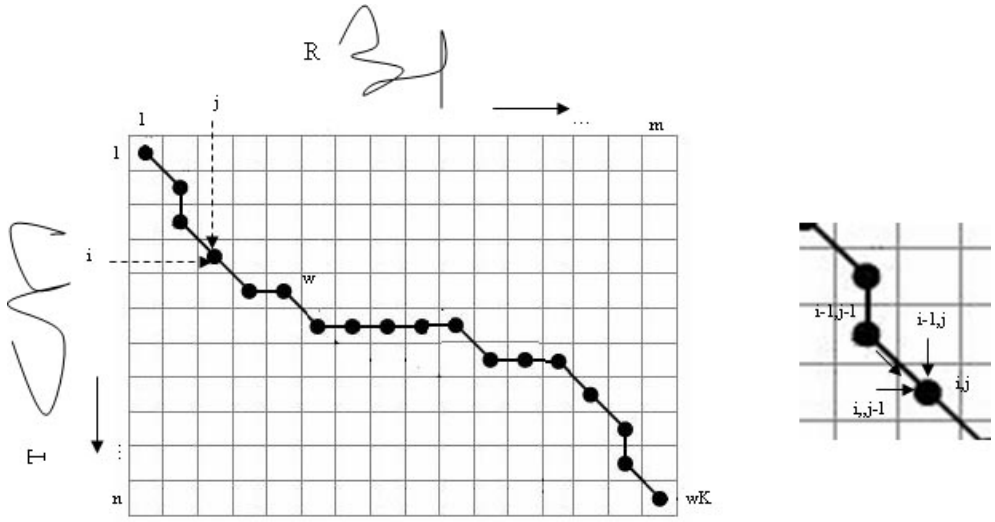

Figure 2.33: An example of warping path during the alignment of two non-linear sequences by using DTW

$D(i, j)$ is the cumulative distance between two sequence T and R from three adjacent cells and Euclidean distance in the (i, j), which is shown in Fig. 2.33. The element at the end of the matrix gives the distance between the sequences. Euclidean distance is the special case of DTW where the k-th element of W is constrained such that $w_k = (i, j)_k$, $i = j = k$. However, it is defined only in case the sequences are of same length. One of the main properties of DTW is time complexity. The time complexity is defined as $O(nm)$.

Typically, two situations are faced in matching the sequences.

- *Whole Matching*:

  Matching takes place between two sequences having identical lengths. In such a case, Euclidean distance can be possible.

- *Subsequence Matching*:

  It is time consuming to match two sequences having different lengths. Consider a test sequence T is smaller in length in comparison to reference sequence R. The test sequence slides along the subsections of every possible subsection of R to find the best match.

2.6.5 Clustering

Machine learning is divided into two categories, supervised learning and unsupervised learning (Fung, 2001). In supervised learning, the algorithm is provided with both the case (data points) and the labels that represent the concept of learned for each case. That is, learn the concept in the sense that when an unseen case comes to classified, the algorithm should predict a label for those. On the other hand, in unsupervised learning, the algorithm is provided with just the data points but no labels, the task is to find a suitable representation of the underlying distribution of data. One major approach is 'Data Clustering'. When both

supervised and unsupervised category are combined, then it is called semi-supervised learning. A common technique used in statistical data analysis is data clustering, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bio-informatics. Clustering in a sense, is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait-often proximity according to some defined distance measure. Clustering is often confused with classification, but there is some difference between two. The basic difference between these two concepts is that in clustering the data points are unlabeled - they have no previous taxonomy as far as the mathematical analysis is concerned. In classification, however, the data points are labeled. Generally, one can say that unsupervised methods perform the job of clustering whilst supervised methods are more suited to classification of datasets. Machine learning typically regards data clustering in the form of unsupervised learning.

What are the main requirements that a clustering algorithm should satisfy? This is the important part to look after in order to choose the best among the existing ones. Ideally, main requirements are,

a. It should have scalability.

b. It should deal with any types of attributes.

c. It should have minimal requirements for domain knowledge to determine input parameters.

d. It should have ability to deal with wide range of noises and outliers.

e. It should be insensitive to order of records.

f. It should have high dimension.

g. It should have interpretability and usability.

Most of the clustering algorithm do not maintain all the requirements adequately. Some of the problems are,

a. Dealing with large number of dimensions and large number of data items can be problematic as time time complexity is considered.

b. The effectiveness of the algorithm depends on the similarity measure (for instance, distance calculation for distance-based clustering)

c. Distance measure can be a problem because of the variation of dimension spaces.

In general, clustering algorithm can be broken down into,

- Parametric Clustering.
- Non-parametric Clustering.

Parametric approach minimizes the cost function, which associates a cost to each instance-cluster assignment. The main aim of this algorithm is to solve an optimization problem to satisfy the optimality criterion, often called minimizing the cost function. While, non-parametric approach is based on the similarity/dissimilarity measure among the clusters. Distance measure is the fundamental parameter to check whether two clusters are near to each other (similar/dissimilar). A rough introduction of most of the distance metrics is presented in the previous section.

Parametric clustering includes,

- Exclusive Clustering: K-means Algorithm.
- Overlapping Clustering: Fuzzy C-means Algorithm.
- probabilistic Clustering: Mixture of Gaussians Algorithm.

While, non-parametric clustering includes,

- Hierarchical Clustering: Agglomerative and Divisive Algorithms.

K-means Clustering Algorithm

Algorithm

a. Assign K points in the space represented by the objects that are to be clustered. K number of points represent initial groups of centroids.

b. Group each object to the group that has the closest centroid.

c. Re-calculate the positions of the centroids.

d. Repeat b. and c. until centroids no longer move.

Suppose, there are n-sample feature vectors from three categories, $p_1, p_2, \ldots, p_i, \ldots, p_n$, $q_1, q_2, \ldots, q_i, \ldots, q_m$ and $r_1, r_2, \ldots, r_i, \ldots, r_k$ are to be clustered. Initially, K numbers of clustered are assigned $(K < n)$. One should aware that the initial partitions that are 'given' to the algorithm have a sizeable impact on the way the data is assigned (Smet et al., 2002) and thereby one must be careful when performing this method. The clustering result is entirely depend on the value of K. How significant is the clustering result, is depend on the value of initial partition. Basically, it is recommended that the value of K is equal to the number of categories. For instance, the value of K is 3 because it has three categories of samples, expecting that each category has similar samples to each other.

For two dimensional (x, y) samples, centroid in each group (partition)can be calculated by,

$$Centroid = (\sum_{i=1}^{n} \frac{x_i}{n}, \sum_{i=1}^{n} \frac{y_i}{n}) \tag{2.54}$$

where, n is the number of data points in each group. A detail explanation can be made easy for the readers from presentation slides by Andrew Moore, 2001.

Fuzzy C-Means Clustering Algorithm (FCM)

It is method of clustering which allows one piece of data to belong to two or more clusters. It is based on minimization of the following objective function,

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{i,j}^m . d_{i,j} \qquad 1 \leq m < \infty \tag{2.55}$$

where, d is the distance metric, usually, $d_{i,j} = \|x_i - c_j\|^2$, m is any real number greater than 1, $u_{i,j}$ is the degree of membership of x_i in the cluster j, x_i is the i-th d-dimensional measured data, c_j is the d-dimension center of the cluster, and $\| * \|$ is any norm expressing the similarity between any measured data and the center.

Fuzzy partitioning is carried out through an iteration optimization of the objective function, with the update of membership $u_{i,j}$ and the cluster centers c_j by,

$$u_{i,j} = \frac{1}{\sum_{k=1}^{C} (\frac{(d_{i,j})^{1/2}}{(d_{i,k})^{1/2}})^{\frac{2}{m-1}}} \tag{2.56}$$

$$c_j = \frac{\sum_{i=1}^{N} u_{i,j}^m . x_i}{\sum_{i=1}^{N} u_{i,j}^m} \tag{2.57}$$

The iteration will stop when $max_{i,j}\{\|u_{i,j}^{k+1} - u_{i,j}^k\|\} < \varepsilon$, where ε is a termination criterion between 0 and 1, where k are the iteration steps. This procedure converges to a local minimum or a saddle point of $J - m$.

Algorithm:

a. Initialize $U = [u_{i,j}]$ matrix, $U^{(0)}$.

b. At k step, calculate the center vectors $C^{(k)} = [c - j]$ with $U^{(k)}$.

c. Update $U^{(k)}$, $U^{(k+1)}$.

d. If $\|U^{(k+1)} - U^{(k)}\| < \varepsilon$, then stop; otherwise return to b.

Mixture of Gaussians Clustering Algorithm

A model-based approach, consisting of certain models for clustering and attempting to optimize the fit between the data and the model. Each cluster can be mathematically represented by a parametric distribution like, gaussian (continuous) or poisson (discrete). Hence the entire set is modeled by a mixture of these distributions.

Firstly, it chooses the component at random with probability $P(w_i)$ and samples a point is $N(\mu_i, \sigma^2 I)$. Suppose, we have,

- $x_1, x_2, \ldots, x_i, \ldots, x_N$.

- $P(w_1), \ldots, P(w_K), \sigma$.

Then, the likelihood of the sample is,

$$P(x|w_i, \mu_1, \mu_2, \ldots, \mu_K) \tag{2.58}$$

What we really want to maximize is, $P(x|\mu_1, \mu_2, \ldots, \mu_K)$. It can be written as,

$$P(x|\mu_i) = \sum_i P(w_i) P(x|w_i, \mu_1, \mu_2, \ldots, \mu_K)$$

Now,

$$P(data|\mu_i) = \prod_{i=1}^{N} \sum_i P(w_i) P(x|w_i, \mu_1, \mu_2, \ldots, \mu_K) \tag{2.59}$$

The point is to maximize the likelihood function by calculating $\frac{\delta L}{\delta \mu_i} = 0$, which is difficult that's why EM algorithm is in use.

Algorithm:

a. Initialize the parameters.
$\lambda_0 = \{\mu_1^0, \mu_2^0, \ldots, \mu_K^0, p_1^0, p_2^0, \ldots, p_K^0\}$

b. E - step:
$P(w_j|x_k, \lambda_t) = \frac{P(x_k|w_j, \lambda_t) P(w_j|\lambda_t)}{P(x_k|\lambda_t)} = \frac{P(x_k|w_i, \mu_i^{(t)}, \sigma^2) p_i^{(t)}}{\sum_k P(x_k|w_j, \mu_j^{(t)}, \sigma^2) p_j^{(t)}}$

c. M - step:
$\mu_i^{t+1} = \frac{\sum_k P(w_i|x_k, \lambda_t) x_k}{\sum_k P(w_i|x_k, \lambda_t)}$, $p_i^{t+1} = \frac{\sum_k P(w_i|x_k, \lambda_t)}{R}$, where R is the number of records.

Hierarchical Clustering Algorithm: Agglomerative and Divisive

In hierarchical clustering, data are not grouped/partitioned into a particular number of cluster(s) in a single step, instead a series of steps take place. It may run from all objects, n number of clusters to single object/cluster in case of agglomerative algorithm, while a series of partitions take place from a single object to all objects to n clusters in divisive algorithm.

A two dimensional diagram known as dendrogram, which is shown in Fig. 2.34 gives an overall idea to differentiate agglomerative and divisive clustering algorithms. In contrast to divisive, agglomerative hierarchical clustering is widely used.

In agglomerative methods, two closest (most similar) clusters are fused/merged to form a cluster. This pair-wise merging takes place until a cluster. How the effective similarity can be measured? Based on the similarity measure (distance calculation) in order to find the closest pair of clusters, agglomerative clustering can be broken down into three.

- *Single Linkage Clustering*:

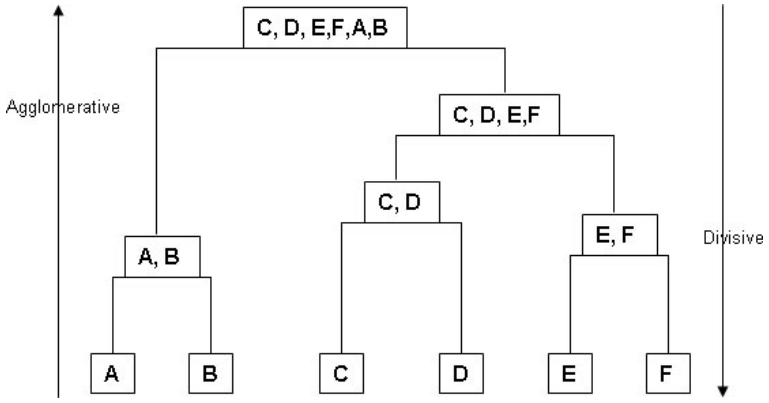

Figure 2.34: A sample of dendrogram providing an idea of agglomerative and divisive clustering algorithms

The distance between two clusters is defined as the distance between the closest pair of objects, one object from each cluster. Mathematically, it can be expressed as,

$$D(A,B) = min\{d(i,j)\} \tag{2.60}$$

where, objects i and j are in cluster A and B respectively. The distance between every pair is computed. The minimum value of these distances is the distance between two clusters. Two clusters are merged from which $D(*,*)$ is minimum. It is also popular by the name 'Nearest Neighbor Clustering Technique'. Fig. 2.35 gives an idea of single-linkage method.

- *Complete Linkage Clustering*:

  The distance between two clusters is the distance between the most distant pair of objects, one from each cluster. Mathematically, it can be expressed as,

$$D(A,B) = max\{d(i,j)\} \tag{2.61}$$

  where, objects i and j are in cluster A and B respectively. The distance between every possible object pair (i,j) is calculated and the maximum value is of these distances gives the distance between two clusters. Two clusters are merged from which $D(*,*)$ is minimum. It is also called by the name 'Farthest Neighbor Clustering Technique'. Fig. 2.35 gives an idea of complete-linkage method.

- *Average Linkage Clustering*:

  The distance between two clusters is calculated by using the average values (distances). The average distance is calculated as,

$$D(A,B) = T_{AB}/(N_A \times N_B) \tag{2.62}$$

  where, T_{AB} is the sum of all pairwise distances between the cluster A and B. N_A and N_B are the sizes of the clusters A and B respectively. from the distance between each object in a cluster and all other objects in another cluster. This is repeated to every object. The lowest average distance gives the distance between two clusters. Two clusters are merged from which $D(*,*)$ is minimum. Fig. 2.35 gives an idea of average-linkage method.

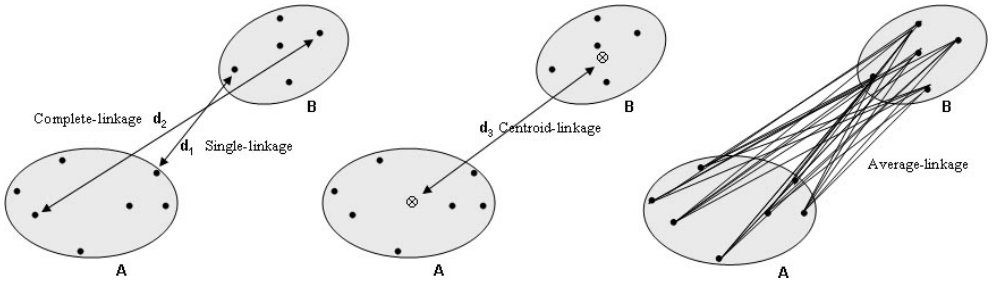

Figure 2.35 Types of agglomerative hierarchical clustering

- *Centroid Linkage Clustering*:

  The distance between two clusters is the distance between two centroids. Mathematically, it can be expressed as,

  $$D(A,B) = d(c_A, c_B) \tag{2.63}$$

  where, c_A and c_B are the centroids of two clusters A and B respectively. Two clusters are merged from which $D(*,*)$ is minimum. Fig. 2.35 gives an idea of centroid-linkage method.

- *Ward's Method*:

  Cluster membership is assigned by calculating the total sum of the squared deviations from the mean of a cluster. The criterion of merging is that it should produce the smallest possible increase in the error sum of squares.

The clustering algorithm is entirely depend on the distance measurement techniques. One can use simple Euclidean or other techniques to align the two sequences having equal length, however, DTW is preferred for non-linear sequences. While most combinations of clustering algorithm and distance metrics provide meaningful results, there are a few combinations that are difficult to interpret. In particular, combining K-Means clustering with the Pearson Squared distance metric can lead to non-intuitive centroid plots since the centroid represents the mean of the cluster and Pearson Squared can group anti-correlated objects. In these cases, visually drilling into clusters to see the individual members through the use of Cluster Plots produce better results.

2.6.6 Cross Validation (XV)

Cross validation is a model evaluation method that is better than residuals. Cross-validation can be simply used either to estimate the generalization error of a given model or it can be used for model selection by choosing one of several models that has the smallest estimated generalization error. The problem with residual evaluation is that they do not give an indication of how well the system will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is not to use the entire data set when training a system. Some of the data is removed before training begins. The removed

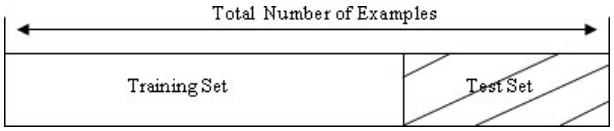

Figure 2.36 Holdout Cross Validation

data ('new' data) is used to test the performance of the learned model. Broadly, we can say validation techniques are motivated by two fundamental problems, model selection and performance estimation. Almost invariably, all pattern recognition techniques have one or more free parameters, number of neighbors, network size, learning parameters and weights. While performance is typically measured by the true error rate. It means, choosing the model that provides the lowest error rate on the entire population. Some of the ways are discussed to select the best model in the following.

In general, XV (http://courses.cs.tamu.edu/rgutier/ceg499_s02/l13.pdf) can be broken down into three.

- *Holdout Cross Validation (H-XV)*:

  It is the simplest XV approach. The data is divided into two sets, training and test datasets. The model is trained with training dataset only by using some functions. It means the function approximator fits a function. Then the error is predicted by testing the test data. The training model never seen the test data before. This is usually preferable because it takes no longer to compute. However, it has a great chance to have high variance. Further, the evaluation is depend on how fair the division is made for training and test data. The limitations of this techniques can be overcome with the family of re-sampling methods at the expense of computational complexity.

- *K-Fold Cross Validation (KF-XV)*:

  K-fold cross validation is one of the ways to improve the holdout method. The data set is divided into K subsets, and the holdout method is repeated up to (K) times. Each time, one of the k subsets is used as the test set and the other $K-1$ subsets are put together to form a training set. Then the average error across all K trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $K-1$ times. However, the disadvantage of this method is that the training algorithm has to be return from K scratch times, which means it takes K times to evaluate the complete model. The true error rate is estimated by the average error rate,

  $$E = \frac{1}{K}\sum_{i=1}^{K} E_i \tag{2.64}$$

  How can we choose the value of K? Superficially, the variance of the resulting estimate is reduced as K is increased. However, it does not follow strictly. The bias and variance of the true error rate estimator and computational time are totally depend on the number of folds that one designed. In practice, choice of number of folds (K) depends on the size of the dataset. It should neither be large nor be small. One has to think about the bias and variance of the true error estimator whenever K is to be chosen.

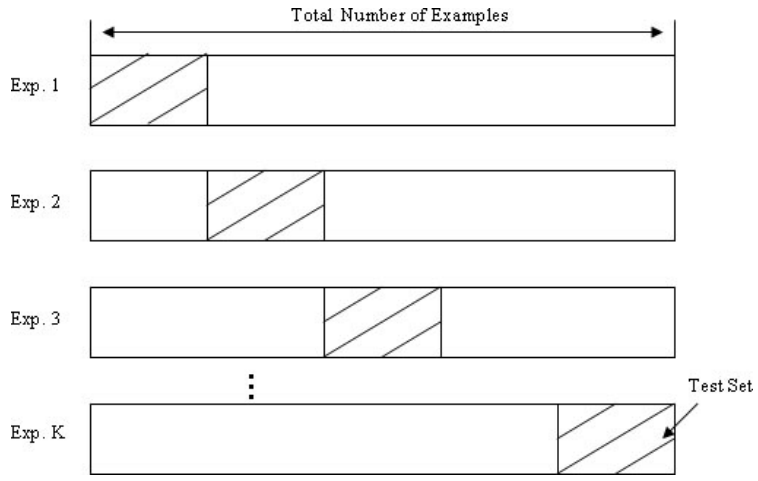

Figure 2.37 K-Fold Cross Validation

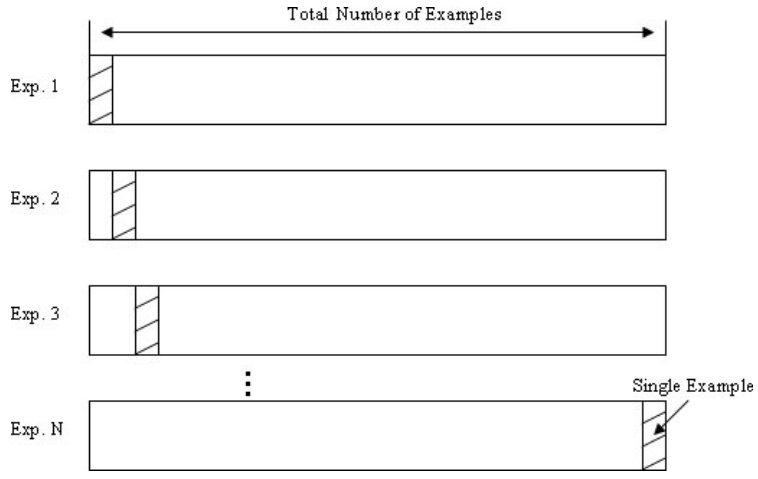

Figure 2.38 Leave-One-Out Cross Validation

- *Leave-One-Out Cross Validation (LOO-XV)*:
  A logical K-fold XV is termed as LOO-XV, where K is equal to N. N is the number of data points in the set. For N separate items, the model is trained with the use of some functions except for one point and prediction is made for that point. As before, the average error is computed to evaluate the model. It is good enough for evaluating the model with the help of LOO-XV. It is known that computing the Leave-One-Out cross validation error (LOO-XVE) takes no more time than computing the residual error and it is much better way to evaluate the model. The true error rate can be expressed as,

$$E = \frac{1}{N}\sum_{i=1}^{N} E_i \tag{2.65}$$

Chapter 3

A Structural Approach on a Template-based Handwritten Character Recognition

3.1 System Design: Recognition of Nepali Stroke Number and Order Free Natural Handwritten Character

Building an automatic recognition of writer independent Nepali natural handwritten alphanumeric character is the aim from the beginning of the task. Among the large sets of symbols used in Nepali, we take the most common type of alphanumeric characters that are frequently used in daily lives. In this task, the alphanumeric characters includes in both training and testing the prototype classifier are 31 classes of consonants, 5 classes major vowels and 10 classes of numerals. The algorithm is implemented to construct a classifier in such a way that it has flexibility in adding rest of the symbols. The complete task at a glance can be summarized as follows.

Writing one's own style brings unevenness in writing units, which is the most difficult part. Stroke numbers, their order, shape and size, tilting angle and similarity among characters are carefully considered in this task. A complete character is broken down into the number of strokes and fed into the pre-processor stroke by stroke basis. A string of pen-tip position and direction at every pen-tip's position as a feature of a stroke is used. Only the direction at every pen-tip's position of a sequence of coordinates of a stroke is also considered to determine the appropriate feature between two. The number of strokes within a class of character from many users are merged based on their similarity, we cluster them accordingly in order to reduce the number of prototype training samples (templates). Agglomerating takes place hierarchically between the strokes, where it uses DTW Algorithm to align the them. Classification of characters is done by the use of feature matching procedure. Matching takes place in between the test feature and the reference ones (templates). The test feature is said to be matched with the template if it produces the smallest distance as compared to the distance from other pairs. The machine is trained with the help of a number of Nepalese natives and yielded an efficient and competitive intelligence.

3.1.1 On-line Handwritten Data

Appropriate data collection is a major factor in handwriting recognition systems. It is typically a dynamic and digitized representation of the electronic pen movement, which is generally describing a sequential information about position, velocity, acceleration and even pen angles at every instant. In this work, a simple Graphite Tablet (WACOM Co.Ltd): model-

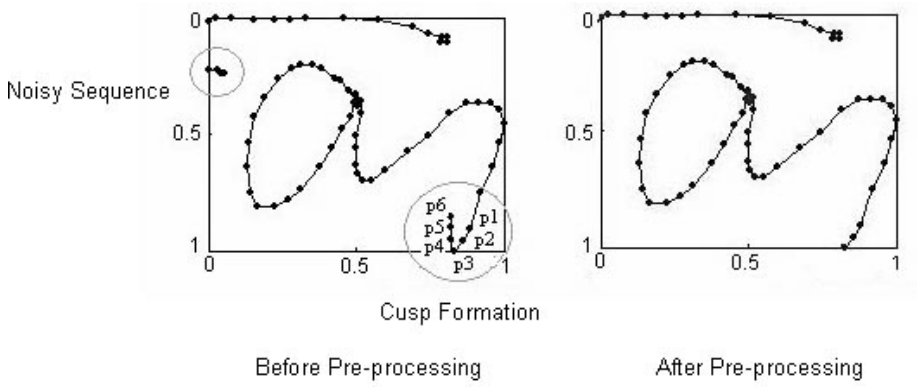

Figure 3.1 A sample of complete specific stroke pre-processing

ET-0405A-U, US patent, which is working under 5 V DC and 40 mA captures the pen tip's position in the form of 2D coordinates at the sampling rate of 20 Hz. A stroke comprises a sequence of 2D coordinates during pen down to pen up movement. A series of strokes following the trajectory of the pen tip's position is presumed to produce a complete letter. An orderly sequence of a number of strokes is stored. The number of strokes used in a character often varies from two to four even within a class of character (dataset). No directions, constraints and limitations were given to the user in their writing styles, but they were encouraged to write as if they were written on a piece of paper. A possible ranges of writing styles were collected.

3.1.2 Data Pre-processing

Real time data, directly collected from users are often incomplete and noisy. Basic tools for pre-processing are, noise elimination, sampling, discretization, integration, transformation (size normalization). Different systems use a variety of different pre-processing techniques before feed into feature extraction block (Blumenstein et al., 2003; Verma et al., 2004). The techniques used in one system may not exactly fit to the other systems because of the different writing styles and the script themselves. For instance, way of writing Nepali is different from English. It utilizes some simple strategies, consisting of size normalization, noise eliminations and disturbing coordinates deletion.

- *Size Normalization*:

  Most of the tasks considered size normalization (Guerfali et al., 1993; Homayoon et al., 1994; Keogh et al., 1999). In the similar manner, size normalization is used in this task because of variable size of writing samples (sometimes very big and sometimes very small). The newly designed window size for every character is,

$$x_{new} = \left(\frac{x - x_{min}}{x_{max} - x_{min}}\right)(x_{max'} - x_{min'}) + x_{min'}$$
$$y_{new} = \left(\frac{y - y_{min}}{y_{max} - y_{min}}\right)(y_{max'} - y_{min'}) + y_{min'} \quad (3.1)$$

  where, x_{max}, y_{max} and x_{min}, y_{min} are the maximum and minimum coordinate points of all strokes within a letter along x axis and y axis respectively. $x_{max'} = 1$, $x_{min'} = 0$

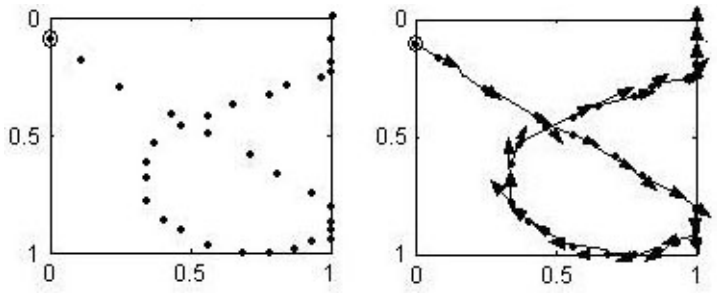

Figure 3.2 Feature selection

and $y_{max'} = 1$, $y_{min'} = 0$ gives the size of new standard window in which every writing sample resides. This helps in determining the strokes' spatial relation by using the boundary condition.

- *Noise Elimination*:

  Some sequences do not carry any information. These are needed to eliminate for both enhancing accuracy and speed of the system. Elimination of noisy sequences in cursive writing is very difficult task; however, we delete some sequences that are unnecessarily written. Such sequences are often written at the end as re-writing strokes. These sequence do not give any information about the character. The pre-processor deletes such a shorter sequence (a sequence of 5 coordinates), having very small width and height (0.2×0.2). Moreover, the pre-processor chops cusps or undesirable hooks at both ascender and descender of a sequence in case it is present. This is done by the use of tangent angles along sequence of 5-10 2D coordinates. If the angle changes drastically ($80^0 - 100^0$ is considered) along the sequence of consecutive coordinates, then that sequence from that coordinate is chopped.

- *Co-occurrence Coordinates Deletion*:

  It deals with the minimization of strokes' sequence by deleting both repeated coordinates i.e., $p_i = p_{i+1} = p_{i+d}$ for some i and d. This co-occurrence of coordinates can disturb the shape of the character, in case tangent is taken for every consecutive coordinates as the feature.

A complete specific stroke pre-processing is graphically demonstrated including size normalization, noise and cusp elimination in Fig. 3.1. The noisy sequence formed by a sequence of 3 coordinates is deleted. The encircled sequence of coordinates demonstrates a sample of a cusp, formed by p2, p3 and p4 (local points). Along the points p1, p2 and p3, there is an infinitesimal change in successive slopes and similar along p3, p4, p5 and p6 whereas a drastic change in the tangent angle from the point p3 gives a birth of a cusp. Therefore, the sequence is chopped from point p3 up to last point.

3.1.3 Feature Extraction

We are familiar with the characters having variable number of strokes from time to time writing and even from one user to another. Therefore, relevant feature with sufficient distinguishing information is important and it has a significant place in recognizer. In this task,

classifier uses the number of strokes, the length of a sequence of 2D coordinates in a stroke, their direction, size of the stroke as the basic features.

A set of strokes is applied to the feature extraction coordinator. At first, the feature extraction coordinator separates a complete character into number of strokes used from a set of strokes. For better understanding, consider any character C_i represented by its strokes S_i and its label L_i from i-th user is

$$\mathbf{C_i} = (\mathbf{S_i}, L_i) \tag{3.2}$$

Any character C_i comprises of a m-set of strokes,

$$\mathbf{S_i} = [\mathbf{s_{i,1}}, \mathbf{s_{i,2}}, \ldots, \mathbf{s_{i,m}}] \tag{3.3}$$

Now, j-th stroke be,

$$\mathbf{s_{i,j}} = [p_{i,j,1}, p_{i,j,2}, \ldots, p_{i,j,l}] \tag{3.4}$$

where, $p_{i,j,k} = (x_{i,j,k}, y_{i,j,k})$. This task extracts two different features from the same temporal information; only the sequence of slopes along the sequence of 2D coordinates is taken in the first part and the sequence of both 2D coordinates and slopes is taken in the second part.

$$\mathbf{F^1_{i,j}} = [f(p_{i,j,1}, p_{i,j,2}), f(p_{i,j,2}, p_{i,j,3}), \ldots, f(p_{i,j,l-1}, p_{i,j,l})] = \theta_{i,j} \tag{3.5}$$

$$\mathbf{F^2_{i,j}} = [(p_{i,j,1}, f(p_{i,j,1}, p_{i,j,2})), (p_{i,j,2}, f(p_{i,j,2}, p_{i,j,3})), \ldots, (p_{i,j,l-1}, f(p_{i,j,l-1}, p_{i,j,l}))] \tag{3.6}$$

where, $f(p,q) = \arctan(\frac{y_q - y_p}{x_q - x_p})$ is used as a tangent function. In another way, the tangent function at time t can be expressed by, $\theta_t = \arctan(\frac{dy_t}{dx_t})$, where coordinate feature at time t is (x_t, y_t) and $dx_t = x_{t+1} - x_t$. In the second feature, we have one less number of tangents than a number of 2D coordinates in a sequence of stroke. So, we omitted the last coordinate of a sequence to ensure that the size of coordinates and slopes of that sequence of stroke are equal. Practically, omitting either first or last coordinate does not affect the shape of the stroke. Both the feature are tested separately to determine the appropriate one in terms of both carrying intelligence and speed. Fig. 3.2 shows an example of how we extract successive tangents along the sequence 2D coordinates.

In both the features, directional property of the writing is preserved. In most cases, angle correction is necessary. However, this task utilizes directional property of strokes' sequence as the feature and thereby no angle correction is in need even though the writings are tilted by some angle either to the left or to the right.

3.1.4 Data Reduction

Representing a large number of similar items with the specific number of representatives is data reduction. Similar Items of one group are dissimilar with another items belonging to another groups. This technique helps in two aspects. Firstly, it reduces the number of similar items into one (representative), i.e. compact system. Secondly, it increases the speed due to a small number of representative training samples (templates). It consists of two stages: firstly, similarity measure, where DTW is used and secondly, single-linkage agglomerative hierarchical clustering, which is based on similarity. An engine for merging the similar sequences pair wise is designated as follows. The question is, how can similarity between two sequences be measured?

Similarity Measure/Distance Calculation

Distance finding technique is the common way of determining the similarity in between two feature vector sequences of strokes. There has been much interest in adapting data mining algorithm to time series massive databases. Typically, extension of Euclidean distance is used. However, Euclidean distance can be extremely brittle measure (Keogh et al., 1999), which produces the pessimistic dissimilarity results. We propose to use a DTW algorithm at the cost of computational complexity, which overcomes the shortcomings of local distance metrics in terms of similarity determination. As compared to the cost of time complexity in local distance metric; $O(n)$, DTW has as high as $O(n^2)$. However, in some cases, lower bounding techniques can be applied to reduce it to the possible smallest value. The possibility of aligning two different nonlinear sequences is enjoyed by this task, which is not possible by other local distance metrics. Strictly speaking a Euclidean distance metric is used only for aligning two linear sequences. The time required to align two different nonlinear sequences is completely based on their lengths (number of coordinates in sequences). The larger the number of coordinates in the sequences, the slower is the speed to align. The bottleneck of DTW lies in its time complexity, which impedes the applications of DTW to a high degree. Dynamic programming (DP) is used in this task as we have large pool of strokes' sequences.

To illustrate mathematically, consider two strokes' sequences A and B of size N and M, respectively. At first, a matrix of size $N \times M$ is constructed. $d(n,m)$ is an element of the local distance metric (Euclidean distance) between the events e_n^A and e_m^B , which can be expressed as,

$$d(n,m) = |e_n^A - e_m^B| \tag{3.7}$$

$D(n,m)$ is the global distance up to (n,m),

$$D(n,m) = \min[D(n-1,m-1), D(n-1,m), D(n,m-1)] + d(n,m) \tag{3.8}$$

with an initial condition $D(1,1) = d(1,1)$. The global distance between A and B is,

$$D(A,B) = D(N,M) \tag{3.9}$$

It means the last element of the $N \times M$ matrix gives the distance between two feature vector sequences. This is often called DTW-matching score in recognition process. A complete distance calculation between two non-linear sequences is shown in Fig. 3.3. The score at the end of the matrix (dark shadow) is the cumulative global distance. It estimates how similar they are. Two strokes' sequences are said to be similar if the global distance/cumulative distance in between them is smaller than from other pairs. Further, a graphical comparison of two numeral pairs is presented in Fig. 3.4. Each black dot represents a feature event. Numerals pair, formed by ? and ? on the left are similar because distance between them is smaller than the pair formed by ? and ? on the right in Fig. 3.4.

Single-Linkage Agglomerative Hierarchical Clustering

This task utilizes the distance based single-linkage agglomerative hierarchical clustering in which distance measure is the important factor to guess their likeness. Cluster analysis is

		B_m					
		1.578	0.000	-1.560	1.500	0.000	0.120
	1.578	0.000	1.578	4.716	4.794	6.372	7.830
	1.278	0.300	1.278	4.716	4.338	5.616	6.774
	0.000	1.878	0.300	1.860	3.360	3.360	3.480
	-1.345	4.801	1.645	0.515	3.360	4.705	4.825
	1.678	4.901	3.323	3.753	0.693	2.371	3.929
A_n	1.340	5.139	4.663	6.223	0.853	2.033	3.253
	0.000	6.717	4.663	6.223	2.353	0.853	0.973
	1.200	7.095	5.863	7.423	2.653	2.053	1.933
	1.450	7.223	7.313	8.873	2.703	3.503	3.263
	1.670	7.315	8.893	10.54	2.873	4.373	4.813
	0.120	8.773	7.435	9.115	4.253	2.993	2.993

Figure 3.3: An example to determine the distance between two non-linear sequences using DTW technique

Only the directional property are taken.
Feature vector sequences are:
$A_n = [1.578, 1.278, 0, -1.345, 1.678, 1.34, 0, 1.2, 1.45, 1.67, 0.12]$ and
$B_m = [1.578, 0, -1.56, 1.5, 0, 0.12]$

related to grouping or segmenting a collection of objects into subsets or clusters, such that those within each cluster are at a shorter distance than objects assigned to different clusters. Data are not grouped into a particular cluster in a single step. Instead, a series of fusion takes place from initially separated data. This algorithm is called *agglomerative hierarchical clustering*. Initially, a distance matrix is constructed. It contains all the distance from every possible pair within a class of character. The total number of strokes gives the size of the distance matrix. For example, if there are N number of strokes from all users while drawing the first consonant क, then the total number of distances produced after matching from one another and within themselves is N. Therefore, the distance matrix is of size $N \times N$. The element (i, j) in the matrix is the distance between the i-th stroke to the j-th stroke. Two strokes are merged if the distance from one another is the smallest one within the whole matrix for that particular character. This task erases rows and columns in the proximity matrix as old cluster are merged into new ones. This new cluster is the representative of the merged clusters, which is computed by averaging clusters' members pair wise via the use of discrete warping path along the diagonal of DTW-matrix (Somervuo et al., 1999). A simple way to find out the average between the two merged clusters is explained in the following.

Let us take a look at Fig. 3.3. The shaded part in the matrix demonstrates the DTW diagonal matrix. Let a_i and b_j be two corresponding feature vectors in the warping path of two feature vector sequences A and B. The weighted average is defined as,

$$c_k = qa_i + (1-q)b_j \tag{3.10}$$

where q is a real number between 0 an 1. The value of q is 0.5 in this task in order to preserve half of the information from one feature vector sequence. Now, the sampling instants of a_i

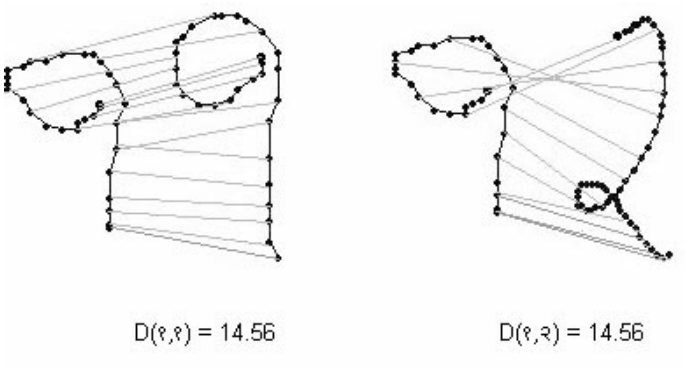

Figure 3.4: A graphical measurement of similarity/dissimilarity between the numerals pairs १ and २

and b_j are denoted by t_i and t_j, the corresponding sampling instant c_k is,

$$t_k = qt_i + (1-q)t_j \tag{3.11}$$

The length of the consecutive feature vectors is determined from the warping diagonal path. In order to define the consecutive vectors in the average sequence $C = [c_1, c_2, \ldots, c_i, \ldots, c_K]$ at regularly spaced sample points, the average feature vector corresponding to the desired time instant can be interpolated between the nearest time instants t_k and t_{k+1} as in the discrete warping path.

This process of merging is repeated until it reaches the stopping threshold (cluster threshold). The value of the threshold gives the significant number of cluster representatives after clustering. This is repeated until the desire number of cluster representatives. This is the basic idea of single-linkage hierarchical clustering. Designing a cluster threshold is another difficult task. As the writing styles is variable from time to time writing from not only different users but also from the same user, the cluster threshold to stop the clustering process is equal to or a few number less than the number of characters involved in processing is found to be appropriate. Use of the cluster threshold like this is appropriate in the sense that the significant merging takes place among the strokes what it needs to be. In other words, if the value of the cluster threshold is very very small, then a chance of getting false representatives from many merged strokes would be higher. It means, strokes which are not much similar to each other are also merged into a group where only similar strokes are to be merged. However, getting a very few representatives improves the speed but it does not guarantee that it receives higher accuracy. Fig. 3.5 gives an idea of agglomerative hierarchical clustering based on the similarity (distance calculation). The basic algorithm for agglomerative hierarchical clustering is as follows.

Algorithm:

Given N items to be clustered, a square matrix (similarity matrix) of size $N \times N$ is constructed first. Let $D(i, j)$ be the distance between the stroke i and stroke j.

a. Start by assigning each item to a cluster, so that if there are N items, then N clusters are assigned. Distance between the clusters determine their similarity.

b. For each pair of clusters (i, j), compute $D(i, j)$.

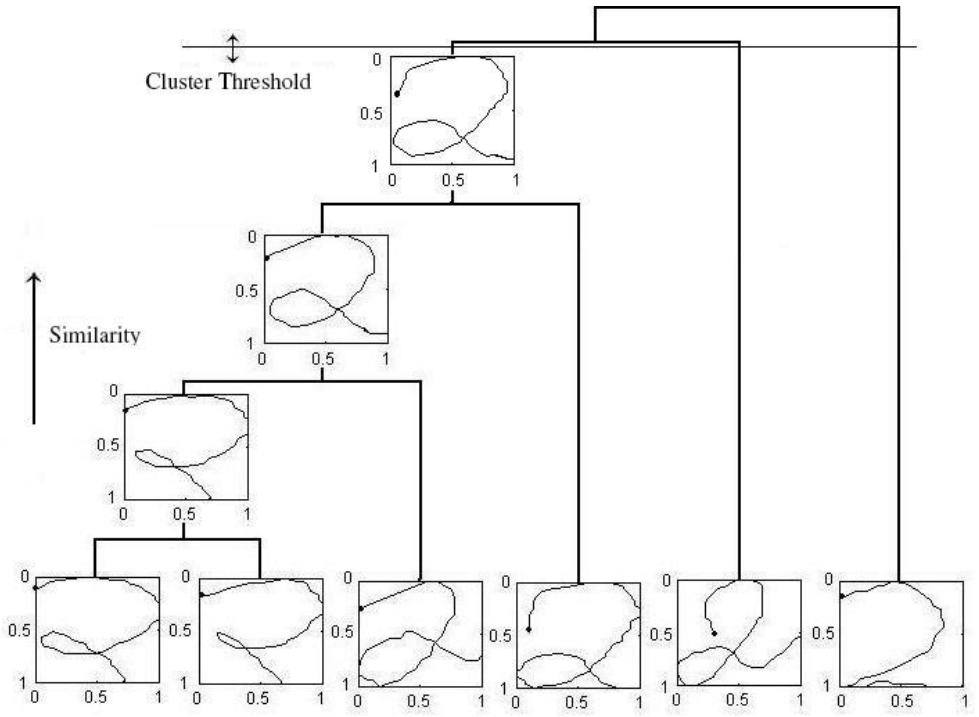

Figure 3.5: A graphical agglomerative hierarchical clustering demonstration with significant cluster threshold by taking one of the numeral classes २

c. For each cluster i, determine the closest pair.

d. Agglomerate two similar clusters i and j.

e. Determine the representative of two merged clusters. It reduces one cluster.

f. Compute the distances between the new cluster and each of the old clusters.

g. Repeat the process from c. until the desired number of clusters' representatives results.

3.1.5 Templates' Management

Managing all training samples (strokes' representatives after clustering) from every class of alphanumeric character is the main problem of a template-based approach. The number of strokes' representatives are fixed in every class for easy management. Each frame is designed to store strokes' representatives of the alphabets and numerals. Therefore, there are altogether forty six frames for forty six classes of alphanumeric characters. First frame is assigned for numeral class १, second frame is assigned for another numeral class २, sixteenth frame is for character class क and so on up to forty sixth frame. In each frame, only the strokes related to that specific character/numeral are stored. According to the characters' and numerals' codes as shown in Fig. 3.1, all strokes' representatives are systematically stored in their own frames. In addition, the frames are categorized into one-stroke-frame and multi-stroke-frame based on the number of strokes used in character and numerals. If only one stroke is used to complete a character/numeral from every users, then it is categorized

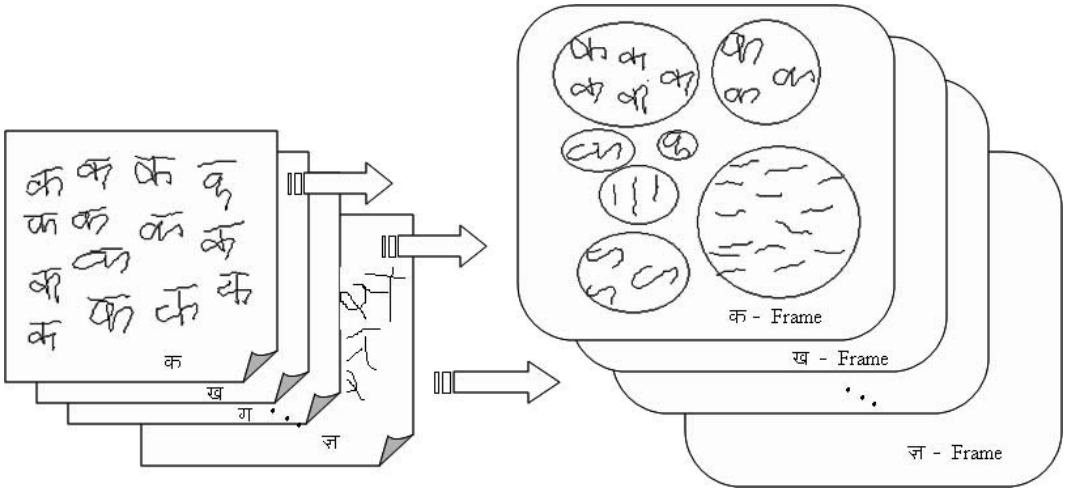

Figure 3.6: A graphical demonstration of grouping similar strokes from every character within a class क. Each encircled group in क-frame produces a representative stroke

under one-stroke-frame otherwise multi-stroke-frame. More specifically, multi-stroke-frame includes thirty six classes of characters (used two or more than two strokes to complete a character) whereas ten classes of numerals (used only one stroke to complete a numeral) are under one-stroke-frame.

3.1.6 Classification/Recognition

Assigning a digitized character into its symbolic class at the same time is a common definition of on-line character recognition. We grouped training templates into two categories: one-stroke-frame and multi-stroke-frame, based on the number of strokes used to make complete letter. Therefore, The classification starts with feature-matching process.

a. Find the number of strokes and separate shirorekha and texts employed in a test character.

b. Feed every test stroke into pre-processing engine (if available).

c. Based on the number of strokes in a test alphanumeric character, feature matching is carried out from test feature with every stored feature. If number of stroke is only one, then feature matching takes place with the templates of one-stroke-frame otherwise with the templates of multi-stroke-frame. Every matching produces a matching score. Mathematically, the distance matrix when n set of test strokes matched is,

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_1 \\ \mathbf{D}_2 \\ \cdots \\ \mathbf{D}_n \end{pmatrix}$$

where, $\mathbf{D_i} = [\mathbf{F_i^1}, \mathbf{F_i^2}, \ldots, \mathbf{F_i^{36}}]$. $\mathbf{F_i^k}$ contains some matching scores from specific group of the k-th frame when the i-th test stroke is matched. We have 36 frames for 36 classes of character. Even though we have 25 strokes' templates in each frame, the numbers of

matching scores are from only the specific group, i.e. $\mathbf{F_i^k} = [D_{i,1}^k, D_{i,2}^k, \ldots, D_{i,m}^k]$. $D_{i,m}^k$ is the matching score from the m template whenever the i-th test stroke is matched. $D_{i,m}^k = \infty$, if the m template is not used in matching.

d. Determine the threshold by taking all matching scores from all frames (either one-stroke-frame or multi-stroke-frame). Threshold for every i-th test stroke is,

$$Thshld_i = min(\mathbf{D_i}) + C \tag{3.12}$$

where, C is the fixed constant.

e. Count the number of matching scores below the threshold in every frame. This number determines how many similar strokes are available as templates. Mathematically, weight determined for every i-th test stroke in the k-th frame is,

$$W_i^k = \sum_{j=1}^{m} C(D_{i,j}^{k(w)}, Thshld_i) \tag{3.13}$$

where,

$$c(x,y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$$

There is a chance of nil matching score in some frames below the designed threshold, if so, we put an earmark for those frames at this instant, but consider in recognition in the next time. The condition for the frames to be included in classification is; there should be at least one matching score below the designed threshold in every frame in every matching from test features otherwise the recognizer excluded those frames.

f. Find the lowest matching score from every matching and from every frame if matching takes place. It should be kept in mind that the lowest matching score is produced only from the most similar template. Mathematically,

$$lms_i^k = min(\mathbf{F_i^k}) \tag{3.14}$$

g. Finally, the character's label is displayed as,

$$L = argmin_k \sum_{i=1}^{n} \frac{lms_i^k}{W_i^k} \tag{3.15}$$

For better understanding, take a test letter क. Firstly, the system determines what number of strokes employed to complete the test letter. As two strokes are used, it concludes that test letter is under multi-stroke-frame, i.e. identified as a character. Secondly, it begins with feature-matching process from both feature vector sequences of test strokes independently with respect to all templates of every class of character, from which DTW-matching score from every template results. The DTW-lowest matching score from every frame is determined. The recognizer calculated the sum of the two separate fractions of DTW-lowest matching scores and their particular weights from every class of character (within multi-stroke-frame). The character recognizes as क only when the sum of two these separate fractions of DTW-lowest matching scores and their particular weights from the frame of क is minimum than from other frames otherwise it misclassifies. Our recognition system displays the character code 01 for १, 02 for २ and so on.

Table 3.1 Experimental Results (System I)

Char. Type (Classes)	Training Chars.	Test Chars.	Misrecog. Chars.	Err.	Overall Err.
Consonant (31)	620	1240 (+60)	274 (+15)	22.09% (25.00%)	22.23%
Vowel (5)	100	200 (+60)	28 (+13)	14.00% (21.67%)	15.76%

No. of Users: 20

Pre-processing: Nil

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

Accuracy: 78.84%

Average Speed: 18 Seconds/Character

Total Bad Data: 4.66% (Training and Test)

3.2 Experimental Set Up

This important section in research to commit the performance of the designed system is the experiment in various ways. In other words, the investigation of the methodologies used in the system is with the help of the experimental results, which includes both accuracies and errors. With the help of the results and error analysis, an overall reliability of different system is explored. A series of experiments from the help of Nepalese natives were performed, which are appeared in the following sections. At first, a simple experiment was done to test whether the proposed methodology was appropriate and then moved to more sophisticated one using various techniques and tools. A simple experiment proved that the methodology was strong enough to carry further (Experiment I). We had experienced that the number of samples played a crucial role in recognition, second experiment was came up with the increase in the number of users (Experiment II). After a depth analysis, pre-processing step was considered with the hope of better reliability of the classifier. On the way to improve classifier's accuracy, many tools were added step by step to the existing methodology; we used more informative feature from the same temporal sequence in order to distinguish from one another and compared with the previous feature (Experiment III). Finally, spatial information about the strokes is handled at the end along with the use of many heuristic rules (Experiment VI).

All the experiments were run under the developed prototype system by using MATLAB 7.0.4 on a 1.81 GHz, 1.00 GB RAM, running Microsoft Windows XP Professional (Version 2002) computer.

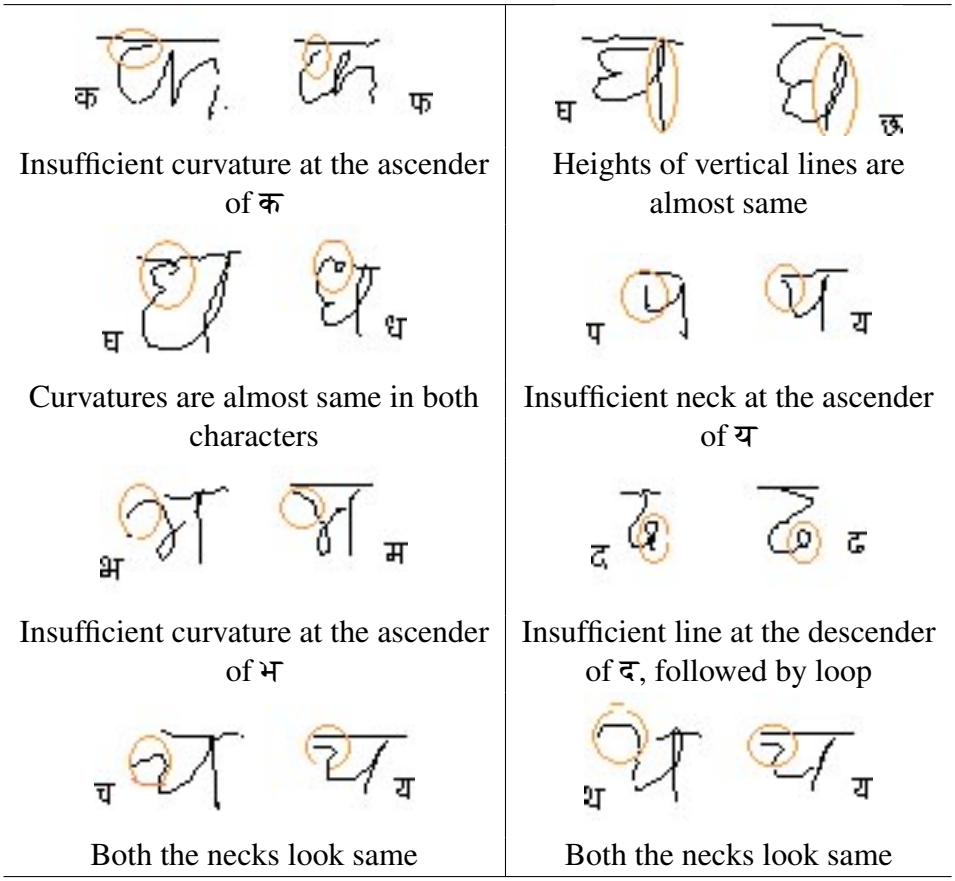

Figure 3.7 Confusion pairs and their reasons of similarity

3.3 Experiment I: Character Recognition

3.3.1 Dataset

In the first experiment, twenty users were employed in which each user has a chance to write three times per class of character. Thirty one pure consonants and five major vowels were taken. Therefore, the complete dataset was composed of 2160 characters (20 users $\times$ 36 classes of characters $\times$ 3 times). Two datasets of 36 classes of characters comprising 20 users were built. We conducted two experiments separately from two datasets, shown in Table 3.1 and Table 3.2.

In first dataset, one-third of every class of character (consonants and vowels) from a complete dataset was taken training the system and remaining two-third characters of every class character from every user are taken as test data. Note that the system was trained with original (without pre-processing) writings (samples). In training the system, each character in a frame was composed of twenty clusters' representatives and hence the total number of feature vector sequences of the strokes from all classes training characters were 720 (36 classes $\times$ 20 clusters' representatives) as training samples (templates). In addition to this, 120 characters were randomly taken from those users who were not included in collecting the data. It contained 60 consonants and 60 vowels, i.e. they missed some characters in the set of consonants. They wrote those characters upon which they were interested. These users were

Table 3.2 Experimental Results (System II)

Char. Type (Classes)	Training Chars.	Test Chars.	Misrecog. Chars.	Err.	Overall Err.
Consonant (31)	1240	620 (+60)	95 (+13)	15.32% (21.67%)	15.88%
Vowel (5)	200	100 (+60)	12 (+10)	12.00% (16.67%)	13.75%

No. of users: 20

Pre-processing: Nil

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

Accuracy: 84.52%

Average Speed: 32 Seconds/Character

Total Bad Data: 4.66% (Training and Test)

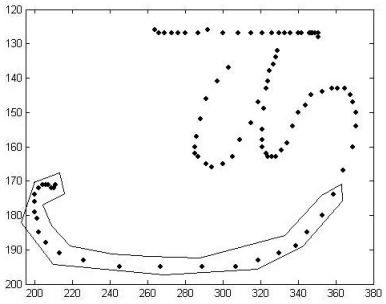

Figure 3.8 A rejected sample (क) due to very long descender

very anxious to check during the experimenting time in the laboratory (KIND laboratory, Information and Computer Technology, Sirindhorn International Institute of Technology, Thammasat University, Thailand).

In second dataset, training data and test data from the first dataset were reversed. So, training dataset was composed of two-third of a complete dataset and remaining one-third were for testing the system. In training the system, each class of character was composed of 40 clusters, due to the large number of samples along with variability in writing. Therefore, we have 1440 (36 classes $\times$ 40 clusters' representatives) feature vector sequences as training samples (templates). As in the first system, 120 additional characters from the same source were tested.

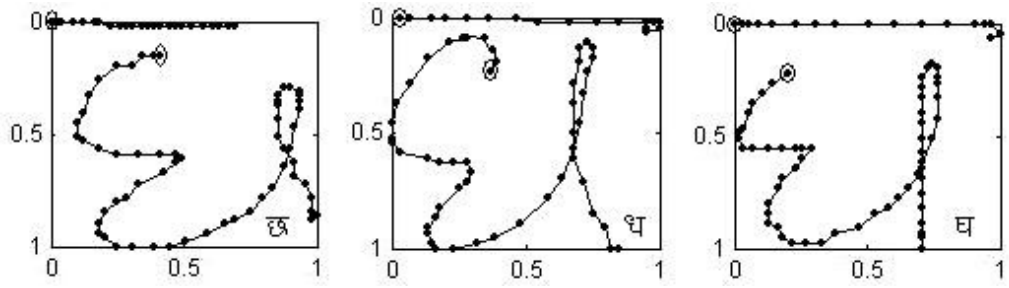

Figure 3.9 A sample of similar characters for humans to read

3.3.2 Results

The experimental results are shown in Table 3.1 and Table 3.2 for both system I and system II respectively. It uses 1 Fold-Cross Validation experiment. System I was trained with small dataset and tested with large dataset. In contrast system II used large dataset for training. Due to this reason, System II received higher recognition rate than system I. Correct classification rates are 78.84% and 84.52% from system I and system II, respectively. The recognition speed of the system is also counted under the classifier's performance. As the classifiers used matching procedure for identification of test strokes, the speed was variably determined. The number of strokes in the test letter, sizes of the feature sequences of the test strokes and the number of templates affect the recognition speed. The average speeds of two classifiers are 18 seconds and 32 seconds per character, respectively.

3.4 Experiment II: Alphanumeric Character Recognition

3.4.1 Dataset

We were interested in building a stroke number and order free handwritten alphanumeric character recognition, and hence it encouraged us to collect much data from many users as far as possible such that it can cover a wide ranges of writing styles. Users were encouraged to write with possible variable stroke number and stroke order. In this experiment, 46 alphanumeric characters were taken for training the system. It included thirty one classes of pure consonants, five classes of major vowels and ten classes of numerals. Twenty five users (Nepalese natives) were requested to write two times for each class of character and numeral. Therefore, the dataset was composed of 2300 (25 users $\times$ 46 classes $\times$ 2 times per class) alphanumeric characters. The data were taken in different days even from one user. Sixty percentages of total data were taken for testing the system while the remaining forty percentages were for testing. It means 15 users are used from training and remaining 10 users are for testing. Altogether, 1380 (30 $\times$ 46 classes) and 920 (20 $\times$ 46 classes) alphanumeric characters were training data and test data respectively. As, there were 25 strokes' representatives in one frame, 1150 (25 strokes' representatives $\times$ 46 classes) strokes' representatives were stored systematically from all classes of alphanumeric characters.

Table 3.3 Experimental Results

Dataset	Char. Type (Classes/Chars.)	Misrecog. Chars. (Err.)	Overall Err.
Training (15 Users)	Consonant (31/930)	34 (03.65%)	03.69%
	Vowel (5/150)	3 (08.66%)	
	Numeral (10/300)	04 (01.34%)	
Test (10 Users)	Consonant (31/620)	98 (15.80%)	13.37%
	Vowel (5/100)	17 (17.00%)	
	Numeral (10/200)	08 (04.00%)	

No. of users: 25

Pre-processing: Nil

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

Accuracy: 86.63%

Average Speed: 27 Seconds/Character

Total Bad Data: 2.7% (Test data)

3.4.2 Results

Table 3.3 shows the experimental results of alphanumeric characters test. Overall correct classification rate is 87% for all types of characters including consonant, vowel and numeral. Separately, recognition rates are 84.20%, 83% and 96% recognition rates from consonant, vowel and numeral category respectively. Accuracy is significantly higher in numeral category in comparison to others. The average recognition speed is 27 seconds per character.

3.5 Discussions

One of the biggest problems is the similarity among the characters due to the script itself and writing styles, for example: character pairs थ↔य, क↔फ, च↔थ, छ↔ध, य↔प, स↔झ, ढ↔द, इ↔ड, ठ↔ट etc. A few confusions pairs and their reasons of similarity/dissimilarity are demonstrated in Fig. 3.7. Among these confusion pairs, many times प, च, थ are confused with य. This is the poorest performance of the system. Looking into the experimental errors (Table 3.1 and Table 3.2), it has a record that 4.66% of the complete data set was found to be bad data. But it does not mean that all the misclassified characters are grouped

Table 3.4 Error Analysis (Test Data)

Error Type	Chars. (Err.)
Feature Similarity	69 (07.50%)
Diminished and/or very long Ascender and Descender	16 (01.73%)
Mis-writing	18 (01.95%)
Re-writing	13 (01.41%)
Miscellaneous	07 (00.76%)

Pre-processing: Nil

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

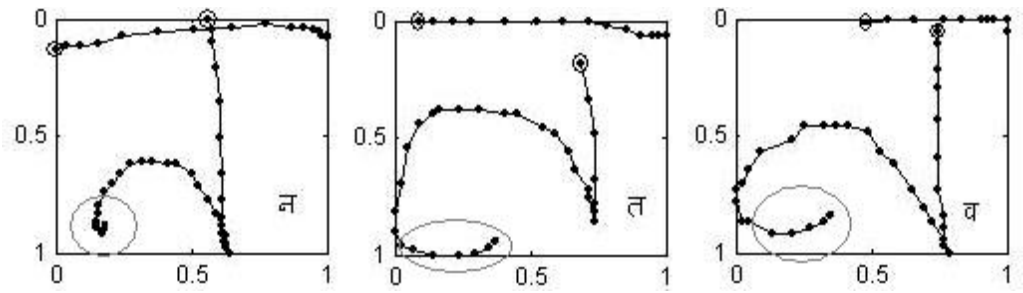

Figure 3.10 Three samples of characters having diminished descenders

under the category of bad dataset. Broadly, bad data refers to those characters, on which humans are also confused to read. The misclassified characters are also given for humans to read. In some cases, human fails to read those characters. Even human can not able to read, how one can imagine about the system's performance with these misclassified characters. In another word, all characters, which do not have any information about the characters' shape, used unnecessary number of strokes to complete the characters and are not read by humans, are categorized under mis-writing and miscellaneous error type. Due to the lack of experiences in writing characters by using digital tablet, writing with shivering hands during the pen movement drops the recognition rate. Therefore, the drawings of characters are unique, and they loose their characteristics, finally, mis-recognition do exist. Not only these, some characters have diminished and/or very long ascender and descender (tremor handwriting), which were also confused. One of the examples of such drawings (having long descender) is shown in figure 3.8, where series of unnecessary dots inside the polygons deform the shape of क.

Considering the statistic of bad data from the complete dataset appeared in the experiment, the system's reliability can be more than 78.84% in system I and more than 84.52% in system II. Both the systems suffered from the additional testing characters. This is because of

the lack of training samples (templates) of such kinds in the recognition systems. However, the experiments prove that the accuracy can be improved by increasing the number of training samples (templates) at the cost of time. In contrast to system I, the number of training templates is two times greater and hence system II performed better.

With the help of Table 3.3, it is apparent that the recognition rate is higher in numeral's category in comparison to consonant and vowel. This is due to the fact that most of the users used only one stroke to complete a numeral. Having a stroke for one to complete a numeral, the recognition time is also reduced significantly. A stroke contains a complete information about the structure of numeral has less chance of misclassification. However, it can be misclassified in case the stroke does not contain the information about the shape of the numeral. Noise, due to the erratic hand motion, digitizing process and inaccuracy in pen down indication draws the recognition rate. Human performance is also considered as in the previous experiments. Some of the confusions' pairs are shown in Fig. 3.9. Along with this, some similar pairs in numerals are, (०, ४ and (०, ९). The speed in writing also affects the system's performance. The speed determines how densely the coordinates along the sequence during the pen movement are collected. For slow pen movement, the number of coordinates along the sequence are densely populated and some times overlapping while, rarely distributed in fast handwriting. In case of the structural feature of the stroke, fast handwriting does not give an overall skeleton of the stroke what the users try to draw. Table 3.4 analyzes the possible error types during the experiment. Characters with diminished descenders are shown in Fig 3.10. This confusions are now given to the reader for distinguishing. Similarly, miswriting and rewriting characters are also appeared. This is due to non-confidence in writing (write two or more than times for the same stroke). Mis-writing character does not have any information about the shape but it contains a lot of strokes. Along with these problems, recognizer does not recognize the letter with rewriting strokes to complete previous stroke at the end of the writing (Fig. 3.12), even though the character with re-writing strokes is readable for human. User writes third stroke at the end of writing in order to isolate the drawings from their similar characters, i.e. to isolate भ from म and क from फ. Some of the rejected alphanumeric characters are under the miscellaneous error type, because their actual types are not identified clearly. Addition of 10 classes of numerals to 36 classes of characters in a system does not have any effect in recognition because numerals and characters are categorized in separate frames, based on the number of strokes required to complete a letter. Due to this, problem of some numerals having features similar with characters like, ८→ट, ७→ढ, ६→छ, २→र etc. are eliminated. Nevertheless, the numerals with more than one stroke are misclassified, which are very few in this dataset.

Considering the bad data appeared in the test dataset during the experiments (2.61%), the reliability of the system is greater than 86.63%. A numeral completed with a stroke is often recognized, while numerals with two or more than two strokes are misclassified. One stroke numeral is sometimes misclassified due to insufficient information about the numeral. Not surprisingly, human intelligence is also taken for reading only misclassified characters during the entire experiment period.

In a conclusion, classifier uses the data directly from the users without using any prescreening techniques . As real time data are noisy, incomplete, inconsistent and bulky as well, most of the times test characters are misclassified due to various causes like, similarity among the large pairs of characters, diminished or very long ascender and descender of the

Table 3.5 Experimental Results

Dataset	Test Char. Type (Classes/Chars.)	Misrecog. Chars. (Err.)	Overall Err.
Training (15 Users)	Consonant (31/930)	31 (03.33%)	03.40%
	Vowel (5/150)	3 (08.66%)	
	Numeral (10/300)	03 (01.00%)	
Test (10 Users)	Consonant (31/620)	84 (13.54%)	11.63%
	Vowel (5/100)	18 (18.00%)	
	Numeral (10/200)	05 (02.50%)	

No. of users: 25

Pre-processing: Yes

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

Accuracy: 88.37%

Average Speed: 23 Seconds/Character

Total Bad Data: 2.7% (Test data)

stroke and re-writing stroke. Most of the characters misclassified, which are to be classified correctly. Further, some characters are rejected due to very small noise which can be eliminated. This has been seen as all the misclassified characters are plotted in case of misclassification/rejection during the entire experiment. This is why the need of pre-processing (pre-processing: to enhance the quality of data, accuracy of the classifier and the recognition speed) is there to solve common problems in natural on-line handwritten character recognition to some extent.

3.6 Experiment III: Inclusion of Pre-processing and its Effect in Recognition

The use of appropriate pre-processing has made the substantial effort in recognition. In other words, every technique used in stroke pre-processing is significantly placed in designing the recognition system. Up to here, the proposed system does not include pre-processing technique. This time, inclusion of pre-processing is demonstrated with the expectation of reducing bad data, which were often appeared in on-line data. Therefore, accuracy would be improved.

In this task, we explore the efficacy of various stroke-based handwriting analysis strategies

Table 3.6 Error Analysis (Test Data)

Error Type	Chars. (Err.)
Feature Similarity	54 (05.86%)
Diminished and/or very long Ascender and Descender	15 (01.63%)
Mis-writing	18 (01.95%)
Re-writing	13 (01.41%)
Miscellaneous	07 (00.76%)

Pre-processing: Yes

Feature: Direction at every point along the pen trajectory ($f_t = \theta_t$)

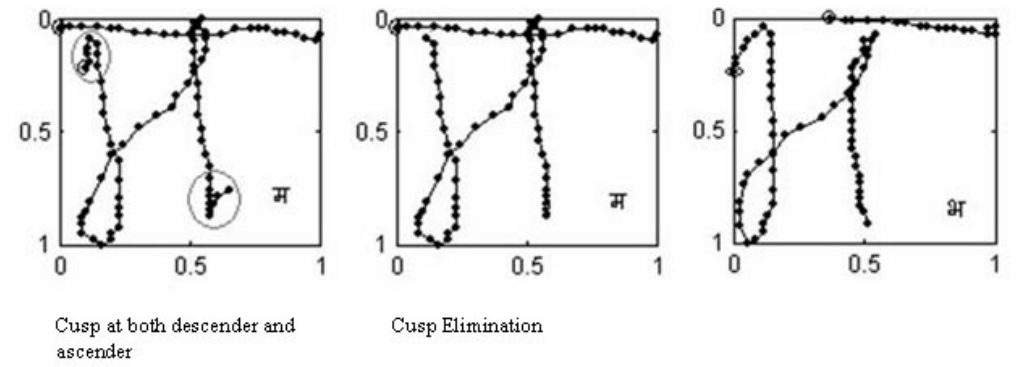

Figure 3.11 Effect of pre-processing in correct recognition

in classifying Nepali handwritten characters. We constructed a prototype classifier and then performed an experiment by using pre-processed samples. The effect of Nepali stroke-pre-processing is carefully demonstrated.

3.6.1 Dataset

The same dataset (Table 3.3) was used to compare the result with the experiment, where original samples were taken for training and testing the system. Our database, collected in separate days from 25 natives was composed of 2300 alphanumeric characters from 46 classes (25 users $\times$ 2 times $\times$ 46 classes). The dataset was comprised of natural handwritings. As in the previous training samples, there were altogether 1150 (25 strokes' representatives $\times$ 46 classes) pre-processed strokes' representatives were stored systematically from all classes of alphanumeric characters.

3.6.2 Results

Table 3.5 illustrates the effects of specific stroke pre-processing by using both training and test data separately. We measured recognition accuracies of every character type (consonant, vowel and Numeral) and compared one another. In contrast to experiment II (Table 3.3), recognition accuracy is increased from 86.63% to 88.37% for test alphanumeric characters with the use of pre-processing techniques, which is shown in Table 3.5. How classification rate improved is explained in the discussions section. The recognition speed of the classifier is 23 seconds per character, which is reduced from 27 due to the removal of noisy sequences.

3.7 Discussions

In contrast to system with original samples (Error Analysis: Table 3.4), the system with pre-processed writing samples (Error Analysis: Table 3.6) performed better. Through the experimental results, the investigation is carried out in major problems faced as well as mis-classified alphanumeric characters and their reasons in both the system (Table 3.3 and Table 3.5). Table 3.4 and Table 3.6 analyze the error occurred in the experiments. We receives a little bit higher recognition rate in pre-processed writing samples. Nevertheless, difficulties arise when pre-processing because the improvement of one class of alphanumeric character sometimes lead to the deterioration of another. For example, chopping cusps at ascender and descender is some times change the shape of the writing such as: भ→म, ध→घ, थ→य, न→त, व→त, ढ→ट etc. whereas reverse confusions are reduced. The reverse confusions are happened in system, where pre-processing is not used. An example of how it works is presented in Fig. 3.11. म is often confused with भ because of the similarity in structure (having very small distinguishing zone (curve)). However, it is recognized correctly after pre-processing. This is what the attempt of using pre-processing techniques, and is useful to other similar pairs too. In comparison to Table 3.4, this classifier reduces the number of confusions due to characters' similarity. As before, tremor handwriting produces diminished ascenders or descenders and sometimes very long descenders that makes systems confused. Fig. 3.10 shows some samples of diminished descenders. Along with these, the rejection samples due to re-writing strokes do exist as before. In a similar way, mis-writing alphanumeric characters affect in correct classification.

Looking at a glance into the experimental results (Table 3.5) and error analysis (Table 3.6), the reliability of the classifier is more than 88.37% in case the bad data is considered. Recall that bad data includes mis-writing characters and character under the category of miscellaneous error type.

It is concluded that specific stroke pre-processing performs better for some of the similar pairs of characters, however, it has no substantial effect on character having diminished and or very long ascender and descender, mis-writing character, re-writing character and so forth. This may be due to the fact that the used feature may not have enough information to classify. Therefore, we come with the idea that the feature with much information about the strokes will obviously distinguish from others. Inclusion of pen-tip position with tangent angle at every position along the trajectory of the pen movement as the feature is proposed in the next section.

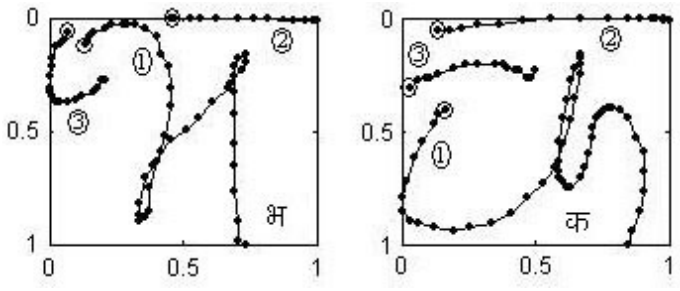

Figure 3.12 Two rejected samples (भ and क) due to rewriting strokes

3.8 Experiment IV: Inclusion of Pen Position with Pen Direction as the Feature

In this section, pen coordinate feature is added with the previous tangent angle feature instead of using only the tangent angle. Here, one dimensional feature vector sequence of a stroke become two dimensional. Inclusion of pen-tip position provides a bit more information about the stroke. In a sense, feature vector sequence now contains two dimensional feature event: $f_t = (x, y, \theta_t)$. The effect of using this feature is demonstrated with the help of experimental results (Table 3.7) and error analysis (Table 3.8).

3.8.1 Dataset

It used the same dataset as in Table 3.3 and Table 3.5 for comparison and improvement from the previous analysis. As in Table 3.5, there were altogether 1150 (25 strokes' representatives $\times$ 46 classes) pre-processed strokes' representatives were stored systematically from all classes of alphanumeric characters. Recall that it was comprised of natural handwriting.

3.8.2 Results

Table 3.7 provides the complete experimental results for both training and test data set. Both data set were tested separately on to the same prototype classifier. We measured recognition accuracies of every character type (consonant, vowel and Numeral) and compared one another. Performance of the classifier is more than 89.79% in case the bad data are taken in consideration. How classification rate improved is explained in the discussions section. The average recognition speed is 35 seconds per character. In contrast to a one-dimensional feature ($f_t = \theta_t$: Table 3.5), it consumes more time during matching for a two-dimensional feature ($f_t = (x, y, \theta_t)$ even when noisy sequences are removed.

3.9 Discussions

In contrast to experiment III (Table 3.5), recognition accuracy is increased from 88.37% to 89.79% for test alphanumeric characters with the use of new feature (2D feature), which is shown in Table 3.7. For better visualization, a graphical demonstration including a complete comparison of the classifiers is provided in Fig. 3.13. Use of 2D feature has made

Table 3.7 Experimental Results

Dataset	Test Char. Type (Classes/Chars.)	Misrecog. Chars. (Err.)	Overall Err.
Training (15 Users)	Consonant (31/930)	19 (02.04%)	02.02%
	Vowel (5/150)	(04.67%)	
	Numeral (10/300)	02 (00.67%)	
Test (10 Users)	Consonant (31/620)	78 (12.58%)	10.21%
	Vowel (5/100)	12 (12.00%)	
	Numeral (10/200)	05 (02.00%)	

No. of users: 25

Pre-processing: Yes

Feature: Pen-tip position and direction at every position along the pen trajectory ($f_t = (x, y, \theta_t)$)

Accuracy: 89.79%

Average Speed: 35 Seconds/character

Total Bad Data: 2.48% (Test data)

the substantial improvement in correct classification. It reduces the confusion in most of the similar pairs of characters, classifies correctly some of the characters having diminished and or very long ascender and descender. However, the number of misclassification of mis-writing characters, characters with re-writing strokes are remain same. The number of misclassified characters under the feature similarity is reduced to 46 (Table 3.8) from 63 (Table 3.6). Only the feature is different from one experiment to another. Both classifiers use same pre-processed training samples. With a series of experimental results (from Table 3.3 to Table 3.8), the on-line handwriting recognition systems are evaluated, which are based on pre-processing strategies and feature selection. Up to here, the influence of feature selection and pre-processing techniques on recognition performance with a template-based approach for Nepali handwritten alphanumeric characters is explored. Either the system with feature: $f_t = \theta_t$ or $f_t = (x_t, y_t, \theta_t)$ of each stroke sequence, use of stroke pre-processing performed better. Moreover, the system using feature event: $f_t = (x_t, y_t, \theta_t)$ achieved the higher recognition rate for both original and pre-processed writings. The system combining preprocessing with both the position and direction of each point in each stroke performs better than other combinations. Applying a combination of both feature: $f_t = (x_t, y_t, \theta_t)$ and pre-processing receives the best result (89.79%) among other combinations. Experimental results are analogical, competitive and encouraging.

Table 3.8 Error Analysis (Test Data)

Error Type	Chars. (Err.)
Feature Similarity	46 (05.00%)
Diminished and/or very long Ascender and Descender	13 (01.41%)
Mis-writing	18 (01.95%)
Re-writing	13 (01.41%)
Miscellaneous	04 (00.43%)

Pre-processing: Yes

Feature: Pen-tip position and direction at every position along the pen trajectory ($f_t = (x, y, \theta_t)$)

Despite of using pre-processing technique and quality feature of stroke ($f_t = (x_t, y_t, \theta_t)$), alphanumeric characters with similar features holds high error than other types. The tremor handwritings, children handwritings and writing with insignificant strokes for completing previous stroke (rewriting strokes) are also considered in these systems too, which are difficult to classify correctly.

Can the system having the combination of feature: $f_t = (x_t, y_t, \theta_t)$ and pre-processing receive the best result for all kinds of dataset, in comparison to other combinations? One way experiment (Table 3.5 and Table 3.6) is not enough to answer the question. The question is now answered with the help of K-fold cross validation experiment. The system is compared with another system having the combination of feature: $f_t = (\theta_t)$ and pre-processing. A complete comparative experiments are shown in the following section.

3.10 Experiment V: Appropriate Classifier Selection: Cross Validation (XV)

The anxiety to know the appropriate feature in the running classifiers is the primary question whenever pre-processing techniques are used. The effective evaluation standards and criteria are important to provide the the confidence results from both classifiers in order to select the appropriate one. The assessment should be an objective and have no preferences to any one. The standard to test the systems should answer the questions very clearly, whether one of the classifiers has meaningful difference in terms of reliability of the classifiers. One of the standard methods to test is Cross Validation (XV).

Leave-one-out cross-validation may run into trouble with various model-selection methods. However, one problem is lack of continuity-a small change in the data can cause a large change in the model selected (Breiman, 1996). Leave-one-out cross-validation often works well for estimating generalization error for continuous error functions such as the mean

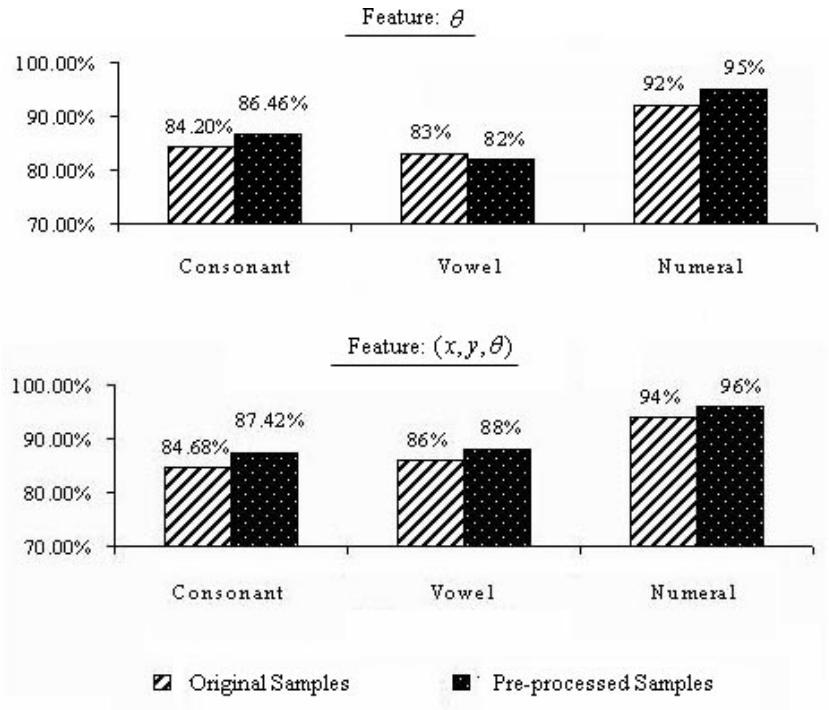

Figure 3.13 A complete comparison of four classifiers based on their accuracies

squared error, but it may perform poorly for discontinuous error functions such as the number of misclassified cases. In the latter case, K-fold cross-validation is preferred. But if K gets too small, the error estimate is pessimistically biased because of the difference in training-set size between the full-sample analysis and the cross-validation analysis. In this case, K-fold cross validation is used to test the systems, where $K = 5$. Therefore, data set is divided into (K) 5 subsets, and the holdout method is repeated up to (K) 5 times. Each time, one of the (K) 5 subsets is used as the test set and the other $(K-1)$ 4 subsets are put together to form a training set. Then the average error across all (K) 5 trials is computed.

3.10.1 Dataset

The complete dataset, collected in several days from 25 natives was composed of 2300 alphanumeric characters from 46 classes. A complete data was broken down into five subsets, each consisting of 460 characters. The same dataset used in Table 3.3, Table 3.5 and Table 3.7 was taken.

3.10.2 Results: Five-fold Cross Validation

Two better classifiers are tested with the help of cross validation technique. Table 3.9 shows the error rates of two classifiers using different features. Both the classifiers uses pre-processed samples, but the feature is different from one another. In the first case, feature: $f_t = (\theta_t)$ is used while, $f_t = (x_t, y_t, \theta_t)$ is in the second one. Through the a series of experimental results, the average error rate is 12.09% with the standard deviation of 0.762 from first classifier. In contrast, second classifier performed with an average error rate of

Table 3.9 Experimental Results (Five-fold Cross Validation)

Experiment	Classifier's Err.	
	Classifier I	Classifier II
	Feature: $f_t = (\theta_t)$	Feature: $f_t = (x_t, y_t, \theta_t)$
1	11.09%	09.57%
2	11.74%	11.09%
3	12.61%	11.74%
4	11.96%	11.53%
5	13.05%	11.31%
Average	12.09%	11.05%
Std. Dev.	0.763	0.861

11.05% and standard deviation of 0.861. With the help of one tailed T-test, two classifiers perform significantly differently with 95% confidence.

3.11 Discussions

Through the experimental results analysis, second classifier perform better in comparison to first classifier. Note that the better classifier uses both position and direction at every position along the trajectory of pen movement as the feature of the stroke ($f_t = (x_t, y_t, \theta_t)$). Thus, it is concluded that the inclusion of position of the pen in the plane with the direction at every sample point as the feature captures more information than only the directional feature. Feature selection and the pre-processing techniques play vital role in the classifier's performance. The feature used in this system may fit with other scripts but not all the pre-processing techniques.

3.12 Major Difficulties, Limiting factors, Needs and Possible Improvements

Firstly, the major difficulties mentioned in section 3.2 are considered to check how far they are recovered using the proposed methodologies. Taking this in mind, the classifier is developed using appropriate techniques such as, stroke pre-processing (size normalization, cusp elimination, overlapping coordinates deletion), pen position and tangent angle at each sample point as the feature of the sequence and DTW for similarity measure. Cusp elimination improves the classification rate (Fig. 3.11) and co-occurrence coordinates deletion improves the speed of the classifier. Directional property of writing is preserved in every writing. The

feature produces the correct skeleton of the character's shape (Fig. 3.2). As it preserves the complete skeleton along with the order, there is no problem with the tilted handwriting. No skew transformation is necessary. DTW algorithm is used to solve the problem of non-linear sequences alignment. The classifier takes an advantage of a template-based approach in classification, i.e. larger the number of templates with variability in writing style greater the classification rate.

Tremor Handwriting and or children handwriting, mis-writing and character with re-writing strokes are some of the limiting factors in classification. In addition, fast handwriting are often misclassified.

One of the biggest problems is similarity in drawing among many classes from many of the users (due to scripts itself and writing styles) is still in error hand to minimize. Specifically, this is the need to design a classifier for correcting such a problem in this script. Some of the similarity pairs ((भ, म), (थ, य) and (ध, घ) etc.), which are often confused to each other, is the point to improve. The above mentioned confusion pairs have two dissimilarity features, one is the curve (loop) in the text and second one is the position of the horizontal line (shirorekha). The placement of shirorekha in case of भ, थ and ध is always on the top-right position with respect to the text, while it is on top in case of म, य and घ. However, it is not guaranteed that the shirorekha is always written on the same position with respect the text. Instead, classifier may take another dissimilarity feature to differentiate the characters. This is how, the performance of the classifier can be increased by using the spatial relation between/among the strokes within a class of character.

A novel approach to Nepali character recognition system based on spatial relation between/among the strokes is proposed in the following chapter. How can the spatial relation between/among the strokes be determined? How can it help in clustering? What will be the performance of the classifier? These questions are answered sequentially in the following chapter. Moreover, improvements would be achieved by using some post-processing techniques to those which have less confidence in correct classification.

Chapter 4

A Structural Approach on a Template-based Handwritten Character Recognition along with the Additional Use of Strokes' Spatial Information

4.1 System Design: Spatial Relation between/among the Strokes based Clustering - A Modern Approach to Writer Independent Character Recognition.

In this task, a new scheme is purposed in building a prototype classifier for classifying Nepali natural handwritten characters in order to overcome the shortcomings in the previous classifiers. It uses a novel idea for analyzing a character based on both the number of strokes used and the strokes' spatial relation within a character for Nepali. Handling spatial information about the strokes is not the new technique in handwriting recognition field (Murakatat, et al., 2004; Bouteruche et al., 2005), however, this task provides different strategy in collecting spatial information and different aspects in designing a prototype recognition engine.

The novel idea different from previous related tasks goes like this.

Firstly, the classifier determined the number of strokes used to complete a character and then separated based on this into different groups. It means those characters having identical number of strokes are collected in one group whereas the dissimilar are in other groups.

Secondly, captured strokes are separated and locations from each sequence to another within a character are determined.

Thirdly, strokes' representatives from a number of similar strokes are determined using DTW based on the distance calculation. Only those strokes of a specific class of character of the specific group are sent to clustering block instead of using all strokes of every character (previous case). Use of both number of strokes and their locations in clustering enjoys the advantages of not merging two different strokes from different groups. However, it has a chance of having different features in a same space of a same character from different users, but is negligible.

Fourthly, strokes' representatives after merging are stored as the templates from every class of character. This ends the training strategy.

Finally, a character is tested stroke-by-stroke basis. A simple template matching procedure is carried out in stroke identification as in the previous classifier. However, the way of matching

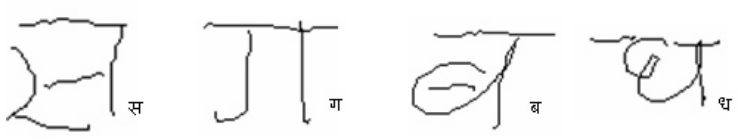

Figure 4.1 Four samples having multiple straight sequences

the test stroke with templates is different. Each stroke is aligned with those templates, which have both identical spatial properties and number of strokes used to complete a character as that of a test character. Hence, the basic idea to recognize a complete test character is to identify all the components (strokes) according to their order. Creating spaces for the templates and matching are the common approach; however, this task explores different aspects in templates' placement and matching for easier classification.

4.1.1 Spatial Relation between/among the Strokes Based Clustering

Spatial information has been a great effort in stroke identification along with the use of feature in case of Nepali characters. Once all strokes are identified, character is classified. Spatial information gives both location and the size of the string from one another. Every character has one 'shirorekha' in which text is suspended. Some times two (Fig. 4.1), which depends on the users' style in writing and appear in some characters only. Having at least one 'shirorekha' in every character has been a unique feature sequence in Nepali in comparison to other scripts. Looking into the users' natural handwriting, the 'shirorekha' may not always be horizontal (it may be either a line with negative or positive slope) and always not on the top exactly (sometimes top-left, sometimes top-right and sometimes it may intersect with the text), what it should be. Not only this, it may also be a small curve sometimes, which acts as a 'shirorekha' but not the text. As locations of both 'shirorekha' and texts vary widely, it is difficult to build a general rule for the determination of spatial relation among them. In general, location of the 'shirorekha' is assumed on the top portion (not specific like, top, top-right and top-left) and then the text(s) in reference to 'shirorekha' is/are determined.

Sequences are provided with six regions: top-left, top, top-right, bottom-left, bottom and bottom-right. In order to determine the location of the stroke, two main criterions are carried out, which are; boundary and angle criterions. Boundary condition uses both maximum and minimum values of both horizontal and vertical axis, while center of gravity of the strings are used in case of angle condition to be checked. Fundamentally, every character has at least two kinds of sequences, one is straight sequence and another is curve. Naturally, a straight sequence resembles a 'shirorekha'. In order to identify a sequence either a straight or a curve sequence, the following conditions are checked.

$$St-CS = \frac{d(\mathbf{p_1}, \mathbf{p_l})}{\sum_1^{l-1} d(\mathbf{p_i}, \mathbf{p_{i+1}})} \tag{4.1}$$

where, $d(\mathbf{p_1}, \mathbf{p_2}) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$, gives the distance between two coordinates $\mathbf{p_1}$ and $\mathbf{p_2}$.

- *For Straight sequence*: $0.8 \leq St-CS \leq 1$

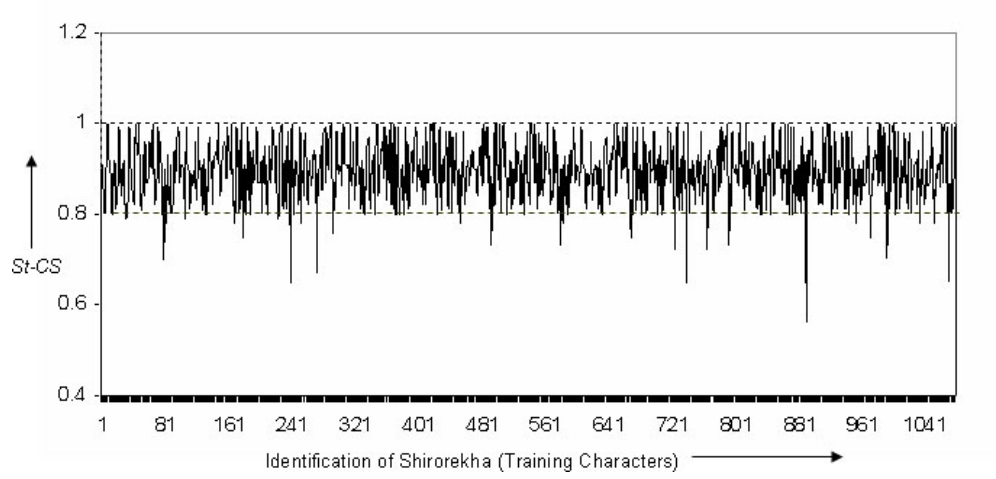

Figure 4.2 Threshold range $(St - Cs)$ determination for shirorekha

- *For Curve sequence*: $0 \leq St - CS < 0.8$

How did we determine the range of $St - CS$ for straight sequence ('shirorekha') is shown in the Fig 4.2. It demonstrates that approximately 98% of 'shirorekha' from all training characters are lying within the range mentioned above. Finding straight sequence does not mean that the 'shirorekha' is determined. For the two-stroke characters, it is easy to separate the straight and curve sequence. But, in case of characters having more than two strokes, there may be chance of multiple sequences containing the feature of straight sequence, which are shown in Fig. 4.1. In such a case, we need to determine the 'shirorekha' from many straight sequences because not every straight sequence is the 'shirorekha'. It makes sense that a straight sequence can be a text sometimes. The only difference is, where does it lie? Here, we design extra conditions, which are,

a. Find direction from initial to the end coordinate.
   $Dir.(Angle) = \arctan(\frac{y_l - y_1}{x_l - x_1}) \times (\frac{180^\circ}{pi})$

b. Find width of the sequence along x-axis.
   $Width = x_{max} - x_{min}$

c. Find positional relation among the straight lines. (a concrete boundary condition)

The straight sequence, which has small change in direction either to positive or to negative from initial to final coordinate, has larger width along the x-axis and has positioned on the top in reference to other straight sequences is the 'shirorekha'.

After separating the 'shirorekha' from other curve texts, their positions (based on their size) are determined by using both boundary and angle conditions. Fig. 4.3. shows the basic boundary and angle conditions to test the strokes locations. Assuming that the 'shirorekha' is on the top portion, location(s) of text(s) is/are estimated. Finally, the location of the 'shirorekha' along with its size is confirmed, once the location(s) of the text(s) is/are determined. A real example of two strokes character क is shown in Fig. 4.4. In addition, the character with two 'shirorekha' is also possible to determine the location in the provided regions.

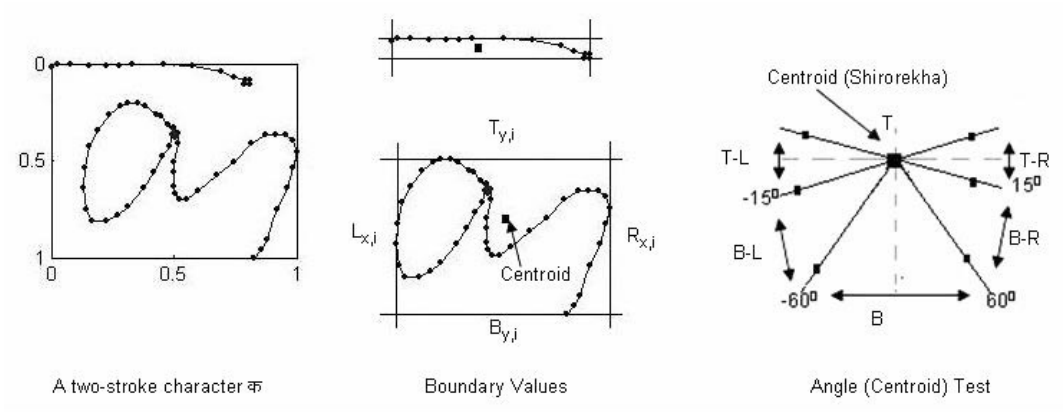

Figure 4.3 Spatial relation between two strokes of the character क

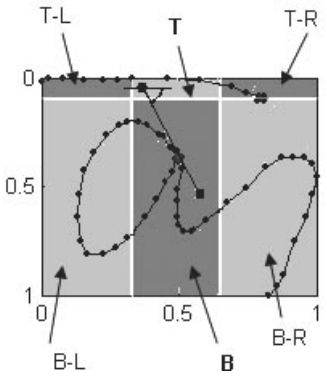

Figure 4.4 A real example of how spatial relation is determined by taking a character क

In such a case, the first 'shirorekha' is the reference for another according to the order of strokes. Three regions: Top-right, top and top-left are provided for 'shirorekha'.

After separating strokes based on the number and the location, a modern approach in clustering is explored. Broadly, a clustering is a technique for collecting items into specific groups, which are similar in some way. Items of one group are dissimilar with another items belonging to another groups. This technique helps in two aspects. Firstly, it reduces the number of similar items into one (representative), i.e. compact system. Secondly, it increases the speed. Clustering consists of two steps. The first step is to organize a number of characters into different groups based on the number of strokes used in completing a character, which is shown in Fig. **??**. It is assumed that the characters having identical number of strokes have almost same way of writing even from different users. The number of strokes used is varied from user to user even within a specific character. However, it is known that at least two strokes are used in completing a character from any of the users. In the second step, a number of similar strokes are agglomerating pair wise within a group. We used single-linkage agglomerative hierarchical clustering. Based on the spatial relation between/among the strokes within a character, agglomerating has taken place. It means a number of strokes which are at specific location (bottom, bottom-right, bottom-left, top, top-right and top left) are feed to the clustering block. How can similarity between two sequences be measured? The answer

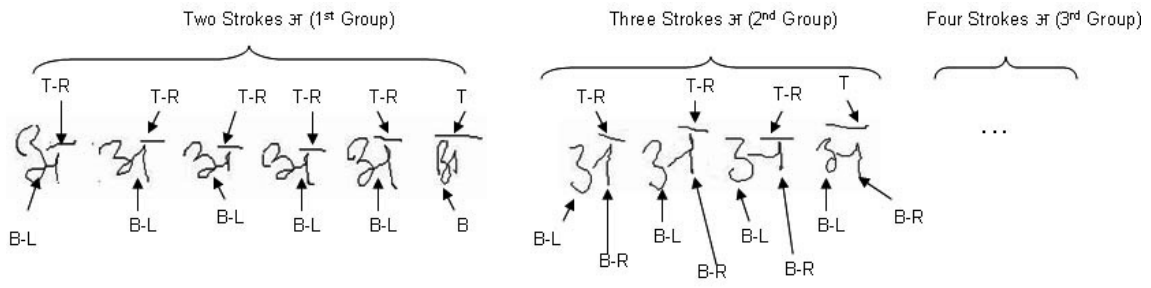

Figure 4.5: Grouping of characters of a specific class अ based on the number of strokes used

is 'distance measurement technique' to estimate the the similarity as in the previous chapter.

4.1.2 Templates' Management

It is important to note that a frame is created for a class of character. More over, there are variable numbers of groups inside each frame because it entirely depends on the number of strokes in completing a character. Fig. 4.6 shows two samples of hierarchical clustering for two strokes अ, where the strokes are clustered based on their locations. In other words, strokes are clustered based on the spatial relation between/among them (texts at bottom-left and bottom are clustered separately and similarly shirorekha on top-right and top). Only six users are employed for the demonstration. It is important to note that the threshold is designed only for text strokes but not the shirorekha. For shirorekha, the number of templates is equal to the number of different locations, but it is variable for texts. More specifically, only those strokes,which are at bottom-left position, are clustered in one group and those strokes on top-right are clustered separately in another group. However, the frame allocated for अ contains every cluster representative from every group in different spaces. For instance, the strokes' representatives from two strokes characters, three strokes characters and four strokes characters are stored in first group, second group and third group respectively. Fig. 4.7 shows a sample of template management strategy by taking few groups. A sample of how we created spaces for those strokes' representatives in each group is demonstrated. A space is used for storing a template. Spaces for both the text and the shirorekha are designed based on where they are located. This is the situation for all classes of characters happened in storing strokes' essence representatives. As the merging of similar strokes has taken place based on the location, it does not have a chance of merging three kinds of shirorekha (i.e. top, top-right and top-left), store even in one group even though they have identical feature. It is because of different locations, as location of shirorekha has a significant place in recognition.

One importance of preserving the locations of shirorekha is illustrated here in some confusion pairs of characters: (ध, घ), (भ, म), (थ, य) etc.

4.1.3 Weight Determination

This section deals with the number of strokes and their spatial relation in reference to one another with the use of probability. In other words, how many strokes are at specific location

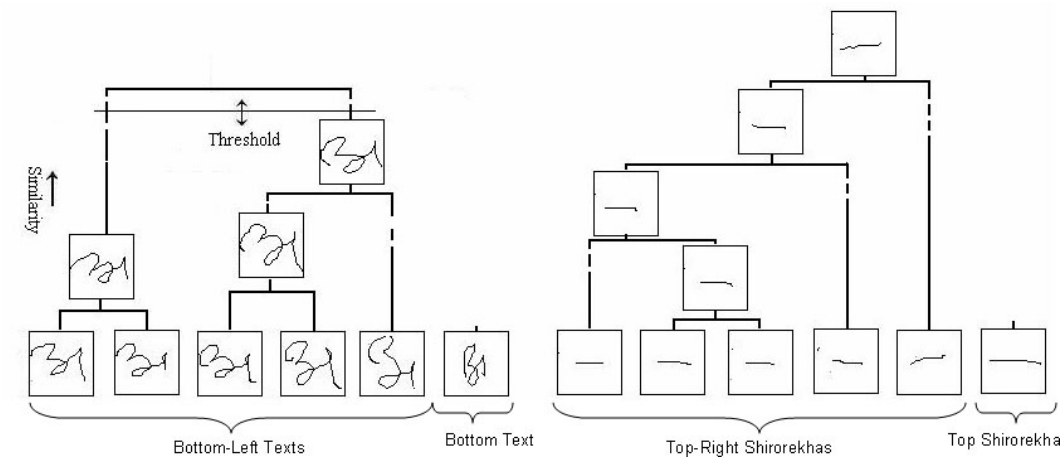

Figure 4.6 Clustering technique for two-stroke अ

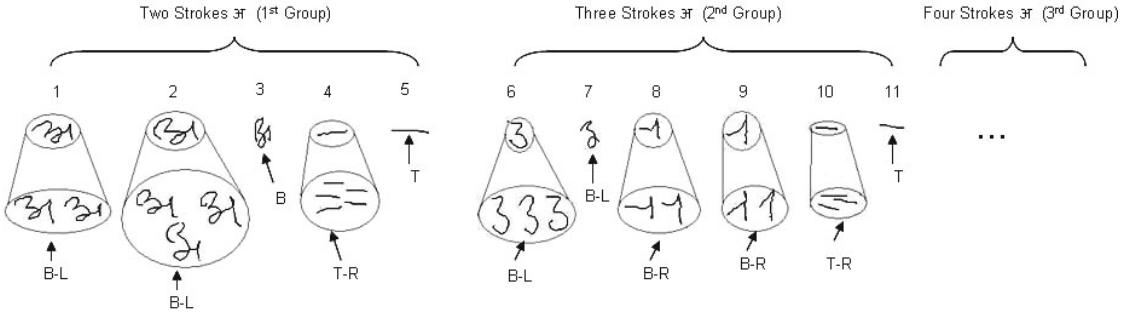

Figure 4.7 Template management for a class of character अ having many groups

in a specific group is the aim to find out in this section. Specifically, it deals with every stroke's representative after clustering in every group.

For instance, let us take a look into Fig. 4.5 for better understanding. Consider the first group having two strokes अ only.

- *For shirorekha*:

  Probability of being on top = No. of shirorekha on top / No. of users = 1/6 = 16.7%
  Probability of being on top-right = 5/6 = 83.4%
  Probability of being on top-left = 0/6 = 0%

- *For Text*:

  Probability of being at bottom = 1/6 = 16.7%
  Probability of being at bottom-right = 0/6 = 0%
  Probability of being at bottom-left = 5/6 = 83.4%

This is the way of calculating weight of each template of every group inside every class of character. Further more, it provides us an idea about how people write and it is important to know the writing pattern of every class of character from many users. These probabilities are considered in template matching in order to find the test strokes in classification block.

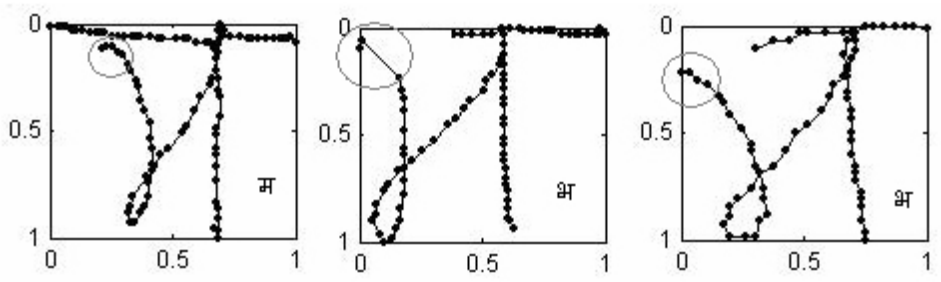

Figure 4.8: Correct recognition of confusion character म (भ) by handling spatial information of the strokes

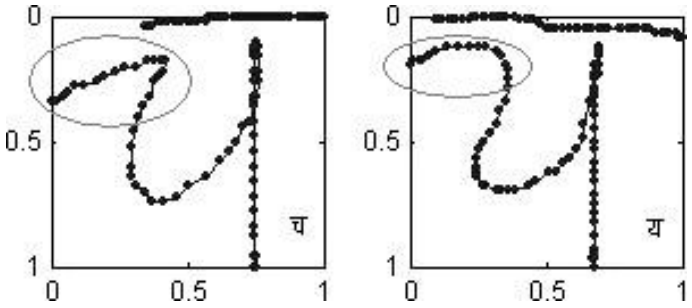

Figure 4.9 Confusion between य↔च

4.1.4 Classification/Recognition

A character is recognized using stroke-by-stroke matching phenomenon. It means any number of strokes used in test character is matched with the templates independently. The stroke is said to be similar with the template from which the lowest distance is produced.

As we have an idea about template management for all class of character in the previous section, we are now able to understand the algorithm used in recognizing the test character. This simple algorithm gives a complete idea of how test strokes are matched with the templates.

a. Determine the stroke' spatial relation.

b. Each stroke is matched only with specific templates. The templates, which are to be matched with the test strokes, are determined based on the number of strokes used in a test character and their spatial relation. For instance, for two strokes test character, matching is involved with only those templates of those groups, which are created from two strokes character. More specifically, shirorekha is matched with only the templates of shirorekha of those groups, where two strokes characters are used for making templates and texts are also matched in the similar manner. Every matching produces a matching score. Mathematically, the distance matrix from n set of test

Table 4.1 Experimental Results

Dataset	Char. Type (classes) (Test chars.)	Misrecog. Chars.	Rejections	Avg. Err.
Training (15 Users)	Consonants (31) (930)	9	2	1%
	Vowels (5) (150)	1	1	
Test (10 Users)	Consonants (31) (620)	29	5	5%
	Vowels (5) (100)	4	3	

No. of Users: 25

Training Samples: Pre-processed

Feature: Position and direction at every point along the pen trajectory ($f_t = (x_t, y_t, \theta_t)$)

Accuracy: 95%

Average Speed: 12 Seconds/Character

Total Bad Data: 1.12%

strokes matching is,

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_1 \\ \mathbf{D}_2 \\ \cdots \\ \mathbf{D}_n \end{pmatrix}$$

where, $\mathbf{D_i} = [\mathbf{F_i^1}, \mathbf{F_i^2}, \ldots, \mathbf{F_i^k}]$. $\mathbf{F_i^k}$ contains some matching scores from specific group of k-th frame when i-th test stroke is matched. We have 36 frames for 36 classes of character. Even though we have fixed strokes' templates in every frame, the numbers of matching scores are from only the specific group, i.e. $\mathbf{F_i^k} = [D^k_{i,1}, D^k_{i,2}, \ldots, D^k_{i,m}]$. $D^k_{i,m}$ is the matching score from m template whenever i-th test stroke is employed. $D^k_{i,m} = \infty$, if m template is not used in matching.

c. Each matching score is divided by the weight of the template based on location. Then, the weighted matching score is,

$$D^{k(w)}_{i,m} = D^k_{i,m} / weight(k,m) \tag{4.2}$$

d. Determine the threshold by taking all matching scores from all frames. Threshold for every i-th test stroke is,

$$Thshld_i = min(\mathbf{D_i}) + C \tag{4.3}$$

where, C is the fixed constant.

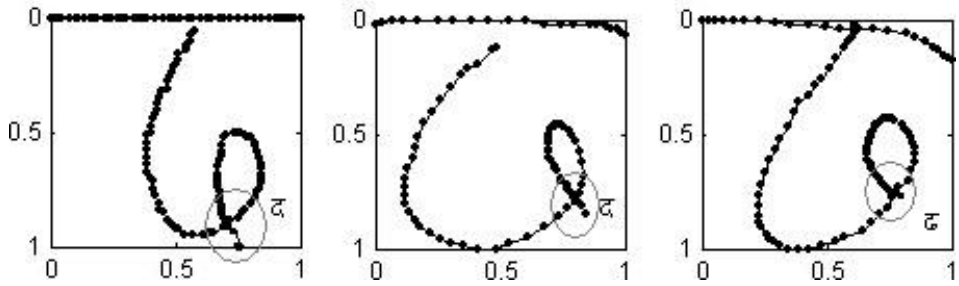

Figure 4.10 Two confusions pairs: द→ढ

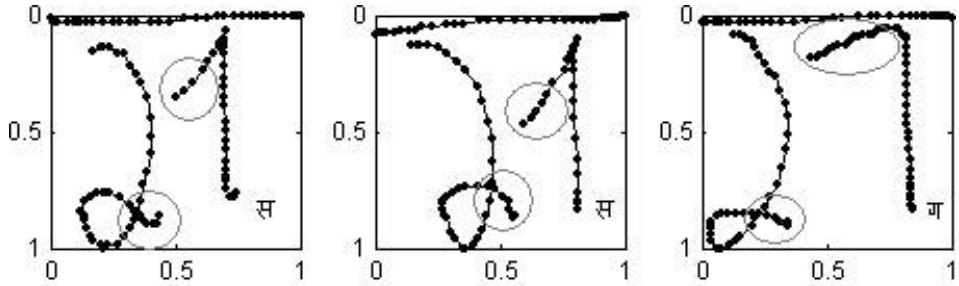

Figure 4.11 Two confusions pairs: स→ग

e. Count the number of matching scores below the threshold in every frame. This number determines how many similar strokes are available as templates. Mathematically, for every i-th test stroke in k-th frame, weight can be determined as,

$$W_i^k = \sum_{j=1}^{m} C(D_{i,j}^{k(w)}, Thshld_i) \tag{4.4}$$

where,

$$c(x,y) = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$$

f. Find the lowest matching score from every matching and from every frame if matching takes place. It should be kept in mind that the lowest matching score is produced only from the most similar template. Mathematically,

$$lms_i^k = min(\mathbf{F_i^k}) \tag{4.5}$$

g. Finally, the character's label is classified as,

$$L = argmin_k \sum_{i=1}^{n} \frac{lms_i^k}{W_i^k} \tag{4.6}$$

The template matching is straight forward when spatial information is not considered. It refers to previous works (K.C. et al., 2006).

Table 4.2 Error Analysis (Test Data)

Error Type	Chars. (Err.)
Feature Similarity	19 (02.63%)
Diminished and/or very long Ascender and Descender	7 (00.97%)
Re-writing	8 (01.12%)
Mis-writing	8 (01.12%)

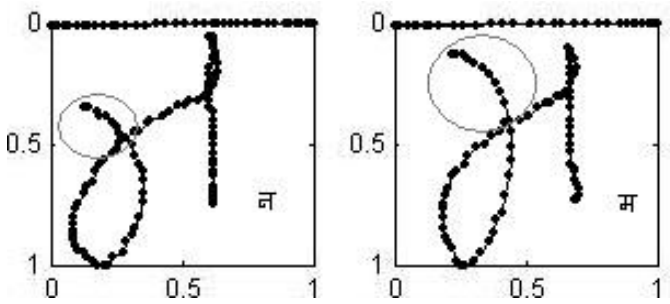

Figure 4.12 Misclassified example: न→म

4.2 Experiment VI

4.2.1 Dataset

The database was composed of 1800 characters from 36 classes of characters, where 25 Nepalese natives were used. Each writer had given a chance of writing two times per class of character. 15 writers were employed for training the system and remaining 10 writers were for testing. As, no directions, constraints and instructions were given to the users, the database was set up of completely from natural handwritings.

4.2.2 Results

Table 4.1 reveals the experimental results for both training and test datasets. Characters were tested one to one basis. Testing of same training characters on to the system confirmed that how good it was trained. Only 1% error rate was received from those training characters. This error rate confirmed that the classifier was trained with 99% intelligence. In the similar manner, 5% error rate was obtained from both consonants and vowels testing. An average recognition speed is 12 seconds per character. The recognition speed of the classifier is significantly reduced due to the technique used in template management, where spatial information about the strokes along with the number of strokes used in completing a character is considered and matching algorithm.

Table 4.3 Class-wise Experimental Results (Test Data)

Class	**Recog.** Char.s	**Confusion**	**Rejection**	**Class**	**Recog.** Char.s	**Confusion**	**Rejection**
Consonant							
क	17	2 (फ)	1	ख	20	0	0
ग	18	2 (स)	0	घ	20	0	0
च	19	1 (य)	0	छ	20	0	0
ज	20	0	0	झ	13	5(स)	2
ट	20	0	0	ठ	20	0	0
ड	20	0	0	ढ	20	0	0
त	18	1(व), 1(न)	0	थ	18	1(य), 1(च)	0
द	16	2(ढ), 1(ट)	1	ध	19	1 (घ)	0
न	18	1(म), 1(त)	0	प	20	0	0
फ	19	1(क)	0	ब	19	0	1
भ	19	1(म)	0	म	19	1(भ)	0
य	18	1(थ), 1(प)	0	र	20	0	0
ल	19	1(त)	0	व	19	1(त)	0
स	18	1(झ), 1(ग)	0	श	20	0	0
ह	20	0	0	क्ष	20	0	0
ज्ञ	20	0	0				
Vowel							
अ	18	1(भ)	1	ए	18	1(ग)	1
उ	19	0	1	ऊ	20	0	0
इ	18	2(ड)	0				

4.3 Discussions

One of the biggest problems, character similarity among many classes is significantly reduced by the help of spatial relation between/among the strokes within every class of character. Some of the similar characters' pairs like, (भ, म), (थ, य) and (ध, घ) etc. are classified correctly by using the strokes' spatial relation in addition to the small curve (dissimilar feature in pairs). It means the positions of the shirorekha of these pairs are at different locations. An example of improvement in accuracy of the classifier is shown in Fig. 4.8, where भ is not confused with another similar character म due to the distinguishing spatial information of 'shirorekha' even when the curve is not clear in the text. But, sometimes, the location of shirorekha is same for both characters, which can be found in Fig. 3.10. In such a case, the idea used in pre-processing: cusp elimination at both ascender and descender of a sequence, has advantages in classifying these similar characters. That is to say, this is helpful in case shirorekha in both the class of characters are at same location. Fig. 3.11 demonstrates the correct classification of similar characters by chopping at ascender. Similarly, Fig. 4.9 and Fig. 4.10 provide an idea how the confusion takes place. Not surprisingly, some of the characters pairs, having a lot of dissimilarity feature in printed format are also confused to one another. For example, Fig. 4.11, Fig. 4.12, Fig. 4.13 and 4.14 are the some of the examples to demonstrate.

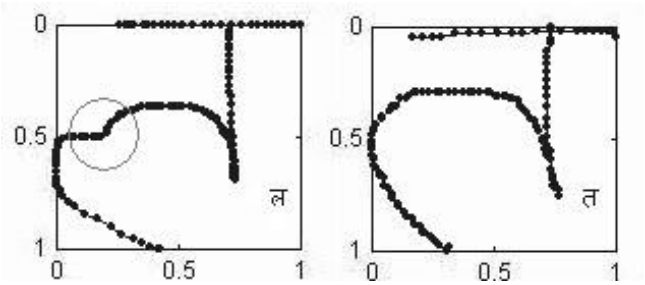

Figure 4.13 Misclassified example: ल→ त, which are looking different in printed format

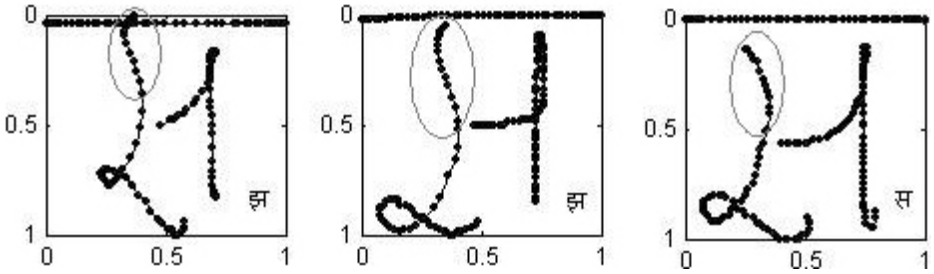

Figure 4.14: Confusion of झ with स even though a lot of dissimilar feature in between them in printed format (झ→स)

Writing a stroke at the end to complete the previous strokes is often called re-writing stroke. These strokes are helpful in giving perfect/complete graphical (off-line) representation but the information carried by only a re-writing stroke is nothing. This is still a problem in this classifier too. Complete information of the specific stroke will be exist if we combine the previous strokes with re-writing stroke. However, it is difficult for the characters, which have large number of strokes i.e., how to know which one is the re-writing stroke and which stroke is to be connected to the re-writing stroke. We only know the re-writing strokes after having 2D plot. Since re-writing strokes are not counted as the complete strokes, we did not have templates. Fig. 3.12 shows two samples of characters having re-writing strokes, which are misclassified. One of the ideas used in pre-processing: noisy sequence elimination, sometimes deletes the re-writing strokes, from which it has a chance of recognizing in testing. Due to this effect, number of misclassified characters with re-writing strokes is reduced.

With the help of statistical information about the strokes, some of the characters which are misclassified before due to their diminished and or very long ascender and descender feature are now classified.

Mis-writing character is another problem for the classifier, which does not give any shape of the character having unnecessary number of strokes.

Having both the accuracy of the classifier (Table 4.1) and the number of misclassified characters (Table 4.2) along with their types in hand, it is concluded that the reliability of the classifier is more than 95%.

In comparison to the classifier, where only the structural properties of the stroke are consid-

ered, it has higher intelligence with better reliability. However, the classifier is also limited to tremor handwriting, children handwriting and writing with re-writing strokes.

Chapter 5

Issues in Classifier's Performance

5.1 Difficulties, Limitations and Scope

Collecting enough data in Thailand is the most difficult part. It took almost three months to collect from the help of Nepalese natives.

Major challenging problems associated with Nepali on-line handwriting are focussed and the proposed methodology involved in the classifier is taken into account based on the performance with peoples' natural writing styles. In chapter 3, section 3.2, a comprehensive difficulties are explained. The following paragraphs provide the way how the existing problems are fixed. Many of the techniques used in developing a prototype classifier are focussing below along with their problem fixing strategies.

- *Size normalization, cusp elimination and overlapping coordinates deletion*:

  As the size (big or small) of the character is variable from time to time writing, the technique like size normalization is used to transform the character of any size into a standard window (designed in eqn. 4.1) with preserving the strokes' information. This is useful in determining the statistical relation in between/among the strokes. Use of spatial relation in designing a classifier is explained in chapter 4. Cusp elimination improves the classification rate (Fig. 3.11) and co-occurrence coordinates deletion improves the speed of the classifier.

- *Directional property*

  The feature used in the classifier to classify the character contains the directional property of writing. Tangent angle at every position along the trajectory of pen movement provides the way how the symbol/stroke is written and the order as well. This feature produces the correct skeleton of the character's shape (Fig. 3.2). Feature vector sequence stores all information about the symbol. As it preserves the complete skeleton along with the order, there is no problem with the tilted handwriting. No skew transformation is necessary.

- *DTW*

  Alignment of two non-linear sequences is the major problem in few decades before. To some extent, with the introduction of DTW algorithm solved the problem by measuring the distance between them at the cost of time complexity. However, the degree of dimension in feature plays a crucial role in recognition speed. The higher the degree of dimension, the larger is the recognition speed.

- *A Template-Based Approach*

  The approach itself defines the way how it works. A classifier with large number of templates with variability in writing styles leads to receive the higher classification rate. It is claimed that some of the misclassified characters would be classified correctly in case the number of templates are increased. Therefore, it can be concluded that the recognition rate and speed of the system depends on number and size of the templates and test strokes, which are variably determined from a series of experiment.

- *Spatial Information*

  One of the biggest difficulties in natural Nepali handwritten character recognition is the similarity between/among the classes of character. In addition, peoples' writing styles aiding extra contribution to become one character similar to another. (छ, ध), (य, प), (ढ, द), (भ, म), (ध, घ), (थ, य) are some of the confusions pairs, which are often classified due to the consideration of spatial information about the strokes. More specifically, spatial information about the strokes within a character has a significant effect on confusion pairs: (भ, म), (थ, य) and (ध, घ). in correct classification. There are two distinguishing features, one is the location of the 'shirorekha' and next is the small curve in the text part. One of two distinguishing features is enough to classify correctly in these pairs. In many of the cases, small curve in the text is not appeared. Therefore, the location of the 'shirorekha' is the main classifying feature. It is assumed that users certainly left one of the two distinguishing features. The spatial information is carried out based on this assumption. However, it has a record that a very few characters do not show such distinguishing features.

During the test, every character is plotted along with the result (code) and hence it is easy to investigate how the test character is misclassified/rejected. A character is tested stroke by stroke basis. Therefore, it has a chance of misclassification/rejection if one of the strokes does not preserve the right information of the specific symbol. This is how the prototype classifier is related to stroke's features (information). In one hand, the developed prototype classifier recovers some of the challenging problems appearing in the natural handwriting (Nepali), however on the other hand, it has remarkable limitations. Some of the limitations are,

- *Tremor Handwriting*

  The system may not apply for handwriting of children and for tremor handwriting.

- *Fast Handwriting*

  The system is limited to very fast handwriting too. In case of fast handwriting, the plot from the features (the collection of coordinates along the pen trajectory) will not be exactly fit with the off-line graphical representation of the stroke (i.e. the shape can be changed), while it can be possible in case of slow handwriting.

- *Feature similarity*

  Most of the times, characters are misclassified due to the similarity in structure from one class of character to another. This is due to the nature of the script and writing styles too.

- *Two stroke Numeral*

  Two or more than two strokes numerals are often misclassified. The system is trained mostly with (98%) uni-stroke numerals. However, two or more than two strokes numerals can be classified correctly by using the system where such numerals are trained.

Globally, it is believed that there are some of the issues in handwriting technology from which it is always lagging behind. Some of them are,

- For transcription, the keyboard is faster than handwriting for most of the languages like Nepali, English, Urdu and so on. Therefore, handwriting technology is not appropriate here.
- Existing on-line handwriting equipments are not as comfortable and natural as pen and paper, what it should be.

The methodology utilized in building a prototype classifier is based on a template-based approach. Dynamic Programming is used to classify the characters into their classes based on their strokes' information (feature). The scope of the thesis is limited to classify Nepali discrete alphanumeric characters. This methodology can be useful not only in adding other symbols in Nepali but also in extending to other scripts such as, Roman, Chinese, Japanese and other Indian Scripts etc.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

A novel approach on a template-based on-line writer independent Nepali handwritten character recognition by using both structural and spatial information is conducted through a series of experiments.

Firstly, only the structural approach is utilized along with the use of Dynamic Programming (DP) for classification of alphanumeric character. The proposed template-based on-line Nepalese natural handwritten alphanumeric character recognition is demonstrated through a series of experiments manipulating two factors in 2×2 design. The first evaluation factor is the inclusion of a pre-processing step incorporating specific knowledge about the Nepalese alphanumeric characters. The second factor is the set of features used for stroke identification. It means, a separate classifier is built for original and pre-processed writing samples in both features: $f_t = (\theta_t)-$one dimensional and $f_t = (x_t, y_t, \theta_t)-$two dimensional. The highest recognition rates are 88.37% and 89.79% for pre-processing samples from the systems using feature events: $f_t = (\theta_t)$ and $f_t = (x_t, y_t, \theta_t)$ along the stroke's sequence respectively. The question of which feature is the best feature is answered by the help of five-fold cross validation. The inclusion of pen-tip's positions with the slopes of consecutive coordinates as a feature along the trajectory of the user's drawing provides much information in stroke classification than from only the sequence of slopes. Therefore, the system combining stroke pre-processing with the position and direction at every point in each stroke as a feature performed better than other systems having different combinations. Hereby, a conclusion has been made that significant places are required for pre-processing and feature selection blocks in character classification engine.

Secondly, a novel idea is explored based on the number of strokes and their spatial relation with each other within a character. Interestingly, the unique feature of Nepali character: the 'shirorekha' is used as the reference feature sequence in order to determine the spatial relation, which helps to achieve the accuracy more than 95%. One of the biggest problems is the similarity in structures between the classes of characters (similarity pairs: (भ, म), (थ, य) and (ध, घ) etc.) is reduced significantly by the use of spatial information about the strokes within a character.

Beside the recognition rate of the classifier, the recognition speed is to be taken under the classifier's performance. As the classifiers use matching procedure for identification of test strokes, the speed is variably determined from one stroke to another. The number and size of strokes in the test letter and templates affect the recognition speed. In addition, de-

gree of dimension in feature vector sequence plays a crucial role. Alignment is faster in a one-dimensional feature than in a two dimensional. Hence, different recognition speed is achieved. However, 12 seconds per character is the fastest average speed of the classifier from the entire experiments.

The superiority of the present work over several related works on Devanagari script is the recognition of writer independent natural handwritten alphanumeric characters, which are stroke number and stroke order free. The methodology used in this classification engine is general and hence flexible in adding extra symbols in Nepali and can be extended to other scripts as well.

6.2 Future Directions

Most of the problems faced during the system design are related to the cursive nature of the scripts and natural writing styles too. It not only brings the problems but also gives a birth of solution techniques. We proposed few extra contributions in this script as future plans by looking into the performance of recognition system based not only on the recognition accuracy but also on the computational complexity and speed. Pre-processing is applied on those characters, which do not have high recognition confidence such that it will enhance the speed. Improvement will be achieved by extracting structural features of all loops, curves, hooks/cusps and intersections etc.

In the first step, building a syllable level recognition in Nepali is the aim to reach. The existing methodologies will work better, once the segmentation of a complete syllable into a character and modifier(s). A concrete technique is needed for segmenting the syllable. As we believe that this methodology is general and can be extended to many of the scripts and symbols, a multilingual recognition system is the plan to build in the second step. How good it will be for those, who are non-natives to English, computer novices and feel inconvenient in using keyboard and keypad (old people), if a system (multi-lingual) having the intelligence in recognizing the natural handwriting for all possible scripts around the world exists.

Bibliography

[1] Andrew Moore, 2001. K-Means and Hierarchical Clustering- Tutorial Slides. *http:// www-2.cs.cmu.edu/ awn/tutorials/kmeans.html*, School of Computer Science, Carnegie Mellon University.

[2] Anoop M. Namboodiri, Student Member, IEEE, Anil K. Jain, Fellow, IEEE, 2004. On-line Handwritten Script Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, No. 1, pp. 124–130.

[3] Ashutosh Malaviya, Cristoph Leja, Liliane Peters, 1996. Multi-script Handwriting Recognition with FOHDEL. *New Frontiers in Fuzzy Logic and Soft Computing Biennial Conference of the North American Fuzzy Information Society-NAFIPS, IEEE*, pp. 147–151.

[4] B. B. Chaudhuri, U. Pal, 1997. An OCR System to Read Two Indian Language Scripts: Bangla and Devanagari (Hindi). *4th International Conference on Document Analysis and Recognition*, Vol. 2, pp. 1011–1015.

[5] B. B. Chaudhuri, U. Pal, 2003. Skew Angle Detection of Digitized Indian Script Documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, pp. 182–186.

[6] Brijesh Verma, Jenny Lu, Moumita Ghosh, Ranadhir Ghosh, 2004. A Feature Extraction Technique for Online Handwriting Recognition. *IEEE International Joint Conference on Neural Networks*, pp. 1337–1341.

[7] C.R. Dyer, 5/3/2007. CS540 Lecture Notes: Neural Networks. http://www.cs.wisc.edu/∼dyer/cs540/notes/nn.html, University of Wisconsin-Madison.

[8] C.Y. Suen, Jinho Kim, Kyekyung Kim, Qizhi Xu, Louisa Lam, 2000. Handwriting Recognition-The last Frontiers. *IEEE 15th International Conference on Pattern Recognition*, Vol. 4, pp. 1–10.

[9] Francios Bouteruche, Eric Anquetil, Nocolas Ragot, 2005. Hanwritten Gesture Recognition Driven by the Spatial Context of Strokes. *8th International Conference on Document Analysis and Recognition(ICDAR)*.

[10] Hangai S., Yamanaka S., Hammamoto T., 2000. Writer Verification Using Altitude and Direction of Pen Movement. *15th International Conference on Pattern Recognition*. Vol. 3, pp. 479–482.

[11] C. C. Tappert, 1982. Cursive Script Recognition by Elastic Matching. *International Business Machine Journal Research and Development: Image processing and Pattern Recognition*, Vol. 26, No. 6, pp. 765–771.

[12] C. C. Tappert, C. Y. Suen, T. Wakahara, 1990. The State of Art in Online Handwriting Recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 8, pp. 787–808.

[13] Chun Lei He, Ping Zhang, Jian Xiong Dong, Ching Y. Suen, Tien D. Bui, 2005. The Role of Size Normalization on the Recognition Rate of Hanwritten Numerals. *The 1st IAPR TC3 NNLPAR workshop*, pp. 8–12.

[14] Claus Bahlmann and Hans Burkhardt,Member, IEEE 2004. The Writer Inependent Online Handwriting Recognition System frog on hand and Cluster Generative Statistical Dynamic Time Warping. *IEEE transactions and Pattern Analysis and Machine Intelligence*, Vol. 26 No. 3, pp. 299–310.

[15] Daiki Okumura, Seiichi Uchida, Hioraki Sakoe, 2005. An HMM Implementation for On-line Handwriting Recognition based on Pen-Coordinate Feature and Pen-Direction Feature. *Bioinformatics*, Vol. 18 No. 5, pp. 735–46.

[16] De Smet F, Mathys J, Marchal K, Thijs G, De Moor B, Moreau Y., 2002. Adaptive Quality-basedClustering of Gene Expression Profiles. *8th IEEE International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 1, pp. 26–30.

[17] Eammon J. Keogh, M. J. Pazzani, 1999. Scaling Up Dynamic Time Warping to Massive Datasets. *3rd European Conference on Principles and Practice of Knowledge Discovery in Databases*, Vol. 1704, pp. 1–11.

[18] E. J. Bellegarda, J. R. Bellegarda, D. Nahamoo, L.S. Nathan, 1994. A Fast Statistical Mixture Algorithm for On-line Handwriting Recognition. *IEEE Transaction PAMI*, Vol. 16, No. 12, pp. 1227–1233.

[19] Esra Vural, Hakan Erdogan, Kemal Oflazer, Berin Yanikoglu, 2005. An Online Handwriting Recognition System for Turkish. *Document Recognition and Retrieval XII: In proceedings of SPIE Electronic Imaging Symposium*, Vol. 5676, pp. 56–65.

[20] Glenn Fung, 2001. A comprehensive Overview of Basic Clustering Algorithms.

[21] Henry A. Rowley, Manish Goyal, John Bennett, 2002. The Effect of Large Training Set Sizes on Online Japanese Kanji and English Cursive Recognizers. *8th IEEE International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 36–40.

[22] Homayoon S.M.Beigi, Krishna N., Gregory J. Clary, Jayashree S., 1994. Size Nomralization in On-line Unconstrained Handwriting Recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 30, No. 9, pp. 1489–1500.

[23] H. J. Kim, K. H. Kim,S. K. Kim, J. K. Lee, 1997. On-line Recognition of Handwritten Chinese Character based on Hidden Markov Models. *Pattern Recognition*, Vol. 1704, pp. 1–11.

[24] Ian Scholey, 2006. Digitizer technology Comparison: Inductive and Resisitive Technology. *Pen Computing Magazine: http://pencomputing.com/features/wacom_digizer_comparison.html (11/12/2006)*.

[25] I. J. Sethi, B. Chatterjee, 1977. Machine Recognition of Constrained Handwritten printed Devanagari. *In Pattern Recognition* , Vol. 9, pp. 69–75.

[26] Jay J. Lee, Jahwan Kim, Jin H. Kim, 2000. Data Driven Design of HMM Topology for On-line Handwriting Recognition. *7th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 239–248.

[27] Jiang Gao, Xiaoqing Ding, Jing Zheng, 2000. Image Pattern Recognition Based on Examples– A combined Statistical and Structural-Syntactic Approach. *In Proceedings of Joint IAPR International Workshops, SSPR 2000 and SPR 2000.*, Vol. 1876, pp. 57–66.

[28] Kenichi Toyozumi, Naoya Yamada, Takayuki Kitasaka, Kensaku Mori, Yasuhito Suenaga, Kenji Mase Tomoichi Takahashi, 2004. A Study of Symbol Segementation Method for handwritten Mathematical Formula Recongition using Mathematical Structure Information. *IEEE 17th International Conference on pattern Recogntion (ICPR)*, pp. 630–633.

[29] Kanad Keeni, Hiroshi Simodaira, Tetsuro Nishino, Yasuo Tan, 1996. Recogntion of Devanagari Characters Using Neural Networks. *IEICE Transactions on Information and Systems*, pp. 523–538.

[30] Krishna S. Nathan, Homayoon S. M. Beigi, Jayashree Subrahmonia, Gregory J. Clary, Hiroshi Maruyama, 1995. Real-time On-line Unconstrained Handwriting Recognition Using Statistical Methods. *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vol. 4, pp. 2619–2622.

[31] L. Breiman, 1996. Heuristics of Instability and Stabilization in Model Selection. *Annals of Statistics*, Vol. 24, pp. 2350–2383.

[32] Lippmann R., 1989. Pattern Classification using Neural Networks. *IEEE Communications Magazine*, pp. 48.

[33] M. Blumenstein, B. Verma, H. Basli, 2003. A Novel Feature Extraction Technique for the Recognition of Segmented Handwritten Characters. *7th IEEE International Conference on Document Analysis and Recognition*, pp. 137.

[34] M. Nakai, N. Akira, H. Shimordaria, S. Sagayama, 2001. Sub-stroke Approach to HMM-based on Kanji Handwriting Recognition. *Proceeding on ICDAR*, pp. 491–495.

[35] Mohammad Asad Ronee, Seiichi Uchida, Hiroaki Sakoe, 2001. Handwritten Character Recognition Using Linear Two-Dimensional Warping. *6th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 39–43.

[36] Nicolas Alvertos, Ivan D'cunha, 1989. Optical Machine Recognition of Greek Characters of Any Size. *IEEE Proceedings: Energy and Information Technologies*, Vol. 2, pp. 623–627.

[37] Niranjan Joshi, G. Sita, A. G. Ramakrishnan, Deepu V., Sriganesh Madhavanath, 2005. Machine Recognition of On-line Handwriteen Devangari Characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 12, No. 8, pp. 787–807.

[38] Niranjan Joshi, G. Sita, A.G. Ramakrishnan, Sriganesh Madhavanath, 2004. Comparison of Elastic Matching Algorithms for Online Tamil Handwritten Character Recognition. *9th IEEE International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 444–449.

[39] Ogla Veksler, 2/2/2007. Pattern Recognition, Lecture 8: Fisher Linear Discriminant. http://www.csd.uwo.ca/∼olga/Courses//CS434a_541a//Lecture8.pdf

[40] Ogla Veksler, 3/2/2007. Pattern Recognition, Lecture 11: Support Vector Machine. http://www.csd.uwo.ca/∼olga/Courses//CS434a_541a//Lecture11.pdf

[41] Panu Somervuo, Teuvo Kohonen, 1999. Self Organizing Maps and Learning Vector Quantization for Feature Sequences. *Neural Network Processing Letters*, Vol. 10, No. 2, pp. 151–159.

[42] P. M. Lallican, C. Viard-gaudin, S.Knerr, 2000. From Off-line to On-line Handwriting Recognition. *7th International Workshop on Frontiers in Handwriting recognition, International Unipen Foundation*, pp. 303–312.

[43] Qiang Gan, Ching Y. Suen, 1996. Neural Networks for Handwritten Character Recognition. *Fuzzy Logic and Neural Network Handbook: McGraw-Hill Series on Computer Engineering, C.H.Chen*, Chapter 16.

[44] R. M. K. Sinha, H. N. Mahabala, 1979. Machine Recognition of Devanagari Script. *IEEE Transactions Systems, Man and Cybernetics*, Vol. 9, No. 8, pp. 435–441.

[45] R. M. K. Sinha, Veena Bansal, 1995. On Devanagari Document Processing. *IEEE International Conference on Systems, Man and Cybernetics, 'Intelligent Systems for the 21st Century'* , Vol. 2, pp. 1621–1626.

[46] Rudolf Freund, Markus Neubauer, Martin Summerer, Stefan Gruber, Jurgen Schaffer, Ronald Swoboda, 2000. A Hybrid System for the Recognition of Hand-Writtern Characters. *Joint IAPR International Workshops on Advances in Pattern Recognition: Lecture Notes in Computer Science, Springer Publication*, Vol. 1876, pp. 67–76.

[47] Roshan Thapaliya, Hirokazu Koizumu, Kashiko Kodate, Takeshi Kamiya, 1998. Parallel Optical Recognition of Devanagari Script with Feature Extraction. *IEEE Conference on Lasers and Electro-optics Europe- Technical Digest.*, pp. 412–413.

[48] S. C. Johnson, 1967. Hierarchical Clustering Schemes. *Psychometrika*, pp. 241–254.

[49] S. palit, B. B. Chaudhari, 1995. A Featur-based Scheme for the Recognition of Printed Devanagari Script. *In Pattern Recognition, Image Processing and Computer Vision.*, pp. 163–168.

[50] Sandip S., 2004. The Digital Pen as a Human Computer Interface. *IEEE International Symposium on Consumer Electrnoics*, pp. 629–634.

[51] Sankoff D., Kruskal J. B., 1983. Time Warps, String Edits and Macromolecules: The theory and Practice of Sequence Comparison. *Addison-Wisely Publishing.*, pp. 125–160.

[52] Sanparith Marukatat, Rudy Sicard, Thierry Artieres, Patrick Gallinari, 2003. A Flexible Recognition Engine for Complex On-line Handwritten Recognition. *7th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1048–1052.

[53] Sanparith Marukatat, Thierry Artieres, Patrick Gallinari, 2004. A Generic Approach for On-line Handwriting Recognition. *9th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 401–406 .

[54] Sanparith Marukatat, Thierry Artieres, 2004. Handling Spatial Information in On-line Handwriting Recognition. *9th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pp. 14–19.

[55] Santosh K. C., Cholwich Nattee, 2006. Structural Approach on Nepalese Natural Handwriting Recognition. *2nd IEEE International Conference on Cybernetics Intelligent Systems*, pp. 711–716.

[56] Santosh K. C., Cholwich Nattee, 2006. Stroke Number and Order Free Hanwriting Recogntion for Nepali. *Trends in Aritificical Intelligence: 9th Pacific Rim International Conference on Aritficial Intelligence, Lecture Notes in Computer Science, Springer Publication*, Vol. 4099, pp. 990–994.

[57] Santosh K. C., Cholwich Nattee, 2006. Effect of Pre-processing and Feature Selection in Recognition for Nepali. *The first International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, pp. 139–146

[58] Scott D. Connell, R. M. K. Sinha, Anil K. Jain, 2000. Recognition of Unconstrained Online Devanagari Characters. *15th IEEE International Conference on Pattern Recognition*, pp. 368–371.

[59] Songl Albayrak, Fatih Amasyal, 2003. Fuzzy C-Means Clustering on Medical Diagnostic Systems. *International XII, Turkish Symposium on Artifical Intelligence and Neural Networks (TAINN)*.

[60] Stephen M. Watt, Xiaofang Xie, 2005. Recognition of Large Sets of Handwritten Mathematical Symbols. *IEEE International Conference on Document Analysis and Recognition (ICDAR)*, pp. 39–43.

[61] Sumaila Malik, Shoab A. Khan, 2005. Urdu Online Handwriting Recognition. *IEEE International Conference on Emerging Technolgies*, pp. 27–31.

[62] Sung-Hyuk Cha, Yong-Chul Shin, Sargur N. Srihari, 1999. Approximate Sequence String Matching Algorithm for Character Recognition and Analysis. *Fifth International Conference on Document Analysis and Recognition (ICDAR)*, pp. 53–56.

[63] Tom M. Micthell. Machine Learning. The McGraw-Hill Companies, Inc.

[64] T. Artieres, P. Gallinari, 2002. Stroke Level HMMs for On-line Handwriting Recognition. *8th International Workshop in Handwriting Recognition*, pp. 227–232.

[65] T. Hasegawa, H. Yasuda, T. Motsumoto, 2000. Fast Discrete HMM Algorithm for On-line Handwriting Recognition. *Proceeding on ICPR*, Vol. 4, No. 4, pp. 535–538.

[66] T. Starner, J. Makhoul, R. Schwartz, G. Chou, 1994. On-line Cursive Handwriting Recognition Using Speech Recognition Methods. *Proceeding on ICASSP*, Vol. 5, No. 6, pp. 125–128.

[67] Utpal Garain, B. B. Chaudhari, 2004. Input of Handwritten Mathematical Expressions into Machine Coded Indian Language Documents. *Indo-European Conference on Multilingual Technologies (IECMT)*, pp. 3–12.

[68] Utpal Garain, B. B. Chaudhari, 2004. Recognition of On-line Handwritten Mathematical Expressions. *IEEE Transactions on Systems, Man and Cybernetics* , Vol. 34, No. 6, pp. 2366–2376.

[69] Veena Bansal, R. M. K. Sinha, 2002. Segmentation of Touching and Fused Devanagari Characters. *Pattern Recognition*, Vol. 35, No. 4, pp. 875–893.

[70] W. Guerfali, R. Plamondon, 1993. Normalizing and Restoring Online Handwriting Pattern Recognition. *Pattern Recognition*, Vol. 2, No. 3, pp. 419–431.

[71] A tutorial on Clustering Algorithms, 30/11/2006. *http://www.elet.polimi.it/upload/matteucc/Clustering/tutorial_html/index.html*.

[72] Fujitsu laboratories, 2005. Fujitsu - The Possibilities are Infinite, Ultrasound Electronic Pen.

[73] Wacom Technology, 12/12/2006. Wacom Components: Technology: EMR®Technology. *http://www. wacom-components.com/english/technology/emr.html*

[74] Statistical Classification, 2/1/2007. *http://en.wikipedia.org/wiki/Statistical_classification*

[75] Lecture 13: Validation, 5/3/2007. http://courses.cs.tamu.edu/rgutier/ceg499_s02/l13.pdf, University of Wisconsin-Madison.

Appendix A

Nepali

A.1 Language Overview

The national language of Nepal is known as 'Nepali'. According to the most recent official census, conducted by His Majesty's Government of Nepal (HMG) in 2001, Nepali is the mother tongue for 11 million people. Outside of Nepal, Nepali is also spoken in northeast India and in much of Bhutan. On account of its widespread use in the states of West Bengal (particularly in the district of Darjeeling) and Sikkim, the Indian Constitution recognizes Nepali as a major language of India. In Bhutan, while Dzongkha is the national language, Nepali is widely spoken by people from diverse ethnic backgrounds. South Asian languages such as Hindi, Bengali, Marathi and Gujarati used Nepali. Modern Indo-Aryan languages are related to Sanskrit, much as modern European languages are related to Latin.

The use of Nepali covers much of the countries (states), which reflects that the need of education system to enhance for the unity in their developments in various matters. The most biggest problem in this world is the language. People used sign language to communicate even though the English is considered as an international language. Most of the population do not know much about the English for instance, south asian. This would be great for those, who are not touch with the English in case the learning Nepali course is designed internationally. The need is solved to some extent by some of the professors in the previous days.

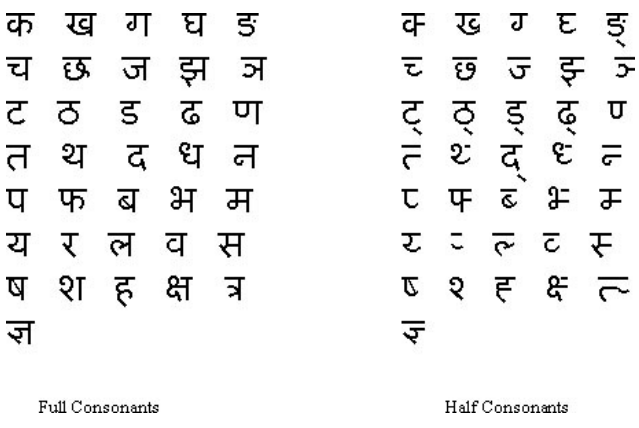

Figure A.1 Both full and half Consonants

क् + ष = क्ष, त् + र = त्र,
ज् + ञ = ज्ञ, श् + र = श्र,
द् + य = द्य, द् + ध = द्ध,
द् + य = द्य, त् + त = त्त,
त् + क = त्क etc.

Figure A.2 A Few samples of Conjunct Consonants

अ आ इ ई उ ऊ ऋ ए ऐ ओ औ अं अँ अः

Primary Vowels Secondary Vowels Other Symbol

Figure A.3 Vowels

० १ २ ३ ४ ५ ६ ७ ८ ९

Figure A.4 Numerals

क का कि की कु कू कृ के कै को कौ कं कँ कः
ख खा खि खी खु खू खृ खे खै खो खौ खं खँ खः
… … … … … … … … … … …
… … … … … … … … … … …
घ घा घि घी घु घू घृ घे घै घो घौ घं घँ घः
म मा मि मी मु मू मृ मे मै मो मौ मं मँ मः

Figure A.5 Sample of Vowels, as a modifier for consonant

ऋषी मुनि छहारी अन्नपूर्ण पोष्ट
सन्तोष प्रशन्न बिनिशा दिपिका
मेरो काली अन्जू जसुधा मन्जु
संध्या ……

Figure A.6 A few samples of words formation

मलाई माया गर्ने मान्छेलाई मैले सधै छोड्नु पर्छ, किन?

नेताहरु कुर्सिको लागि लडिरहेछन् के रे।

अम्ममम…आजभोलि त नेपाली हुँ भन्न पनि लाज लाग्ने भैसक्यो।

हे भगवान्! म कति भाग्यमानि रहेछु है।

Figure A.7 A few samples of complete sentences

A.1.1 Learning Strategies: Review

The Nepali script is available in the existing computer in case the devanagari font is installed.

- Dr. John Peterson maintained the 'South Asian Linguistics' web site in order to know much about south asian languages.
- 'Teach Yourself Nepali' by Dr. Michael Hutt and Professor Abhi Subedi (Hodder and Stoughton, first published in 1999) is available widely.
- Beginner's courses include basic course in Spoken Nepali in order to learn conversational Nepali, and Dr. David Matthews' 'Course in Nepali', which focuses more on literary Nepali and is useful for advanced students. Not only inside the country, but also learning methodologies was designed.

Nepali is offering at many locations all over the world.

- 'School of Oriental and African Studies' (SOAS), in London, Cornell University's summer intensive Nepali language course and the semester courses in Nepali offered throughout the year by the Department of Asian Studies at Cornell.
- The University of Wisconsin-Madison's 'South Asia Summer Language Institute' (SASLI) offers Nepali language courses for beginners.
- Cornell's Nepali language instructors, Banu Oja and Shambhu Oja, have written a course book and dictionary. Banu Oja, Shambhu Oja, Mark Turin and Elisabeth Uphoff's Nepali - English and English - Nepali Glossary has recently been updated.
- Nepali is also taught at the 'India Instituut' in Amsterdam, the Netherlands, by Rene Huysmans and Mark Turin. Turin and Huysmans have written and published a Nepali language course in Dutch, entitled Nepali voor Beginners.

A.1.2 Dictionaries

- Nepali Dictionary by Ralph Lilley Turner (first published in 1931 by the Royal Asiatic Society, but still reprinted in India every couple of years).
- Nepali-English-Nepali Dictionary by Professor Babulall Pradhan (Ratna Pustak Bhandar, Kathmandu, 2001).
- Nepali-English/English-Nepali dictionary and phrase book by Prakash A. Raj.
- Revised Nepali-English, English-Nepali Glossary by Banu Oja, Shambhu Oja, Mark Turin and Elisabeth Uphoff.
- Nepali-English/English-Nepali Dictionary by Cornell.
- Thangmi-Nepali-English Dictionary by Mark Turin.
- 420-word Unicode Nepali dictionary by Bruce Adcock.
- Nepal Homepage's English to Nepali Dictionary.
- Practical Dictionary of Modern Nepali by Ruth Laila Schmidt.

A.2 Characters, Numerals, words and Sentences

Nepali consonants, vowels and numerals are clearly demonstrated in the printed format for the reader to understand. Fig. A.1 shows the consonants in both full and half forms. A few sets of conjunct consonants are provided in Fig. A.2. Primary and secondary vowels are shown in Fig. A.3. Fig. A.4 demonstrates the numerals used in Nepali. Use of vowel as modifier for consonant to build a syllable is provided in Fig. A.5. Finally, Fig. A.6 shows how the word can be built up and Fig. A.7 provides a few samples of writing sentences.

Appendix B

Glossary

B.1 Natural Input

The writing is said to be completely natural as if user writes on a piece of paper. Two or more than two strokes can be connected to one another to form a word, such as in cursive handwriting. In natural writing, there are no any constraints, directions and limitations in writing. One can write with one's style. This is highly preferable for all kinds of people and has many applications like, note-making, letter writing etc. Natural writing is shown in Fig. B.1.

B.2 Uni-stroke Input

A character is created by the use if single ink stroke is called uni-stroke input. The shape of the character is selected in a way to increase the differentiation from the other characters. However, possibility of having similar drawing of two character occurs. The method is highly suitable for small devices on which it is impossible to write naturally or by using many strokes. How difficult for the people who has to write many strokes to make a character, for instance, Chinese, Japanese and Korean etc. An uni-stroke numeral is demonstrated in Fig. B.1.

Box Input Characters: पशल

Uni-stroke Numeral: २

Multi-stroke Character: क

Figure B.1 Natural Writings

B.3 Boxed Input

The input can be made by implementing a natural handwriting style, but each character has to be drawn within the defined box layout. Therefore it is far more easier to segment the character based on the location and number of box. Usually, it is appropriate in form-filling applications. It is conscious handwriting style. A purely discrete characters is known by the name boxed input characters, which is shown in Fig. B.1.

Appendix C

Source Code

C.1 Input Frame Design

```
* MainFrame.java
* Created on August 26, 2005, 10:20 AM

package Test1;

import java.awt.*;
import java.awt.geom.*;
import java.util.*;
import java.io.*;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
* @author Cholwich and Santosh

public class MainFrame extends javax.swing.JFrame
{
/** Creates new form MainFrame */
        public MainFrame()
```

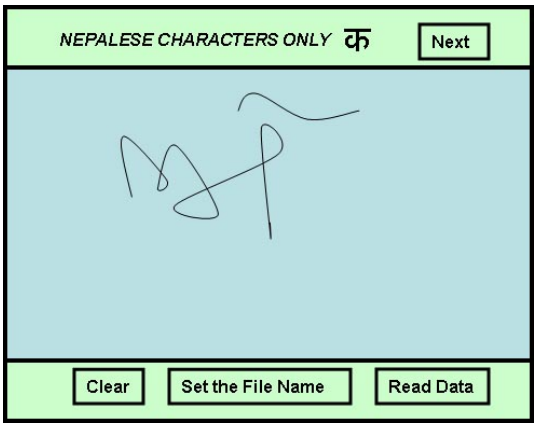

Figure C.1 Input Window

```
        {
        initComponents();
        this.setSize(600,500);
        }
/** This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code.
The content of this method is always regenerated by the Form Editor.*/

// editor-fold defaultstate="collapsed" desc="Generated Code "
private void initComponents()
{
jPanel1 = new javax.swing.JPanel();
clearButton = new javax.swing.JButton();
jButton1 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
mainPanel = new javax.swing.JPanel();
jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
jButton2 = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
addWindowListener(new java.awt.event.WindowAdapter()
{
        public void windowClosing(java.awt.event.WindowEvent evt)
        {
        formWindowClosing(evt);
        }
});
clearButton.setBackground(new java.awt.Color(255, 0, 51));
clearButton.setText("Clear");
clearButton.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        clearButtonActionPerformed(evt);
        }
});
jPanel1.add(clearButton);
jButton1.setBackground(new java.awt.Color(255, 153, 0));
jButton1.setText("Set File Name");
jButton1.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        jButton1ActionPerformed(evt);
        }
});
jPanel1.add(jButton1);
```

```
jButton3.setBackground(new java.awt.Color(153, 153, 0));
jButton3.setText("read data");
jButton3.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        jButton3ActionPerformed(evt);
        }
});
jButton3.addMouseListener(new java.awt.event.MouseAdapter()
{
        public void mouseClicked(java.awt.event.MouseEvent evt)
        {
        jButton3MouseClicked(evt);
        }
});
jPanel1.add(jButton3);
getContentPane().add(jPanel1, java.awt.BorderLayout.SOUTH);
mainPanel.setBackground(new java.awt.Color(204, 204, 255));
mainPanel.setBorder(new javax.swing.border.MatteBorder(new java.awt.Insets(1, 1, 1, 1),
new java.awt.Color(28, 189, 189)));
mainPanel.addMouseMotionListener(new java.awt.event.MouseMotionAdapter()
{
        public void mouseDragged(java.awt.event.MouseEvent evt)
        {
        mainPanelMouseDragged(evt);
        }
});
mainPanel.addMouseListener(new java.awt.event.MouseAdapter()
{
        public void mouseReleased(java.awt.event.MouseEvent evt)
        {
        mainPanelMouseReleased(evt);
        }
});
getContentPane().add(mainPanel, java.awt.BorderLayout.CENTER);
jLabel1.setBackground(new java.awt.Color(0, 153, 153));
jLabel1.setFont(new java.awt.Font("Arabic Transparent", 3, 11));
jLabel1.setForeground(new java.awt.Color(0, 102, 102));
jLabel1.setText("NEPALESE CHARACTERS ONLY");
jLabel1.setMaximumSize(new java.awt.Dimension(155, 20));
jLabel1.setMinimumSize(new java.awt.Dimension(155, 20));
jLabel1.setPreferredSize(new java.awt.Dimension(155, 20));
jPanel2.add(jLabel1);
jLabel2.setFont(new java.awt.Font("Mangal", 1, 36));
jLabel2.setText("ŏ915");
jPanel2.add(jLabel2);
jButton2.setBackground(new java.awt.Color(204, 0, 204));
```

```
jButton2.setText("Next");
jButton2.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
        jButton2ActionPerformed(evt);
        }
});
jPanel2.add(jButton2);
getContentPane().add(jPanel2, java.awt.BorderLayout.NORTH);
pack();
}
//editor_fold

private void jButton3MouseClicked(java.awt.event.MouseEvent evt)
{
// TODO add your handling code here:
JFileChooser fileChooser = new JFileChooser();
fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
int w = fileChooser.showOpenDialog(this);
if (w ==JFileChooser.CANCEL_OPTION)
return;
File fileName = fileChooser.getSelectedFile();
if (fileName ==null —— fileName.getName().equals(""))
JOptionPane.showMessageDialog(this,"Invalid File Name","Invalid File Name",
JOptionPane.ERROR_MESSAGE);
else
        {
          try
          {
          String line;
          BufferedReader input = new BufferedReader(new FileReader(fileName));
          while ((line= input.readLine())!=null);
          }
          catch (IOException ioException)
          {
          JOptionPane.showMessageDialog(this,"Error Opening File","Error",
          JOptionPane.ERROR_MESSAGE);
          }
        }
}

private void formWindowClosing(java.awt.event.WindowEvent evt)
{
// TODO add your handling code here:
if (output!=null)
        {
        output.flush();
```

```
        output.close();
        }
}

private PrintWriter output=null;
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
// TODO add your handling code here:
JFileChooser fileChooser = new JFileChooser();
fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
int w =fileChooser.showSaveDialog(this);
if(w == JFileChooser.CANCEL_OPTION)
return ;
File fileName = fileChooser.getSelectedFile();
if (fileName == null —— fileName.getName().equals(“”))
JOptionPane.showMessageDialog(this,“InvalidFile Name”,“Invalid File Name”,
JOptionPane.ERROR_MESSAGE);
else
        {
         try
         {
         output = new PrintWriter (new FileWriter(fileName));
         }
         catch (IOException ioException)
         {
         JOptionPane.showMessageDialog(this,“Error opening file”,“Error”,
         JOptionPane.ERROR_MESSAGE);
         }
        }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
// TODO add your handling code here:
String error;
error=“”;
if (output!=null)
        {
        clearButtonActionPerformed(null);
        for(int i=0; i¡points.size(); i++)
            {
            Point2D.Double p = (Point2D.Double)points.get(i);
            output.println(p.x+“,”+p.y);
            }
        output.println(“ ”);
        output.flush();
        points.clear();
        }
```

```
else
        {
        System.err.println("Set File Name First");
        JOptionPane.showMessageDialog(null, error,"ERROR ..!! Please, Set filename first",
        JOptionPane.INFORMATION_MESSAGE);
        }
        mainPanel.setBackground(Color.GREEN);
}

private void mainPanelMouseReleased(java.awt.event.MouseEvent evt)
{
// TODO add your handling code here:
first = true;
points.add(new Point2D.Double(-1,-1));
}

private void clearButtonActionPerformed(java.awt.event.ActionEvent evt)
{
// TODO add your handling code here:
Graphics g = mainPanel.getGraphics();
g.clearRect(0,0, mainPanel.getWidth(), mainPanel.getHeight());
mainPanel.repaint();
}
private boolean first = true;
private Point2D.Double curP = new Point2D.Double();
private Point2D.Double newP = new Point2D.Double();
private Vector points = new Vector();

private void mainPanelMouseDragged(java.awt.event.MouseEvent evt)
{
// TODO add your handling code here:
Graphics2D g2 = (Graphics2D)mainPanel.getGraphics();
g2.setStroke(new BasicStroke(1));
if (first)
        {
        curP.setLocation(evt.getX(), evt.getY());
        first = false;
        points.add(new Point2D.Double(evt.getX(), evt.getY()));
        }
else
        {
        newP.setLocation(evt.getX(), evt.getY());
        g2.setColor(Color.RED);
        Line2D.Double l = new Line2D.Double(curP,newP);
        g2.draw(l);
        curP.setLocation(newP);
        points.add(new Point2D.Double(evt.getX(), evt.getY()));
        }
```

```
System.out.println(evt.getX()+“,”+evt.getY());
}

public static void main(String args[])
{
        java.awt.EventQueue.invokeLater(new Runnable()
        {
        public void run()
                {
                new MainFrame().setVisible(true);
                }
        });
}

// Variables declaration _ do not modify
private javax.swing.JButton clearButton;
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JPanel mainPanel;
// End of variables declaration
}
```

C.2 Alignment of Two Non-linear Sequences and Averaging

Alignment of two sequences of different lengths (a string of coordinates)is possible with the use of Dynamic Time Warping (DTW). How it was done in the experiment is demonstrated below with the full source code in MATLAB.
Dist is un-normalized distance between T and R
D is the accumulated distance matrix
T is the vector you are testing against
R is the vector you are testing
k is the normalizing factor
w is the optimal path

function $[D, Dist, w, average] = dtw(T, R)$
$[rows, N] = size(T)$;
$[rows, M] = size(R)$;
for $n = 1 : N$
    for $m = 1 : M$
    $d(n,m)$ =sqrt($(T(n) - R(m))^2$);
    end

```
end
D = zeros(size(d));
D(1,1) = d(1,1);
for n = 2 : N
D(n,1) = d(n,1) + D(n-1,1);
end
for m = 2 : M
D(1,m) = d(1,m) + D(1,m-1);
end
for n = 2 : N
    for m = 2 : M
    D(n,m) = d(n,m) + min([D(n-1,m), D(n-1,m-1), D(n,m-1)]);
    end
end
// Distance between T and R,
Dist = D(N,M);

n = N;
m = M;
k = 1;
w = [ ];
w(1,:) = [N,M];
while ((n+m) = 2)
if (n-1) == 0
m = m-1;
    elseif (m-1) == 0
    n = n-1;
    else
    [values, number] = min([D(n-1,m), D(n,m-1), D(n-1,m-1)]);
    cord = [values];
    switch number
    case 1
    n = n-1;
    case 2
    m = m-1;
    case 3
    n = n-1;
    m = m-1;
    end
end
k = k+1;
w = cat(1, w, [n,m]);
end
w1 = rot90(w);
w2 = rot90(w1);
// Averaging two sequences T and R
for i = 1 : length(w)
average(i) = (R(w2(i,1)) + T(w2(i,2)))/2
```

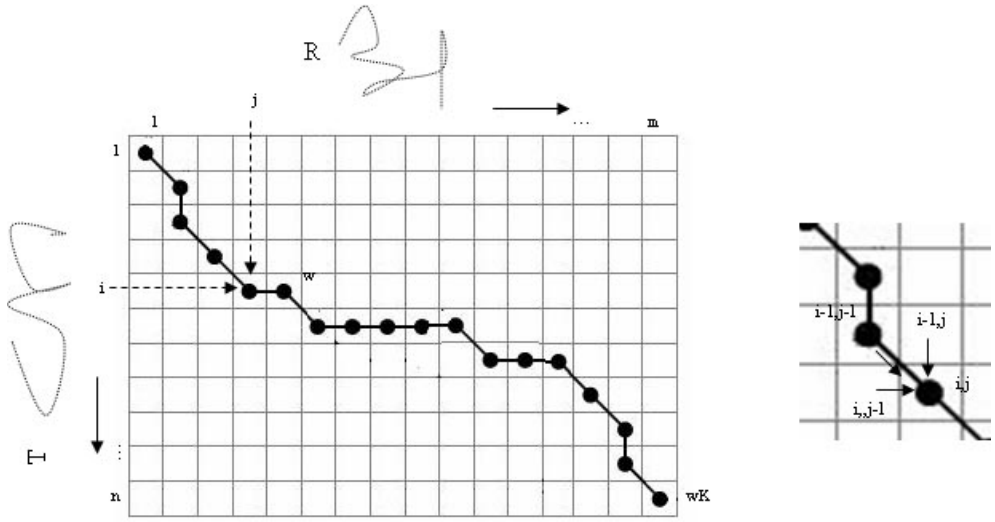

Figure C.2 Discrete Warping Path

	C1	C2	C3	C4	C5	C6
	A1	A2	A3	A4	A5	A6
A1	0	662	877	255	412	996
A2	662	0	295	468	268	400
A3	877	295	0	754	564	138
A4	255	468	754	0	219	869
A5	412	268	564	219	0	669
A6	996	400	138	869	669	0

Index: C1 = cluster 1, C2 = cluster 2,..., C6 = cluster 6

Figure C.3 Distance Matrix

end
average;

C.3 Agglomerative Clustering

The following source code provides an idea about the single-linkage, complete- linkage and centroid agglomerative hierarchical clustering.
Cluster = agg(Distance, Method, k)
Distance is square dissimilarity matrix, with Inf on leading diagonal
Method is one of 'single', 'complete' or 'centroid', and
k is the intended number of clusters
Cluster is a cell array showing which entities belong to which cluster

```
cluster = [1]  [2]  [1x2 double]  [4]  [5]  [6]        // c3 and c6 are going to be merged.

cluster = [1]  [2]  [1x2 double]  [4]  [5]             // c3 and c6 are merged.

cluster = [1]  [2]  [1x2 double]  [1x2 double]  [5]    // c4 and c5 are going to be merged

cluster = [1]  [2]  [1x2 double]  [1x2 double]         // c4 and c5 are merged.
```

Figure C.4 Clustering Results - Method: 'Single' and k =4

function $cluster = agg(Distance, Method, k)$

$Distance = matrix$;
$Method =$ '$single$';
k;

if $nargin < 3$
error('agg requires three arguments');
help(filename)
return
end

$[R, C] = size(Distance)$;

//Put every point is in its own cluster

$cluster = num2cell(1 : R)$;
for $i = 2 : (R - k + 1)$

//Find closest clusters

$[MinRow, IdxDow] = min(Distance)$;
$[temp, MinJ] = min(MinRow)$;
$MinI = IdxDow(MinJ)$;

if $MinI > MinJ$
$t = MinI$;
$MinI = MinJ$;
$MinJ = t$;
end

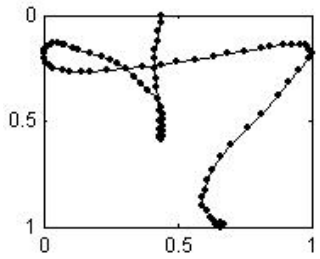

Figure C.5 A sample of a stroke

//Merge cluster j into cluster i, then delete j

$clusterMinI = [clusterMinIclusterMinJ]$
$cluster(MinJ) = [\,]$

//Calculate new Distance matrix

switch $Method$
case $'single'$
$Distance(:,MinI) = min(Distance(:,MinI), Distance(:,MinJ));$
$Distance(MinI,:) = min(Distance(MinI,:), Distance(MinJ,:));$
case $'complete'$
$Distance(:,MinI) = max(Distance(:,MinI), Distance(:,MinJ));$
$Distance(MinI,:) = max(Distance(MinI,:), Distance(MinJ,:));$
case $'centroid'$
$Distance(:,MinI) = (Distance(:,MinI) + Distance(:,MinJ))/2;$
$Distance(MinI,:) = (Distance(MinI,:) + Distance(MinJ,:))/2;$
otherwise
error('Unsupported Method in agg!');
end

$Distance(MinJ,:) = [\,];$
$Distance(:,MinJ) = [\,];$
$Distance(MinI,MinI) = inf;$
end
$cluster;$

C.4 Feature Vector Sequence

A sample of feature vector sequence is demonstrated in the following. It contains both pen tip positions and pen directions at every position. Feature is designated in the format of (x, y, θ).
Feature vector sequence of the stroke shown in Fig. C.5 is,

f = [(0.4406, 0.0053, 1.5708), (0.4406, 0.0370, 1.5708), (0.4266, 0.0794, -1.2517), (0.4266, 0.1217, 1.5708), (0.4126, 0.1640, 1.2517), (0.4126, 0.2011, 1.5708), (0.4126, 0.2487, 1.5708), (0.4196, 0.2963, 1.4250), (0.4196, 0.3386, 1.5708), (0.4266, 0.3968, 1.4512), (0.4336, 0.4550, 1.4512), (0.4336, 0.5026, 1.5708), (0.4336, 0.5397, 1.5708), (0.4336, 0.5714, 1.5708), (0.4336, 0.5820, 1.5708), (0.4406, 0.5873, 0.6477), (0.4476, 0.5714, -1.1558), (0.4476, 0.5503, -1.5708), (0.4476, 0.5238, -1.5708), (0.4476, 0.4921, -1.5708), (0.4476, 0.4656, -1.5708), (0.4406, 0.4339, 1.3540), (0.4266, 0.3968, 1.2097), (0.3916, 0.3598, 0.8142), (0.3636, 0.3228, 0.9239), (0.3357, 0.2910, 0.8485), (0.3077, 0.2487, 0.9868), (0.2727, 0.2275, 0.5443), (0.2517, 0.2116, 0.6477), (0.2168, 0.1958, 0.4261), (0.1748, 0.1799, 0.3617), (0.1329, 0.1640, 0.3617), (0.1119, 0.1587, 0.2471), (0.0769, 0.1429, 0.4261), (0.0699, 0.1376, 0.6477), (0.0559, 0.1323, 0.3617), (0.0350, 0.1323, 0), (0.0280, 0.1429, -0.9868), (0.0140, 0.1481, -0.3617), (0, 0.1640, -0.8485), (0, 0.1799, 1.5708), (0, 0.2011, 1.5708), (0.0070, 0.2275, 1.3124), (0.0210, 0.2381, 0.6477), (0.0350, 0.2487, 0.6477), (0.0699, 0.2593, 0.2939), (0.0979, 0.2646, 0.1869), (0.1399, 0.2698, 0.1254), (0.1748, 0.2698, 0), (0.2378, 0.2593, -0.1666), (0.3077, 0.2540, -0.0755), (0.3706, 0.2434, -0.1666), (0.4406, 0.2328, -0.1502), (0.4965, 0.2222, -0.1869), (0.5664, 0.2116, -0.1502), (0.6364, 0.1958, -0.2232), (0.6853, 0.1799, -0.3136), (0.7483, 0.1693, -0.1666), (0.8112, 0.1534, -0.2471), (0.8462, 0.1481, -0.1502), (0.8951, 0.1429, -0.1077), (0.9301, 0.1376, -0.1502), (0.9580, 0.1376, 0), (0.9790, 0.1376, 0), (0.9860, 0.1481, 0.9868), (0.9930, 0.1640, 1.1558), (1.0000, 0.1799, 1.1558), (0.9930, 0.2063, -1.3124), (0.9510, 0.2593, -0.9003), (0.9161, 0.3122, -0.9868), (0.8741, 0.3810, -1.0231), (0.8112, 0.4656, -0.9315), (0.7622, 0.5344, -0.9523), (0.6993, 0.6085, -0.8665), (0.6643, 0.6720, -1.0674), (0.6294, 0.7302, -1.0298), (0.6084, 0.7831, -1.1933), (0.6014, 0.8254, -1.4071), (0.5944, 0.8624, -1.3842), (0.5944, 0.8995, 1.5708), (0.6084, 0.9259, 1.0845), (0.6224, 0.9577, 1.1558), (0.6364, 0.9735, 0.8485), (0.6434, 0.9894, 1.1558), (0.6573, 0.9947, 0.3617), (0.6643, 1.0000, 0.6477), (0.6713, 0.9894, -0.9868), (0.6643, 0.9841, 0.6477), (0.6643, 0.9788, -1.5708)].

Appendix D

List of Publications

D.1 International Journal

1. K.C., Santosh and Nattee, Cholwich, 2006. Stroke Number and Order Free Handwriting Recognition for Nepali, *9th Pacific Rim International Conference in Artificial Intelligence (PRICAI)*, Lecture Notes in Computer Science (LNCS), Subseries: Lecture Notes in Artificial Intelligence (LNAI), Guilin, China, Vol. 4099, pp. 990-994.

D.2 National Journal

1. K.C., Santosh and Nattee, Cholwich, 2007. Template-based Nepali Natural Handwritten Alphanumeric Character Recognition, *Thammasat International Journal of Science and Technology (TIJSAT)*, Vol. 12, No.1, pp. (to be appeared).

D.3 International Conferences

1. K.C., Santosh and Nattee, Cholwich, June 7-9, 2006. Structural Approach on Writer Independent Nepalese Natural Handwriting Recognition. *IEEE International Conference on Cybernetics Intelligent Systems (CIS)*, Bangkok, Thailand, pp.711-716.

2. K.C., Santosh and Nattee, Cholwich, August 1-4, 2006. Effect of Pre-processing and Feature Selection in Recognition for Nepali. *The first International Conference on Knowledge, Information and Creativity Support Systems (KICSS)*, Autthaya, Thailand, pp. 139-146.–[***Best Student Paper Award***]