# Fast, Language Independent, Handwriting Recognition Algorithms for Handheld Devices

A Project Report

Submitted in partial fulfilment of the

requirements for the Degree of

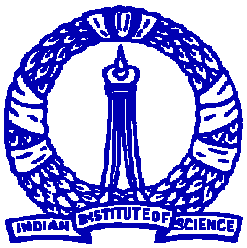
**Master of Engineering**

in

Computer Science and Engineering

by

**Karthik K**




Computer Science and Automation

Indian Institute of Science

BANGALORE – 560 012

June 2006

# Acknowledgements

I would like to thank my advisor Dr. Chiranjib Bhattacharyya for all the things he has taught me in Machine Learning and the other aspects of research. I also wish to thank him for his motivational words, his accessibility, and his guidance in all the fields during my stay in IISc. I am grateful for his continuing support.

I was fortunate to receive valuable help from Dr Subramanya of Pico Peta Ltd. during this project. I would like to thank him for his help and support. I would also like to thank Prof Vinay and Prof Swami Manohar from Pico Peta for their help and the new ideas they provided.

I am grateful to Prof M Narasimha Murthy for getting me interested in Pattern Recognition and teaching me the basics of the subject and helping me start out on this project. I am also indebted to all the faculty of IISc.

I would also like to thank my lab mates Sanjay, Saketha, Sourangshu, Krishnan, Mehul and Rashmin for their suggestions and for providing an excellent environment for working in the lab.

Finally I would like to thank my parents for their support and guidance during the project.

# Abstract

*In this project we propose two new approaches for classifying multivariate time-series with applications to the problem of writer independent online handwritten character recognition. In the first approach, each time-series is approximated by a sum of piecewise polynomials in a suitably defined reproducing kernel Hilbert space (RKHS). Using the associated kernel function a large margin classification formulation is proposed which can discriminate between two such functions belonging to the RKHS. The associated problem turns out to be an instance of convex quadratic programming which can be solved with standard tools.*

*Another approach for classifying multivariate time-series involves modelling the multivariate series as a linear function of a hidden univariate series with an additive gaussian noise. This approach deals with extracting the hidden series and training standard classifiers on the same. A Kalman filter was used to model the evolution of the pen position with respect to a hidden wrist position. The classification was performed on a time series obtained by the wrist position rather than the position of the pen. Experiments performed on the above gave promising results.*

*The above described algorithms can be applied to other Machine Learning tasks such as clustering and regression. The Kernels defined can also be used with other kernelized algorithms.*

# Contents

# Chapter 1

# Introduction

***This Chapter introduces the problem and provides a brief insight into time series analysis and handwriting recognition.***

The mobile phone revolution in India has been largely restriceted to the urban and English speaking population. Even though the penetration of phones is increasing, other services like SMS, Email, Chat, Phone books etc are not accessible to non English literate population as interfaces for these languages are not available on mobile phones. The problem can be solved by providing these services in local languages. The challenge is to develop language neutral algorithms for mobile phones that can provide interface to these services through handwriting. In this project, we aim to develop language neutral algorithms for handwriting recognition that can work on the limited resources available on mobile phones.

This chapter starts with an introduction to time series and temporal sequences. The following sections deal with the problem of handwriting recognition in detail and enumerate the challenges posed by the problem. The chapter goes on to provide an outline for the entire thesis.

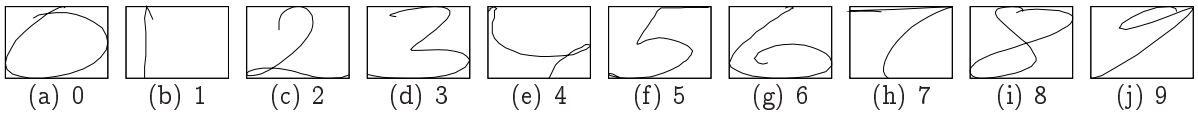



Figure 1.1: English Numerals from the UCI dataset.

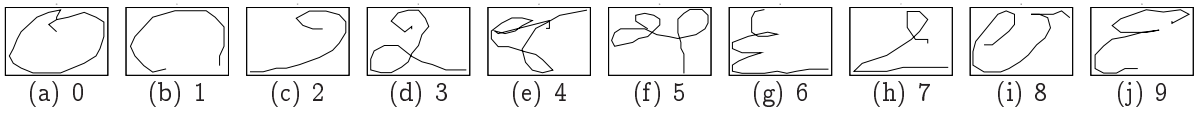



Figure 1.2: Kannada Numerals for the data collected on the simputer

## 1.1 Time Series

Understanding temporal sequences is an important task having many applications in diverse domains like online handwriting recognition, speech recognition etc. The task of temporal sequence understanding often boils down to assigning proper labels to such sequences.

In general a time series is represented as a collection of time ordered vectors $\mathcal{F} = \{\mathcal{F}(t_1), \mathcal{F}(t_2), ..., \mathcal{F}(t_n)\}$ where $\mathcal{F}$ is the time series, $\mathcal{F}(t_i)$ denotes a vector sampled at the time instant $t_i \in \mathbb{R}$, $t_i < t_j$ whenever $i < j$ and $n$ is the number of sampled points in the time series. In this project, we study the problem of classifying such multivariate time-series and develop algorithms suitable for application to online handwriting recognition on a handheld device like a PDA. The characters written on a touch sensitive screen of a PDA are available as time series. An example of English and Kannada numerals written on a PDA are given in figures 1.1 and 1.2.

## 1.2 Online handwriting recognition

Online handwriting recognition refers to handwriting recognition performed on a handheld device immediately after the character is written. On the other hand, Offline handwriting recognition deals with batch mode processing of handwriting data on a powerful computer. The challenges involved in online handwriting recognition are different from the challenges involved in offline handwriting recognition. The immediate

recognition of handwriting in online recognition systems coupled with limited availability of resources on handheld devices require the recognition algorithm to be light and fast.

### 1.2.1 Why online handwriting recognition

Usage of mobile phones and handheld devices are rapidly rising in India, however the interface for most of the mobile phones are based on keypads and keyboards which can only cater to English. Input of characters from Indian languages is impossible on the small keypads provided on the mobile phones. The only viable solution is to provide a touch sensitive screen where users can write the character and a fast recognition algorithm that can immediately recognize the character entered. Providing these features opens up possibilities for SMS and e-mail in Indian languages.

Indian languages provide the ability to combine multiple vowels and consonants to obtain characters with different flavors. This leads to a combinatorial explosion of possibe characters. For instance, in Kannada there are 53 consonants and vowels which form the base set. But the total possible distinct characters that can be written is more than 5000. This makes it inevitable for handheld devices to have handwriting recognition for handling Indian languages.

The challenges involved in building such an algorithm are many. No Machine learning classification algorithm is presently available that can handle thousands of classes accurately. The limited processing power on a handheld device also requires the algorithm to be fast and have a small memory footprint. The algorithms developed also have to be language neutral and training on a new language should require minor tweaking. It should be simple enough for the user to train the phone on any new language.

The experiments for this project were conducted on a Simputer [1]. Simputer is a hand held device with a touch sensitive screen. Simputer runs on a Arm processor with

[1] http://www.amidasimputer.com/

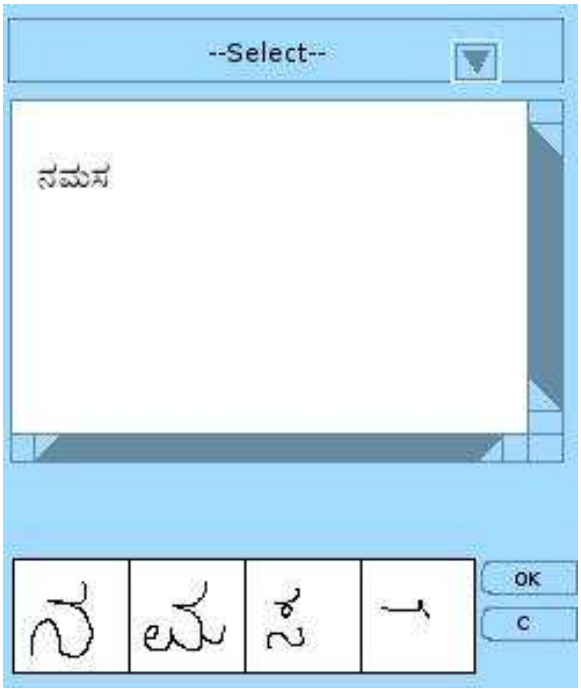



Figure 1.3: A mock up of the Kannada SMS application developed on the Simputer

a clock speed of 200MHz. It lacks a floating point coprocessor but emulation of floating point operations are supported. Other commercially available phones in the market have a similar configuration. For example, the Nokia 9300 runs on a Texas Instruments OMAP 1510 processor with a clock speed of 150MHz. Hence the algorithms developed for the Simputer can be ported to any other mobile phones. The challenges are to build classifiers capable of classifying a character into one of 5000 classes in less than a second without considerable strain on the low power processor.

## 1.3 Outline

The organization of the manuscript is as follows. In Chapter 2 some of the work previously done in this area are mentioned and is followed by detailed reviews of works relevant to the project. Chapter 3 introduces a set of kernels for classifying time series. Chapter 4 provides details about our effort in fitting a generative model for handwriting. We end the manuscript with a brief summary in Chapter 5

# Chapter 2

# Previous work

***Some of the exisiting algorithms for handwriting recognition and time series analysis are introduced in this chapter. Few of the more relavant algorithms are explained in more detail.***

Time Series classification and Handwriting recognition have been studied in great detail. There are many applications and algorithms that are being commercially used. This chapter provides a brief insight into some of the existing methods for time series classification. The chapter starts with an summary of classification schemes used on time series previously. A summary of relevant machine learning techniques is also provided. The chapter explores in detail some of the state of the art techniques in this area like GDTW. Finally work done by Sivaramakrishnan [1] and Rahul Agrawal [2] are explained in detail as these are prerequisites for the solution provided.

## 2.1 Time Series Analysis

The problem of time series classification has been well studied in the area of speech processing where Hidden Markov Models(HMMs) [3] have emerged as a powerful tool.

However its performance on other temporal domains is not well known. The applicability of HMMs to online handwriting recognition was explored in [4,5] with mixed results. HMMs are a special case of Bayesian network while Dynamic Bayesian Networks are more suited for modelling temporal sequences. DBNs was explored in [6] for time series classification and can be another interesting alternative. These algorithms rely on the Expectation Maximization algorithm which turns out to be computationally expensive making them unsuitable for implementation for online handwriting recognition on a PDA. There are also other approaches to online handwritten character recognition tasks, for a detailed review please see [7].

In recent times Support Vector Machines (SVMs) [8] have emerged as a powerful tool for classifying fixed length vectors. Its applicability to other kinds of data like sequences, curves, time-series etc is an open issue. The main research issue in extending the SVM formulation to such data is the design of kernel function. The Fisher kernel [9] was an important breakthrough which was used successfully applied to classifying protein sequences. Each kernel computation requires two passes of a forward backward algorithm on a pre-trained HMM and hence is computationally heavy. Alternatively it is possible to derive kernels based on dynamic programming based alignment for classification of sequences [10]. Dynamic Time Warping(DTW) is an instance of such a approach, which was explored by [11] for online handwriting recognition with encouraging results over HMM based classifiers. A variant of DTW which uses clustering techniques [12] seems to be the state of the art.

## 2.2 The gaussian dynamic time warping kernel

Recently a new kernel called GDTW was proposed by Bahlmann et al. in [11] for Support Vector classification of online handwriting data. This is one of the state of the art algorithms for online handwriting recognition and hence we have chosen to compare the empirical performance of our algorithm against GDTW. A GDTW kernel uses the score provided by a Dynamic Time Warp alignment [3] with the gaussian

kernel for Support Vector Machines.

$$K(P_i, P_j) = e^{-\gamma D(P_i, P_j)} \tag{2.1}$$

Where $P_i$ and $P_j$ are input sequences and $D$ is the DTW distance defined as.

$$D(P_i, P_j) = \min_{\phi_i, \phi_j} \frac{1}{N} \sum_{k=1}^{N} d(P_i(\phi_i(k)), P_j(\phi_j(k))) \tag{2.2}$$

where $\phi$ is the alignment path defined as $\phi = (\phi(1), \ldots \phi(N))$ with $\phi(n) \in \{1, \ldots, N\}$ where $N$ is the length of the sequence. $\phi$ is introduced in order to align the corresponding regions of the sequences $P_i$ and $P_j$. $d$ is the local distance measure, we have used the Euclidean distance in our experiments. (See [3] for more details).

One of the troubles with GDTW as explained above is that the time complexity of kernel calculation is $O(mn)$ where $m$ and $n$ are the length of the sequences. Windowing functions have been proposed to simplify the kernel calculation but use of such a system on PDA are still a concern.

## 2.3 Kernel Interpolation

In [1] a scheme for time series classification based on Kernel interpolation was proposed. We start this section by briefly describing an interpolation scheme due to Moore [13], which uses piecewise polynomial functions as basis functions. The interpolation scheme is used to represent each time-series as a function and using a large margin approach we show how to discriminate between such functions.

### 2.3.1 Kernel interpolation

Let $0 \leq s_1 < s_2 \ldots s_{n-1} < s_n \leq 1$ be a sequence of points and let $\mathcal{F}_i = \mathcal{F}(s_i)$ be the function $\mathcal{F} : [0, 1] \rightarrow \mathbb{R}$ evaluated at $s_i$. The interpolation problem can be viewed as approximating $\mathcal{F}$ given $\mathcal{D} = \{(s_i, \mathcal{F}_i) | 1 \leq i \leq n\}$.

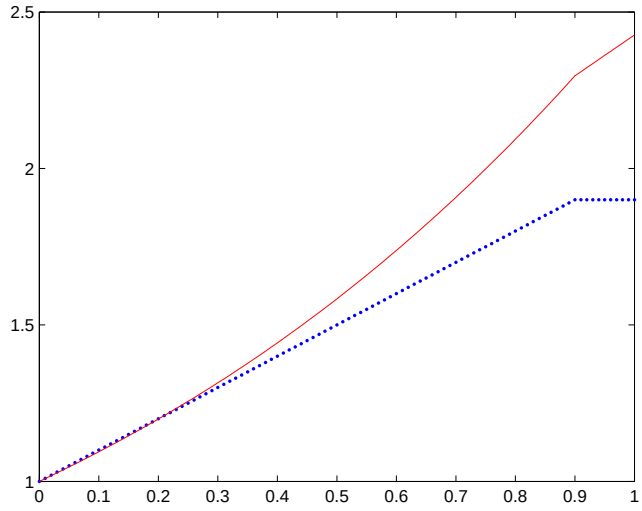



Figure 2.1: **Basis functions, $R^1_{0.9}(t)$ and $R^2_{0.9}(t)$**

Denote by $\mathcal{H}^q$ the space of all functions $f : [0,1] \to \mathbb{R}$, whose $q^{th}$ derivatives are in $\mathcal{L}_2(0,1)$. It can be shown that $\mathcal{H}^q$ is a RKHS [13] with the inner product defined as,

$$\langle f, g \rangle = \sum_{j=0}^{q-1} \frac{f^{(j)}(0) g^{(j)}(0)}{j!} + \int_0^1 f^{(q)}(t) g^{(q)}(t) dt, \tag{2.3}$$

where $f \in \mathcal{H}^q$, $g \in \mathcal{H}^q$ and $f^{(r)}(t)$ denotes the $r^{th}$ derivative. The reproducing kernel for the RKHS is,

$$R^q_s(t) = \begin{cases} \sum_{j=0}^{q-1} \frac{s^j t^j}{(j!)^2} + \int_0^{min(s,t)} \frac{(s-u)^{q-1}(t-u)^{q-1} du}{((q-1)!)^2} & q > 2 \\ \left.\begin{array}{ll} 1 + st + \frac{st^2}{2} + \frac{t^3}{6} & if\ t < s \\ 1 + st + \frac{s^2 t}{2} + \frac{s^3}{6} & if\ t > s \end{array}\right\} & q = 2 \\ 1 + min(s,t) & q = 1 \end{cases}$$

Consider the function $\bar{\mathcal{F}}$,

$$\bar{\mathcal{F}}(s) = \sum_{j=1}^{n} c_j R^q_{s_j}(s), \tag{2.4}$$

where the basis function $R^q_{s_j}(s)$ are piecewise $q^{th}$ degree polynomials. For any $0 \leq s \leq 1$ and $f \in H^q$, the basis functions satisfy the following,

$$\langle f, R^q_s \rangle = f(s),$$

which is also referred as the RKHS property. To ensure $\bar{\mathcal{F}}$ best approximates $\mathcal{F}$, one can choose $c$ by requiring that,

$$\langle (\bar{\mathcal{F}} - \mathcal{F}), R^q_{s_i} \rangle = 0 \ \forall i = 1, 2, ...n.$$

Using equation (2.4) and the RKHS property, the above set of equations reduces to,

$$\sum_{i=1}^{n} \langle R^q_{s_i}, R^q_{s_j} \rangle c_i = \mathcal{F}(s_j).$$

This is a set of linear equations in $c$ and can be efficiently solved. It can be shown that such a choice of $c$ is optimal and the resulting $\bar{\mathcal{F}}$ best approximates $\mathcal{F}$ given $\mathcal{D}$.

## 2.4 A SVM classifier based on kernel interpolation

The classifier proposed by [1] consists of resampling the interpolated time series at constant intervals and use of the vectors so obtained with an SVM classifiers.

Choosing the set of $c$ as described in the previous section, the interpolated time series can be represented as,

$$\bar{\mathcal{F}}(s) = \sum_{i=1}^{n} c_i R^q_{s_i}(s), \tag{2.5}$$

The function is resampled $M$ times represented by $t_1 \dots t_M$. This provides the interpolated vector.

$$\bar{f} = \sum_{i=1}^{n} \sum_{j=1}^{M} c_i \langle R^q_{s_i}(s), R^q_{s_j}(s) \rangle.$$

Which can be rewritten as,

$$\bar{f} = Ac,$$

Where $A_{ij} = \langle R^q_{s_i}(s), R^q_{s_j}(s) \rangle$ . The vectors so obtained are used with a SVM classifier.

$$\begin{aligned} \min_{w \in R^n, b} \quad & \tfrac{1}{2}||w||^2 + C \textstyle\sum_i \xi_i \\ \text{subject to} \quad & y_i(w^T \bar{f}^{(i)} + b) \geq 1 - \xi_i \\ & 1 \leq i \leq N \\ & \xi_i \geq 0. \end{aligned} \tag{2.6}$$

Where $\bar{f}$ is obtained by resampling $x$ and $y$ coordinates of the time series $M$ times and concatenating the sequences to obtain a vector of dimension $2M$.

## 2.5 Large Margin formulation for time series classification

The formulation in the previous section provides a disjoint process for classification. Rahul Agrawal [2] provides a more elegant single step approach for time series classification.

A univariate time series $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$ $\mathcal{F}_i = \mathcal{F}(t_i)$, evaluated at time instants $\mathcal{F}(t_i) \in \mathbb{R},\ 0 \leq t_1 < t_2 \ldots < t_n$, can be approximated by a continuous function of the form,

$$\bar{\mathcal{F}}(s) = \sum_{i=1}^{n} c_i R^q_{s_i}(s), \tag{2.7}$$

where,

$$s \in [0, 1], s_i = \frac{t_i}{t_n}. \tag{2.8}$$

Using the method outlined in the previous section the basis coefficients $c$ can be chosen to satisfy,

$$Gc = [\mathcal{F}_1 \ldots \mathcal{F}_n]^T, \tag{2.9}$$

where $G_{ij} = \langle R^q_{s_i}, R^q_{s_j} \rangle$. is the Gram Matrix for the basis elements $R^q_{s_i}$.

The learning problem can be posed as that of computing a classifier on the dataset $\mathcal{T} = \{(\bar{\mathcal{F}}^i, y_i) | \bar{\mathcal{F}}^i \in \mathcal{H}^q, y_i \in \pm 1, \forall 1 \leq i \leq N\}$. More precisely the problem can be stated as finding a $w \in \mathcal{H}^q$ and $b \in \mathbb{R}$ so that the decision function given by,

$$y = sign(\langle \bar{\mathcal{F}}, w \rangle + b),$$

can correctly predict the class label of a given time-series.

The norm of $f \in \mathcal{H}^q$ is given by $\|f\| = \sqrt{\langle f, f \rangle}$ and the distance between any two functions $f_1 \in \mathcal{H}^q$ and $f_2 \in \mathcal{H}^q$ is defined as $d(f_1, f_2) = \|f_1 - f_2\|$. Consider two sets $S_1$ and $S_2$,

$$\begin{aligned} S_1 &= \{f \in \mathcal{H}^q | \langle w, f \rangle + b \geq 1, \ w \in \mathcal{H}^q, b \in \mathbb{R}\} \\ S_2 &= \{f \in \mathcal{H}^q | \langle w, f \rangle + b \leq -1, \ w \in \mathcal{H}^q, b \in \mathbb{R}\}. \end{aligned} \tag{2.10}$$

The distance between two such sets is defined as

$$D(S_1, S_2) = \min_{f_1 \in S_1, f_2 \in S_2} d(f_1, f_2) = \frac{2}{\|w\|}. \tag{2.11}$$

To see this note that $\mathcal{H}^q$ is a Hilbert space so Cauchy Schwartz inequality applies and hence the following inequality

$$|\langle w, f_1 - f_2 \rangle| \leq \|w\| \|f_1 - f_2\|$$

holds. By definition for any $f_1 \in S_1$ one can find $c_1 \geq 0$ such that $\langle w, f_1 \rangle = 1 - b + c_1$. Similarly for any $f_2 \in S_2$ one can find $c_2 \geq 0$ so that $\langle w, f_2 \rangle = -1 - b - c_2$. This immediately gives $|\langle w, f_1 - f_2 \rangle| = 2 + c_1 + c_2 \geq 2$ which when plugged into the Cauchy-Schwartz inequality one obtains

$$\frac{2}{\|w\|} \leq \|f_1 - f_2\|$$

and hence (2.11) follows.

Let all timeseries having label $y = 1$ be in $S_1$ and all time series having the label $y = -1$ be in $S_2$(see (2.10)). A way of choosing the optimum $w$ could be by maximizing the margin, i.e. $D(S_1, S_2)$ which leads to the following formulation.

$$\begin{aligned} &\min_{w \in \mathcal{H}^q, b} && \tfrac{1}{2}\langle w, w \rangle \\ &\text{subject to} && y_i(\langle w, \bar{\mathcal{F}}_i \rangle + b) \geq 1 \\ &&& 1 \leq i \leq N \end{aligned} \tag{2.12}$$

This formulation can also be derived from a structural risk minimization framework [8].

As $w \in \mathcal{H}^q$, one can express $w$ as linear combination of the basis functions in $\mathcal{H}^q$, specifically,

$$w(t) = \sum_{j=i}^{M} d_j R^q_{m_j}(t),$$

where, $m_j$ are normalized time instants such that $m_j \in [0, 1]$ The vector $m = [m_1, \ldots, m_M]$ needs to be chosen appropriately depending on the data.

The inner product between two functions $\bar{\mathcal{F}}_1 = \sum_{i=1}^{n1} c_i R^q_{s_i}$ and $\bar{\mathcal{F}}_2 = \sum_{j=1}^{n2} d_j R^q_{s_j}$ is given by

$$\langle \bar{\mathcal{F}}_1, \bar{\mathcal{F}}_2 \rangle = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_i d_j \langle R^q_{s_i}, R^q_{s_j} \rangle. \tag{2.13}$$

Using the above definition the inner product of $w$ and data $\bar{\mathcal{F}}$ is,

$$\langle w, \bar{\mathcal{F}} \rangle = \sum_{i=1}^{M} \sum_{j=1}^{n} d_i c_j \langle R^q_{m_i}, R^q_{s_j} \rangle = d^T A c,$$

where $c = [c_1, \ldots, c_n]$ is a vector determined by (2.9). The $A$ matrix is given by

$$A_{ij} = \langle R_{m_i}, R_{s_j} \rangle \; 1 \leq i \leq M, 1 \leq j \leq n, \tag{2.14}$$

where $M$ is the number of basis functions describing $w$. Again from the definition of

the inner-product (2.13), the objective can be stated as,

$$||w||^2 = \langle w, w \rangle = \sum_{i=1}^{M} \sum_{j=1}^{M} d_i d_j \langle R^q_{m_i}, R^q_{m_j} \rangle. \tag{2.15}$$

Using (2.15) the optimization problem (2.12) can be restated as,

$$\begin{array}{ll} \min_{d,b} & \frac{1}{2} d^T B d \\ \text{subject to} & y_i(d^T A^{(i)} c^{(i)} + b) \geq 1 \\ & 1 \leq i \leq N, \end{array} \tag{2.16}$$

where $B$ is the Gram matrix for the basis functions given by,

$$B_{ij} = \langle R^q_{m_i}, R^q_{m_j} \rangle \tag{2.17}$$

from equation (2.15), $A^{(i)}$ and $c^{(i)}$ is the $A$ matrix and $c$ vector respectively for the $i^{th}$ data point.

The matrix $B$ is positive semi-definite and hence the optimization problem (2.16) is an instance of convex quadratic programming and can be solved by standard tools. The decision function can now be evaluated as,

$$y = sign(d^T A c + b).$$

The above formulation (2.16) provides a unified scheme for time series classification.

In this chapter a large margin method for time series was introduced, in the next chapter, time series kernels are extracted for time series classification and other machine learning tasks. Some variants of the kernels are used for classification of some standard time series data and the results are reported.

# Chapter 3

# Time Series Kernel

***Kernels for classification of time series are proposed in this chapter. Experiments and results are provided in the latter part of this chapter.***

One of the most important contribution of this project is explained in this Chapter. The aim of this work was to extract time series kernels from the large margin formulation proposed in the previous chapter. Variants of kernels are used on a online handwriting recognition problem and a speech recognition problem.

In the previous chapter, a large margin formulation for time series classification was provided. This chapter starts with the large margin formulation and extends the formulation to obtain general kernels for time series. The kernel is extended to operate on multivariate data. Later a final algorithm is provided for classification of time series and few methods are discussed for speeding up the calculations. The details of the experiments performed and the results follow. The chapter ends with a discussion of scheme and the results obtained.

## 3.1 A time series kernel

The Gram matrix $B$ (2.17) can be factorized as $B = U^T \Sigma U$. Where $U$ is the matrix formed by normalized eigen vectors and $\Sigma$ is a diagonal matrix, the eigen values being the diagonal elements. It can be shown that $UU^T = U^T U = I$. Based on this we can extract a kernel from (2.16) that can be used for comparing similarity between sequences of time series. Let $d = U^T \Sigma^{-\frac{1}{2}} u.$, where $\Sigma^{-\frac{1}{2}}$ is the inverse of the matrix square root of the diagonal matrix $\Sigma$. Substituting in (2.16) one obtains the following formulation.

$$\begin{aligned} \min_{u,b} \quad & \tfrac{1}{2} u^T u \\ \text{subject to} \quad & y_i (u^T X_i + b) \geq 1 \ , \\ & 1 \leq i \leq N, \end{aligned} \tag{3.1}$$

where

$$X_i = \Sigma^{-\frac{1}{2}} U A^{(i)} c^{(i)}. \tag{3.2}$$

The matrix $\Sigma^{-\frac{1}{2}} U$ can be precomputed for faster operation.

The formulation (3.1) is equivalent to an SVM [8] with the Kernel,

$$K(X_i, X_j) = X_i^T X_j. \tag{3.3}$$

## 3.2 Extension to the multivariate case

In the case of handwritten data the PDA gives two timeseries, one for each coordinate or in case of speech data there can be multiple time varying features. The theory developed so far can handle only univariate time-series. In the following we generalize the approach to handle multivariate time-series.

In general consider a vector of functions with $L$ dimensions, $\bar{\mathcal{F}} = [\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_L]^T$. The inner product of two such functions $F$ and $G$, each with $L$ dimension

$$\langle F, G \rangle = \sum_{l=1}^{L} \langle F_l, G_l \rangle.$$

Using this definition one can compute the kernel as,

$$K(i,j) = \sum_{l=1}^{L} X_{i_l}^T X_{j_l} = X_i^T X_j, \tag{3.4}$$

where $X_i = [X_{i_1}^T, \ldots, X_{i_l}^T]^T$, which is equivalent to concatenating the univariate vectors $X_{i_l}$.

The dual of the formulation (3.1) is,

$$\begin{array}{ll} \min_\alpha & \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j X_i^T X_j - \sum_i \alpha_i \\ \text{subject to} & \sum_i y_i \alpha_i = 0 \\ & 0 \leq \alpha_i, \\ & 1 \leq i \leq N. \end{array} \tag{3.5}$$

For a derivation of this please see [8]. As in standard SVM procedure one can relax the above formulation to handle non-separable data by using any positive definite kernel and restricting the $\alpha_i$'s to be less than a user defined constant $C > 0$ [8].

$$\begin{array}{ll} \min_\alpha & \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j K(X_i, X_j) - \sum_i \alpha_i \\ \text{subject to} & \sum_i y_i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \\ & 1 \leq i \leq N. \end{array} \tag{3.6}$$

We have experimented with the Radial Basis kernel,

$$K(X_i, X_j) = e^{-\gamma \|X_i - X_j\|^2}. \tag{3.7}$$

Finally we are ready to state our classification algorithm.

**Algorithm** *Given*: Training Data $\mathcal{T}$. Parameters: Sampling Instants $m$, Kernel Order $q$.

1. Re-scale time axis to [0,1] for each time series $\mathcal{F}^i$.

2. Find the coefficients $c^{(i)}$ by solving (2.9).

3. Find $A^{(i)}$ for the $i^{th}$ time series at time instants specified by $m$ (see 2.14).

4. Use the appropriate kernel (3.4) or (3.7) and solve using SVM.

The above defined algorithm has a time complexity of $O(M)$ for kernel calculation, where $M$ is a constant. This is a large improvement from the time complexity of GDTW which is $O(mn)$.

## 3.3 A time series Kernel based on innerproduct

In 2.13, a form for the innerproduct of two unsampled time series was provided. This innerproduct is based on interpolation and can be directly used as a kernel for time series analysis. The kernel is given by

$$K(i,j)\langle \bar{\mathcal{F}}_i, \bar{\mathcal{F}}_j \rangle = \sum_{l=1}^{n_i} \sum_{m=1}^{n_j} c_l d_m \langle R^q_{s_l}, R^q_{s_m} \rangle. \tag{3.8}$$

The kernel defined in equation (3.8) can be used with SVM. The limitation of the above method is the complexity of kernel calculation which is $O(n_i n_j)$.

## 3.4 Fast inversion scheme for piecewise linear basis functions

The calculation of vectors $X_i$, depends on time series interpolation, which is a costly task. However, for piecewise linear basis functions, $q = 1$ (see equation (2.3)), it is possible to do the interpolation cheaply. Note that the basis functions can be written as,

$$R_t(s) = \begin{cases} 1+s & 0 \leq s \leq t \\ 1+t & t \leq s \leq 1. \end{cases}$$

The structure of the basis functions can be exploited to obtain a recursive formula for $c_i$ which leads to a fast algorithm for interpolation.

Consider a function $f$ whose values are available at various time instants, $s_i, 1 \leq i \leq n$. We are interested in finding $c_i$ such that $f(s) = \sum_i c_i R_{s_i}(s)$. See that $s_i < s_j$ whenever $i < j$ which gives

$$f(s_j) = \sum_{i<j} c_i(1 + s_i) + (1 + s_j)(\sum_{i \geq j} c_i).$$

Also note that,

$$f(s_{j-1}) = \sum_{i<j-1} c_i(1 + s_i) + (1 + s_{j-1})(\sum_{i \geq j-1} c_i).$$

Subtracting one from the other we have,

$$f(s_j) - f(s_{j-1}) = (s_j - s_{j-1}) \sum_{i \geq j} c_i.$$

The coefficients $c$ can be calculated as follows.

$$\begin{aligned} c_n &= \frac{f(s_n) - f(s_{n-1})}{s_n - s_{n-1}}. \\ c_j &= \frac{f(s_j) - f(s_{j-1})}{s_j - s_{j-1}} - \sum_{i>j} c_i. \\ c_1 &= \frac{f(s_1)}{1 + s_1} - \sum_{i>1} c_i. \end{aligned} \tag{3.9}$$

This sets up a recursive formula for calculating $c_i$, see that it proceeds backwards starting from $n$. The worst case time complexity of the above algorithm is $O(n)$ which is again considerably cheaper than the matrix inversion step.

## 3.5 A comparison with resampled approach

In [1] a two step scheme for interpolation and resampling was used with SVM classification. The classification kernel in such a case is a simple L2 product of the resampled

vector.

$$K_{ij} = (A^{(i)}c^{(i)})^T(A^{(j)}c^{(j)}),$$

where $c$ is the coefficients of basis functions (2.9), and $A$ is as defined in (2.14). In the proposed formulation, the kernel obtained after optimization is given by,

$$K_{ij} = X_i^T X_j = (A^{(i)}c^{(i)})^T B^{-1}(A^{(j)}c^{(j)}).$$

Thus if the matrix $B$ is identity, then the two formulations are identical.

In this section, we have proposed a formulation for classifying time-series and discussed several ways in which the formulation can be practically applied to real world problems. One of the strengths of our formulation is that we can employ polynomials of any positive order. It must be noted that as the order increases, the piecewise polynomials attain increasingly complicated shapes. Trade off must be decided between the complexity of the piecewise polynomial and the extent of fitting that is done.

## 3.6 Experiments

The large margin formulation (2.16) proposed in the previous section was tested on three freely available real world timeseries datasets. In this section we provide empirical results for the experiments and discuss the merits of the proposed kernels.

### 3.6.1 Online handwriting recognition

Experiments were conducted to determine the performance of the proposed formulation on different versions of online handwriting such as digits, upper case alphabets and lower case alphabets. The results obtained with the proposed formulation are compared with the state of the art handwriting recognition techniques such as CSDTW [12], GDTW [11] and HMM [14].

### Data

We have used the *pendigits* data from the UCI Machine learning repository [15] for our experiments. The data is freely available for download and is easily accessible. Results from other authors [16] are available on the data for comparison. The data consists of English numerals written on a Wacom tablet PC by 44 writers. Data from 30 writers was used for training and data from the other 14 was used as the test set. A more detailed introduction to the data can be found in [16].

Apart from the UCI dataset, the formulation was tested on another freely available dataset from MIT [1] [17]. The data consists of words, digits, upper case and lower case English alphabets written by 159 writers on a Wacom tablet PC. We have chosen to conduct experiments on the digits, lower and upper case alphabets.

### Preprocessing

The data is available in the unipen format which needs to be preprocessed before applying (2.16) . The first step involves converting the unipen form data into a time series containing a series of $[x, y, s]$ for each character. This was achieved by extracting strokes from the data, and obtaining the time series for each of the strokes. Time series for a character is obtained by concatenating the time series obtained by individual strokes in the character. Each of the series was further normalized in $x, y$, such that the character lies in a fixed size box of $50 \times 50$ points. The scaling was done such that the aspect ratio of the character was maintained. The character was also translated on both $x$ and $y$ axes such that the minimum value for these axes is zero. An example of the normalized 8 is given in figure 3.1. The code for preprocessing, training and testing are available for download from our site [18].

[1] available at ftp://lightning.lcs.mit.edu/pub/handwriting/mit.tar.Z

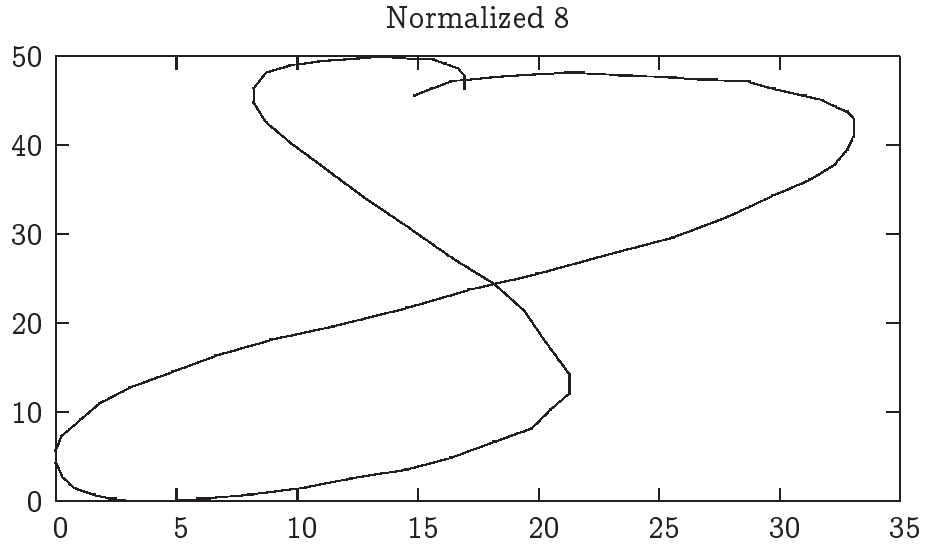



Figure 3.1: **An illustration of the preprocessing on the characters. The character is translated such that the minimum of the $x$ and $y$ coordinates are 0. The character is scaled such that axes with the higher span is exactly 50 units.**

### Results

In the previous section it was shown that (2.16) can be solved by standard SVM solvers. The experiments were carried out by adding these custom kernels to the libsvm package [19]. The experiments with linear kernel (3.4) are labelled as **QP Linear** and those with Gaussian kernel (3.7) are labelled as **QP Gaussian**. The kernels ((3.4) or (3.7)) take different forms with different order of interpolation. We have limited our experiments upto second order interpolants ($q = 1, q = 2$ in (2.3)). The experiments can be easily extended to higher order interpolants. The classifiers were tuned to the data at hand by tuning the parameters on the validation set (see appendix). For comparison, results from the literature on different versions of the unipen data set are provided. The results of the experiments are tabulated in table 3.1.

The results show that in case of digits, the performance of the proposed formulation with a gaussian kernel is comparable to the state of the art. The performance of the linear interpolation is better than that of the quadratic interpolation scheme.

Similarly the proposed formulation with a Gaussian kernel provides a comparable performance with other schemes for classification of uppercase and lower case handwritten characters. It must be noted that as the dataset used in our experiments are different from those that were used in results from other approaches, an exact comparison is not possible. Overall, the proposed formulation with a Gaussian kernel and linear interpolation is suggested for writer independent handwriting recognition.

## 3.7 Speaker Recognition

### 3.7.1 Data

The second set of experiments were conducted on the Japanese Vowel data freely available from UCI KDD repository [20]. The data was introduced in [21]. The data consists of 12 LPC coefficients of utterances of the vowel *ae* by 9 Japanese speakers. The classification task is to identify the speaker from these sequences.

### 3.7.2 Results

We used the proposed formulation for classification of the vowel data. Both QP Linear and QP Gaussian kernels were tried. The results are compared with the 5 State HMM quoted in [21]. The algorithms were compared based on the error rate on the dataset at hand. The error rate denotes the percentage of data that was not correctly labelled by the particular classifier. The result of the experiments are provided in table 3.1.

In this chapter various kernels were proposed and the efficacy of the kernels were tested on classification problems. It was found that the performance of the kernels were inferior to state of the art method in handwriting while some of the kernels performed better than state of the art for speech. The kernels can also be tried out on other machine learning tasks such as regression, clustering etc. The next chapter looks at a generative model for online handwriting recognition.

Table 3.1: Results

| Data | Approach | Error (%) | $q = 1$ | $q = 2$ |
|---|---|---|---|---|
| Digits, UCI KDD (0-9) | QP Linear | 8.35 | 8.35 | 9.25 |
| *pendigits* | QP Gaussian | **5.15** | 5.15 | 7.26 |
| Digits (0-9), | QP Linear | 10.95 | 10.95 | 11.38 |
| MIT Unipen dataset | QP Gaussian | **6.34** | 6.34 | 8.42 |
| Digits (0-9), | CSDTW [12] | 2.9 | | |
| Unipen dataset(R01/V07) | DAG-SVM-GDTW [11] | 3.8 | | |
| | HMM [14] | 3.2 | | |
| Upper case alphabets (A-Z), | QP Linear | 17.02 | 17.02 | 19.89 |
| MIT Unipen dataset | QP Gaussian | **8.25** | 9.1 | 8.25 |
| Upper case alphabets (A-Z), | CSDTW [12] | 7.2 | | |
| Unipen dataset (R01/V07) | DAG-SVM-GDTW [11] | 7.6 | | |
| | HMM [14] | 6.4 | | |
| Lower case alphabets (a-z), | QP Linear | 16.06 | 16.06 | 20.48 |
| MIT Unipen dataset | QP Gaussian | **9.1** | 9.1 | 10.6 |
| Lower case alphabets (a-z), | CSDTW [12] | 9.3 | | |
| Unipen dataset (R01/V07) | DAG-SVM-GDTW [11] | 12.1 | | |
| | HMM [14] | 14.1 | | |
| Japanese Vowel, | QP Linear | **3.22** | 3.22 | 3.25 |
| | 5-state continuous HMM | 3.8 | | |

# Chapter 4

# Generative modeling of handwriting using Kalman filter

***In this chapter, handwriting is modelled as generated by movement a hidden state in a single dimension. The hidden state is inferred using Kalman filter and classifiers were trained on the hidden states. The efficacy of this procedure is determined with experiments***

Handwriting consists of the motion of a stylus or a pen on a two dimensional plane. The $(x, y)$ position of the pen at various instants is available as a multivariate time series. The series can be modelled as produced by a linear evolution of a hidden state, the $x$ and $y$ position of the stylus being manifestation of the hidden state transitions. Initially, two Kannada numerals were selected and the hidden states were inferred using a Kalman filter. The time evolution of the hidden states as well as the $x$ and the $y$ coordinates are shown in figure 4.2. From the figure, it is clear that the separation provided by the hidden states is better than that provided by the concatenated $x$ and $y$ series. Using this as the motivation, a set of algorithms were developed for classification of multivariate time series using hidden states.

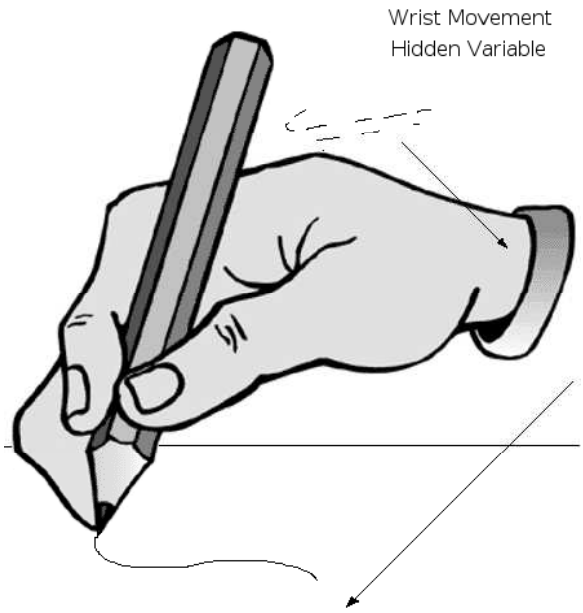



Figure 4.1: **The motion of the hand as a hidden state.**

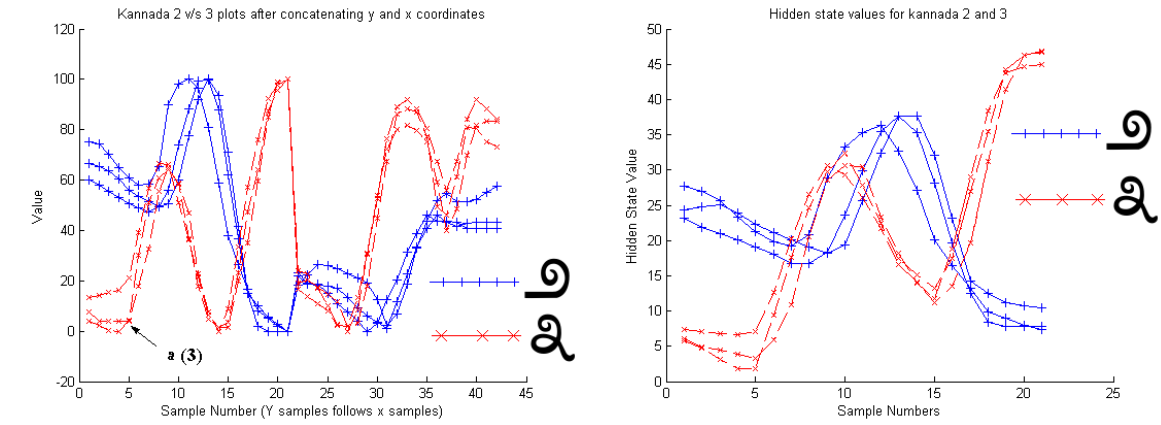



Figure 4.2: **The top figure denotes two sets of unfiltered Kannada numerals, the second figure contains the equivalent hidden states obtained from a Kalman filter. The difference is more marked in case of the hidden states as compared to the observed state.**

## 4.1 Kalman Filter

A Kalman filter is a time linear dynamic system with gaussian noise. (An excellent review of linear Gaussian models can be found in [22]). The state of the system, $X$ is a $k$ dimensional continuous variable which cannot be measured directly. The state $X$ is assumed to evolve according to a simple first order Markov dynamics. The observed sequence $Y$ which is a $p$ dimensional continuous variable is linearly dependent on the state of the system. The basic generative model can be written as:

$$\begin{aligned} x_{t+1} &= Ax_t + w_t \quad & w_t &\sim N(0, Q), \\ y_t &= Cx_t + v_t \quad & v_t &\sim N(0, R), \end{aligned} \tag{4.1}$$

where $A$ is a $k \times k$ state transition matrix and $C$ is a $p \times k$ observation, measurement, or generative matrix. The $k$ vector $w$ and the $p$ vector $v$ are random variables representing the state evolution and observation noises respectively, which are independent of each other and of the values of $x$ and $y$. Both the noise sources are Gaussian with zero mean and covariance represented by the $Q$ and $R$ matrices respectively.

The observed vector $Y$ in case of handwriting is a vector with $p = 2$. We assume that the hidden state is a single dimension variable, i.e $k = 1$. This simplifies (4.1) to

$$\begin{aligned} x_{t+1} &= ax_t + w_t \quad & w_t &\sim N(0, \sigma^2), \\ y_t &= Cx_t + v_t \quad & v_t &\sim N(0, R), \end{aligned} \tag{4.2}$$

where $\sigma^2$ is the variance of the noise on the hidden state. The initial state of the system is assumed to be random with a Gaussian distribution with a mean $\mu_1$ and variance $\sigma_1$.

$$x_1 \sim N(0, \sigma_1^2). \tag{4.3}$$

The conditional expectations for the hidden state and observables can be written

as:

$$\begin{aligned} P(x_{t+1}) &\sim N(ax_t, \sigma^2), \\ P(y_t) &\sim N(Cx_t, R). \end{aligned} \tag{4.4}$$

Because of the Markov assumption on the hidden state, the joint probability of a sequence of $\tau$ states and outputs is given by:

$$\begin{aligned} &P(x_1 \cdots x_\tau, y_1 \cdots y_\tau) \\ &= P(x_1) \prod_{t=1}^{\tau-1} P(x_{\tau+1} \mid x_\tau) \prod_{t=1}^{\tau} P(y_\tau \mid x_\tau). \end{aligned} \tag{4.5}$$

The negative log probability (cost) is just the sum of matrix quadratic forms:

$$\begin{aligned} &-2logP(x_1 \cdots x_\tau, y_1 \cdots y_\tau) \\ &= \textstyle\sum_{t=1}^{\tau} [(y_t - Cx_t)^T R^{-1} (y_t - Cx_t) + log|R|] \\ &+ \textstyle\sum_{t=1}^{\tau} [\frac{||x_{t+1} - ax_t)||^2}{\sigma^2}] + log(\sigma) \\ &+ \frac{||x_1 - \mu_1||^2}{\sigma_1^2} + log(\sigma_1^2) + 3\tau log(2\pi). \end{aligned} \tag{4.6}$$

In this project, we are interested in learning the parameters of the model, i.e $\theta = [a, C, \sigma, R, \sigma_1, \mu_1]$ and the problem of filtering, which is to determine the sequence of hidden states $x$ for a sequence of observations $y$ given the model $\theta$.

## 4.2 Parameter Learning

Before the Kalman filter is used for testing, the parameters $\theta = [a, C, \sigma, R, \sigma_1, \mu_1]$ have to be learnt from the training set. Learning is also called as system identification. This is achieved by using expectation maximization on the training set as with Hidden Markov Models [22, 23]. The algorithm for the single dimension case $k = 1$ is provided in algorithm 1.

**Algorithm 1** Algorithm for determining the parameters $A, C, Q, R, x_1^0, v_1^0$

.

Learn$(Y, \epsilon)$

Initialize $a, C, \sigma^2, R, x_1^0, v_1^0$ {In our experiments we initialize these randomly}.

$\alpha \leftarrow \sum_t Y_t Y_t^T$

**while** change in likelihood $> \epsilon$ **do**

InferStates$(Y, a, C, \sigma^2, R, x_1^0, v_1^0)$ {E Step.}

Initialize $\gamma \leftarrow 0$, $\beta \leftarrow 0$, $\delta \leftarrow 0$

**for** $t = 1$ to $T$ **do**

$\delta \leftarrow \delta + \hat{x}_t Y_t$

$\gamma \leftarrow \gamma + \hat{x}_t^2 + \hat{v}_t$

$\beta \leftarrow \beta + \hat{x}_t^2 + v_{t,t-1}$

**end for**

$\gamma_1 \leftarrow \gamma - \hat{x}_T^2 - \hat{v}_T$

$\gamma_2 \leftarrow \gamma - \hat{x}_1^2 - \hat{v}_1$

$C \leftarrow \frac{1}{\gamma}\delta$

$R \leftarrow \frac{(\alpha - C\delta^T)}{T}$

$a \leftarrow \frac{\beta}{\gamma}$

$\sigma^2 \leftarrow \frac{(\gamma_2 - a\beta)}{(T-1)}$

$x_1^0 \leftarrow \hat{x}_1$

$v_1^0 \leftarrow \hat{v}_1$

**end while**

return $a, C, \sigma^2, R, x_1^0, v_1^0$

## 4.3 Inference

Inference is the problem of determining the latent states given the fixed model $\theta$ and the observations. If $y_1 \dots y_n$ is the observed sequence, the problem can be more formally stated as:

$$P(x_1, \dots, x_n | y_1, \dots, y_n, \theta). \tag{4.7}$$

To be of any use for the classification problem, the values of $x_i$, with the highest likelihood has to be determined. Since we are assuming the distribution on $x_i$s to be Gaussian, we need to determine the mean $\mu_i$ and variance $\sigma_i$. For a Gaussian distribution, the maximum likelihood is achieved when $x_i = \mu_i$, so in the inference problem, we are interested in determining $\mu_i$s for all $i$. The problem can be solved by a variant of the forward backward algorithm called the Rauch-Tung-Streibel Smoother [22]. The algorithm for the single dimension case is provided in algorithm 2. The quantities $x_t^s$ and $v_t^s$ denote the mean and the variance of $x_t$ given observation $y_1, \dots, y_s$, $\hat{x}_t \equiv x_t^T$ and $\hat{v}_t \equiv v_t^T$ The above provided algorithm has a time complexity of $O(T)$. Where $T$ denotes the number of samples in the sequence.

The trained parameters of a Kalman filter are heavily dependent on the initial parameter values, which are chosen at random. Hence training has to be repeated $N$ times and the model that best describes the data, that is the model with the highest likelihood is chosen. The Kalman training algorithm is given in algorithm 3.

The time series data provided is preprocessed before using the algorithm. Preprocessing involves linear interpolation of the time series and resampling the interpolated function $M$ times. Thus for a multivariate time series, after preprocessing each dimension is a vector of $M$ samples.

The algorithm 3 is used on the resampled vectors to obtain the trained Kalman filter.

The preprocessed training data is now filtered to obtain a univariate series of $M$

**Algorithm 2** Algorithm for determining the hidden states.

InferStates$(Y, a, C, \sigma^2, R, x_1^0, v_1^0)$

**for** $t = 1$ to $T$ **do**

$x_t^{t-1} \leftarrow a x_{t-1}^{t-1}$ {if $t > 1$}

$v_t^{t-1} \leftarrow a^2 v_{t-1}^{t-1} + \sigma^2$ {if $t > 1$}

$K_t \leftarrow v_t^{t-1} C^T (v_t^{t-1} C C^T + R)^{-1}$

$x_t^t \leftarrow K_t (y_t - C x_t^{t-1})$

$v_t^t \leftarrow v_t^{t-1} - K_t v_t^{t-1} C$

**end for**

Initialize $\hat{v}_{T,T-1} = a v_{T-1}^{T-1} (I - K_T C)$

**for** $t = T$ to 2 **do**

$j_{t-1} \leftarrow a \frac{v_{t-1}^{t-1}}{v_t^{t-1}}$

$\hat{x}_{t-1} \leftarrow x_{t-1}^{t-1} + j_{t-1} (\hat{x}_t - a x_{t-1}^{t-1})$

$\hat{v}_{t-1} \leftarrow v_{t-1}^{t-1} + j_{t-1}^2 (\hat{v}_t - v_t^{t-1})$

$\hat{v}_{t,t-1} \leftarrow j_{t-1} v_t^t + j_t j_{t-1} (\hat{v}_{t+1,t} - a v_t^t)$

**end for**

return $\hat{x}_t, \hat{v}_t, \hat{v}_{t,t-1} \forall t$

**Algorithm 3** Training algorithm for Kalman filter.

TrainKalman($Y, N$)

Initialize maxlikelihood $\leftarrow 0$

**for** $i = 1$ to $N$ **do**

$[A, C, Q, R, x_1^0, v_1^0, L] \leftarrow Learn(Y)$

**if** $L > maxlikelihood$ **then**

$maxlikelihood \leftarrow L$

$A_{max} \leftarrow A$

$C_{max} \leftarrow C$

$Q_{max} \leftarrow Q$

$R_{max} \leftarrow R$

$x^0_{1max} \leftarrow x^0_1$

$v^0_{1max} \leftarrow v^0_1 L$

**end if**

**end for**

samples $x_i$. A SVM classifier is trained on these samples. The formulation is given by,

$$\begin{aligned} \min_{w \in R^n, b} \quad & \tfrac{1}{2}||w||^2 + C \textstyle\sum_i \xi_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & 1 \leq i \leq N \\ & \xi_i \geq 0. \end{aligned} \tag{4.8}$$

The final training algorithm is given in algorithm 4

**Algorithm 4** Training algorithm for SVM.

TrainSVM($\mathcal{D}$)

$\mathcal{D} \leftarrow PreProcess(\mathcal{D})$

$KalmanFilterParameters \leftarrow TrainKalman(\mathcal{D}, 10)$

$\mathcal{D}' \leftarrow InferStates(\mathcal{D}, KalmanFilterParameters)$

$Classifier \leftarrow TrainSVM(\mathcal{D}')$

| Method | UCI | Kannada |
|---|---|---|
| Kalman Filter (4.8) | 92.63% | 77.11% |
| SVM on observed (2.6) | 97.79% | 82.8% |

Table 4.1: **The table compares the accuracies obtained by using SVM on the observed states to the accuracies obtained by using SVM on the hidden states.**

For classification of a test pattern, the time series is preprocessed to obtain a vector of fixed length. The obtained vector $y$ is filtered using the previously trained Kalman filter to get the vector $x$. The decision is taken by,

$$label = sign(w^T x + b)$$

## 4.4 Experiments and Results

The Kalman filter formulation proposed in the previous section was tested on the UCI and the Kannada data. For comparison, a formulation proposed in [1] was used. The formulation is given in equation 2.6.

The two algorithms were compared based on the accuracy provided by the algorithm with the data set at hand. Accuracy was defined as the percentage of patterns correctly classified from all the classes in the dataset.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 252 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 12 | 1 |
| 1 | 225 | 10 | 7 | 0 | 0 | 0 | 2 | 0 | 1 |
| 0 | 5 | 233 | 2 | 0 | 0 | 3 | 5 | 0 | 0 |
| 0 | 1 | 0 | 234 | 0 | 3 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 253 | 0 | 10 | 2 | 0 | 3 |
| 1 | 0 | 0 | 29 | 0 | 204 | 0 | 0 | 2 | 2 |
| 1 | 0 | 7 | 0 | 0 | 1 | 239 | 0 | 1 | 0 |
| 0 | 1 | 6 | 0 | 0 | 0 | 1 | 252 | 2 | 4 |
| 21 | 1 | 9 | 0 | 1 | 1 | 1 | 8 | 194 | 0 |
| 0 | 4 | 1 | 0 | 2 | 0 | 0 | 0 | 1 | 224 |

From the results, it can be seen that the performance of the Kalman filter does not match upto the expectation. The performance may improve with use of higher order models.

In this chapter, a generative model for handwriting was proposed. The model was used to infer a hidden state series for every character. Experiments showed that the Kalman filters modelled were not able to provide results better than the state of the art. More sophisticated versions of the Kalman filter like the extended Kalman filter can be tried in future.

# Chapter 5

# Discussion and Future work

***A discussion of the previous chapters and the project is provided in this chapter. Some possible areas of research are outlined.***

In this project, algorithms for time series data were analyzed and particular solutions for handwriting recognition were proposed. It was shown that time series analysis on system with constraints on processing power was still an open problem. Solving this problem will enable users to have handwriting recognition on mobile phone in a language of their choice.

In the first part of the project, a large margin classification algorithm was studied in detail. Based on the algorithm, several kernels were proposed for time series analysis. A faster method for time series interpolation was also proposed. The kernels were shown to work satisfactorily in different domains like handwriting and speech. However the proposed method was found to lack the accuracy provided by some of the state of the art methods. The error rates obtained from the algorithm was around 3% more than the state of the art techniques. This may be attributed to the decomposition of the $B$ matrix. More decompositions can be tried out to determine the fault with the algorithm. The kernels can also be tested on other machine learning tasks such as regression and clustering.

A technique to classify online handwriting was proposed by modelling handwriting as a linear gaussian system. The technique involved training a Kalman filter on the training data and inferring a hidden state series for each of the training data and a classifier was trained on the hidden states. Initial experiments with Kannada numerals showed that the visual separation provided by using the hidden states on characters of different types was greater than than that obtained by directly using the observed states. The experimental results show an increase of 5% in the error rates by using the hidden states. The variation in the visual separability and the separability in the feature space can be a topic of future research. Research can be also performed by using more sophisticated forms of the Kalman filter.

The problem of handling multiple combination of Indian characters was solved by using different symbols for the modifier and the base character. So the number of distinct possible characters is easily manageable. For instance in Kannada the total number of characters to be learnt is now around 70 instead of 5000. The flavored characters are written by writing the base characters and the modifiers in separate boxes as shown in figure 1.3. All the modifiers are applied to the preceding base character. This model is not too constraining and was successfully used in many commercially available mobile phones [1] for English as shown in figure 5.1.

The problem of Handwriting Recognition on mobiles is far from solved. A true solution will involve complete recognition of any language without any external training. A true solution has a potential to take the mobile phone to millions of Indians who are not English literate.

[1] www.3g.co.uk/PhoneReviews/MotoA1000/MotoA12.htm

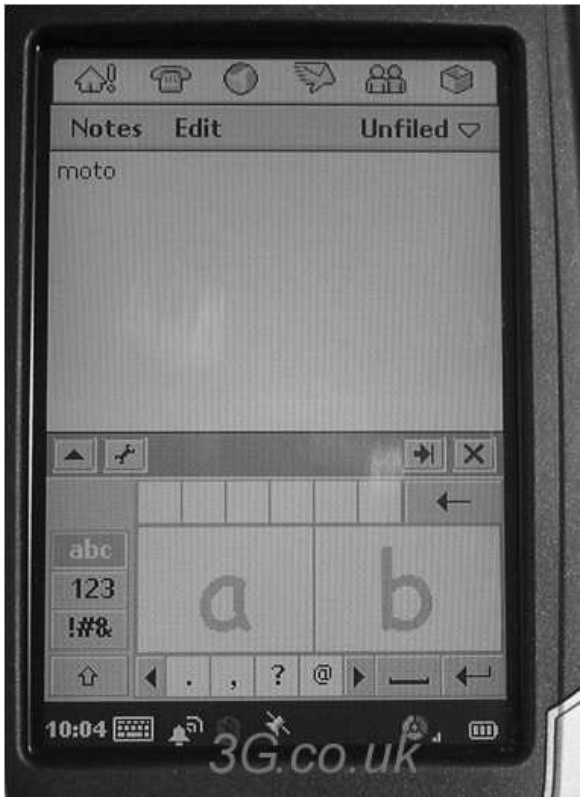



Figure 5.1: A phone with characters written in separate boxes

# References


[1] K. R. Sivaramakrishnan and Chiranjib Bhattacharyya. Time series classification for online tamil handwritten character recognition - A kernel based approach. In *ICONIP*, pages 800–805, 2004.

[2] Rahul Agrawal. *Machine Learning Techniques for DNA Sequence Data and Time Series*. ME. Thesis, Department of CSA, IISc, September 2004.

[3] L. Rabiner and B-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[4] H Binsztok and Thierry Artières. Learning HMM structure for on-line handwriting modelization. In *IWFHR*, Tokyo, October 2004.

[5] Jianying Hu, Michael K. Brown, and William Turin. Invariant features for HMM based on-line handwriting recognition. In Carlo Braccini, Leila De Floriani, and Gianni Vernazza, editors, *Image Analysis and Processing, 8th International Conference, ICIAP '95, San Remo, Italy, September 13-15, 1995, Proceedings*, volume 974 of *Lecture Notes in Computer Science*, pages 588–593. Springer, 1995.

[6] V. Pavlovic, B. J. Frey, and T. S. Huang. Time-series classification using mixed-state dynamic bayesian networks. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition (CVPR-99)*, pages 609–617, Los Alamos, June 23–25 1999. IEEE.

[7] Réjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell*, 22(1):63–84, 2000.

[8] N. Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York., 2000.

[9] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*. MIT Press, 1998.

[10] Chris Watkins. Dynamic alignment kernels, July 15 1999.

[11] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines: A kernel approach. In *Frontiers in Handwriting Recognition*, pages 49–54, 2002.

[12] Claus Bahlmann and Hans Burkhardt. The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Anal. Mach. Intell*, 26(3):299–310, 2003.

[13] R.E Moore. *Computational Functional Analysis*. Academic Press, NY, 1985.

[14] Jianying Hu, Sok Gek Lim, and Michael K. Brown. Writer independent on-line handwriting recognition using an HMM approach. *Pattern Recognition*, 33(1):133–147, 2000.

[15] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[16] Fevzi Alimoglu and Ethem Alpaydin. Combining multiple representations and classifiers for pen-based handwritten digit recognitio. In *ICDAR*, pages 637–640. IEEE Computer Society, 1997.

[17] Robert H Kassel. *A comparison of approaches to on-line handwritten character recognition*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1995.

[18] Datasets.

[19] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[20] S. Hettich and S.D. Bay. The uci kdd archive, 1999.

[21] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11-13):1103–1111, 1999.

[22] Sam T. Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–345, 1999.

[23] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. Technical Report (Short Note) CRG-TR-96-2, Department of Computer Science, University of Toronto, 22 February 1996.