CLASSIFICATION OF CHIRPS USING HIDDEN MARKOV MODELS

BY

NIKHIL BALACHANDRAN, B.E.

A thesis submitted to the Graduate School

in partial fulfillment of the requirements

for the degree

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

December 2006

"Classification of chirps using Hidden Markov Models," a thesis prepared by Nikhil Balachandran in partial fulfillment of the requirements for the degree, Master of Science in Electrical Engineering, has been approved and accepted by the following:

Linda Lacey
Dean of the Graduate School

Charles D. Creusere
Chair of the Examining Committee

Date

Committee in charge:

Dr. Charles D. Creusere, Chair

Dr. Phillip DeLeon

Dr. Joe Lakey

DEDICATION

To my parents.

ACKNOWLEDGMENTS

Although every aspect of life entails research, this experience is new at every stage. It was in the Electrical Engineering Department of New Mexico State University that I had my first encounter with research in academia. I have thoroughly enjoyed the Master's program in DSP here. I would like to thank everybody in NMSU who strive to make it a better university. I would like to thank Dr. Charles D. Creusere and Dr. Phillip DeLeon for imparting their profound knowledge in this field. In addition, I am especially grateful and deeply indebted to Dr. Creusere for his immense support, encouragement and advice with which I have built a strong research and philosophical foundation. It is only now that I have understood the true meaning of *Sir Isaac Newton's* words, "*If I have seen further, it is by standing on the shoulders of Giants.*". I would also like to thank Dr. Joe Lakey for participating on my committee. I wish to express my gratitude towards Los Alamos National Laboratory, New Mexico for their support in this research and for providing us with the FORTE data without which this research would have been incomplete. I would like to thank my parents, my brother's family and my grandmother for having a strong belief in me. I would also like to thank Hrishikesh Tapse and Vijendra Raj for their invaluable comments and suggestions.

What science cannot tell us, mankind cannot know.

– Bertrand Russell

VITA

August 4, '81	Born in Hyderabad, Andhra Pradesh, India.
May '99	Graduated from Ratna Junior College, Hyderabad, Andhra Pradesh, India.
June '03	Bachelor of Engineering in Electrical Engineering, Rashtreeya Vidya College of Engineering (R.V.C.E.) Viswesvaraiah Technological University, Belgaum, Karnataka, India.
July '03-July '04	Software Engineer, Robert Bosch India Ltd., Bangalore, Karnataka,India.
Fall '04-Spring '05	Research Assistant, Fisheries and Wildlife Sciences Department, New Mexico State University, Las Cruces, NM.
Fall '05-Spring '06	Teaching Assistant, Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM.
Summer '05	Intern, Hytec Inc., Los Alamos, NM.
Summer '06-Fall '06	Research Assistant, Dr. Charles Creusere, Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM.

Publications

[1]. Nikhil Balachandran, Dr. Charles D. Creusere, "Chirp classification using Hidden Markov Models, Fortieth ASILOMAR conference, in press.

Field of Study

Major Field: Electrical Engineering

Communications & Signal Processing

ABSTRACT

CLASSIFICATION OF CHIRPS USING HIDDEN MARKOV MODELS

BY

NIKHIL BALACHANDRAN, B.E.

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 2006

Dr. Charles D. Creusere, Chair

Chirps are present everywhere in nature, in calls of bats, whales, etc. and man-made in the form of radar, sonar, gravitational waves, etc. By identifying different classes of chirps, the source can be characterized. Our basic approach combines a Short Time Fourier Transform (STFT) with a Hidden Markov Model (HMM) to track the frequency progression versus time. Here, two feature extraction methods are considered to compute a compact representation of the chirp signal. One of them entails finding a best-fit polynomial of the resulting discrete Viterbi path while the other estimates the central moments of the Viterbi path from its distribution. Our experimental results show that if either of these methods is used as a feature extraction process, separable clusters in the feature space are formed for broad classes of chirps. A Bayesian Classifier can then be applied effectively to classify the different families of chirps. Experiments have been carried out on both synthetically generated

chirp signals and naturally occurring lightning discharges as recorded by the FORTE satellite.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Introduction

Chirps are ubiquitous in nature, occurring in manmade signals like radar and sonar as well as in nature in calls of birds, whales, bats, etc. The gravitational waves that Einstein proposed in his General Theory of Relativity also occur in the form of chirps. Thus, the ability to classify chirps could be helpful in understanding the nature of the source. Chirp signals are essentially non-stationary, in that the spectral content of the signal either increases (*up-chirp*) or decreases (*down-chirp*) with increasing time in accordance to an underlying relationship between the instantaneous frequency and time. Broadly, chirps can be classified according to this relationship: i.e., linear, quadratic, exponential, etc. Moreover, within a class, chirps can be further classified based on the rate at which the instantaneous frequency changes with time, a parameter known as the chirp rate. These are the primary factors which determine whether or not a received signal belongs to a particular family of chirps. Figure 1.1 shows the *spectrogram* of a lightning discharge recorded by the FORTE (Fast On-orbit Recording of Transient Effects) satellite. The figure also shows various narrowband interferences as well as a highly noisy background.

In the method that follows, the spectrogram of the non-stationary chirp signal is transformed into a simple 1-dimensional form using a *Hidden Markov Model* along with the *Viterbi decoding algorithm* as the first step in the feature extraction

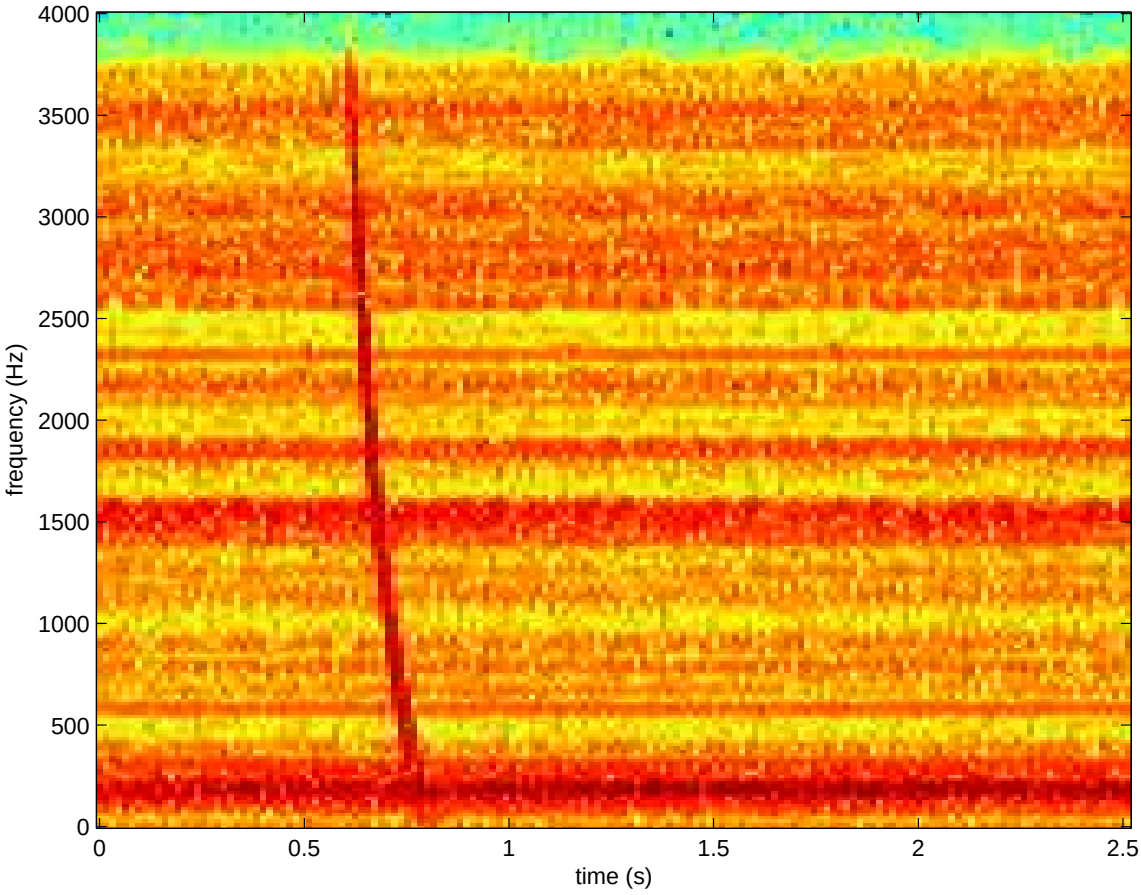


Figure 1.1: Spectrogram of lightning discharge captured by FORTE satellite.

process. The optimal state sequence[1] which maximizes the *likelihood* of the observed chirp signal occurring forms the basis for two different feature extraction approaches. One of them involves finding a *best fit curve* in the least squares sense to the state sequence and using the polynomial coefficients to form the feature vector. The second approach involves the computation of statistical moments about the mean of the state sequence's histogram. Here, these *central moments* form the feature vector. Results for both the methods are presented in Chapter 4. A limited set of chirps from each class have their features extracted using one of the methods briefly mentioned above

[1]State sequence, path and track refer to the output of the Viterbi algorithm and are used interchangeably throughout this thesis report

and a *Bayesian Classifier* is *trained* using these features. The Hidden Markov Model is also designed from training sets that characterize the different classes. During the *evaluation* stage, a signal that was not used in the training process is input to the system for feature extraction, and the resulting feature vector is classified by a Bayesian classifier previously designed from the training set.

In this thesis, the effectiveness of classifying chirp signals using Hidden Markov Models along with the two different feature extraction methods, namely, curve fitting and central moments from the histogram, is explored. The algorithms are applied to both synthetically generated chirps and lightning discharges as recorded by the FORTE (Fast On-orbit Recording of Transient Effects) satellite. The polarization properties of lightning produced VHF (Very High Frequency) signals and man-made VHF signals were studied by Shao and Jacobson in [1] and [2] respectively. These signals, amidst a highly dense VHF environment, resemble chirps and thus, the research presented in this thesis should be effective with such signals.

1.2 Literature Survey

Chirp signals occur in numerous applications, ranging from radar and sonar to seismography to the calls of certain species like bats and birds. Due to their non-stationary behavior, analyzing chirp signals has always been a difficult task. Many techniques have been developed and explored in the quest for finding a good solution.

This thesis addresses the problem of classifying isolated chirp signals of any type (e.g. Linear, Quadratic, etc.) with different chirp rates originating from different sources. The identification of chirps helps characterize the nature of the source. In [3], Davy and his colleagues explored the problem of classifying chirp signals. Their paper discusses the classification of multi-component linear and quadratic chirp sig-

nals using hierarchical Bayesian learning and Markov Chain Monte Carlo (MCMC) methods. Their algorithm requires evaluation of complicated multi-dimensional integrals, approximating them in closed form using MCMC methods. Moreover, this method needs a large number of samples to estimate the class conditional density functions. Since chirp signals fall under the category of non-stationary signals, an insight into the classification techniques of non-stationary signals would help with this research. Optimized time-frequency representations and distance measures for classification have been developed in [4] to minimize the probability of classification error. However, only linear chirp signals and real speech signals have been tested. The authors of [5] provide an improved non-stationary signal classifier using Support Vector Machines (SVM) and time-frequency representations (TFR). Unfortunately, only a few statistical parameters differentiate the two classes of linear chirps that have been used to test the method. In [6], Candès identifies chirps in amplitude-phase form and he finds that for classification purposes, the phase information is not required. In [7], time frequency representations and Maximum Likelihood (ML) estimation have been used for chirp detection with emphasis on power-law chirps. These results illustrate the robustness of using a time-frequency representation. Since the concentration of energy in the frequency domain changes slowly with time in a chirp signal, the Short-Time Fourier Transform (STFT) provides a very natural analysis framework. The authors of [8] and [9] treat the phase of the signal as a polynomial, and linear least squares schemes and Maximum Likelihood techniques are then adopted to estimate polynomial coefficients from unwrapped phases. Rank reduction and sample covariance methods are used in [10] and [11] respectively, to estimate the chirp signal's bandwidth and center frequency, but, only linear chirp signals are considered.

Another class of solutions focusses on the *frequency tracking* and the estimation of instantaneous frequency. The literature discussed thus far focusses on a non-stationary signal classification, of chirp signals in particular, and on parameter estimation. Specifically, frequency line tracking can be modeled as a state estimation problem as discussed in [12]. Estimation of the current modulating frequency using the Extended Kalman Filter (EKF), adaptive methods like the Adaptive Line Enhancer (ALE) and the Recursive Least Squares (RLS) methods and others based on time-frequency distributions have been compared in [8]. In addition to that, Boashash states that applications for frequency tracking using smoothing approaches based on Wigner-Ville distributions have been explored in [13] for speech signals, in [14] for seismic surveying and in [15] for identification of machine sounds and biological signals. Finally, Hidden Markov Models are used to track the frequency of non-stationary signals in [16]. An HMM provides a novel way of reconstructing the true frequency history from the measurement sequence contaminated by noise. The authors of [16] propose the use of HMM with the measurement sequence given by a short-time Fourier Transform where the states in the HMM coincide with the frequency bins of the FFT. The Viterbi decoding algorithm then estimates the optimal state sequence corresponding to the observed data sequence. A zero state is also provided to indicate the absence of the signal. In addition to this, mean cell occupancy (MCO) and gate occupancy probability (GOP) parameters are also defined. The MCO is a measure of variation in frequency from the mean cell frequency and it becomes large when the Viterbi track ends. The GOP, on the other hand, estimates the probability of the signal being present. The performance of the tracker is compared to the alpha-beta, the Kalman and the fixed-lag Kalman tracking algorithms, and it has is found to out-

perform the other trackers in most of the cases. Although, the work of [16] was the first recognized application of HMM to the frequency line tracking problem, HMMs were applied to a number of related applications prior to this. In [12], Boashash states that Kopec applied the HMM to the problem of tracking formants in [17], but rather than using a short-time Fourier Transform as an input to the tracker, Kopec used the codebook vectors created by a vector quantization algorithm. In [18], Jaffer and others formulate a recursive Bayesian method for tracking dynamic signals in noise, resulting in a technique that is similar to the finite-state HMM. Furthermore, they also define the states of their system as FFT bins. We note, however, that a zero state to indicate the absence of the signal is not included in their algorithm.

The work of [16] forms the starting point for the research reported in this thesis. In addition to the basic frequency line tracker, this thesis goes one step further and performs feature extraction, using one of the methods discussed previously above, on the output of the HMM tracker. Most of the previous research in this area has been focussed on a narrow set of classes of chirps, mostly linear, with few statistical differences between the classes. Hence, this thesis broadens the perspective of classification of chirps by considering ten different families of known synthetically generated chirps as well as unknown naturally occurring chirps.

1.3 Overview

This thesis has been organized in the following manner. Chapter 2 gives a brief tutorial on Hidden Markov Models, highlighting the three main problems of HMMs and giving solutions to them. Chapter 3 focusses on the algorithm used in this research and describes the different stages of the classifier. The spectrogram is described briefly, moving on to the Viterbi algorithm, and finally the two parameter

extraction techniques- polynomial curve fitting and central moments. The training of the HMM and Bayesian Classifier as well as the testing of the classifier are also discussed. Chapter 4 details the experiments performed and interprets the results obtained when our algorithm is applied to synthetically and naturally generated data sets. Finally, Chapter 5 concludes the thesis and highlights the ways in which this thesis can be extended in the future.

Chapter 2

TUTORIAL ON HIDDEN MARKOV MODELS

2.1 Introduction

In many signal processing applications, we encounter signals that are buried in noise, and extracting the relevant data from these signals is of great importance. Reference [19] provides an excellent overview of Model-based signal processing techniques. Model-based signal processing takes into account the underlying physical phenomenon creating the signal, the measurements and the noise process by using a mathematical modeling framework. This approach enables researchers to achieve better performance than common heuristic techniques. As many signals can be characterized as the results of real-world processes, characterizing such signals using mathematical models of these processes is of great interest. Signal models are broadly classified as deterministic or statistical. Deterministic models capitalize on known properties of the signal and by estimating the parameters of such models, one can completely characterize the signal. Alternatively, one can create a statistical model which characterizes the statistical properties of the signal. The underlying assumption here is that the signals these model characterize are random processes and that the parameters of these random processes can be estimated in a straight-forward manner. One such statistical model is the Hidden Markov Model (HMM). Our review in this chapter is based on the information provided in [20] and [21]. Reference [20] gives a good insight into modeling systems with Hidden Markov Models and contains a detailed

description of how they can be applied to the problem of speech recognition. The basic theory of Hidden Markov Models was established in a series of seminal papers by Leonard E. Baum and his collaborators in [22] and [23]. The focus of this chapter is to help the reader develop a basic understanding of Hidden Markov Models and to familiarize him/her with its parameters and its solutions to the three fundamental problems, as explained in more detail in the following sections.

2.2 Hidden Markov Models

A Hidden Markov Model is a statistical model which can be used to characterize the statistical properties of a signal. The underlying system being modeled is assumed to be a Markov process. In a conventional Markov process, the states are visible and the parameters of the state transition probability matrix have to be estimated, whereas, in a Hidden Markov Model, the states are hidden from the observer. The elements of an HMM are:

- N, the number of states in the model. The states of a Hidden Markov Model usually have a physical significance; they are, however, hidden. When any state can be reached from any other state, the HMM is called ergodic (Figure 2.1). Other types of HMM are Left-Right HMM (Figure 2.2), Right-Left HMM, etc. The individual states of the HMM are denoted as $W = \{S_1, S_2, S_3, \ldots, S_N\}$. The state at time t is denoted as $S_k(t)$, where $1 \leq k \leq N$ and a sequence of T hidden states are denoted as $S^T = \{S(1), S(2), S(3), ..., S(T)\}$.

- M, the number of distinct observational symbols. More often than not, the observation symbols correspond to the physical output of the system. The individual symbols are denoted as $V = \{v_1, v_2, v_3, \ldots, v_M\}$.

- **A**, the state transition probability matrix given by $\mathbf{A} = \{a_{ij}\}$, where,

$$\{a_{ij}\} = P(S_j(t+1)|S_i(t)), 1 \leq i, j \leq N. \tag{2.1}$$

with the constraint $0 \leq \{a_{ij}\} \leq 1$ and $\sum_{j=1}^{N}\{a_{ij}\} = 1,\ 1 \leq i \leq N$. For an ergodic HMM, $\{a_{ij}\} > 0$ for all i, j. Whereas, for other types of HMM, $\{a_{ij}\} = 0$ for some i, j.

- **B**, the observation symbol probability matrix in state j given by $\mathbf{B} = \{b_{jk}\}$, where,

$$\{b_{jk}\} = P(v_k(t)|S_j(t)), 1 \leq j \leq N, 1 \leq k \leq M. \tag{2.2}$$

The symbols from the set V are observable but the states of the HMM from the set S are hidden. This corresponds to a discrete HMM, where the observations are discrete symbols selected from a finite alphabet V. In a continuous HMM, the states are characterized by continuous observation probability density functions, generally represented as a *mixture* of Gaussians of the form,

$$b_i(\mathbf{O}) = \sum_{k=1}^{M} c_{ik} N(\mathbf{O}, \mu_{ik}, \mathbf{\Sigma}_{ik}), 1 \leq i \leq N. \tag{2.3}$$

c_{ik} is the mixture weight for the k^{th} mixture in state i and $\mathbf{O}$ is the observation sequence and $N(\mathbf{O}, \mu_{ik}, \mathbf{\Sigma}_{ik})$ is the normal pdf with mean μ_{ik} and $\mathbf{\Sigma}_{ik}$ for the k^{th} mixture in state i.

Figure 2.1 shows an ergodic continuous HMM with three hidden states. Figure 2.2 shows a left-right HMM with five hidden states. This kind of HMM has been shown to be helpful in [24] in modeling faces in facial recognition problems. In this example, the states from left to right could correspond to the hair, forehead, eyes, nose and mouth respectively.

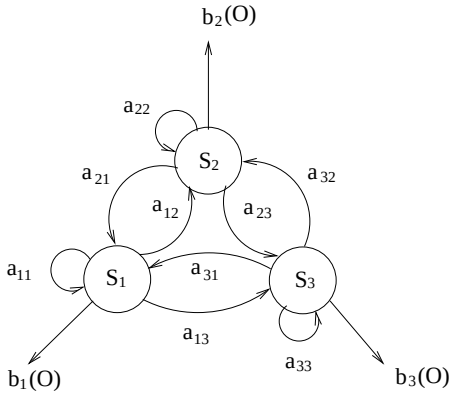


Figure 2.1: Three state continuous HMM.

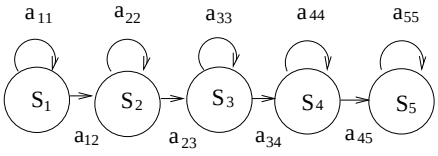


Figure 2.2: Five state Left-Right HMM.

- $\mathbf{\Pi}$, the initial state distribution given by $\mathbf{\Pi} = \{\pi_i\}$, where,

$$\{\pi_i\} = P[S_i(1)], 1 \leq i \leq N. \tag{2.4}$$

Hence, a HMM is completely characterized by $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$.

2.3 The Three Problems of HMM

Most applications involving HMMs concern the following three basic problems.

1. **The Evaluation Problem:** Given the observation sequence $\mathbf{O}$ and the model $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$, how do we compute $P(\mathbf{O}|\lambda)$, the probability of occurrence of the observation sequence?

2. **The Decoding Problem:** Given the observation sequence $\mathbf{O}$ and the model λ, how do we choose a state sequence such that $P(\mathbf{O}|\lambda)$ is maximized for that state sequence?

3. **The Learning Problem:** Given the observation sequence $\mathbf{O}$, how do we adjust the model parameters λ to maximize $P(\mathbf{O}|\lambda)$?

2.3.1 Solution to the Evaluation Problem:

The information in this section has been adapted from [21]. The probability that the HMM produces an observation sequence $\mathbf{O} = O_1, O_2, O_3, ..., O_T$ given the model parameters λ is

$$P(O|\lambda) = \sum_{r=1}^{r_{max}} P(\mathbf{O}|\mathbf{S}_r)P(\mathbf{S}_r). \tag{2.5}$$

where each r indicates a particular sequence $\mathbf{S}_r = \{S(1), S(2), S(3), ..., S(T)\}$ of T hidden states. If there are N hidden states in the HMM, the number of possible terms in (2.5) is $r_{max} = N^T$. Hence, the probability that the observation sequence is generated by the model is computed by taking every conceivable state sequence, calculating the probability that a given sequence produced $\mathbf{O}$ and summing them up. Now, the second term in (2.5) can be written as

$$P(\mathbf{S}_r) = \prod_{t=1}^{T} P(S(t)|S(t-1)). \tag{2.6}$$

which is the product of the state transition probabilities of the state sequence $\mathbf{S}_r$. Similarly, the first term in (2.5) can be written as

$$P(\mathbf{O}|\mathbf{S}_r) = \prod_{t=1}^{T} P(v(t)|S(t)). \tag{2.7}$$

which is the product of observation density probabilities given the state sequence $\mathbf{S}_r$.

We can clearly see that the computational complexity of this calculation is $\mathbf{O}(N^T T)$[1]. Duda et al. in [21] state that even for small values like $N = 10$ and $T = 20$, this requires approximately 10^{21} calculations which would take ages even for

[1]Here, $\mathbf{O}$ denotes the Big O notation for complexity.

a powerful computer. Fortunately, a computationally more efficient algorithm exists and is called the *Forward-Backward algorithm.*

2.3.1.1 Forward Algorithm

$P(\mathbf{O})$ is computed recursively since each term

$$P(v(t)|S(t))P(S(t)|S(t-1)) \tag{2.8}$$

involves only $v(t)$, $S(t)$ and $S(t-1)$. To this end we define a variable,

$$\alpha_j(t) = P(O_1, O_2, O_3, \ldots, O_t, S_j(t)|\lambda). \tag{2.9}$$

where O_t, $1 \leq t \leq T$ is the observation at time t and

$$\alpha_j(t) = \begin{cases} 0 & t = 0,\ j \neq initialstate \\ 1 & t = 0,\ j \neq initialstate \\ b_{jk}v(t)\sum_{i=1}^{N}\alpha_i(t-1)a_{ij} & \text{else} \end{cases} \tag{2.10}$$

where $b_{jk}v(t)$[2] means the observation symbol probability emitted at time t. Hence, $\alpha_j(t)$ represents the probability that the model is in state S_j at time t. The following algorithm is used to implement the forward procedure:

Algorithm 1 Forward Algorithm

1: $\alpha_j(1) = \pi_j b_{jk}v(1),\ 1 \leq j \leq N$

2: For $t = 1, 2, 3, \ldots, T$

$\alpha_j(t) = [\sum_{i=1}^{N}\alpha_i(t-1)a_{ij}]b_{jk}v(t)$

3: $P(\mathbf{O}|\lambda) = \sum_{j=1}^{N}\alpha_j(T)$

[2]Without loss of generality, this theory can be extended to a continuous HMM with continuous observation density functions.

Compared to the previous method, this procedure has a computational complexity of $\mathbf{O}(N^2T)$. Again, Duda et al. in [21] state that for $N = 10$, $T = 20$, only 200 calculations are needed as opposed to 10^{21} computations in the exhaustive approach of (2.5).

2.3.1.2 Backward Algorithm

In a similar fashion, we define a backward variable,

$$\beta_j(t) = P(O_{t+1}, O_{t+2}, O_{t+3}, \ldots, O_T | S_j(t), \lambda). \tag{2.11}$$

i.e. the probability of the observation sequence from $t+1$ to T given that the model is in state j at time t with parameters λ. The algorithm is as follows:

Algorithm 2 Backward Algorithm

1: $\beta_j(T) = 1,\ 1 \leq j \leq N$

2: For $t = T-1, T-2, T-3, \ldots, 1,\ 1 \leq j \leq N$

$\beta_j(t) = \sum_{i=1}^{N} \beta_j(t+1) a_{ij} b_{jk} v(t+1)$

3: $P(\mathbf{O}|\lambda) = \sum_{j=1}^{N} \pi_j b_{jk} v(1) \beta_j(T)$

2.3.2 Solution to the Decoding Problem

The solution to the decoding problem is the most probable sequence of states given the observation sequence $\mathbf{O}$ and the model parameters λ. In other words, the problem is to find the state sequence that maximizes $P(\mathbf{O}, \mathbf{S}|\lambda)$, where $\mathbf{S}$ is consistent with our previous notation for a sequence of states. A full exhaustive search through all the paths would yield $O(N^T T)$ calculations. A much faster algorithm was developed by Andrew J. Viterbi in 1967 in his seminal paper [25]. The algorithm is as follows:

Algorithm 3 Viterbi Algorithm

1: Initialize Path as an empty array

2: For $t = 1, 2, 3, \ldots, T$, $j = j + 1$

3: For $j = 1, 2, 3, \ldots, N$

$\alpha_j(t) = [\sum_{i=1}^{N} \alpha_i(t-1)a_{ij}]b_{jk}v(t)$

4: Compute $j^{'} = \text{argmax}_j\, \alpha_j(t)$

5: Append $S_{j'}$ to Path

6: Return Path

A similar algorithm can be developed by using instead the logarithm of probabilities. The complexity of this algorithm is also $\mathbf{O}(N^2T)$.

2.3.3 Solution to the Learning Problem

The primary goal here is to extract a set of model parameters, a_{ij} and b_{jk}, from the observation sequence starting from an initial estimate of these parameters. The forward-backward algorithm is revisited as it is an example of the generalized Expectation-Maximization (E-M) algorithm, also known as the Baum-Welch Re-estimation algorithm developed by Baum and his colleagues in [22] and [23]. The usual technique is to initialize the parameters with an estimate and then update the weights in an iterative fashion until a condition is satisfied. We have already defined $\alpha_j(t)$ and $\beta_i(t)$ as the forward and backward variables in section 2.3.1 on page 12. In addition to the above variables, we define another variable $\gamma_{ij}(t)$, the probability of jumping from $S_i(t-1)$ to $S_j(t)$ as follows:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(\mathbf{O}|\Theta)}. \tag{2.12}$$

where Θ denotes the model parameters to be estimated. The improved estimate for a_{ij} denoted by $\hat{a}_{ij}$ is given by:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)}. \tag{2.13}$$

In a similar fashion, an improved estimate $\hat{b}_{jk}$ for b_{jk} can be computed as follows:

$$\hat{b}_{jk} = \frac{\sum_{t=1, v(t)=v_k}^{T} \sum_{l} \gamma_{jl}(t)}{\sum_{t=1}^{T} \sum_{l} \gamma_{jl}(t)}. \tag{2.14}$$

The algorithm is as follows, where θ is the convergence condition under which the

Algorithm 4 Re-estimation Algorithm

1: Initialize a_{ij} and b_{jk}

2: $t = t + 1$

3: Compute $\hat{a(t)}$ using (2.13) and $\hat{b(t)}$ from (2.14)

4: $a_{ij} = \hat{(a_{ij})}$ and $b_{jk} = \hat{b_{jk}}$

5: Perform Steps 2-4 until $\text{argmax}_{i,j,k}[a_{ij}(t) - a_{ij}(t-1), b_{jk}(t) - b_{jk}(t-1)] < \theta$

6: Return $a_{ij}(t)$ and $b_{jk}(t)$

iterations stop. An alternative criterion that can be tested for convergence includes computation of the probability of the observations given the estimated parameters. The critical factors in the re-estimation algorithm are the initial estimates of a_{ij} and b_{jk}. Different initializations almost always lead to different results because the algorithm is not guaranteed to converge to a global minima.

Chapter 3

ALGORITHM DESCRIPTION

3.1 Introduction

Now that the reader has some idea about the power of Hidden Markov Models and how it is indeed possible to use them to model data, we are ready to develop an algorithm that uses an HMM to extract features peculiar to particular families of chirps. The goal of the feature extraction process is to represent the signal in reduced dimensionality by parameters that characterize the signal. The HMM detects and tracks the frequency of the chirp signals with time, but it does not directly produce a feature vector. Hence, an additional parameter extraction block is necessary: either polynomial curve fitting or a set of central moments in this work. These will be explained in detail in the sections that follow. The other vital processes involved in classification are *training* and *testing*. Since we are using a statistical pattern classifier with *supervised learning*, in the form of a Hidden Markov Model and a Bayesian Classifier, both of these need to be trained using the observations from known classes- i.e., a training set. After the classifiers have been trained and some model convergence has been achieved, chirps from a different testing set must be processed by the system to evaluate its performance. Result for the two different feature extraction processes used here are compared in Chapter 4. Figure 3.1 shows a block diagram of the classifier.

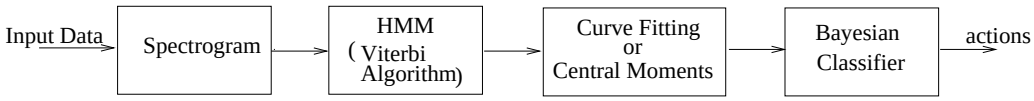


Figure 3.1: Paradigm for feature extraction, training and evaluation of the classifier.

3.2 Feature Extraction

Feature extraction forms the basis of any classification algorithm. It is also the most difficult part, and it is consequently very important to do a thorough analysis of the characteristics of each class in order to exploit its salient features. Feature extraction is the process of creating a compact representation of the information in the signal that can be used to effectively differentiate that signal from other similar signals, be they speech signals, images, gravitational waves or the calls of bats and whales. In this thesis, the signals are chirps. Once the feature vectors have been extracted, any appropriate classifier can be used to categorize the data. A good feature extraction method can simplify the design of the classifier, and often, time-frequency decompositions are an important preprocessing step. The highly non-stationary nature of chirps and the fact that they appear to be narrowband signals at any fixed time instant leads one to believe that the Short-Time Fourier Transform (STFT) is a natural starting point here. Following the STFT, the Hidden Markov Model portion of the feature extraction process detects and tracks the frequency of the chirp signal with time. The result of this stage is basically a sketch of the detected chirp. The parameter extraction block utilizes this sketch to create a low dimensional feature vector using either a polynomial curve fitting algorithm or a set of central moments. Each of these tools is discussed in detail in the following sections.

3.2.1 Short-Time Fourier Transform

Although the discrete-time Fourier Transform (DTFT) provides good frequency resolution, it cannot be used to characterize signals whose spectrum varies largely with time, due its poor time resolving properties. To overcome this problem, time-frequency distributions are used instead. The simplest of them is an extension to the DTFT, known as the Short-Time Fourier Transform (STFT). A very interesting compilation of the research in time-frequency analysis can be found in [12]. In addition to that, [26] provides a good overview for the two different approaches of the STFT we considered: (**1**) Fourier Transform and (**2**) Filter bank views. This chapter focusses only on the Fourier Transform viewpoint since, this is what we have implemented in our algorithm.

The data to be transformed is split up into frames, which usually overlap with each other. A DFT is performed on the data in each of these sections after multiplication with a window function. This can be expressed mathematically as follows:

$$X(n, w) = \sum_{m=-\infty}^{\infty} x[m]w[n-m]e^{-jwn} \tag{3.1}$$

where *w[n]* is referred to as the window function or analysis window which is non-zero over a small time interval. Typically, a Hamming or a Hann window is chosen for most of the applications. The following Figure 3.2 illustrates the operation of an STFT.

An important tool derived from the STFT used in many signal processing applications is the *spectrogram*. It is given by the magnitude-squared of the STFT: i.e., $|X(n, w)|^2$. The proposed algorithm uses the spectrogram to analyze the time-frequency content of the chirp as a preprocessing step prior to applying the HMM.

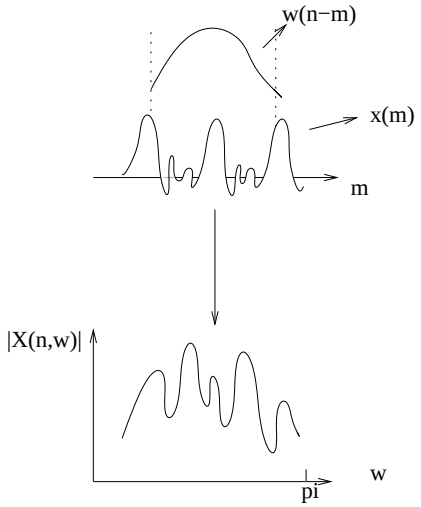


Figure 3.2: One frame of the Short-time Fourier Transform.

Using a short window yields a good time resolution but a poor frequency resolution and is called a *wideband spectrogram*, whereas, a long window gives a good frequency resolution but a poor time resolution and is referred to as a *narrowband spectrogram*. This fixed resolution property of an STFT is one of its limitations. Hence, researchers must often resort to using other time-frequency distributions such as the Wigner-Ville or the wavelet transform, which provide non-uniform tilings of the time-frequency space.

For our purposes, however, the fixed resolution of the STFT is not an issue, and we use it here in the form of the spectrogram for the remainder of the work. The spectrogram of a typical chirp signal is shown in Figure 4.1. Specifically, the spectrogram containing the time-frequency data is used to both train the HMM and detect/track the signal.

3.2.2 Viterbi Decoding

In Chapter 2, the Viterbi decoding algorithm was presented in detail. The main focus in this section is to explain how this algorithm can be used for chirp

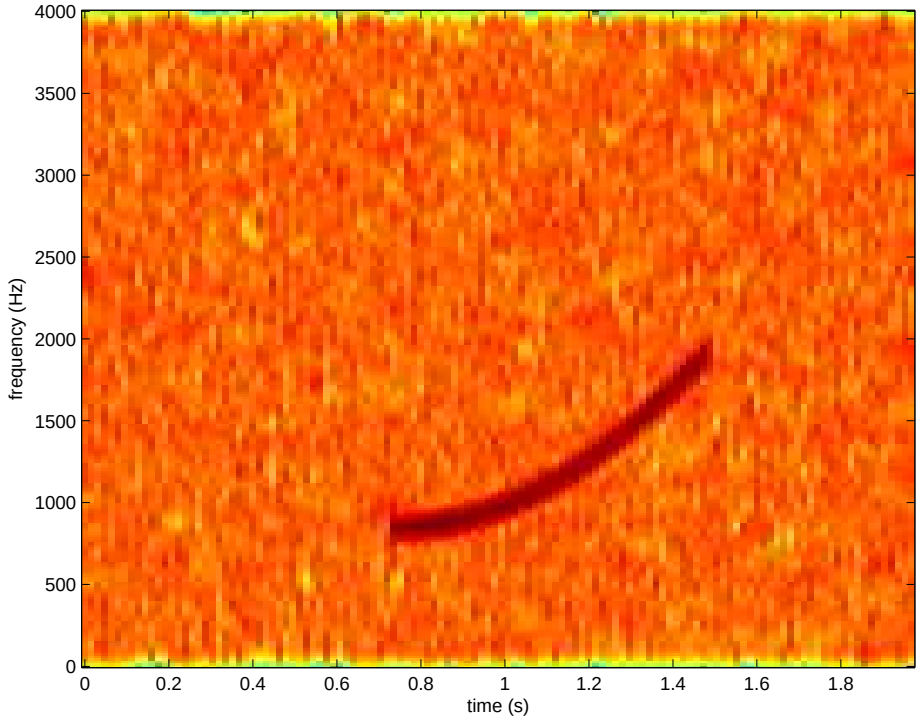


Figure 3.3: Spectrogram of a quadratic up-chirp.

detection/tracking. As we mentioned earlier, feature extraction is an integral part of both the training and testing processes. Feature extraction plays the same role in both these processes, namely representing the salient features of the data in a more compact manner. In Chapter 2 it was shown how the Viterbi algorithm solves one of the basic problems encountered in HMMs, specifically the Decoding Problem in which the state sequence is found that maximizes the probability of the observation sequence $\mathbf{O}$ given a particular model λ. The observation sequence in question here is the spectrogram of the data normalized to the unit-norm, making the energy in the sequence equal to 1. The optimal state sequence for this normalized data is computed using the Viterbi algorithm which, in effect, tracks the frequency of the signal as time progresses. The result of this operation is a state sequence or *path* indicating the presence of high energy in a corresponding time-frequency bin. Thus,

the 2-dimensional spectrogram is converted to a 1-dimensional vector while keeping the information necessary for classification, namely the time-frequency content. The following Figure 3.4 shows the result of this algorithm for the spectrogram of Figure 1.1.

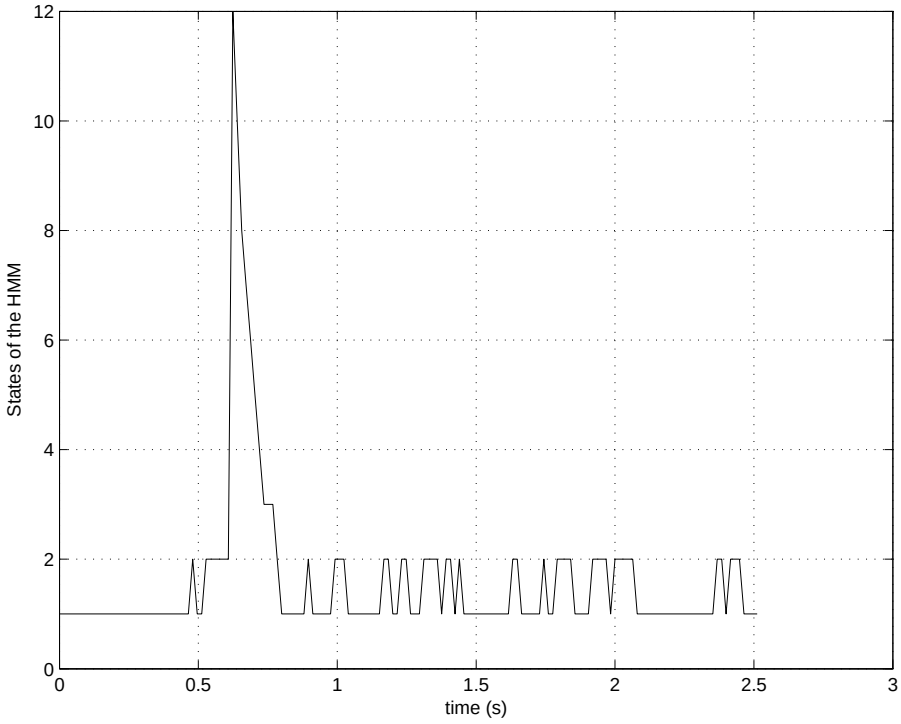


Figure 3.4: Viterbi decoded signal of Figure 1.1.

Clearly, the Viterbi algorithm tracks the frequency of the chirp as time progresses. It also attempts to track other narrrowband interferences after the chirp ends. Viterbi decoding is followed by a parameter extraction block using one of the two methods- polynomial curve fitting or central moments, as discussed in the next couple of sections.

3.2.3 Curve Fitting

Curve fitting is a well known problem in the mathematics world. Most of the measurements (observations) result in an ordered pair (x_i,y_i). An easy way to predict

the output at future times is to fit a curve through the observations such that the squared error of points between it and these observations is minimized. The key is then to find the coefficients of a polynomial corresponding to the curve that best fits the observations. Let

$$\hat{y} = \sum_{j=0}^{N} a_j x^j \tag{3.2}$$

and

$$MSE = \sum_{j=0}^{N} (y - a_j x^j)^2 \tag{3.3}$$

where MSE is the defined as the mean squared error of y with respect to $\hat{y}$.

The solution to the least squares problem described above is used to find a curve that best fits the path or the state sequence of the spectrogram of the data. Hence, in the case of chirp signal, this block yields coefficients of a polynomial that optimally fits the path of the chirp in the least squares sense. The polynomial coefficients thus form the feature vector. Figure 3.5 below shows the curve that best fits the observations in Figure 3.4. The polynomial approximation is marked with circles at intermittent points. The coefficients of this polynomial, a quadratic in this case, characterize the chirp signal of Figure 1.1 and yet can be represented by feature vectors containing only two elements. The constant term in the polynomial approximation is neglected here when forming a feature vector so that this vector will be independent of vertical frequency translation. A Bayesian Classifier is trained with these feature vectors belonging to a particular class. While testing the classifier, each of these feature vectors is also classified as belonging to each particular class. Note, that the degree of the polynomial can be increased to accommodate chirps with more complicated time-frequency relationships, like the power-law chirps, than linear or quadratic chirps.

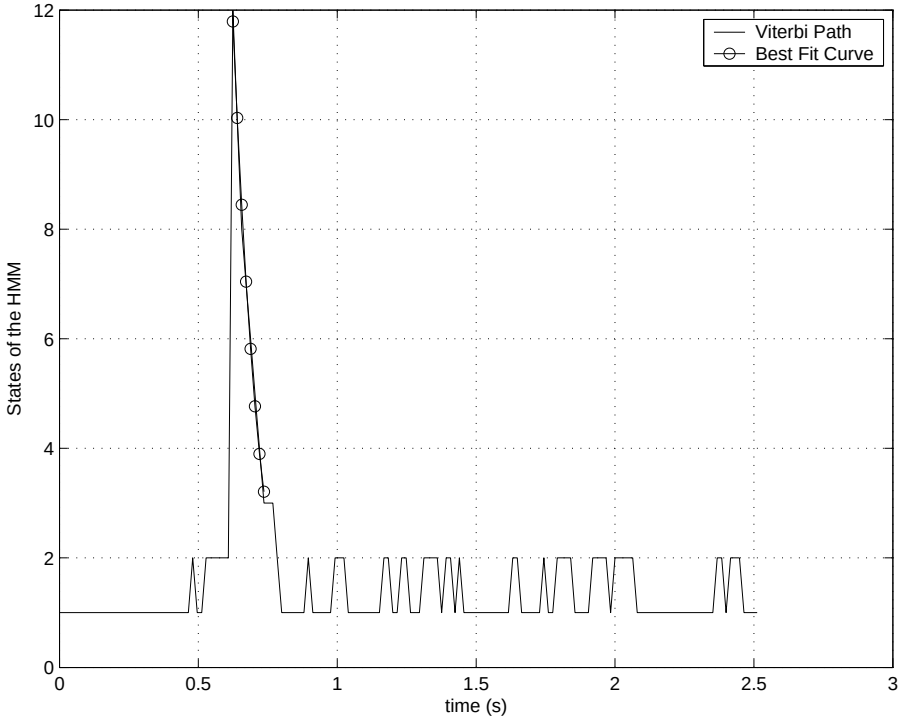


Figure 3.5: Best curve fit for Viterbi Path in Figure 3.4.

3.2.4 Central Moments

As an alternative to the above mentioned feature extraction approach, we also consider a different approach based on central moments. Here, a histogram of the probability of occurrence of the chirp in all the states in the Viterbi path is computed and the central moments, moments about the mean, of the distribution are calculated. These central moments characterize the time-frequency relationship of the chirp, independent of translation. The n^{th} central moment is given by

$$\mu_n = E\big((X - \mu)^n\big) \tag{3.4}$$

where E is the expectation operator defined as

$$E[x] = \int_{-\infty}^{\infty} x f(x) dx \tag{3.5}$$

which is also the mean (μ) of the Random Variable X and $f(x)$ is the probability density function (pdf) of X. In the discrete case,

$$E[x] = \sum_{n=-\infty}^{\infty} XP[X] \tag{3.6}$$

where X is now a discrete Random Variable and $P[X]$ is its probability mass function (PMF). The first central moment is obviously zero. The second central moment is also known as the *variance*, denoted by σ^2. The third and fourth central moments are usually normalized by σ^n. The normalized moments are dimensionless quantities. The third central moment is also known as *skewness* because it gives a measure of how tilted the distribution is with respect to a normal distribution. The fourth central moment, on the other hand, is known as *kurtosis* and is a measure of whether the data is peaked or flat near the mean compared to the normal the distribution.

Thus we use this property of moments to extract features peculiar to the distribution of the states of a chirp signal. Specifically, chirps with different characteristics are found to have different central moments as will be shown in the next chapter. Again, depending on whether the training or the testing stage follows, a Bayesian Classifier is either trained with these feature vectors belonging to a particular class or each of these feature vectors is classified as belonging to a particular class.

3.3 Training

We briefly mentioned the use of supervised learning for training HMM earlier in this chapter. This process consists of presenting the classifier with a set of signals whose classes are already known. This set of data is known as the *training set*. This section deals with training a Hidden Markov Model and Bayesian Classifier with

different training sets. Both these operations are explained in detail, including the type of data used in both the training sets.

3.3.1 Training Hidden Markov Models

In Chapter 2, we mentioned that an HMM is completely characterized by $\lambda = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$, where $\mathbf{A}$ corresponds to the state probability transition matrix, $\mathbf{B}$ is the observation probability matrix or distribution and $\mathbf{\Pi}$ is the initial state of the HMM. With respect to our problem and following the methodology of [16], the states in the HMM correspond to frequency bins of the Short-Time Fourier Transform of the original data. Specifically, sets of contiguous frequency bins correspond to a particular state in the HMM. The initial estimate of the matrix $\mathbf{A}$ is formed by generating N N-dimensional random variables having zero mean, unit variance Gaussian distributions, where N, consistent with the notation introduced in Chapter 2, is the number of states in HMM. Observation density functions are modeled as M Gaussian mixtures of dimensions equal to the number of frequency bins. In essence, there are N mean vectors and N covariance matrices, one pair for each of the N states. The initial estimates of the mean vectors and covariances matrices are obtained as a result of passing the training data through the K-means algorithm. The data is comprised of sinusoidal signals with frequencies corresponding to a particular state of the HMM. Figure 3.6 shows the spectrogram of the training signal for the third state of the HMM.

The sum total of states span the entire Nyquist frequency range. The initial state probability $\mathbf{\Pi}$ is estimated to be equally likely, with the chirp signal assumed to start any state. The transformed data (spectrograms) with the appropriate initializations are presented to the Baum-Welch re-estimation algorithm. When the

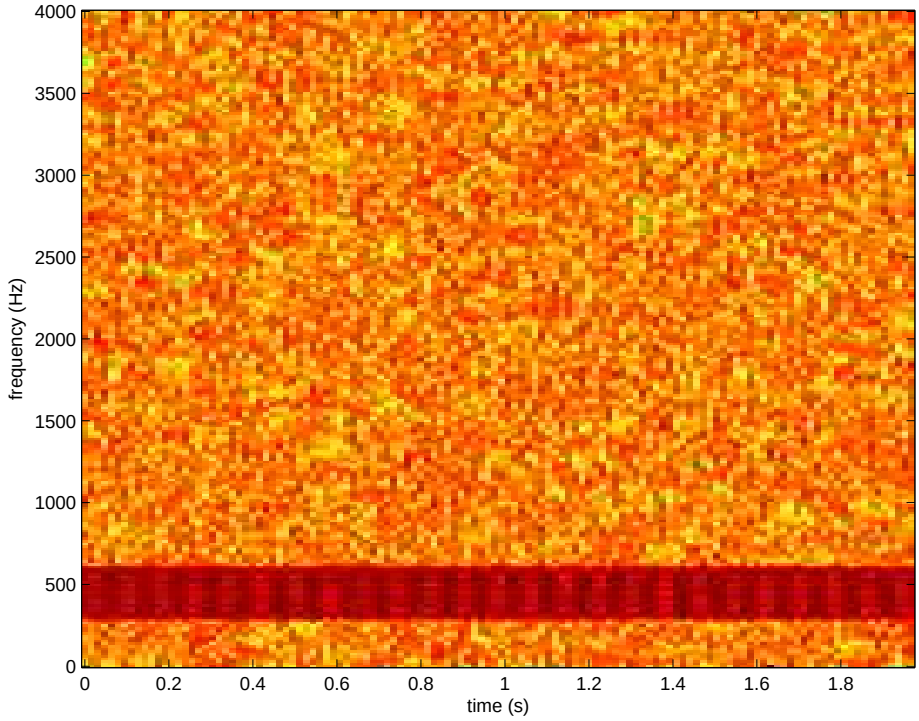


Figure 3.6: Spectrogram of training signal corresponding to state 3 of HMM.

convergence condition is satisfied, the iterations are stopped and better estimates of the state transition probability matrix **A**, mean vectors and covariance matrices of the Gaussian mixtures are obtained and thus, the Hidden Markov Model is created that represents the source to an extent determined by the data used to train the model. Figure 3.7 shows steps taken to train the HMM. The model parameters obtained at the end of this training process are henceforth used for all the other stages in the classifier.

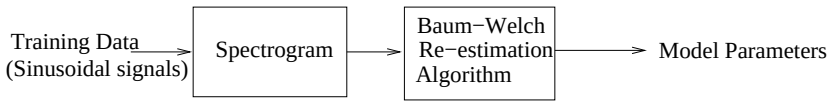


Figure 3.7: Training the HMM.

3.3.2 Training the Bayesian Classifiers

A Bayesian classifier is known to minimize the overall risk in making a decision or, in other words minimizes the average probability of error. Let us familiarize the reader with the terms used in the Bayesian classification approach as followed by the authors of [21]. The Bayes formula in probability theory is given by

$$P(w_j|\mathbf{x}) = \frac{p(\mathbf{x}|w_j)P(w_j)}{p(\mathbf{x})}. \tag{3.7}$$

$P(w_j|\mathbf{x})$, the probability of the state being w_j given the observation $\mathbf{x}$, is also known as the *a posteriori* probability. $p(\mathbf{x}|w_j)$, the probability of the observation $\mathbf{x}$ being generated from a certain state w_j is known as the *likelihood* while $p(w_j)$ is known as the *prior* probability and $p(\mathbf{x}) = \sum_{j=1}^{N} p(\mathbf{x}|w_j)P(w_j)$ is called the *evidence*. The product of the likelihood and the prior is considered to be the most important factor in deciding to which class a particular observation is most likely to belong. The evidence is usually constant for a given set of classes. Moreover, if the all the classes are equally probable, the *a posteriori* probability is dependent only on the likelihood.

3.3.2.1 Discriminant Functions

Discriminant functions are one of the most popular ways of representing classifiers, aiding in making decisions to reduce the average probability of error. As in [21], we denote the discriminant functions as $g_j(\mathbf{x})$ and we decide that an observation vector belongs to class w_i if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}), i \neq j. \tag{3.8}$$

Figure 3.8 shows how discriminant functions are used in a pattern classification system. All the discriminant functions are computed for the input feature vector and then a decision is made based on which discriminant function has the largest value.

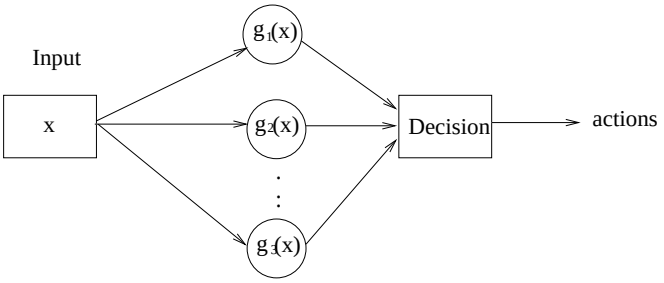


Figure 3.8: Block diagram of a typical pattern classification system.

The discriminant functions divide the space into *decision regions* which are separated by *decision boundaries*. If the discriminant functions turn out to be linear, then the decision regions are hyperplanes which are much easier to analyze than other complicated surfaces. To make computation easier, sometimes a monotonically increasing function of the discriminant function is used instead of the actual function. An example of such a function might be

$$g_i(\mathbf{x}) = \ln p(x|w_i) + \ln P(w_i). \tag{3.9}$$

3.3.2.2 Training

The data which trains the Bayesian Classifier is the result of the feature extraction process. In this thesis we assume that the feature vectors are distributed with a Gaussian pdf having arbitrary mean vectors and covariance matrices for each of the classes. The discriminant function defined in (3.9) is used here. Figure 3.9 shows the complete process involved in training the Bayesian Classifier. Hence, if the class conditional probability densities are multi-variate Gaussians of the form $p(\mathbf{x}|w_j) = N(\mu_i, \mathbf{\Sigma}_i)$, (3.9) reduces to

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \mu_i)^t(\mathbf{\Sigma}^{-1})(\mathbf{x} - \mu_i) - \frac{1}{2}\ln|\mathbf{\Sigma}_i| + \ln P(w_i). \tag{3.10}$$

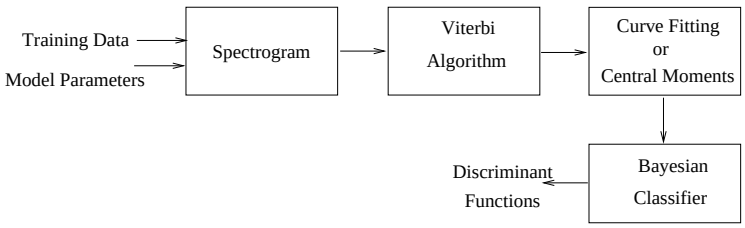


Figure 3.9: Training the Bayesian Classifier.

after neglecting a $(-\frac{M}{2}\ln 2\pi)$ term. For each class, the sample mean and the covariance matrix are estimated from the feature vectors pertaining to that class. This is a direct application of *maximum-likelihood estimation* to the unknown parameters μ_i and $\mathbf{\Sigma}_i$ of a multi-variate Gaussian probability density function and the estimates are formed by

$$\hat{\mu} = \frac{1}{n}\Sigma_{k=1}^{n}\mathbf{x}_k. \tag{3.11}$$

and

$$\hat{\mu} = \frac{1}{n}\Sigma_{k=1}^{n}(\mathbf{x} - \mu_\mathbf{i})^t(\mathbf{x} - \mu_i). \tag{3.12}$$

The μ_i and $\mathbf{\Sigma}_i$ are used to compute the discriminant functions for each class. These discriminant functions are used in the classification process during actual operation of the algorithm. This will be explained in detail in the next section.

3.4 Operation and Testing

Once the classifier has been trained as described in the previous sections, those signals in the database not used to train the classifier are used to test the classifier. Such testing is required to evaluate the performance of any pattern classification system. It is only through testing that the performance of the classifier can be measured and compared with its contemporaries. In actual operation on unknown signals, the classifier operates in exactly the same way as it does for testing, i.e., all model parameters are fixed. Specifically, the optimal sequence of states extracted by the HMM

decoder from the spectrogram of the original data is first computed and then either polynomial curve-fitting or central moments type feature extraction is applied. Each feature vector is next classified as belonging to one of the classes using the trained Bayesian Classifier as indicated in Figure 3.9. The performance of the classifier is quantified by the percentage of correct classifications versus the Signal to Noise Ratio (SNR). The next chapter fills in the missing pieces: it details the performance of the classifier for the two different feature extraction methods.

Chapter 4

EXPERIMENTS AND RESULTS

4.1 Introduction

This chapter presents the experimental results for the HMM-based chirp classification algorithm described in the previous sections. We consider here the selection of parameters and the effect of varying them on both synthetic data generated on a computer and real data recorded by a **F**ast on-**O**rbit **R**ecording of **T**ransient **E**ffects (FORTE) satellite. The real data consists of *broadband VHF signals*, some of them generated by an electromagnetic pulse generator known as **L**os **A**lamos **P**ortable **P**ulser (LAPP) at Los Alamos, New Mexico and others being recorded *lightning emissions*. The signals of interest in this database vary from chirps to impulses and encompass a variety of different chirp rates and types (e.g. Linear and Quadratic). The goal of this research then is to classify the different chirps present in the database, without knowledge of the number of classes present using the algorithm described in Chapter 3. In particular, the classification results for both of the feature extraction methods discussed in Chapter 3 are compared.

4.2 Selection of Parameters

Parameters in the algorithm have to carefully selected because small differences may lead to drastically different results. Also, a few functions like the re-estimation algorithm require proper initialization of its parameters for the algorithm to converge

with meaningful results. Keeping this in mind, we define the following parameters that are of vital importance in achieving good classification performance.

1. *Spectrogram*: The spectrogram or the Short-time Fourier Transform was mentioned briefly in the previous chapter as were also its shortcomings in terms of time-frequency resolution. The length of the *window* used in the spectrogram plays an important role in localizing the energy of the signal in the time-frequency bins. A large window captures the frequencies present with a good resolution with poor time localization while a small window has the opposite trade-off. Since chirps are non-stationary, a good spectrogram is needed where *leakage* of energy from a time-frequency bin into its neighboring frequency bins is avoided. For this reason, a Hamming window length (N_w) of 256 is selected with an *overlap* (L) of 128 samples since a longer window causes the frequencies to spill over into the neighboring bins. A 256 length FFT is thus performed on each set of windowed input samples. The chirps in the synthetic data set have been generated with sampling rate of $8KHz$ whereas the FORTE data is already sampled and thus represented in the normalized frequency domain. Since all of the processing is performed in the digital frequency realm, the underlying sampling rate is immaterial as long as it meets the Nyquist criterion and there is consequently no loss in generality in assuming that our data is sampled at $8KHz$.

2. *Hidden Markov Models*: The two sets of parameters associated with HMMs are those concerned with training the HMM. The parameters obtained after training the HMM are used by the Viterbi Algorithm. Although the window

length is fixed at 256 so that the spill over of frequencies is minimized, some frequency components of chirps with very high chirp rates invariably leak into neighboring bins. This situation can be addressed by dedicating a set of contiguous frequencies to each state of the HMM. The *number of states* (N) used in the HMM for this research is 14 with a null state (labeled *State* 1) also being included. The null state indicates the absence of a signal. States 2 through 14 span the entire normalized frequency spectrum from 0 to π radians/sample. To simplify the model, the underlying distribution of *observations* given a state is assumed to be a *Gaussian* random variable of dimension equal to the number of frequency bins. Hence, only one Gaussian *mixture* ($M = 1$) is considered. The mean vector and covariance matrix for each state is initialized from the training data using a K-means clustering algorithm, and the number of clusters is equal to the number of states in the HMM because the observations in each state are distributed as a Gaussian random variable. The *training data*, as described in chapter 2, consists of 13 signals, each signal containing frequencies for a particular state, and a null state containing white Gaussian noise. The data is presented to the model in the form of linear up and down chirps with different chirp rates. The *transition probability matrix* ($\mathbf{A}$) is initialized with random entries from a zero mean, unit variance gaussian distribution. This has been found to be sufficient since the structure of the training data ultimately defines the final transition matrix. The HMM is trained using the Baum-Welch re-estimation algorithm described in Chapter 2. The resulting model parameters are then stored for use in the classifier.

3. Feature Extraction: Detection and frequency tracking of the signal is performed using the Viterbi Algorithm. Once it is done, either polynomial coefficients or central moments of the track are computed.

    - *Curve Fitting*: An important parameter in this method is the number of *coefficients* used in the polynomial which best describes the track in the least squares sense. Two coefficients have been used in this research, mainly because the chirps generated synthetically are either linear or quadratic. The theory can, however, be easily extended to chirps that have a higher order time-frequency relationship.

    - *Central Moments*: The number of *bins* determining the distribution of the track in terms of the states of the HMM has been chosen here to equal the number of states in the model obviously- 13, in this case. The other parameter of importance is the number of *moments* to be computed. The first central moment is is obviously zero. Hence, the second, third, fourth and fifth central moments have been used to form the feature vector. For a more complicated distribution, higher order moments can also be computed and used in the feature vector.

4. *Bayesian Classifier*: The *mean* and *covariance matrix* for each class in the Bayesian Classifier are estimated from the feature space of the training data. For the real data, where the number of classes are unknown, the mean and covariance matrix for each cluster is determined using a K-means Algorithm. The centers of the clusters are initially picked at random from the data points themselves.

4.3 Simulation Results for synthetically generated data

The synthetically generated data on the computer consists of isolated chirps with background additive zero mean Gaussian random noise whose variance is varied to obtain results for different Signal-to-Noise ratios (SNR). Note, however, that the sinusoidal signals used to train the HMM have been generated with a constant SNR of $10dB$. Ten different classes of chirps are considered for the synthetic data analysis, and each class is different from the others in either the chirp rate or the type of chirp (e.g. linear or quadratic). While the algorithm is sensitive to the length of the chirp, we note that in natural and manmade phenomenon that generate chirps, the variation in the length is generally small. In other words, with respect to our algorithm, chirps with the same specifications but different lengths are considered to be members of different classes. A typical chirp from this database is shown in Figure 4.1. Note

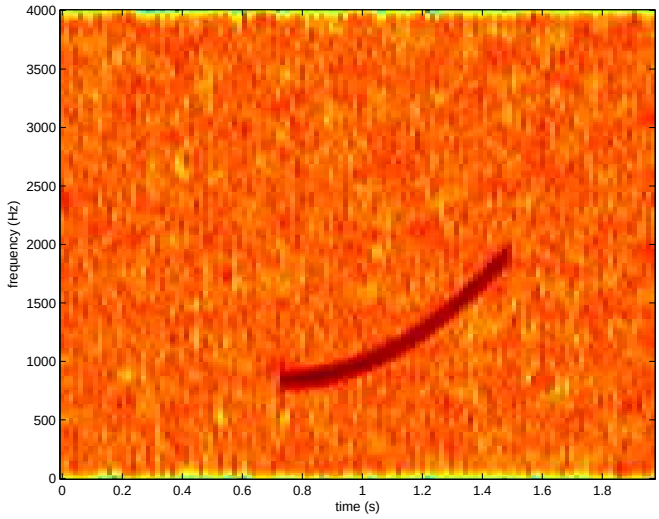


Figure 4.1: Spectrogram of a chirp in synthetic database.

that all the chirps used in this database are up chirps, where frequency increases as time increases. At every time step, the chirp is convolved with a gaussian random variable having zero mean and a random variance in order to spread the energy across

a number of frequency bins. Table 4.1 shows the specifications of the different classes of synthesized chirps used in these experiments. Note that, *classes 1* through *5*

Table 4.1: Specifications of classes for training data.

Class	***Type***	***Change in frequency (Hz)***	***Chirp Rate (s)***	***Length (s)***
1	Quadratic	2000	1	1.5
2	Quadratic	2000	2	1.5
3	Quadratic	1500	0.7	1.5
4	Quadratic	1450	0.5	1.5
5	Quadratic	1500	0.09	0.2
6	Linear	2000	1	1.5
7	Linear	2000	2	1.5
8	Linear	2000	0.7	1.5
9	Linear	1450	0.5	1.5
10	Linear	2000	0.09	0.2

correspond to quadratic chirps (frequency is a quadratic function of time) whereas, *classes 6* through *10* have the same specifications as the former but are linear chirps. The 2-dimensional scatter plot of the feature vectors used in the polynomial method for the training data with an SNR of $10dB$ is shown in Figure 4.2. The figure shows distinct clusters for the different classes. Also noticeable is the distinction between

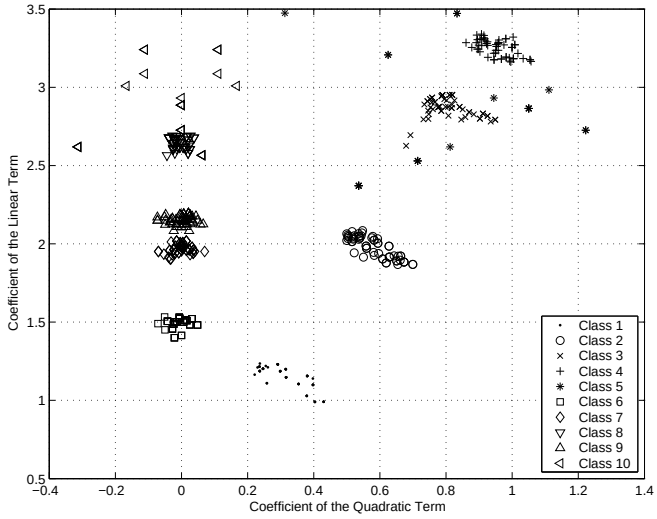


Figure 4.2: Scatter plot of training data using curve fitting method.

the linear and quadratic classes. The coefficient of the quadratic term for the linear chirps is almost zero whereas it increases as the chirp rate increases for *classes 1* through *5*. Similarly, the coefficient of the linear term increases as the chirp rate increases for linear chirps. Hence, this feature extraction method captures the type of the chirp as well as the chirp rate accurately which is the result we want to achieve. The proposed feature extraction method scatters the feature vectors corresponding to signals from *classes 5* and *10* more broadly than for the other classes and therefore the classification performance is likely to be worse. Due to the very high chirp rates in *classes 5* and *10*, the frequency spill over at each time step is large, causing the Viterbi algorithm to fail to track the frequency of the chirp correctly. This problem too can be overcome by training the HMM properly as will be explained in Chapter 5. The resulting feature vectors are used to design the Bayesian Classifier by evaluating the discriminant functions for each class as discussed in Chapter 3. While the feature space for central moments method is 4-dimensional and thus cannot be visualized, we show that similar classification accuracy was achieved.

The classification as described in Chapter 3 is performed on a set of signals held out of the design process, since 100% classification accuracy is achieved when the training data set is classified. Table 4.2 shows the specifications of signals used to test the classifier. The signals start and end at different times than those used for training,

Table 4.2: Specifications of classes for test data.

Class	***Type***	***Change in frequency (Hz)***	***Chirp Rate (s)***	***Length (s)***
1	Quadratic	2000	1	1.5
2	Quadratic	2000	2	1.5
3	Quadratic	1500	0.7	1.5
4	Quadratic	1450	0.5	1.5
5	Quadratic	1500	0.09	0.2
6	Linear	2000	1	1.5
7	Linear	2000	2	1.5
8	Linear	2000	0.7	1.5
9	Linear	1450	0.5	1.5
10	Linear	2000	0.09	0.2

and their additive Gaussian noise is independently generated. Each signal from the table is classified as belonging to one of the ten classes enumerated in Table 4.1. Figure 4.3 shows the 2-dimensional feature space of the test signals to be classified

using the same polynomial method similar as in Figure 4.2. Clusters similar to

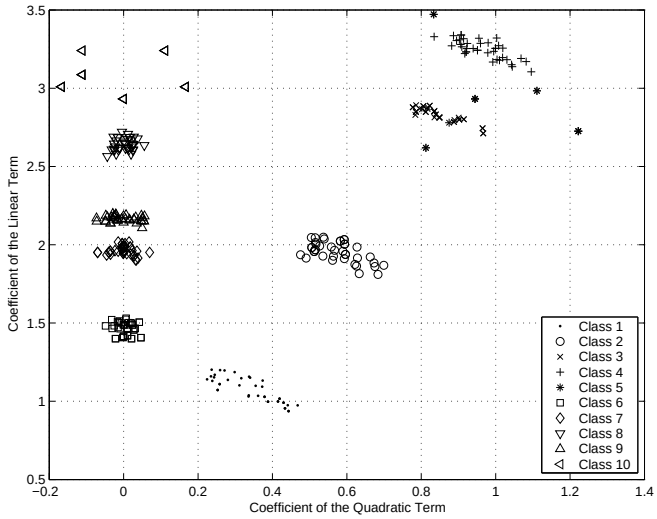


Figure 4.3: Scatter plot of test data using curve fitting method.

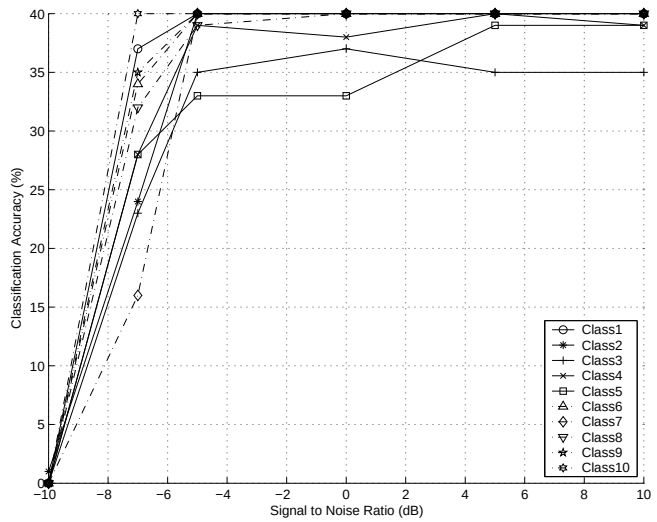


Figure 4.4: Performance of classifier with curve fitting.

those found in Figure 4.2 are observed. Figure 4.4 and 4.5 show the degradation in the percentage of correct classification as SNR decreases for both of the feature extraction methods. At low SNRs, the Viterbi algorithm is overpowered by noise and does not track the chirp accurately. Consequently, the HMM has to be re-trained at lower SNRs to accommodate for the drop in classification. The fact that results are

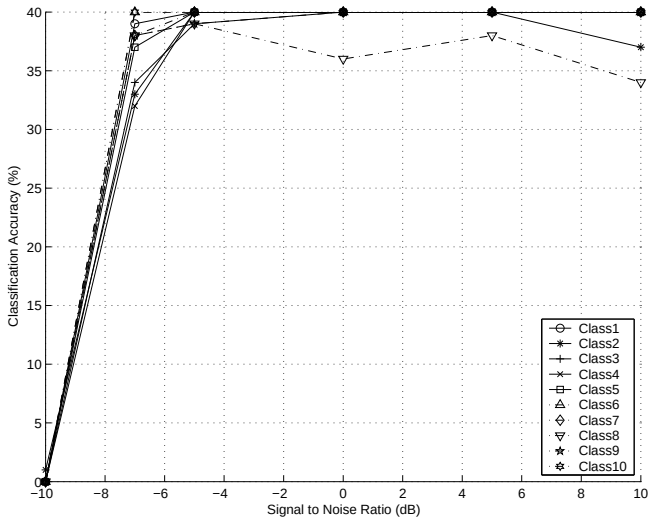


Figure 4.5: Performance of classifier with central moments.

similar for both of the feature extraction methods indicates that the central moments method also effectively clusters samples in its 4-dimensional space. Table 4.3 and 4.4 show the *confusion matrices* indicating the misclassification of signals of one class being classified as belonging to others for a $10dB$ SNR. The columns correspond to the true class whereas the rows correspond to the predicted class. For example, in column 2 of Table 4.3, 92.5% of the signals from *class 2* have been labeled correctly whereas 7.5% of the signals from *class 2* have been classified as *class 5*. The correct classifications are found on the diagonal of the confusion matrix, and we note that the majority of the classes have been labeled correctly. Some of the signals in both the methods have been misclassified as *classes 5* or *10*. As we saw in the scatter plots, the signals belonging to these classes tend to be more widely scattered around in the feature space and consequently more prone to misclassification. The results are similar for classification with central moments as shown in Table 4.4. Hence, this algorithm using either of the proposed feature extraction methods appears to be very effective in classifying synthetically generated chirp signals.

Table 4.3: Confusion Matrix for Curve Fitting: Synthetic Data.

Class	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1	100	0	0	0	0	0	0	0	0	0
2	0	92.5	0	0	0	0	0	0	0	0
3	0	0	100	0	0	0	0	0	0	0
4	0	0	0	100	0	0	0	0	0	0
5	0	7.5	0	0	100	0	0	0	0	0
6	0	0	0	0	0	100	0	0	0	0
7	0	0	0	0	0	0	100	0	0	0
8	0	0	0	0	0	0	0	85	0	0
9	0	0	0	0	0	0	0	0	100	0
10	0	0	0	0	0	0	0	15	0	100

4.4 Results for FORTE data

The real data consists of around 870 signals recorded by the FORTE satellite- [1] and [2]. The data is divided into LAPP signals generated by an electronic pulse generator (LAPP) and natural lightning discharge signals. These signals behave from anything like chirps to impulses, depending on the level of ionization they have undergone. Hence, the number of classes of chirps are virtually unknown. Figure 4.6 shows a signal from this database in which, prominent narrowband interference signals can be discerned amidst a large amount of background noise. The database contains signals ranging from those with no signals of interest (SOI), chirps in this case, to those consisting of high energy chirps. To improve the performance, a de-noising algorithm

Table 4.4: Confusion Matrix for Central Moments: Synthetic Data.

Class	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
1	100	0	0	0	0	0	0	0	0	0
2	0	100	0	0	0	0	0	0	0	0
3	0	0	87.5	0	2.5	0	0	0	0	0
4	0	0	0	97.5	0	0	0	0	0	0
5	0	0	12.5	2.5	97.5	0	0	0	0	0
6	0	0	0	0	0	100	0	0	0	0
7	0	0	0	0	0	0	100	0	0	0
8	0	0	0	0	0	0	0	100	0	0
9	0	0	0	0	0	0	0	0	100	0
10	0	0	0	0	0	0	0	0	0	100

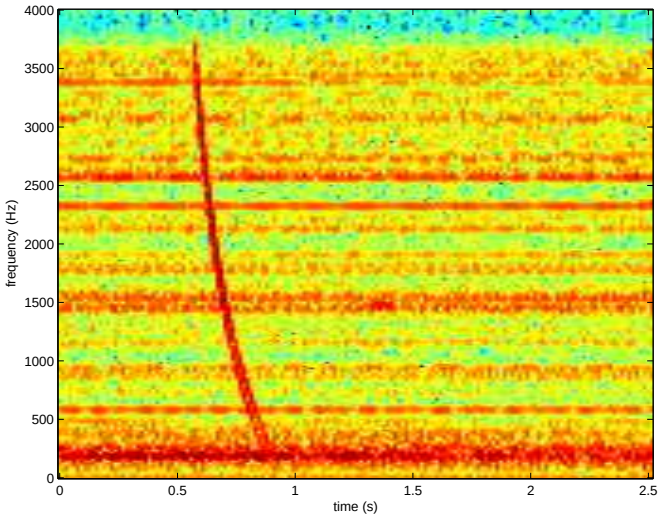


Figure 4.6: Spectrogram of a signal from the FORTE database.

is iterated three times on the raw data as a pre-processing step. Specifically, an FFT is performed over the entire length of the sequence and the coefficients corresponding

to the highest frequencies are hard thresholded to zero. The variance of the background noise is estimated by assuming it to be an additive white Gaussian noise and calculating the sample variance. The HMM is next trained with signals having an SNR corresponding to the estimated SNR of the signals in the database, roughly $0dB$. Note, however, that the SNRs of the signals vary a great deal, so extraction of the chirp signal from the background and narrowband interferences is thus a very difficult task. The algorithm depends highly on the precise extraction of the chirp for accurate predictable results.

Figure 4.7 and 4.8(b) show the result of the Viterbi Algorithm using the polynomial feature extraction method for two signals. Figure 4.7 shows the Viterbi path for the signal whose spectrogram is shown in Figure 4.6.

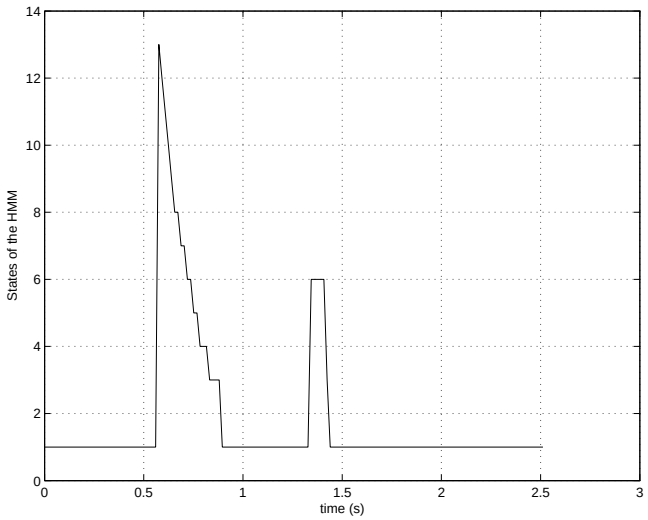
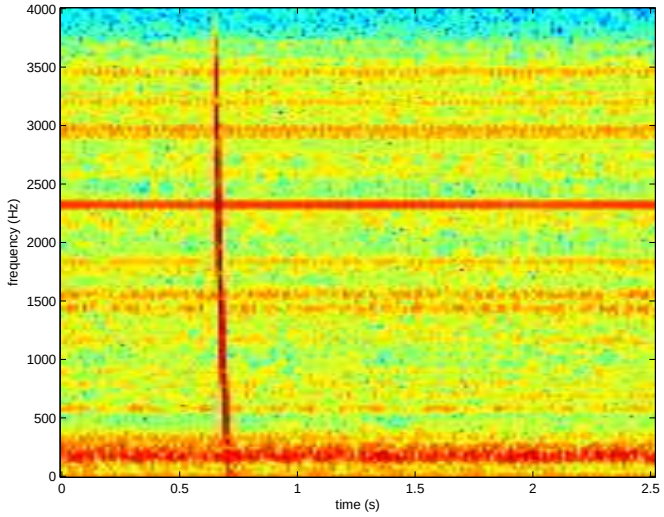
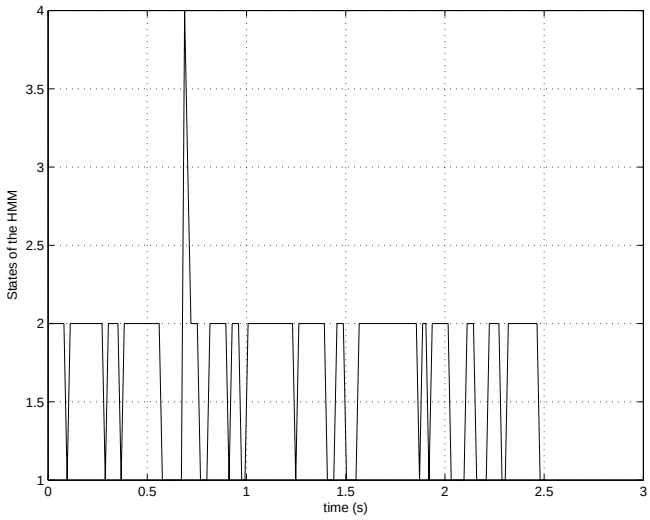


Figure 4.7: Viterbi decoding of a high-energy chirp.

It can be easily seen that the first signal as indicated by Figure 4.6 has a prominent chirp signal, superimposed in time and frequency with other narrowband interferences. The Viterbi algorithm tracks the chirp well although it also makes a spurious detection elsewhere in the interval. The spectrogram of the second signal in



(a)



(b)

Figure 4.8: (a) Spectrogram of the a high rate chirp signal and (b) corresponding Viterbi path.

Figure 4.8(a) shows a very high rate chirp which is not tracked well by the Hidden Markov Model. This occurs because the HMM finds that the chirp's energy spills over to the neighboring frequency bins, making it appear to be almost an impulse. This is one of the inherent difficulties encountered when working with the database. Because of these difficulties, 50 signals have been hand selected and used to train the

classifier while another 48 from the database are used to test it's performance. Since the number of classes is not known *a priori*, a K-means clustering is performed on the feature space after feature extraction. The algorithm's performance is compared for two and three clusters, respectively. Since the classes cannot be clearly distinguished by human observation, a quantitative analysis is difficult. However, when two clusters are selected, chirps with a very high chirp rate, almost impulses, can be distinguished from chirps with a slightly lower rates. Figure 4.9 shows the scatter plot of the training data set for two clusters in the K-means algorithm. Most of the

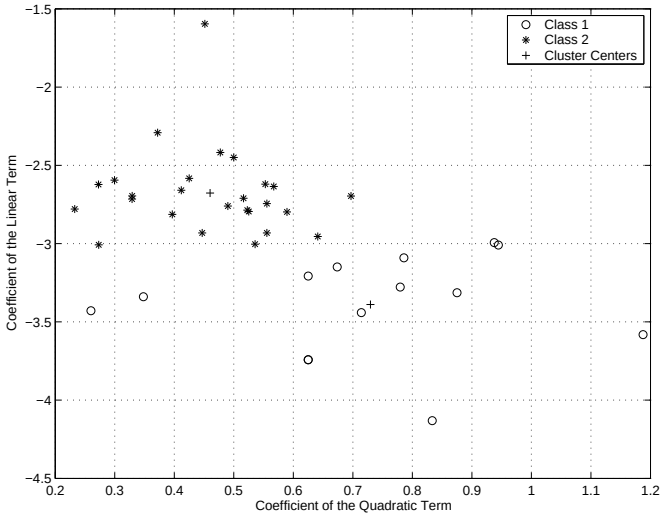


Figure 4.9: Scatter Plot of the training database with 2 clusters.

chirps with a high chirp rate have been labeled *Class 1* and the chirps with a lower chirp rate, *Class 2*. Note that the polynomial coefficients indicate that all of these chirps are quadratic in nature. A physical correspondence between the chirps in the training data set and their clusters enabled us to determine the characteristics of a particular class. Specifically, classes were labeled as characterizing chirps with high rates (*Class 1*) and those with lower rates (*Class 2*). Table 4.5 shows the confusion matrix for classification of the 48 test signals into one of the two classes using curve

fitting method. The confusion matrix shows that 75.68% of the signals in *Class 1*

Table 4.5: Confusion Matrix for Curve Fitting: 2 Clusters.

Class	***1***	***2***
1	75.68	27.27
2	24.32	72.73

have been classified correctly. It should be noted that the length of the chirp needs to be accurately extracted and that incorrect extraction leads to a change in the polynomial coefficients. Moreover, for chirps that differ by a small chirp rate, this can lead to erroneous classification. A similar confusion matrix for the moments method is illustrated in Table 4.6. Although similar, the results obtained using the central mo-

Table 4.6: Confusion Matrix for Central Moments: 2 Clusters.

Class	***1***	***2***
1	67.57	36.36
2	32.43	63.64

ments feature extraction process are degraded relative to the performance of the curve fitting method. The central moments method appear to be less consistent because the classes were defined by observing the scatter plot for the curve fitting method. Two of the cluster centers for the three cluster case have been initialized with points close to the centers in the previous case. The scatter plot with the cluster centers superimposed on the training set data using the curve fitting method is shown in Figure 4.10. By comparing Figure 4.10 with Figure 4.9, we observe that the new class, *Class 2* in

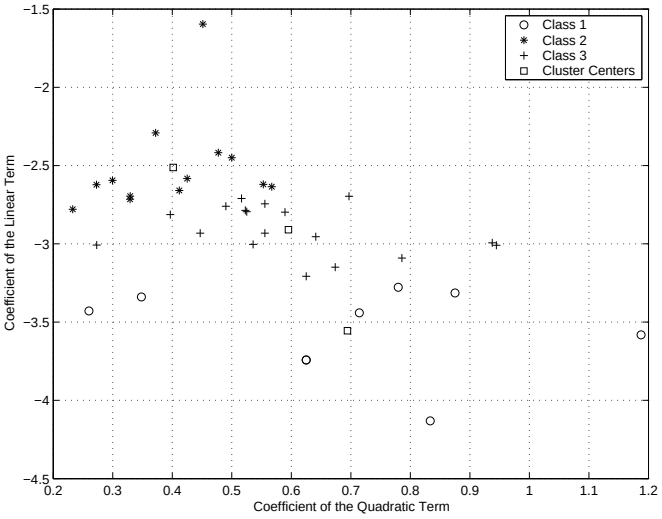


Figure 4.10: Scatter Plot of the training database with 3 cluster centers.

this case, has encroached into the prior region of *Class 3* and that *Class 3* has in turn, encroached into the region, formerly a part of *Class 1*. The fuzzy boundary between the different families of chirps present in the database has become even more obscure. The third class contains signals very similar to both of the previous classes, indicating that it does not correspond to chirps with consistent characteristics. Since a physical relationship between the chirps and their corresponding clusters cannot be discerned, results in the form of confusion matrices are not tabulated. Classification shows that there are two dominating classes to which most of the signals have been classified, *Class 1* and *Class 3*. Hence, proceeding further with the K-means clustering algorithm by increasing the number of clusters does not appear to be warranted. Using two classes gives us insight into the classification process of chirps, separating high rate chirps from lower rate chirps. The results for the three class case indicate that the rates of the chirps in the database are too close to one another for the algorithm to make a good comparison using either of the feature extraction methods.

The following chapter concludes the findings of this research and discusses some scope for future work.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusions

The research reported thus far on the classification of chirp signals has been surveyed in Chapter 1. According to this study, most of the classifiers have been restricted to classifying chirp signals that either fall into a very few classes or very few parameters differentiate these classes. This thesis proposes a classifier capable of identifying chirp signals that has any time-frequency relationship. Two different feature extraction methods are proposed to support this claim. A polynomial curve fitting method models the time-frequency relationship of the chirp signal by modeling the discrete Viterbi path of the spectrogram of the chirp as a polynomial, whereas, the central moments method models the distribution of the Viterbi path. The classifier is evaluated on two data sets, ten classes of synthetically generated linear and quadratic chirps and naturally occurring chirps in the form of lightning discharges as recorded by the FORTE satellite. Chapter 4 discusses these results and shows that the classifier works well in identifying the ten different classes of synthetically generated chirps. For broad classes of chirps, the performance shows consistent results for both the feature extraction methods as separable clusters are formed in the feature space under the assumption that chirps with different lengths are labeled as different classes. Different lengths for the chirp signals signify different lengths for the Viterbi path, which in turn, affects the polynomial curve fitting and central moments methods. This can be

seen as a shortcoming of the classifier. However, chirps occurring naturally or as a result of manmade phenomena vary very little, so length generally should be used as a discriminating parameter. Hence, the ability to extract features is highly dependent on the length of the chirp. In the presence of a high amount of noise (low SNR), the classifier shows a degradation in performance. One way of partially overcoming this problem is by training the Hidden Markov Model with data corresponding to that SNR. The results also show that for chirp signals with very high rates, classification using both the feature extraction methods deteriorates. This difficulty can be at least partly addressed by training the HMM in such a way the state transition matrix characterizes high rate chirps.

The FORTE data set poses a major challenge in identifying different classes of chirps amidst varying background noise and narrowband interferences. The chirp signals in the database vary from impulsive to low rate. Fifty signals were handpicked to train the classifier and another 48 to evaluate it's performance. Clustering the feature space into two clusters separates the chirps that look almost like impulses from those with lower chirp rates. The chirps were hand labeled as belonging to either of the two classes and the classifier's performance was evaluated. The classification with central moments method shows a poorer performance compared to the curve fitting method because the labeling was performed in the curve fitting derived feature space due to the simplicity in the visualizing a 2-dimensional feature space. Most of the misclassifications in both the methods can be attributed to the accuracy of the extraction of the entire length of the chirp, or in other words, accuracy of the frequency tracker in the presence of noise and other interferences. It shall be noted, however, the results show that most of the chirps are quadratic in nature. When the feature

space is divided into three clusters, no meaningful inference can be made due to the existence of a fuzzy boundary between the three classes in the 2-D feature space.

5.2 Future Work

During the course of work on this thesis, it's shortcomings can be viewed as highlighting the scope for future work in the area. The previous section discusses the pros and cons of the research reported in the thesis. Firstly, the need for a robust classifier entails training the Hidden Markov Model in such a way as to take the Signal to Noise Ratio into consideration. Secondly, training the HMM to incorporate signals chirping at any arbitrary rate is an issue that needs to be addressed. Thirdly, after tracking the frequency of the chirp signal, a better feature extraction technique which would be independent of the length of the chirp could be important in some applications. The histogram of the path provides an estimate of the state transition matrix associated with a given chirp signal and we might compare this histogram to those of associated with each possible class to determine the best match. The comparison could be performed directly using the relative entropy or Kullback-Leibler distance. Here, we have instead used an indirect approach of computing the central moments. Finally, a classifier that is independent of the distribution of the feature vectors would yield better classification results. All these deficiencies could possibly be overcome by assigning a Hidden Markov Model to every possible family of chirps. Here, we start with the way the HMM is trained for the classifier presented in this thesis. The Viterbi algorithm, then, tracks the frequency of the chirp for different signals from a particular class and the Hidden Markov Model is re-trained by placing the training data for HMM in the sequence of states exhibited by the Viterbi path. Once it has been trained, the only possible difference would be contained in the state transition

matrix and the observation density functions. For signals from a particular class, all these parameters- generated by computing the optimal state sequence- could be averaged. When computed for all the known classes, these parameters would characterize the different families of chirps. To evaluate the classifier, the class corresponding to the Hidden Markov Model yielding the largest log-likelihood for an unknown chirp signal would be the closest match to the chirp signal. Thus, many of the shortcomings of the classifier discussed in this thesis might be eliminated.

APPENDICES

APPENDIX A
TRAINING THE HMM

Chirp Signal Generation

```
1  % ************************************************************************
2  % chirpgen.m
3  % ************************************************************************
4  % Name: Nikhil Balachandran                                  Date: 02/10/06
5  % Purpose: The program genrates isolated chirp signals in white Gaussian
6  % noise.
7  %
8  % Input: fs = Sampling Frequency
9  %        WIN = window length
10 %        OVER = overlap
11 %        NFFT = FFT length
12 %        f0 = start frequency
13 %        f1 = end frequency
14 %        SNR = Signal to Noise ratio (add_noise function)
15 % Output: classify_chirp*_*_snr_*.mat
16 %
17 % References: Matlab command 'chirp'
18 % ************************************************************************
19 % clc;
20 clear all;
21 % close all;
22 t=0:0.000125:2 - 0.000125;                % sampling
23 fs = 8e3;
24 NFFT = 512;
25 WIN = 256;
26 OVER = 128;
27 randn('state',sum(100*clock))
28 z = randn(length(t),1)';
29 f0 = 200;
30 f1 = 1650;
31 N = 20;
32 for i = 1:N
33 y = chirp(t,f0,0.5,f1);
34 yl = [zeros(1,6000) y(1:6000) zeros(1,4000)];
35 yh = add_noise(yl,z,-10);
36 eval(['[C',int2str(i),',F,T,B] = chirprand(yh,NFFT,fs,WIN,OVER,0);']);
37
38 f0 = f0 + 20;
39 f1 = f1 + 20;
40 end
41 save Z:\Thesis\data\classify_chirp5_8_snr-10.mat C* T F
```

Function for Chirp Signal Generation

```
% ************************************************************************
% chirprand.m
% ************************************************************************
% Name: Nikhil Balachandran                                  Date: 02/22/06
% Purpose: The function convolves the frequencies at each time step of
% the spectrogram of the chirp with a Gaussian.
%
% Input: y = chirp signal
%        fs = Sampling Frequency
%        WIN = window length
%        OVER = overlap
%        NFFT = FFT length
%        option = '1' - display chirp, '0' - nothing
% Output: C = convolved spectrogram of the chirp
%         F = Frequency vector
%         T = Time vector
%         B = original spectrogram
% References:
% ************************************************************************

function [C,F,T,B] = chirprand(y,NFFT,fs,WIN,OVER,option)
% y=chirp(t,0,1,300);          % Start @ DC, cross 150Hz at t=1sec
% specgram(y,256,8E3,256,128); % Display the spectrogram
% plot(y);
% fvar = 0.002;
randn('state',sum(100*clock));

% f = f0+sqrt(fvar)*rsum(100*clock)andn(length(t),1);
[B,F,T] = specgram(y,NFFT,fs,WIN,OVER); % Display the spectrogram
f = -5:1:5; %Range of frequencies for randomness
[M N] = size(B);
L = length(f);
B = abs(B);
B = B.^2;
C = zeros(M + L - 1,N);
C(1:M,:) = B;

for i = 1: 1: N

    s(i) = floor(abs(5 + 0.5*randn));
    mu = 0;%find(abs(B(:,1)) == max(abs(B(:,1))));
    X = 1/sqrt(2*pi*s(i))*exp(-(f.^2)/(2*s(i)));;
    C(:,i) = conv(B(:,i),X);
end
if (option == 1)
    figure;
    IMAGESC(T,F,20*log10(C))
    axis xy
    colormap(jet)
end
% AM
% f1 = 500;
% x = sin(2*pi*f1*t);
% a = modulate(x,3000,10e3,'am');
% specgram(a,256,10E3,256,250); % Display the spectrogram
```

Function for Chirp Signal Generation

```
1  function [y,v] = add_noise(s,v,snr)
2  % ******************************************************************
3  % ADD_NOISE y = add_noise(s,v,snr)
4  %
5  % Description: This function adds additive Gaussian noise v to signal s
6  % resulting in SNR snr
7  %
8  % Input Arguments:
9  %     Name: s
10 %     Type: Vector
11 %     Description: Input  Signal
12 %
13 %     Name: v
14 %     Type: Vector
15 %     Description: Gaussian noise
16 %
17 %     Name: snr
18 %     Type: scalar
19 %     Description: SNR
20 %
21 % Output Arguments:
22 %     Name: y
23 %     Type: Vector
24 %     Description: Output signal
25 %
26 % ******************************************************************
27 s = s - mean(s);
28 Psig = cov(s);
29 Pnoise = Psig/(10^(snr/10));
30 v = v - mean(v);
31 v = (sqrt(Pnoise)/sqrt(cov(v))).*v;
32 y = s + v;
33 y = y./max(abs(y));
34 % y = awgn(s,10);
```

Training sequence for HMM

```
% ************************************************************************
% signal_chirp.m
% ************************************************************************
% Name: Nikhil Balachandran                                  Date: 02/10/06
% Purpose: The program genrates the training signal for the HMM.
%
% Input: fs = Sampling Frequency
%        WIN = window length
%        OVER = overlap
%        NFFT = FFT length
%        f0 = start frequency
%        Q = No. of states in the HMM excluding zero state
%        z = additive white Gaussian noise
% Output: train_sin1.mat
%
% References:
% ************************************************************************
% Sinusoid Training Signals for chirps
clc;
clear all;
close all;
t=0:0.000125:10 - 0.000125;                 % sampling
fs = 8e3;
NFFT = 512;
WIN = 256;
OVER = 128;
Q = 13;
z = sqrt(0.05)*randn(length(t),1);
f0 = 50;

for j = 1 : Q
    y = cos(pi*f0.*(t')) + z ;%+ cos(10*pi*f0.*(t'));
    if j ~= 0
        for i = 1:5
            f0 = f0 + 100;
            y = y + cos(pi*f0.*(t'));
        end
    end
    y = y - mean(y); y = y./sqrt(cov(y));
    % [C1,F,T,B] = chirprand(y,NFFT,fs,WIN,OVER,1);
    [B,F,T] = specgram(y,NFFT,fs,WIN,OVER); % Display the spectrogram
    eval(['C',int2str(j),' = abs(B).^2;']);
    clear y;
    % C1 = C1./(max(sum(C1)));
    f0 = f0 + 100;
end
save train_sin1.mat C* T F


```

Initialization of HMM

```
1  % ********************************************************************
2  % initHMM.m
3  % ********************************************************************
4  % Name: Nikhil Balachandran                                Date: 05/25/06
5  % Purpose: The function initializes the HMM.
6  %
7  % Input: M = No. of Gaussian mixtures
8  %        Q = No. of states in the HMM
9  %        cov_type = type of covaraince matrix (full, diagonal, etc.)
10 %        data = input to the HMM
11 % Output: prior0 = initial probability od the states
12 %         transmat0 = intial estimate of the state transition matrix
13 %         mu_t = mean for each state
14 %         Sigma_t = covariance matrix for each state
15 %         mixmat0 =  mixtures
16 %
17 % References: Kevin Murphy's HMM Toolbox
18 % ********************************************************************
19 function [prior0,transmat0,mu_t,Sigma_t,mixmat0] = initHMM(data,Q,M,cov_type)
20 % initial guess of parameters
21 % prior0 = [1 zeros(1,Q-1)];
22 % prior0 = [0.5 0.5];
23 prior0 = mk_stochastic(rand(1,Q));
24 % transmat0 = 0.0001*ones(Q);
25 % f = [1:Q];
26 % s = 10;
27 % mu = 1;
28 % for i = 1:Q
29 %     transmat0(i,:) = 1/sqrt(2*pi*s)*exp(-((f-mu).^2)/(2*s));
30 %     mu = mu + 1;
31 % end
32 % transmat0 = mk_stochastic(transmat0);
33 % transmat0 = mk_leftright_transmat(Q, 0.9);
34 transmat0 = mk_stochastic(rand(Q,Q));
35 %  transmat0 = [0.9 0.1; 0.1 0.9];
36 [mu0, Sigma0] = mixgauss_init(Q*M, data, cov_type);
37 z = max(mu0);
38 for i = 1 : Q
39     [r(i) c(i)] = find(mu0 == z(i));
40 end
41 [l in] = sort(r);
42 mu_t(:,:) = mu0(:,in);
43 Sigma_t(:,:,:) = Sigma0(:,:,in);
44 %     mu0 = reshape(mu0, [O Q M]);
45 %   Sigma0 = reshape(Sigma0, [O O Q M]);
46 %   mixmat0 = ones(Q,1);
47   mixmat0 = mk_stochastic(rand(Q,M));
```

APPENDIX B
FEATURE EXTRACTION

Code for Feature Extraction (Curve Fitting)

```
% ************************************************************************
% test_hmm_chirpv4.m
% ************************************************************************
% Name: Nikhil Balachandran                                  Date: 05/25/06
% Purpose: The program detcts, tracks and extracts paramters from the chirp.
%          (Polynomial Curve Fitting)
% Input: model5_trans.mat
% Output: chirp*_*_snr*.mat (P has the coefficients.)
%
% References: Kevin Murphy's HMM Toolbox, Matlab command 'polyfit'
% ************************************************************************
% clc;
clear all;
close all;
load model5_trans.mat
N = 20;
M = 15;
addpath(genpath('/home/graduate/nikhilb/Thesis/HMMall'));
for l = [4,5]
    for k = [-10,-7,-5,0,5,10] % SNR
        for j = [1:3,7:12]  % classes
            eval(['load /home/graduate/nikhilb/Thesis/data/test_chirp',...
                    int2str(l),'_',int2str(j),'_snr',int2str(k),'.mat;']);
            % Using Viterbi path algorithm
            for i = 1:N
                eval(['data = mk_unit_norm(C',int2str(i),');']);
                test_data = data(5:261,:);
                % test_data = fliplr(data4(5:261,:));
                [loglik,error,obslik] = mhmm_logprob(test_data, prior1,...
                    transmat1, mu1, Sigma1, mixmat1);
                path = viterbi_path(prior1, transmat1, obslik);

                nnull =  find(path~=1);
                if ~isempty(nnull)
                    if length(nnull) > 3
                        patht = path(nnull);
                        actpath = path(nnull(1):nnull(length(nnull)));
                        actpath(find(actpath == 1)) = mean(patht);
                    end
                else actpath = path;
                end
                X = [1:length(actpath)];
                % Poynomial Coefficients
                [P(i,:),S,MU] = polyfit(X,actpath,2);
                val = polyval(P(i,:),X,S,MU);
                %     figure;hist(path,[1:15]);
                %         plot(actpath);hold;plot(val);hold off;
            end
            %     plot(P5_1_snr0(:,1),P5_1_snr0(:,2),'b*');grid;
            eval(['save /home/graduate/nikhilb/Thesis/data/curvefit/chirp'...
                    ,int2str(l),'_',int2str(j),'_snr',int2str(k),'.mat P;']);
        end
    end
end
% end
```

Code for Feature Extraction (Central Moments)

```
% ************************************************************************
% test_hmm_chirpv5_hist.m
% ************************************************************************
% Name: Nikhil Balachandran                                  Date: 05/25/06
% Purpose: The program detcts, tracks and extracts paramters from the chirp.
%          (Central moments)
% Input: model5_trans.mat
% Output: chirp*_*_snr*_mom.mat (m has the central moments.)
%
% References: Kevin Murphy's HMM Toolbox, Matlab command 'polyfit'
% ************************************************************************
% clc;
clear all;
% close all;
% load C:\data\sample.mat
load model5_trans.mat
N = 20;
M = 15;
addpath(genpath('/home/graduate/nikhilb/Thesis/HMMall'));
for l = [4,5]
    for k = [-10,-7,-5,0,5,10] % SNR
        for j = [1:3,7:12] % classes
            eval(['load /home/graduate/nikhilb/Thesis/data/test_chirp',...
                    int2str(l),'_',int2str(j),'_snr',int2str(k),'.mat;']);
            % Using Viterbi path algorithm
            for i = 1 : N
               eval(['data = mk_unit_norm(C',int2str(i),');']);
               % Removed int2str for data....no point!!
                test_data = data(5:261,:);
                % test_data = fliplr(data4(5:261,:));
                [loglik,error,obslik] = mhmm_logprob(test_data, prior1,...
                    transmat1, mu1, Sigma1, mixmat1);
                path = viterbi_path(prior1, transmat1, obslik);

                nnull =  find(path~=1);
                 if ~isempty(nnull)
                    if length(nnull) > 3
                        patht = path(nnull);
                        actpath = path(nnull(1):nnull(length(nnull)));
                        actpath(find(actpath == 1)) = mean(patht);
                    end
                else actpath = path;
                end
                % Central Moments
                [R,X] = hist(actpath,[1:14]);
                m1 = sum(X.*(R./(sum(R))));
                m2 = sum((X.^2).*(R./(sum(R)))) - m1.^2 ;
                m3 = sum(((X-m1).^3).*(R./(sum(R))));
                m4 = sum(((X-m1).^4).*(R./(sum(R))));
                m5 = sum(((X-m1).^5).*(R./(sum(R))));
                m(i,:) = [m2 m3 m4 m5];
            end
            %     plot3(m(:,1),m(:,2),m(:,3),'ks');grid;
            eval(['save /home/graduate/nikhilb/Thesis/data/mom/chirp',...
             int2str(l),'_',int2str(j),'_snr',int2str(k),'_mom.mat m;']);
```

```
56              end
57          end
58      end
```

APPENDIX C
EVALUATION

Code for evaluating the classifier (Bayesian Classifier)

```
% ********************************************************************
% bayesian_classifier.m
% ********************************************************************
% Name: Nikhil Balachandran                              Date: 06/15/06
% Purpose: The program trains the Bayesian Classifier by computing the
%          discriminant functions. The commented part is used to evaluate
%          the classifier.
%
% Input: test.mat
%        Q = No. of states in the HMM
%        cov_type = type of covaraince matrix (full, diagonal, etc.)
% Output: model6_trans.mat
%
% References: 'Pattern Classification' by Duda, Hart and Stork.
% ********************************************************************
clc;
clear all;
close all;
N = 8;
load C:\data\test.mat
test = test(:,1:2)';test1 = test';
for k = 0
    for j = 1: N
        eval(['load C:\data\Pc',int2str(j),'_snr',int2str(k),'.mat']);
        eval(['P = Pc',int2str(j),'(:,1:2);']);
        M = size(P4,1);
        P4 = P4';
        P5 = P5';
        m1 = mean(P4');m1 = m1';
        m2 = mean(P5');m2 = m2';
        sig1 = zeros(2);
        sig2 = zeros(2);
        for k = 1:M
            sig1 = (P4(:,k) - m1)*(P4(:,k) - m1)' + sig1;
            sig2 = (P5(:,k) - m2)*(P5(:,k) - m2)' + sig2;
        end
        sig1 = sig1/10;
        sig2 = sig2/10;
        eval(['W4',int2str(j),'= (-0.5)*inv(sig1);']);
        eval(['w4',int2str(j),'= inv(sig1)*m1;']);
        m1t = m1';
        eval(['w40',int2str(j),...
         '= (-0.5)*m1t*inv(sig1)*m1 - 0.5*log(det(sig1));']);
        eval(['save C:\data\param4c',int2str(j),' W4',int2str(j),...
                ' w4',int2str(j),' w40',int2str(j),';']);


        eval(['W5',int2str(j),'= (-0.5)*inv(sig2);']);
        eval(['w5',int2str(j),'= inv(sig2)*m2;']);
        m2t = m2';
        eval(['w50',int2str(j),...
         '= (-0.5)*m2t*inv(sig2)*m2 - 0.5*log(det(sig2));']);
        eval(['save C:\data\param5c',int2str(j),' W5',int2str(j),...
                ' w5',int2str(j),' w50',int2str(j),';']);
        %    plot(m1(1,:),m1(2,:),'b*');hold on;plot(m2(1,:),m2(2,:),'m*');
```

```
56      end
57  end
58  % g = 0;
59  % for i = 1: N
60  %     eval(['load C:\data\param4c',int2str(i),'.mat']);
61  %     eval(['load C:\data\param5c',int2str(i),'.mat']);
62  %     eval(['w = w4',int2str(i),';']);
63  %     w = w';
64  %     eval(['g4 = test1*W4',int2str(i),'*test + w*test + w40',int2str(i),';']);
65  %
66  %     eval(['w1 = w5',int2str(i),';']);
67  %     w1 = w1';
68  %     eval(['g5 = test1*W5',int2str(i),'*test + w1*test + w50',int2str(i),';']);
69  %
70  %     g = [g g4 g5];
71  % end
72  %     g = g(2:end);
73
74
75
76
```

REFERENCES

[1] X. M. Shao and A. R. Jacobson, "Polarization observations of lightning-produced vhf emissions by the forte satellite," *Journal of Geophysical Research*, vol. 107, no. D20, pp. 4430, doi:10.1029/2001JD001018, October 2002.

[2] X. M. Shao and A. R. Jacobson, "Polarization observations of broadband vhf signals by the forte satellite," *Radio Science*, vol. 36, no. 6, pp. 1573–1589, November/December 2001.

[3] Manuel Davy, Christian Doncarli, and Jean-Yves Tourneret, "Classification of chirp signals using hierarchical bayesian learning and mcmc methods," *IEEE Transactions of Signal Processing*, vol. 50, no. 2, pp. 377–388, February 2002.

[4] Manuel Davy, Christian Doncarli, and G. Faye Bourdreaux-Bartels, "Improved optimization of time-frequency based signal classifiers," *IEEE Signal Processing Lett.*, vol. 8, no. 2, pp. 52–57, February 2001.

[5] Manuel Davy, Arthur Gretton, Arnaud Doucet, and Peter J. W. Rayner, "Optimized support vector machines for non-stationary signal classification," *IEEE Signal Processing Lett.*, vol. 9, no. 12, pp. 442–445, December 2002.

[6] "Multiscale chirplets and near-optimal recovery of chirps," E. J. Candès, Submitted. [Online], `Available:http://www.acm.caltech.edu/emmanuel/publications.html`.

[7] E. Chassande-Mottin and P. Flandrin, "On the time-frequency detection of chirps," *Applied and Computational Harmonic Analysis*, vol. 6, pp. 252–281, 1999.

[8] B. Boashash, P. O'Shea, and M.Arnold, "Algorithms for instantaneous frequency estimation: a comparitive study," *Proc. of SPIE Conf., Advanced Algorithms and Architectures for Signal Processing*, vol. 1384, no. 4, pp. 126–148, November 1990.

[9] B. Boashash, "Estimating and interpreting the instantaneous frequency of a signal—part 1: Fundamentals," *Proceedings of IEEE*, vol. 80, no. 4, pp. 520–538, April 1992.

[10] B. Volcker and B. Ottersten, "Chirp parameter estimation using rank reduction," *Signals, Syetems and Computers, Thrity-Second Asilomar Conference*, vol. 2, pp. 1443–1446, November 1998.

[11] B. Volcker and B. Ottersten, "Chirp parameter estimation using sample covaraince matrix," *IEEE Transactions on Signal Processing*, vol. 49, no. 3, pp. 603–612, March 2001.

[12] B. Boashash, *Time-Frequency Signal Analysis—Methods and Applications*, Longman-Cheshire, Melbourne, 1992.

[13] E. F. Velez and R. G. Absher, "Segmentation and classification of phonemes in continuous speech based on their time-frequency representations," *Proc. of SPIE Conf., Advanced Signal Processing Algorithms, Architectures and Implementations*, vol. 1348, pp. 188–196, July 1990.

[14] P. J. Boles and B. Boashash, "The cross wigner-ville distribution: A two dimensional analysis method for the processing of vibrosis seismic signals," *IEEE International Conference on Speech, Acoustics and Signal Processing*, vol. 2, pp. 904–907, April 1988.

[15] P. O'Shea and B. Boashash, "Time frequency analysis applied to the identification of machine sounds and biological signals," *IRRECON Digest, Institution of Engineers (Australia) Press*, 1987.

[16] R. F. Barrett and R. L. Streit, "Frequency line tracking using hidden markov models," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 38, no. 4, pp. 586–598, April 1990.

[17] G.E. Kopec, "Formant tracking using hidden markov models and vector quantization," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 34, no. 4, pp. 709–729, August 1986.

[18] A.G. Jaffer, R. L. Stoutenborough, and W.B. Green, "Improved detection and tracking of dyanamic signal by bayes-markov techniques," *IEEE International Conference on Speech, Acoustics and Signal Processing*, vol. 8, pp. 575–578, April 1983.

[19] J. V. Candy, *Model-Based Signal Processing*, Wiley, New York, 2005.

[20] L. Rabiner, "A tutorial in hidden markov models and selected applications in speech recognition," *Proceedings of IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[21] R. O. Duda, P. E. Hart, and D.G. Stork, *Pattern Classification*, Wiley, New York, 2001.

[22] L. E. Baum and T. Petrie, "Statistical Inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[23] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[24] A. V. Nefian and M. H. Hayes, "Hidden markov models for face recognition," *IEEE International Conference on Speech, Acoustics and Signal Processing*, vol. 5, pp. 2721–2724, May 1998.

[25] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[26] T. F. Quatieri, *Discrete-time Speech Signal Processing Principles and Practice*, Prentice-Hall Inc., New Jersey, 2002.