5-2010

Generalized multi-stream hidden Markov models.

Oualid Missaoui
University of Louisville

GENERALIZED MULTI-STREAM HIDDEN MARKOV MODELS

By

Oualid Missaoui
B.S., Signals and Systems, Polytechnic School of Tunisia, June 2003
M.S., Applied Mathematics, Polytechnic School of Tunisia, November 2005

A Dissertation
Submitted to the Faculty of the
Graduate School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2010

GENERALIZED MULTI-STREAM HIDDEN MARKOV MODELS

By

Oualid Missaoui
B.S., Signals and Systems, Polytechnic School of Tunisia, June 2003
M.S., Applied Mathematics, Polytechnic School of Tunisia, November 2005

A Dissertation Approved On

3/2/2010

Date

by the following Dissertation Committee:

Dissertation Director

This dissertation is dedicated to my parents: Khadija and Ahmed, and my brothers: Riadh and Moez, for the love, support and patience.

ACKNOWLEDGEMENTS

Thanks · · ·

First and foremost, to my advisor, Hichem Frigui. At the risk of sounding cliche, Hichem has been a near ideal advisor to me. He helped me cultivate a taste in research problems, taught me how to recognize and design good models and algorithms, and was a constant source of invaluable advice on navigating the academic world. Perhaps most importantly, he helped me to develop confidence in myself as a researcher - no easy feat, I am sure. I feel truly lucky to have had the chance to learn so much from him at the start of my career, and I sincerely hope that we will remain both collaborators and friends for many years to come.

To my dissertation committee, Aly Farag, Ahmed Desoky, Olfa Nasraoui, and Patricia Cerrito, for their feedback and advice, and for being so supportive of me and of this work.

To all my colleagues in the Multimedia Research Laboratory, and the Computer Engineering and Computer Science Department for their support support and friendship.

I all my friends in Louisville for the wonderful time we spent together. They helped me make this city home away from home.

Lastly but most importantly, to my wonderful family for the love, patience, and support they always gave me in this and all my projects.

ABSTRACT

GENERALIZED MULTI-STREAM HIDDEN MARKOV MODELS

Oualid Missaoui

March 2nd, 2010

For complex classification systems, data is usually gathered from multiple sources of information that have varying degree of reliability. In fact, assuming that the different sources have the same relevance in describing all the data might lead to an erroneous behavior. The classification error accumulates and can be more severe for temporal data where each sample is represented by a sequence of observations. Thus, there is a compelling evidence that learning algorithms should include a relevance weight for each source of information (stream) as a parameter that needs to be learned.

In this dissertation, we assumed that the multi-stream temporal data is generated by independent and synchronous streams. Using this assumption, we develop, implement, and test multi-stream continuous and discrete hidden Markov model (HMM) algorithms. For the discrete case, we propose two new approaches to generalize the baseline discrete HMM. The first one combines unsupervised learning, feature discrimination, standard discrete HMMs and weighted distances to learn the codebook with feature-dependent weights for each symbol. The second approach consists of modifying the HMM structure to include stream relevance weights, generalizing the standard discrete Baum-Welch learning algorithm, and deriving the necessary conditions to optimize all model parameters simultaneously. We also generalize the minimum classification error (MCE) discriminative training algorithm to include stream relevance weights.

For the continuous HMM, we introduce a new approach that integrates the stream relevance weights in the objective function. Our approach is based on the linearization of the probability density function. Two variations are proposed: the mixture and state level variations. As in the discrete case, we generalize the continuous Baum-Welch learning algorithm to accommodate

these changes, and we derive the necessary conditions for updating the model parameters. We also generalize the MCE learning algorithm to derive the necessary conditions for the model parameters' update.

The proposed discrete and continuous HMM are tested on synthetic data sets. They are also validated on various applications including Australian Sign Language, audio classification, face classification, and more extensively on the problem of landmine detection using ground penetrating radar data. For all applications, we show that considerable improvement can be achieved compared to the baseline HMM and the existing multi-stream HMM algorithms.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

The greatest challenge to any thinker is stating the problem in a way that will allow a solution

Bertrand Russell

Temporal data is sequential data that is ordered with respect to a given index [1, 2, 3]. The description and modeling of such data is based on the ordering concept which is not necessarily time. Time series form a well-known class of sequential data where records are indexed by time. Other examples of sequential data are text, pixels in an image, gene sequences, protein sequences, and lists of moves in a chess game. Temporal data has been studied extensively in data mining and statistics [4]. Classification is one of the main tasks in temporal data mining [1, 5]. Temporal data classification is needed in many applications such as speech recognition [6], gesture recognition [7] and handwritten word recognition [8].

One of the most commonly used temporal data classifiers is the Hidden Markov Models (HMMs) algorithm [6]. HMMs were introduced and studied in the late 1960s and early 1970s. They have great adaptability and versatility in handling sequential signals [9]. They have also been used for biological sequence classification [10, 11] and finance [12].

A hidden Markov model (HMM) (also known earlier as a probabilistic function of a Markov chain, or as a Markov source, or as a Markov regime model) is a stochastic process generated by two interrelated probabilistic mechanisms. The first mechanism consists of an underlying Markov chain with a finite number of states. At discrete instants of time, the process is assumed to be in some state and an observation is generated by the random function corresponding to the current state. The generated observations form the second probabilistic mechanism. The underlying Markov chain changes its state according to its transition matrix.

In HMM, the observer sees only the output of the random functions associated with each state and cannot observe the states of the underlying Markov chain directly. Hence, the Markov chain is hidden and the name for the model family is hidden Markov model.

In principle, the outputs from the states of the hidden Markov chain may be either multivariate random processes having some continuous joint probability distribution or a discrete finite alphabet [13]. The former model is called Continuous HMM and the latter one is called Discrete HMM.

For complex classification, multiple sources of information may contribute to the generation of sequences. In the standard HMMs, the different features contribute equally to the classification decision. However, these features may have different relevance degrees that depend on different regions of the feature space. Moreover, not all sources are always reliable. Consequently, treating these sources equally important and simply concatenating them and using a standard HMM might lead to a suboptimal classifier. Thus, more complex HMM based structures are needed to handle temporal data with multiple sources of information.

I.1 Related work

Approaches toward the combination of different modalities can be divided into three main categories: feature level fusion or direct identification, decision level fusion or separate identification (also known as late integration) and model level fusion (early/intermediate integration) [14]. In feature level fusion, multiple features are concatenated into a large feature vector and a single HMM model is trained [15]. This type of fusion has the drawback of treating heterogeneous features equally important. It also cannot represent the loose timing synchronicity between different modalities easily. In decision level fusion, the modalities are processed separately to build independent models [16]. This approach completely ignores the correlation between features and allows complete asynchrony between the streams. Also, it is computationally heavy since it involves two layers of decision. In model level fusion, an HMM model that is more complex than a standard one is sought. This additional complexity is needed to handle the correlation between modalities, and the loose synchronicity between sequences. Several HMM structures have been proposed for this purpose. Examples include factorial HMM [17], coupled HMM [18] and Multi-stream HMM [19]. Both factorial and coupled HMM structures allow asynchrony between sequences since a separate state sequence is assigned to each stream [20]. However, this is performed at the expense of an approximate parameter estimation. In fact, the parameters of factorial and coupled HMMs could be estimated via the EM (Baum-Welch) algorithm [6]. However, the E- step is computationally intractable and approximation approaches are used instead [18, 17].

Multi-stream HMM (MSHMM) is an HMM based structure that handles multiple modalities

for temporal data. It is used when the modalities (streams) are synchronous and independent. Since the streams are supposed to be synchronous, MSHMM assumes that for each time slot, there is a single hidden state, from which different streams interpret different observations. The independence of the streams means that their interpretation of the hidden state and their generation of the observations is performed independently.

To the best of our knowledge, no work that attempts to integrate feature discrimination in the discrete HMM. However, limited work has been reported for the continuous HMM had been reported in the literature. In particular, feature weighting was introduced in audio-visual stream weighting in speech recognition using continuous HMM [21, 22, 23]. In the standard continuous HMM, the classification decision is related to the probability density of a given observation sample in each state of the model. The probability density function (pdf) has been treated as a mixture of Gaussians where each Gaussian is weighted by a relevance coefficient. In particular, the overall feature space is partitioned into sub-spaces, and partial Gaussian components are learned in the different sub-spaces. Two general approaches have been proposed. The first one consists of factorizing each mixture into a product of weighted partial pdf(s) [24], where each pdf models a different feature subset. The relevance weight of each subset is learnt via the Minimum Classification Error (MCE) approach or the Generalized Probabilistic Descent (GPD) [24]. There was no reported learning approaches using the Maximum Likelihood (ML) based Baum-Welch algorithm. In fact, it was shown [24] that it is not possible to derive the ML learning equations for the exponent weights. The second approach considers the pdf as a product of exponent weighted mixture of Gaussians [25]. In this case, the pdf is a product of summation of Gaussians, whereas in the former case, the pdf is a summation of a product of Gaussians. In the latter case, the exponent weights are either fixed a priori by the user or learnt via the MCE/GPD approach [26]. The only attempt to learn the exponent weights within the Baum-Welch learning algorithm had been reported in [27]. However, this approach restricts the HMM structure to one Gaussian component per state.

I.2 Contributions

In this dissertation, we argue that since the stream relevance weights are parameters of the HMM structure, it is not meaningful to learn them separately from the rest of the HMM parameters. For the discrete case, we generalize the discrete HMM by introducing a weight matrix that assigns a relevance weight to each feature subset of each codebook. We propose two new approaches to train the discrete HMM. The first one combines unsupervised learning, feature discrimination, standard

discrete HMM and weighted distance to learn the codebook with feature dependent weights for each symbol. In this case, the set of feature weights are optimized using the Simultaneous Clustering and Attribute Discrimination (SCAD) algorithm [28]. SCAD is a clustering algorithm that partitions the data into clusters and learns cluster-dependent feature relevance weights. After learning the feature relevance weights through clustering, the regular discrete HMM parameters are trained via the standard Baum-Welch algorithm. To avoid overfitting, the Baum-Welch training is followed by a discriminative training component using the Minimum Classification Error/Generalized Probabilistic Descent (MCE/GPD) algorithm.

The second approach consists of generalizing the Baum-Welch algorithm to learn the weight of each feature subset. We assume that the probability of each codebook is a combination of the probabilities of the feature subset. Two forms of codebook probabilities are proposed. The first is a linear combination of partial probabilities. The second one is a geometric combination. The standard Baum-Welch learning algorithm is generalized to support both forms. In particular, we formulate the MLE that includes the feature relevance weights, and we derive the necessary conditions to maximize this MLE. The MCE/GPD algorithm is also generalized to include stream relevance weight estimation.

For the continuous HMM case, we treat the pdf as a sum of weighted linear combination of partial Gaussian components. We generalize the Baum-Welch algorithm and derive the necessary conditions to maximize the MLE learning equations. In particular, we introduce new structures of the continuous HMMs based on a linearization of the observation density of probability. This linearization is introduced to permit deriving the learning equations for all of the model parameters simultaneously. Two forms of linear pdf are introduced. The first one consists of a linear combination of the probability densities within different feature subspaces, each of them is a linear combination of mixture of Gaussians. The second one is a mixture of Gaussians, each is a linear combination of probability densities of different feature subspaces. For each method, we formulate the MLE that generalizes the Baum-Welch algorithm to include the necessary conditions to maximize this MLE. The MCE/GPD algorithm is also generalized to include stream relevance weight estimation.

Figure 1 displays a diagram that summarizes the different multi-stream HMM structures that we propose. We will refer to them as Generalized Multi-stream HMM (GMSHMM).

Figure (2) shows the diagram of a multi-modal temporal classifier based one of the structures in figure 1. It sketches the underlying components of a typical classifier. In fact, the data generated from the different streams is then fed to the GMSHMM. The GMSHMM takes into account the

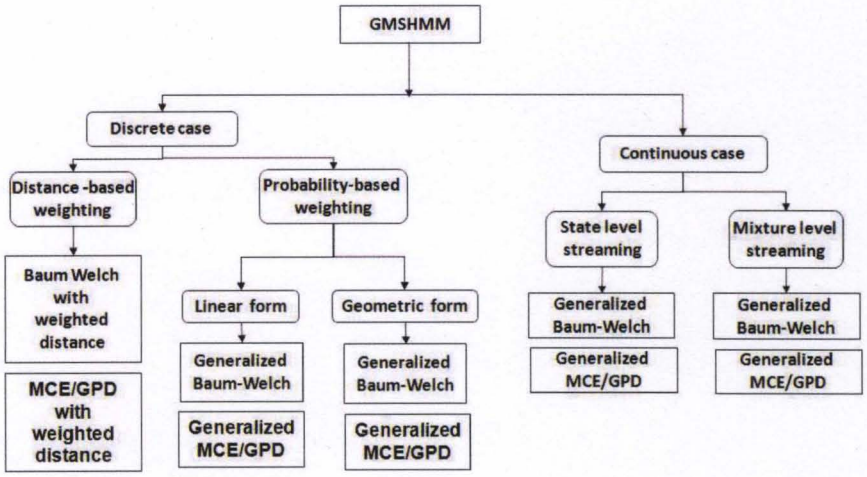


Figure 1. Proposed generalized multi-stream hidden Markov model structures.

multi-modal nature of the available sequences for training. In the case of C classes, the generalized Baum-Welch algorithm learns a separate GMSHMM for each class. A layer of discriminative training is then performed based on the generalized MCE/GPD to tune the parameters of the different GMSHMM. Once the C models are learnt, if a testing point occurs, it is assigned to the classes that produces the higher likelihood.

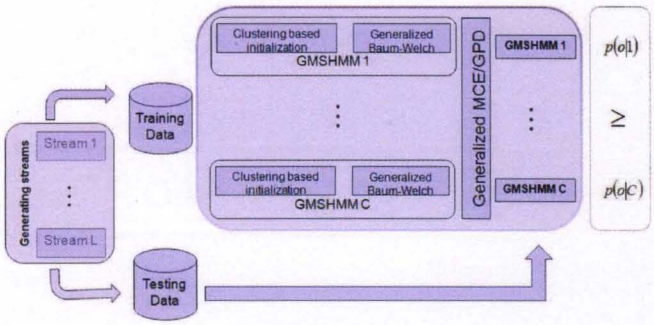


Figure 2. Architecture of the GMSHMM based classifier for multi-modal temporal data.

The proposed GMSHMM structures are applied to multiple domains. First we evaluate these GMSHMM models on synthetic data sets. Then, we apply them to classify sequential data in various applications including landmine detection, sign language classification, music genre classification, and face classification.

I.3 Dissertation overview

The organization of the rest of this dissertation is as follows. In chapter 2, we present the basics of Hidden Markov Models and the related optimization algorithms, namely, the Baum-Welch algorithm (BW) and the MCE/GPD algorithms. In chapter 3, we survey existing methods that handle multi-modal temporal data and their limitations. In chapter 4, we present our new approach, the generalized multi-stream discrete HMM (GMSDHMM), and we validate with synthetic data. In chapter 5, we present and validate, the generalized multi-stream continuous HMM (GMSDHMM). In chapter 6, we apply the proposed models to the problem of landmine detection, as well as to several other benchmark data sets widely used in the machine learning community. Chapter 7 presents the conclusions and future directions.

CHAPTER II

Hidden Markov Models: Fundamentals

Nothing is more practical than a good theory

Vladimir Vapnik

This chapter describes the basics of Hidden Markov Models (HMMs). We start by introducing the characteristics of the HMMs and the different HMM topologies. Then, we present the three main problems involved with HMMs and their correspondent solutions. In particular, we sketch the details of the Baum-Welch and the Minimum Classification Error/Generalized Probabilistic Descent algorithms for learning HMM parameters. These discussions form the base of our proposed generalized multi-stream HMMs.

II.1 Definition of Hidden Markov Models

A hidden Markov model (HMM) is a model of a doubly stochastic process that produces a sequence of random observation vectors at discrete times according to an underlying Markov chain. At each observation time, the Markov chain may be in one of N_s states $s_1, \cdots, s_{N_s}$ and, given that the chain is in a certain state, there are probabilities of moving to other states. These probabilities are called the transition probabilities. An HMM is characterized by three sets of probability density functions: the initial probabilities (π), the transition probabilities ($\mathbf{A}$), and the state probability density functions ($\mathbf{B}$). Let T be the length of the observation sequence (i.e., number of time steps), let $O = [o_1, \cdots, o_T]$ be the observation sequence, and let $Q = [q_1, \cdots, q_T]$ be the state sequence. The compact notation

$$\lambda = (\pi, \mathbf{A}, \mathbf{B}) \tag{II.1.1}$$

is generally used to indicate the complete parameter set of the HMM model. In (II.1.1), $\mathbf{A} = [a_{ij}]$ is the state transition probability matrix, where $a_{ij} = Pr(q_t = s_j | q_{t-1} = s_i)$, for $i, j = 1, \cdots, N_s$; $\pi = [\pi_i]$, where $\pi_i = Pr(q_1 = s_i)$ are the initial state probabilities; and $\mathbf{B} = b_i(o_t), i = 1, \cdots, N_s$, where $b_i(o_t) = Pr(o_t | q_t = s_i)$ is the set of observation probability distribution in state i. To simplify

the notation, we will simply use i to denote s_i.

An HMM is called discrete if the observation probability density functions are discrete and continuous if the observation probability density functions are continuous. In the case of the discrete HMM, the observation vectors are commonly vector quantized into a finite set of symbols, $\{v_1, v_2, \cdots, v_M\}$, called the codebook $\mathbf{V}$. Each state is represented by a discrete probability density function (pdf) and each symbol has a probability of occurring given that the system is in a given state. In other words, $\mathbf{B}$ becomes a simple set of fixed probabilities for each class, that is, $b_i(o_t) = b_{ij} = Pr(v_j|q_t = i)$, where v_j is the symbol of the nearest code book of o_t .

Figure 3 shows a discrete HMM with 3 states and 3 observations as a graphical model. In this model, the solid clear nodes represent the hidden states (q_1, q_2, q_3) and the shaded solid nodes represent the observed vectors (o_1, o_2, o_3). Solid edges represent the conditional dependence between the nodes. The model parameters are shown as transparent nodes linked to dotted arcs.

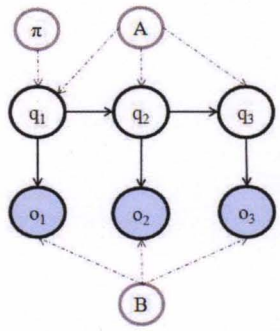


Figure 3. A graphical representation of the standard DHMM with 3 states and 3 observations.

In the continuous HMM, $b_i(o_t)$'s are defined by a mixture of some parametric probability density functions. The most common parametric pdf used in continuous HMM is the mixture of Gaussian density where

$$b_i(o_t) = \sum_{j=1}^{M_i} u_{ij} b_{ij}(o_t), i = 1, \cdots, N_s. \tag{II.1.2}$$

In (II.1.2), M_i is the number of components in state i, u_{ij} is the mixture coefficient for the jth mixture component in state i, and satisfies the constraints $u_{ij} \geq 0$, and $\sum_{j=1}^{Mi} u_{ij} = 1$, for $i = 1, ..., N_s$, and $b_{ij}(o_t)$ is a p-dimensional multivariate Gaussian density with mean μ_{ij} and covariance matrix Σ_{ij}. Without loss of generality, we assume that all the states have the same number of components i.e., $M_i = M, \forall\ 1 \leq i \leq N_s$. Figure 4 shows a continuous HMM with 3 states as a graphical model. In this model, the solid clear nodes represent random variables indicating the hidden states (q_1, q_2, q_3). The random variables (m_1, m_2, m_3) represent the occurring mixture component in each state. The

shaded solid nodes represent the observed vectors (o_1, o_2, o_3). As for the discrete case, solid edges represent the conditional dependence between the nodes, and the model parameters are shown as transparent nodes linked to dotted arcs.

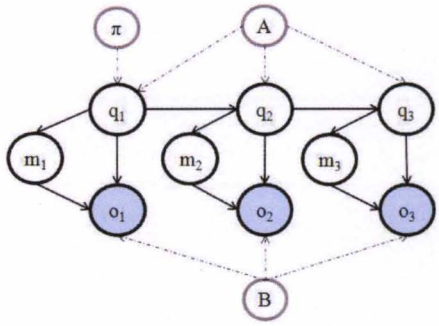


Figure 4. A graphical representation of the standard continuous HMM with 3 states.

II.2 HMM topologies

Depending on the state transition matrix, an HMM can be classified into one of the following types [9]:

II.2.1 Ergodic model

An ergodic model, as shown in figure 5, has full state transition. Being in any state, the model has the flexibility to move toward any other state.

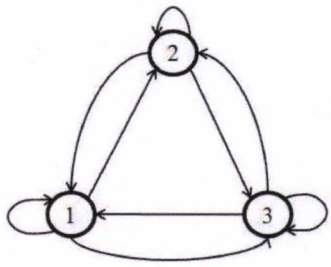


Figure 5. A graphical representation of an Ergodic HMM with 3 states.

II.2.2 Left-to-right model

A left-to-right model has only partial state transition such that $a_{ij} = 0 \; \forall j < i$. This type of model is widely used in modeling sequential signals. Figures 6 represent two varieties of left-to-right HMMs. In figure 6(a), being in a given state, the model can only stay in the same state, or move

forward to the subsequent one. In figure 6(b), a model cannot move backward, and is allowed to move forward to any subsequent state.

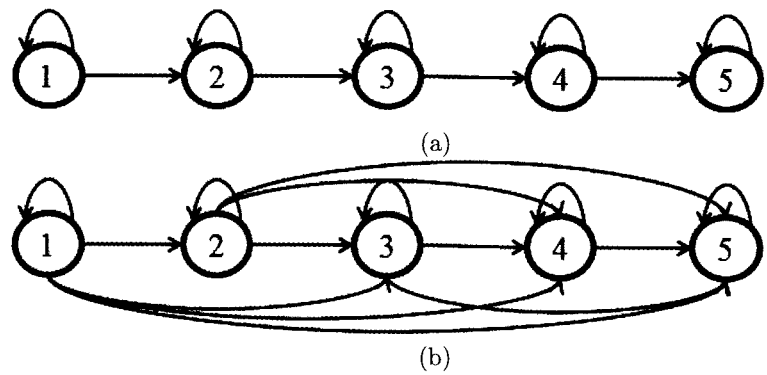


Figure 6. Graphical representations of two types of left-to-right HMM with 5 states. (a) the model stays in the same state or moves to next state. (b) the model can move to any subsequent state.

II.2.3 Cyclic model

As shown in figure 7, cyclic HMM is the model in which a state transition path represents a cycle.

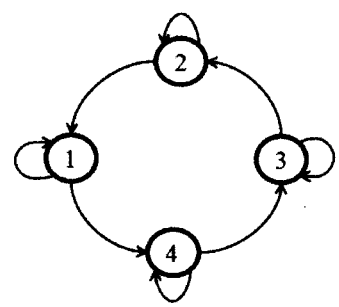


Figure 7. A graphical representation of a Cyclic HMM with 4 states.

II.3 Assumptions in the theory of HMMs

The theory of HMMs is based on three basic assumptions that make inference within the HMM framework tractable.

II.3.1 Markov assumption

Intuitively, the Markov assumption indicates that the future is independent of the past given the present. In fact, the transition to a next state q_{t+1}, given both the present and the past, depends

only on the current state q_t. In particular,

$$Pr(q_t|q_{t-1}, q_{t-2}, \cdots, q_1, \lambda) = Pr(q_t|q_{t-1}, \lambda) \tag{II.3.1}$$

II.3.2 Independence assumption

This assumption means that the observations are statistically independent given their correspondent states. In other words,

$$\begin{aligned} Pr(O|Q, \lambda) &= Pr(o_1, o_2, \cdots, o_T|q_1, q_2, \cdots, q_T, \lambda) && \text{(II.3.2)} \\ &= \prod_{t=1}^{T} Pr(o_t|q_t, \lambda). && \text{(II.3.3)} \\ & && \text{(II.3.4)} \end{aligned}$$

II.3.3 Stationarity assumption

Here, it is assumed that the state transition probabilities are independent from the actual time at which the transitions take place. Formally,

$$Pr(q_{t_1+1}|q_{t_1}, \lambda) = Pr(q_{t_2+1}|q_{t_2}, \lambda), \tag{II.3.5}$$

for any t_1 and t_2 in the range $1, \cdots, T$.

II.4 Main problems of HMMs

Given the form of the hidden Markov model defined in (II.1.1), Rabiner [6] defines three key problems of interest that must be solved for the model to be useful in real-world applications:

II.4.1 Problem 1: evaluation

The classification problem involves computing the probability of an observation sequence $O = [o_1, \cdots, o_T]$ given a model λ, that is, $Pr(O|\lambda)$. Bayesian methods can be used to obtain the probability of the model given the observation. This probability can be computed with $\mathcal{O}(TN_s^2)$ computations.

II.4.2 Problem 2: decoding

The problem of finding an optimal state sequence is also known as the decoding problem. There are several possible ways of finding an optimal state sequence associated with the given observation sequence, depending on the definition of the optimal state sequence. That is, there are

several possible optimality criteria. One that is particularly useful is to maximize $Pr(O, Q|\lambda)$ over all possible state sequences Q. The Viterbi algorithm [29] is an efficient formal technique for finding this maximum state sequence and associate probabilities.In most applications, it often turns out that computing an optimal state sequence is more useful than $Pr(O|\lambda)$.

II.4.3 Problem 3: classification

The classification problem, also called the training problem, consists of learning the optimal model parameters given a set of training data. This problem is difficult because there are several levels of estimation required in an HMM. First, the states themselves must be estimated. This is usually inferred from the physical characteristics of the problem in hands or performed using a model selection technique. Then, the model parameters $\lambda = (\pi, \mathbf{A}, \mathbf{B})$ need to be estimated. In the discrete HMM, first the codebook is determined, usually using clustering algorithms such as the K-means [6], or other vector quantization algorithms. In the continuous HMM, and for the case of Gaussian mixture density functions, the mixture component parameters, μ_{ij}, Σ_{ij}, u_{ij}, are first initialized (usually by clustering the training data). Then for both cases, the parameters $(\pi, \mathbf{A}, \mathbf{B})$ are estimated iteratively. Two strategies can be followed to estimate these parameters $(\pi, \mathbf{A}, \mathbf{B})$. The first one is an iterative method called Baum-Welch algorithm [6] which is an Expectation-Maximization (EM) [30] based algorithm that maximizes the likelihood function. In this maximum likelihood estimation (MLE) approach, the parameters of each class are learned independently. The optimality of the MLE criterion is conditioned on the availability of a large amount of training data and the correct choice of the model. Indeed, it was shown in [31] that if the true distribution of the samples to be classified can be accurately described by the assumed statistical model, and if the size of the training set tends to infinity, the MLE tends to be optimal. However, in practice, neither of these conditions are satisfied as the available training data is limited, and the assumptions made about the HMM structure are often inaccurate. As a consequence, the likelihood based training may not be effective. In this case, minimization of the classification error rate is a more suitable objective than minimization of the error of the parameter estimates. A common discriminative training method is the Minimum Classification Error (MCE)[32]. In fact, it has been reported since the mid nineties that discriminative training techniques were more successful, especially for automatic speech recognition [32]. The optimization of the error function is generally carried out by the Generalized Probabilistic Descent (GPD) algorithm [32], a gradient descent based optimization, and results in a classifier with minimum error probability.

II.5 Observation Evaluation: Forward-Backward Procedure

Let $O = (o_1 o_2 \ldots o_T)$ be an observation sequence where o_t is the observation symbol at time t and let $Q = (q_1 q_2 \ldots q_T)$ be a state sequence where $q_t \in S$ is the state at time t. Given a model λ, and an observation sequence O, we wish to evaluate $Pr(O|\lambda)$. The most straightforward way to determine $Pr(O|\lambda)$ is to find $Pr(O|Q,\lambda)$ for the fixed state sequence Q, multiply it by $Pr(Q|\lambda)$, and then sum up over all possible Q's. We have

$$Pr(O|Q,\lambda) = \prod_{t=1}^{T} b_{q_t}(o_t), \text{ and} \tag{II.5.1}$$

$$Pr(Q|\lambda) = \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}. \tag{II.5.2}$$

Hence,

$$Pr(O|\lambda) = \sum_{Q} Pr(O|Q,\lambda) Pr(Q|\lambda) \tag{II.5.3}$$

$$= \sum_{q_1 q_2, \cdots, q_T} \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t). \tag{II.5.4}$$

From (II.5.4) we see that the summation involves $2T-1$ multiplications, and there exists N_s^T distinct possible state sequences Q. Hence, a direct computation of (II.5.4) will be in the order of $2TN_s^T$ multiplications. Even for small values such as, $N_s = 5$ and $T = 100$, this would involve approximately 10^{72} multiplications which could take eons to complete even for a supercomputer. Hence, a more efficient procedure to solve problem 1 is required. Such a procedure exists and is called the forward-backward procedure [6].

Consider the variable $\alpha_t(i)$, defined as:

$$\alpha_t(i) = Pr(o_1, o_2, \ldots, o_t, q_t = i|\lambda), \tag{II.5.5}$$

i.e. the probability of the partial observation sequence up to time t and the state i at time t, given the model λ. The variable $\alpha_t(i)$ can be computed iteratively using the following three steps:

1.

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N_s \tag{II.5.6}$$

2. for $t = 1, 2, \ldots, T-1, \quad 1 \leq j \leq N_s$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N_s} \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \tag{II.5.7}$$

3. then we have:

$$Pr(O|\lambda) = \sum_{i=1}^{N_s} \alpha_T(i) \tag{II.5.8}$$

In step (2), the goal is the computation of the probability of partial observation sequence up to time $t+1$ and state j at time $t+1$; state j can be reached (with probability a_{ij}) independently from any of the N_s states at time t. The summation in (II.5.7) refers to this fact. In step (3), we just sum up all possible (independent) ways of realizing the given observation sequence.

In the Forward-Backward algorithm, step (1) involves N_s multiplications. Step (2) involves N multiplications plus one for the out of bracket $b_j(o_{t+1})$ term, this has to be done for $j = 1$ to N_s and $t = 1$ to $T-1$, making the total number of multiplication in step 2 $(N_s+1)N_s(T-1)$. Since step (3) involves no multiplications, the total number of multiplications is $N_s + N_s(N_s+1)(T-1)$ i.e. of the order of N_s^2T as compared to the $2T.N_s^T$ required for the direct method.

In a similar way, the backward variable $\beta_t(i)$ is defined as :

$$\beta_t(i) = Pr(o_{t+1}, o_{t+2}, \ldots, o_T | q_t = i, \lambda) \tag{II.5.9}$$

i.e. the probability of the observation sequence from $t+1$ to T given the state i at time t and the model λ. Note that here $q_t = i$ has already been given (it wasn't the case for the forward variable). This distinction has been made to be able to combine the forward and the backward variables to produce useful results. $\beta_t(i)$ could be computed in 3 steps similar to the way $\alpha_t(i)$ was computed:

1.

$$\beta_T(i) = i, 1 \leq i \leq N_s \tag{II.5.10}$$

2. for $t = T-1, T-2, \ldots, 1$, $1 \leq i \leq N$

$$\beta_t(i) = \sum_{j=1}^{N_s} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{II.5.11}$$

3.

$$Pr(O|\lambda) = \sum_{i=1}^{N_s} \pi_i b_i(o_1) \beta_1(i) \tag{II.5.12}$$

The computation of $Pr(O|\lambda)$ using $\beta_t(i)$ also involves the order of N_s^2T calculations. Hence both the forward as well as the backward method are equally efficient for the computation of $Pr(O|\lambda)$. This solves problem 1.

II.6 State Sequence Decoding: The Viterbi Algorithm

Problem 2 consists of finding the optimal state sequence associated with the given observation sequence. In other words, we have to find a state sequence $Q = (q_1, q_2, \ldots, q_T)$ such that the probability of occurrence of the observation sequence $O = (o_1, o_2, \ldots, o_T)$ from this state sequence is greater than that from any other state sequence. The problem is then to find Q that will maximize $Pr(O, Q|\lambda)$. This can be achieved using the Viterbi Algorithm [6]. From (II.5.3) and (II.5.4) we have

$$\begin{aligned} Pr(O, Q|\lambda) &= Pr(O|Q, \lambda) Pr(Q|\lambda) \\ &= \pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \prod_{t=1}^{T} b_{q_t}(o_t) \\ &= \pi_{q_1} b_{q_1}(o_1) \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(o_t) \end{aligned}$$

By introducing a new variable:

$$\begin{aligned} U(Q) &= -\ln(Pr(O, Q|\lambda)) \\ &= -\left[\ln(\pi_{q_1} b_{q_1}(o_1)) + \sum_{t=2}^{T} \ln(a_{q_{t-1} q_t} b_{q_t}(o_t)) \right], \end{aligned} \tag{II.6.1}$$

it can be shown that:

$$Pr(O, Q|\lambda) = \exp(-U(Q)) \tag{II.6.2}$$

Consequently the problem of optimal state estimation, namely,

$$\max_{Q} P(O, Q|\lambda) \tag{II.6.3}$$

becomes equivalent to

$$\min_{Q} U(Q) \tag{II.6.4}$$

This new reformulation enables us to consider terms like $-\ln(a_{q_j q_k} b_{q_k}(o_t))$ as the cost associated with the transition from state q_j to state q_k at time t.

In the following, we describe the Viterbi algorithm which can be used to find the optimum state sequence. Let $-\ln(a_{ij} b_j(o_t))$ be the weight on the path from state i to state j where o_t is the observation symbol selected after visiting state j. Let $-\ln(\pi_i b_i(o_t))$ be the weight corresponding to the selection of the initial state i.

Finding the optimum sequence is equivalent to finding the path (i.e. a sequence of states) with the minimum weight through which the given observation sequence occurs. Thus, the Viterbi

Algorithm is basically a dynamic programming approach for minimizing $U(Q)$. Let $\delta_t(i)$ denote the weight accumulated when we are in state i at time t as the algorithm proceeds and let $\psi_t(j)$ be the state at time $t-1$ which has the lowest cost corresponding to the state transition to state j at time t. The Viterbi algorithm can be summarized by the following four steps:

1. Initialization: for $1 \leq i \leq N_s$

$$\delta_1(i) = -\ln(\pi_i) - \ln(b_i(o_1)) \tag{II.6.5}$$

$$\psi_1(i) = 0 \tag{II.6.6}$$

2. Recursive computation: for $2 \leq t \leq T$ For $1 \leq j \leq N_s$

$$\delta_t(j) = \min_{1 \leq i \leq N_s} [\delta_{t-1}(i) - ln(a_{ij})] - \ln(b_j(o_t)) \tag{II.6.7}$$

$$\psi_t(j) = \operatorname*{argmin}_{1 \leq i \leq N_s} [\delta_{t-1}(i) - \ln(a_{ij})] \tag{II.6.8}$$

3. Termination:

$$P^* = \min_{1 \leq i \leq N_s} [\delta_T(i)] \tag{II.6.9}$$

$$q_T^* = \operatorname*{argmin}_{1 \leq i \leq N_s} [\delta_T(i)] \tag{II.6.10}$$

4. Tracing back the optimal state sequence For $t = T-1, T-2, \ldots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{II.6.11}$$

Hence, $\exp(-P^*)$ gives the required state-optimized probability, and $Q^* = q_1^*, q_2^*, \ldots, q_T^*$ is the optimal state sequence. Computationally, the Viterbi algorithm is similar to the forward-backward procedure except for the comparisons needed to find the maximum value. Therefore, its complexity is also of the order of $N_s^2 T$. This solves problem 2.

II.7 The classification problem

The classification problem is to determine a method to estimate the model parameters $(\pi, \mathbf{A}, \mathbf{B})$ based on the observation sequence O. This is the most difficult problem as there is no analytical solution to this problem. The general approach is to train the model with the available training data following some iterative procedure until its convergence. In particular, after an initial guess, a set of re-estimation formula would be repeated so that the parameter set could gradually approach the optimal values where the likelihood of the observation sequence is maximal. Similar

to problem 2, there are different criteria to interpret the problem. The maximum likelihood (ML) criterion [6] and the minimum classification error (MCE) criterion [32] are the most widely used ones. In the following, a description is given to both objectives for both the discrete and continuous HMM.

II.7.1 Maximum Likelihood criterion: Baum-Welch algorithm

The standard approach to estimate the HMM parameters is to use the expectation-maximization (EM) algorithm [30], also known as the forward-backward or Baum-Welch (BW) algorithm [6] in this context, to find the maximum likelihood (ML) estimator. Let λ represents the current model and let $\overline{\lambda}$ represents a candidate model, i.e. the model that we want to built out of the current model and that optimizes a specific objective function. Our objective is to make $Pr(O|\overline{\lambda}) \geq Pr(O|\lambda)$, or equivalently $\log(Pr(O|\overline{\lambda})) \geq \log(Pr(O|\lambda))$. Due to the presence of stochastic constraints such that $\sum_j a_{ij} = 1$, it is easier to maximize an *auxiliary function* $\mathbb{Q}(.)$ rather than to directly maximize $\log(Pr(O|\overline{\lambda}))$. This auxiliary function, also called E-M auxiliary function, is defined as:

$$\mathbb{Q}(\lambda, \overline{\lambda}) = \sum_{Q} Pr(Q|O, \lambda) \ln(Pr(O, Q|\overline{\lambda})). \tag{II.7.1}$$

The following two propositions show that maximizing of $\mathbb{Q}(\lambda, \overline{\lambda})$ is equivalent to maximizing $Pr(O|\lambda)$.

Proposition II.7.1. *If the value of $\mathbb{Q}(\lambda, \overline{\lambda})$ increases, then the value of $Pr(O|\lambda)$ also increases, i.e.,*

$$\mathbb{Q}(\lambda, \overline{\lambda}) \geq \mathbb{Q}(\lambda, \lambda) \Longrightarrow Pr(O|\overline{\lambda}) \geq Pr(O|\lambda) \tag{II.7.2}$$

Proposition II.7.2. *λ is a critical point of $Pr(O|\lambda)$ if and only if it is a critical point of $\mathbb{Q}(\lambda, \overline{\lambda})$, i.e.,*

$$\frac{\partial Pr(O|\lambda)}{\partial \theta_p} = \left.\frac{\partial \mathbb{Q}(\lambda, \overline{\lambda})}{\partial \overline{\theta}_p}\right|_{\overline{\lambda}=\lambda}, \tag{II.7.3}$$

where θ_p is any individual parameter of λ.

The proofs of the proposition (II.7.1) and (II.7.2) are sketched in appendix A.

The closed form expression of $\mathbb{Q}(\lambda, \overline{\lambda})$ represents the E-step (Expectation step) of the Expectation-Maximization algorithm (EM algorithm).

II.7.1.1 Discrete HMM

The objective function in (II.7.1) can be expanded to:

$$\begin{aligned}\mathbb{Q}(\lambda,\bar{\lambda}) &= \sum_{Q} Pr(Q|O,\lambda)\log(\bar{\pi}_{q_1}) + \\ &\sum_{t=1}^{T}\sum_{Q} Pr(Q|O,\lambda)\log(\bar{a}_{q_t q_{t+1}}) + \\ &\sum_{t=1}^{T}\sum_{Q} Pr(Q|O,\lambda)\log(\bar{b}_{q_t Q_V(o_t) f_t}).\end{aligned} \tag{II.7.4}$$

By applying the Lagrange multipliers optimization method to the objective function in (II.7.4), it can be shown [6] that the parameters π, $\mathbf{A}$, and $\mathbf{B}$ need to be updated using:

$$\begin{aligned}\pi_i &= Pr(q_1 = i|O,\lambda) \\ a_{ij} &= \frac{\sum_{t=1}^{T} Pr(q_t = i, q_{t+1} = j|O,\lambda)}{\sum_{t=1}^{T} Pr(q_t = j|O,\lambda)}, \\ b_{ij} &= \frac{\sum_{t=1}^{T} Pr(q_t = j|O,\lambda)\delta(Q_V(o_t), j)}{\sum_{t=1}^{T} Pr(q_t = j|O,\lambda)}.\end{aligned}$$

In the above, Q_V is the quantization operation defined on an observation vector as $Q_V(o_t) = \text{argmin}_{1\le j\le M}\, d(o_t, v_j)$ and $\delta(.,.)$ is the Kronecker delta function defined as:

$$\delta(i,j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}. \tag{II.7.5}$$

The above expressions of the parameters π, $\mathbf{A}$, and $\mathbf{B}$, since they zero the gradient of the objective function in (II.7.4), represent a critical point. However, it could be easily shown that this critical point makes the second derivative of (II.7.4) negative, and thus is a local maximum.

Let $\gamma_t(i) = Pr(q_t = i|O,\lambda)$, and $\xi_t(i,j) = Pr(q_t = i, q_{t+1} = j|O,\lambda)$. As in [6], we can express $\gamma_t(i)$ and $\xi_t(i,j)$ using the forward and backward variables $\alpha_t(j)$ and $\beta_t(j)$:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N_s}\alpha_t(j)\beta_t(j)} \tag{II.7.6}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N_s}\sum_{j=1}^{N_s}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \tag{II.7.7}$$

Thus, the update equations could be written as:

$$\pi_i = \gamma_1(i), \tag{II.7.8}$$

$$a_{ij} = \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{t=1}^{T}\gamma_t(i)}, \tag{II.7.9}$$

$$b_{ij} = \frac{\sum_{t=1}^{T}\gamma_t(i)\delta(Q_V(o_t), j)}{\sum_{t=1}^{T}\gamma_t(i)}. \tag{II.7.10}$$

II.7.1.2 Continuous Baum-Welch

In the continuous case, we recall that

$$b_i(o_t) = Pr(o_t|q_t = i, \lambda) = \sum_{j=1}^{M} u_{ij}\mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij}). \tag{II.7.11}$$

Let $E = [e_1, \cdots, e_T]$ be the sequence of random variables representing the mixture component indices for each time step. In fact, the random variable e_t identifies the mixture component index within a given state at time t. Thus, if $q_t = i$ and $e_t = k$, then at time t state i and mixture component j occur. The kernel function $\mathcal{N}$ in (II.7.11) is a multivariate Gaussian that represents the probability density of a vector of continuous observation o_t, given that at time t the underlying state and mixture component are respectively i and j. In other words,

$$Pr(o_t|q_t = i, e_t = j, \lambda) = \mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij}). \tag{II.7.12}$$

In the Gaussian case, the kernel function $\mathcal{N}$ is:

$$\mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij}) = \frac{1}{(2\pi)^{p/2}|\Sigma_{ij}|^{1/2}} \exp(-\frac{1}{2}(o_t - \mu_{ij})\Sigma_{ij}^{-1}(o_t - \mu_{ij})') \tag{II.7.13}$$

where $|\Sigma_{ij}|$ is the determinant of the covariance matrix. In practice, the off-diagonal variances are assumed zero. In such case, the determinant $|\Sigma_{ij}|$ is just the product of p scalar variances $|\Sigma_{ij}| = \prod_{d=1}^{p} \sigma_{ijd}^2$, and (II.7.13) reduces to:

$$\mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij}) = \frac{1}{(2\pi)^{p/2}|\Sigma_{ij}|^{1/2}} \exp -\frac{1}{2}\sum_{d=1}^{p} \frac{(o_{td} - \mu_{ijd})^2}{\sigma_{ijd}^2}. \tag{II.7.14}$$

Given a state i, the system randomly chooses one of its M possible mixture components within the state with a mixture emission probability $Pr(e_t = j|q_t = i, \lambda)$. This probability is assumed to be independent of t and thus, it can be represented by a parameter with no time index. In our notation, we let $u_{ij} = Pr(e_t = j|q_t = i, \lambda)$ be the weight of the kth mixture component embedded in state i. The mixture component could be interpreted as low level hidden states e_t embedded within high level hidden states q_t. Thus, the objective function in (II.7.1) is adapted to include the random vectors Q and E representing the high level and low level hidden states:

$$\mathbb{Q}(\lambda, \overline{\lambda}) = \sum_{Q}\sum_{E} \ln(Pr(O, Q, E|\overline{\lambda}))Pr(Q, E|O, \lambda). \tag{II.7.15}$$

It could be easily proven that this new form of $\mathbb{Q}$ still satisfies the propositions in (II.7.1) and (II.7.2). In fact, we only need to consider Q and E as one single hidden vector (Q, E) to satisfy the propositions (II.7.1) and (II.7.2). The objective function in (II.7.15) involves the quantity $Pr(O, Q, E|\overline{\lambda})$

which has the analytical form:

$$Pr(O,Q,E|\overline{\lambda}) = \pi_{q_1} \prod_{t=2}^{T} Pr(q_t|q_{t-1},\overline{\lambda}) \prod_{t=1}^{T} u_{q_t e_t} \mathcal{N}(o_t, \mu_{q_t e_t}, \Sigma_{q_t e_t}). \tag{II.7.16}$$

It follows then that,

$$\begin{aligned}
\mathbb{Q}(\lambda,\overline{\lambda}) &= \sum_Q \sum_E Pr(Q,E|O,\lambda) \ln(\overline{\pi}_{q_1}) + \\
&= \sum_{t=1}^{T-1} \sum_Q \sum_E Pr(Q,E|O,\lambda) \ln(\overline{a}_{q_t q_{t+1}}) + \\
&= \sum_{t=1}^{T-1} \sum_Q \sum_E Pr(Q,E|O,\lambda) \ln(\overline{u}_{q_t e_t}) + && \text{(II.7.17)} \\
&= \sum_{t=1}^{T-1} \sum_Q \sum_E Pr(Q,E|O,\lambda) \ln \mathcal{N}(o_t, \overline{\mu}_{q_t e_t}, \overline{\Sigma}_{q_t e_t}) && \text{(II.7.18)}
\end{aligned}$$

Using the Lagrange multipliers optimization method, it can be shown [6] that the estimates of the parameters π, $\mathbf{A}$, $\mathbf{B}$, μ, and Σ could be computed iteratively using:

$$\overline{u}_{ij} = \frac{\sum_{t=1}^{T} Pr(q_t = i, e_t = j|o,\lambda)}{\sum_{t=1}^{T} Pr(q_t = i|o,\lambda)} \tag{II.7.19}$$

$$\overline{\mu}_{ijd} = \frac{\sum_{t=1}^{T} Pr(q_t = i, e_t = j|o,\lambda) o_{td}}{\sum_{t=1}^{T} Pr_\lambda(q_t = i|o)} \tag{II.7.20}$$

$$\overline{\sigma}^2_{ijd} = \frac{\sum_{t=1}^{T} Pr(q_t = i, e_t = j|o,\lambda)(o_{td} - \overline{\mu}_{ijd})^2}{\sum_{t=1}^{T} Pr(q_t = i|o,\lambda)} \tag{II.7.21}$$

The above expression of the parameters π, $\mathbf{A}$, and $\mathbf{B}$, since they zero the gradient of the objective function in (II.7.15), represent a critical point. It could be easily shown that this critical point is also a local maximum since it makes the second derivative of (II.7.15) negative. Since

$$Pr(q_t = i, e_t = k|o,\lambda) = Pr(q_t = i|o,\lambda) \frac{u_{ik}\phi(o_t, \mu_{ik}, \Sigma_{ik})}{b_i(o_t)} \tag{II.7.22}$$

and $\gamma_t(i,k) = Pr(q_t = i, e_t = k|o,\lambda)$, the learning equations in (II.7.19)-(II.7.21) can be rewritten:

$$\overline{u}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{II.7.23}$$

$$\overline{\mu}_{ijn} = \frac{\sum_{t=1}^{T} \gamma_t(i,j) o_{tn}}{\sum_{t=1}^{T} \gamma_t(i,j)} \tag{II.7.24}$$

$$\overline{\sigma}^2_{ijn} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)(o_{tl} - \overline{\mu}_{ijn})^2}{\sum_{t=1}^{T} \gamma_t(i,j)} \tag{II.7.25}$$

II.7.1.3 Multiple observation sequences

In practice, one single observation sequence is not sufficient to learn all the parameters of an HMM model (discrete or continuous). Typically, multiple observation sequences are available and

are used to obtain reliable estimation of the model parameters [6]. Let

$$\mathbb{O} = [O^{(1)}, O^{(2)}, \ldots, O^{(R)}] \tag{II.7.26}$$

denote the set of R observation sequences, where $O^{(r)} = [o_1^{(r)}, o_2^{(r)}, \cdots, o_{T_r}^{(r)}]$ is the rth observation sequence. Without loss of generality, we assume T_r=T $\forall$ $1 \leq r \leq R$. Usually, one does not know if these observation sequences are independent of each other or not, and a controversy can arise if one assumes the independence property while these observation sequences are statistically correlated. In either case, we have the following expressions without generality:

$$\begin{cases} Pr(\mathbb{O}|\lambda) &= Pr(O^{(1)}|\lambda)Pr(O^{(2)}|O^{(1)}\lambda)\ldots Pr(O^{(R)}|O^{(R-1)}\ldots O^{(1)},\lambda) \\ Pr(\mathbb{O}|\lambda) &= Pr(O^{(2)}|\lambda)Pr(O^{(3)}|O^{(2)}\lambda)\ldots Pr(O^{(1)}|O^{(R)}\ldots O^{(2)},\lambda) \\ . & \\ . & \\ . & \\ Pr(\mathbb{O}|\lambda) &= Pr(O^{(R)}|\lambda)Pr(O^{(1)}|O^{(K)}\lambda)\ldots Pr(O^{(R-1)}|O^{(R)}O^{(R-2)}\ldots O^{(1)},\lambda) \end{cases}$$

Based on the above equations, the multiple observation probability given the model can be expressed as:

$$Pr(O|\lambda) = \sum_{r=1}^{R} w_r Pr(O^{(r)}|\lambda), \tag{II.7.27}$$

where

$$\begin{cases} w_1 &= \frac{1}{R}Pr(O^{(2)}|O^{(1)},\lambda)\ldots Pr(O^{(R)}|O^{(R-1)}\ldots O^{(1)},\lambda) \\ w_2 &= \frac{1}{R}Pr(O^{(3)}|O^{(2)},\lambda)\ldots Pr(O^{(1)}|O^{(R)}\ldots O^{(2)},\lambda) \\ . & \\ . & \\ . & \\ w_R &= \frac{1}{R}Pr(O^{(1)}|O^{(R)},\lambda)\ldots Pr(O^{(R-1)}|O^{(R)}O^{(R-2)}\ldots O^{(1)},\lambda) \end{cases}$$

are weights. These weights are conditional probabilities and, hence, they can characterize the dependence-independence property. Based on the above expression, we can construct the following auxiliary function for model training:

$$\mathbb{Q}(\lambda, \overline{\lambda}) = \sum_{r=1}^{R} w_r \mathbb{Q}_r(\lambda, \overline{\lambda}), \tag{II.7.28}$$

where $\overline{\lambda}$ is the auxiliary variable corresponding to λ and

$$\mathbb{Q}_r(\lambda, \overline{\lambda}) = \sum_{Q} Pr(O^{(r)}, Q|\lambda) \ln(Pr(O^{(r)})|\overline{\lambda}), 1 \leq r \leq R \tag{II.7.29}$$

are Baum's auxiliary functions related to individual observations. Since w_r, are not functions of $\overline{\lambda}$, propositions (II.7.1) and (II.7.2) hold for $\mathbb{Q}(\lambda, \overline{\lambda})$. Now, let us assume that the individual observations are independent of each other, i.e.,

$$Pr(O|\lambda) = \prod_{r=1}^{R} Pr(O^{(r)}|\lambda) \tag{II.7.30}$$

In this case, the combinatorial weights reduce to:

$$w_r = \frac{1}{R}\frac{Pr(O|\lambda)}{Pr(O^{(r)}|\lambda)}, 1 \leq r \leq R. \tag{II.7.31}$$

Using the same procedure as for one observation sequence, we can derive the following training equations for the discrete HMM:

$$\overline{\pi}_i = \frac{1}{R}\sum_{r=1}^{R} \gamma_1^{(r)}(i), 1 \leq i \leq N_s \tag{II.7.32}$$

$$\overline{a}_{ij} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T-1} \xi_t^{(r)}(i,j)}{\sum_{r=1}^{R}\sum_{t=1}^{T-1} \gamma_t^{(r)}(i)} \tag{II.7.33}$$

$$\overline{b}_{ij} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^{(r)}(i)\delta(o_t^r, j)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \gamma_t^{(r)}(i)} \tag{II.7.34}$$

Similarly, the following equations could be derived for the continuous HMM:

$$\overline{u}_{ik} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t(i,k)^r}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t(i,k)^r} \tag{II.7.35}$$

$$\overline{\mu}_{ikn} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t(i,k)^l o_{tn} o^{(r)}}{\sum_{r=1}^{R}\sum_{t=1}^{T_l} \gamma_t(i,k)^r} \tag{II.7.36}$$

$$\overline{\sigma}_{ikn}^2 = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t(i,k)^r (o_{tr} - \overline{\mu}_{ikn})^2}{\sum_{t=1}^{T} \gamma_t(i,k)^r} \tag{II.7.37}$$

II.7.1.4 Forward-Backward variables scaling

As mentioned in [6], the computation of $\alpha_t(i)$ and $\beta_t(i)$ consists of the sum of a large number of terms, each of the form

$$\left(\prod_{s=1}^{t} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(o_s)\right) \tag{II.7.38}$$

with $q_t = s_i$. Since $\mathbf{A}$ is a discrete probability distribution, its value is less than one. In addition, the values for $\mathbf{B}$ are usually less than one. Consequently, as t gets large (e.g., 10 or more), each term of $\alpha_t(i)$ starts to head exponentially to zero. In fact, for sufficiently large t (e.g., 100 or more) the dynamic range of the computed $\alpha_t(i)$ will exceed the precision range of essentially any machine (even in double precision). Hence, it is necessary to incorporate a scaling procedure. A common scaling procedure multiplies $\alpha_t(i)$ by a scaling coefficient that is independent of i (i.e., it depends only on

t). The goal is to keep the scaled $\alpha_t(i)$ within the dynamic range of the computer for $1 \leq t \leq T$. A similar scaling is done to the $\beta_t(i)$ coefficients since these also tend to zero exponentially fast.

For each t, we first compute $\alpha_t(i)$ according to its basic formula, and then we multiply it by a scaling coefficient $c_t = \frac{1}{\sum_{i=1}^{N_s} \alpha_t(i)}$. Similarly, each $\beta_t(i)$ is scaled by $\frac{1}{\sum_{i=1}^{N_s} \beta_t(i)}$. Obviously, the scaling of $\alpha_t(i)$ and $\beta_t(i)$ affects the computation of $Pr(O|\lambda)$. However, we can still compute $Pr(O|\lambda)$ using c_t values. In fact, since $\frac{\sum_{i=1}^{N_s} \alpha_T(i)}{\sum_{i=1}^{N_s} \alpha_T(i)} = 1$, we can write:

$$\frac{\sum_{i=1}^{N_s} \alpha_T(i)}{\sum_{i=1}^{N_s} \alpha_T(i)} = \sum_{i=1}^{N_s} \frac{\alpha_T(i)}{\sum_{i=1}^{N_s} \alpha_T(i)} \tag{II.7.39}$$

$$= \sum_{i=1}^{N_s} \hat{\alpha}_T(i) \tag{II.7.40}$$

However, we can induce that:

$$\hat{\alpha}_T(i) = \left(\prod_{\tau=1}^{T} c_\tau \right) \alpha_T(j) \tag{II.7.41}$$

Thus, we have:

$$\prod_{t=1}^{T} c_t . Pr(O|\lambda) = 1, \tag{II.7.42}$$

or

$$Pr(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}, \tag{II.7.43}$$

or

$$\ln Pr(O|\lambda) = -\sum_{t=1}^{T} \ln(c_t). \tag{II.7.44}$$

Thus, $\ln Pr(O|\lambda)$ can be computed, but not $Pr(O|\lambda)$ since it would be out of the dynamic range of the machine.

II.7.2 Discriminative training: Minimum Classification Error / GPD algorithm

For a C-class classification problem, each random sequence O is to be classified into one of the C classes. We denote these classes by $C_c, c = 1, 2, \ldots, C$. Each class c is modeled by an HMM λ_c. Let $\mathbb{O} = [O^{(1)}, \ldots, O^{(R)}]$ be a set of R sequences drawn from these C different classes and let $g_c(O)$ be a discriminant function associated with classifier c that indicates the degree to which O belongs to class c.

The classifier $\Gamma(O)$ defines a mapping from the sample space $O \in \mathbb{O}$ to the discrete categorical set $C_c, c = 1, 2, \ldots, C$. That is,

$$\Gamma(O) = I \quad \text{iff} \quad I = \underset{c}{\operatorname{argmax}}\, g_c(O). \tag{II.7.45}$$

The parameters of model λ_c can be optimized using several learning methods such as Baum-Welch [33], Segmental k-means [6], Maximum Mutual Information [34], MCE/GPD [32] algorithms etc..

In this dissertation, we are interested in the MCE/GPD algorithm. If sequence $O = [o_1, \ldots, o_T]$ belongs to class c, then

$$g_c(O, \Lambda) = \log[\max_q g_c(O, q, \Lambda)]. \tag{II.7.46}$$

In (II.7.46), q is a state sequence correspondent to the observation sequence O, Λ includes the models parameters, and

$$\begin{aligned} g_c(O, q, \Lambda) &= Pr(O, q; \lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}^{(c)} \prod_{t=1}^{T} b_{q_t}^{(c)}(o_t). \end{aligned} \tag{II.7.47}$$

II.7.2.1 Discrete HMM

For the discrete HMM, in (II.7.47), $b_{q_t}^{(c)}(o_t) = b_{q_t Q_V(o_t)}^{(c)}$, and Q_V is the quantization operation defined on an observation vector as

$$Q_V(o_t) = \underset{1 \le k \le M}{\operatorname{argmin}}\, d(o_t, v_k), \tag{II.7.48}$$

where $d()$ is a distance measure. In this work, $d()$ is taken as the Euclidean distance. Thus, $g_c(O, \Lambda) = \log[g_c(O, \bar{Q}, \Lambda)]$, where $\bar{Q} = (\bar{q}_0, \bar{q}_1, \ldots, \bar{q}_T)$ is the optimal state sequence that achieves $\max_Q g_c(O, Q, \Lambda)$, which could be computed using the Viterbi algorithm [29].

The misclassification measure of the sequence O is defined by:

$$d_c(O) = -g_c(O, \Lambda) + \log \left[\frac{1}{C-1} \sum_{j, j \neq c} \exp[\eta g_j(O, \Lambda)] \right]^{\frac{1}{\eta}} \tag{II.7.49}$$

where η is a positive number, $d_c(O) > 0$ implies misclassification and $d_c(O) \le 0$ means correct decision. When η approaches ∞, the term in the brackets becomes $\max_{j, j \neq c} g_j(O, \Lambda)$.

The misclassification measure is embedded in a smoothed zero-one function, referred to as loss function, defined as:

$$l_c(O; \Lambda) = l(d_c(O)), \tag{II.7.50}$$

where l is a sigmoid function, one example of which is:

$$l(d) = \frac{1}{1 + \exp(-\zeta d + \theta)}. \tag{II.7.51}$$

In (II.7.51), θ is normally set to zero, and ζ is set to a number larger than one. Correct classification corresponds to loss values in $[0, \frac{1}{2})$, and misclassification corresponds to loss values in $(\frac{1}{2}, 1]$. An

equivocal case occurs when $d_c(O) = 0$ or $l_c(O, \Lambda) = \frac{1}{2}$. The shape of the sigmoid loss function varies with the parameter $\zeta > 0$: the larger the γ, the narrower the transition region. Finally, for any unknown sequence O, the classifier performance is measured by:

$$l(O; \Lambda) = \sum_{c=1}^{C} l_c(O; \Lambda) \mathbb{I}(O \in C_c) \tag{II.7.52}$$

where $\mathbb{I}(.)$ is the indicator function.

For a set of training observation sequences O_r, $r = 1, 2, \ldots, R$, the empirical loss function on the entire data set is defined as

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R} \sum_{c=1}^{C} l_c(O; \Lambda) \mathbb{I}(O \in C_c). \tag{II.7.53}$$

The empirical loss above is then used to approximate the total misclassification error. The DHMM parameters are therefore estimated by minimizing $\mathbf{L}(\Lambda)$ using a gradient descent algorithm. In order to ensure that the estimated DHMM parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$ and $b_{ij} \geq 0$, $\sum_{j=1}^{M} b_{jk} = 1$, these parameters are typically mapped using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij} \tag{II.7.54}$$

$$b_{ij} \rightarrow \tilde{b}_{ij} = \log b_{ij} \tag{II.7.55}$$

The parameters are updated w.r.t to $\tilde{\Lambda}$. Then, after updating, the parameters are mapped back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}} \tag{II.7.56}$$

$$b_{ij} = \frac{\exp \tilde{b}_{ij}}{\sum_{j'=1}^{M} \exp \tilde{b}_{ij'}}. \tag{II.7.57}$$

Using a batch estimation mode, the DHMM parameters are iteratively updated using

$$\tilde{\Lambda}(\tau + 1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda) \Big|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}. \tag{II.7.58}$$

It can be shown [32] that the parameters $\tilde{\pi}_i^{(c)}$, $\tilde{a}_{ij}^{(c)}$, and $\tilde{b}_{jk}^{(c)}$ need to be updated using:

$$\tilde{a}_{ij}^{(c)}(\tau + 1) = \tilde{a}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}, \tag{II.7.59}$$

$$\tilde{a}_{ij}^{(c)}(\tau + 1) = \tilde{a}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)} \tag{II.7.60}$$

and

$$\tilde{b}_{ij}^{(c)}(\tau + 1) = \tilde{b}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ij}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)} \tag{II.7.61}$$

where

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O_r)}{\partial d_m(O_r)} \frac{\partial d_m(O_r)}{\partial g_c(O_r)} \frac{\partial g_c^{O_r}}{\partial a_{ij}^{(c)}} \frac{\partial a_{ij}^{(c)}}{\partial \tilde{a}_{ij}^{(c)}},$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ij}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O_r)}{\partial d_m(O_r)} \frac{\partial d_m(O_r)}{\partial g_c(O_r)} \frac{\partial g_c^{O_r}}{\partial b_{ij}^{(c)}} \frac{\partial b_{ij}^{(c)}}{\partial \tilde{b}_{ij}^{(c)}}.$$

The partial derivatives in (II.7.62) and (II.7.62) could be reduced to:

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} &= \sum_{r=1}^{R} \sum_{m=1}^{C} \sum_{t=1}^{T} \gamma l_m(O_r, \Lambda)(1 - l_m(O_r, \Lambda)) \times \\ &\quad \delta(q_t^r = i, q_{t+1}^r = j)(1 - a_{ij}^{(c)}) \frac{\partial d_c(O_r)}{\partial g_m(O_r, \Lambda)}, \end{aligned}$$

and

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{jk}^{(c)}} &= \sum_{r=1}^{R} \sum_{m=1}^{C} \sum_{t=1}^{T} \gamma l_m(O_r, \Lambda)(1 - l_m(O_r, \Lambda)) \times \\ &\quad \delta(q_t^r = j, Q_V(o_t^r) = k)(1 - b_{jk}^{(c)}) \frac{\partial d_c(O_r)}{\partial g_m(O_r, \Lambda)} \end{aligned}$$

In the above,

$$\frac{\partial d_c(O)}{\partial g_m(O, \Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j, j \neq c} \exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}, \tag{II.7.62}$$

and

$$\delta(i = j, k = l) = \begin{cases} 1 & \text{if } i = j \text{ and } k = l \\ 0 & \text{otherwise} \end{cases}. \tag{II.7.63}$$

II.7.2.2 Continuous HMM

For the continuous HMM, the function in (II.7.47) can be expanded to:

$$\begin{aligned} g_c(O, Q, \Lambda) &= P(O, Q; \lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}^{(c)} \prod_{t=1}^{T} b_{q_t}^{(c)}(o_t) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}^{(c)} \prod_{t=1}^{T} \sum_{j=1}^{M} u_{q_t j}^{(c)} b_{q_t j}^{(c)}(o_t) \end{aligned} \tag{II.7.64}$$

In the above, $b_{ij}(o_t) = \mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij})$ where $\mathcal{N}(o_t, \mu_{ij}, \Sigma_{ij})$ represents the normal density function with mean μ_{ij} and covariance Σ_{ij}. The covariance matrix Σ_{ij} is typically diagonal and $\Sigma_{ij} = \left[(\sigma_{ijd})^2\right]_{d=1}^{p}$. Thus, $g_c(O, \Lambda) = \log[g_c(O, \bar{Q}, \Lambda)]$, where $\bar{Q} = (\bar{q}_0, \bar{q}_1, \ldots, \bar{q}_T)$ is the optimal state sequence that achieves $\max_q g_c(O, q, \Lambda)$, which could be computed using the Viterbi algorithm [29], as for the discrete case.

Following similar steps as those outlined for the discrete case, we define the empirical loss function as:

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R} \sum_{c=1}^{C} l_c(O; \Lambda) \mathbb{I}(O \in C_c). \tag{II.7.65}$$

In the above, O_r, $r = 1, 2, \ldots, R$ represent a given set of training observation sequences.

Minimizing the empirical loss is equivalent to minimizing the total misclassification error. The CHMM parameters are therefore estimated by carrying out a gradient descent on $\mathbf{L}(\Lambda)$. In order to ensure that the estimated CHMM parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$ and $u_{ij} \geq 0$, $\sum_{j=1}^{M} u_{ij} = 1$ and $\mu_{ijd} \geq 0$ and $\sigma_{ijk} \geq 0$, these parameters are mapped using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij} \tag{II.7.66}$$

$$u_{ij} \rightarrow \tilde{u}_{ij} = \log u_{ij} \tag{II.7.67}$$

$$\mu_{ijd} \rightarrow \tilde{\mu}_{ij} = \frac{\mu_{ijd}}{\sigma_{ijd}} \tag{II.7.68}$$

$$\sigma_{ijd} \rightarrow \tilde{\sigma}_{ijd} = \log \sigma_{ijd} \tag{II.7.69}$$

Then, the parameters are updated w.r.t to $\tilde{\Lambda}$. After updating, the parameters are mapped back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}} \tag{II.7.70}$$

$$u_{ij} = \frac{\exp \tilde{u}_{ij}}{\sum_{j'=1}^{M} \exp \tilde{u}_{ij'}} \tag{II.7.71}$$

$$\mu_{ijd} = \tilde{\mu}_{ijd} \sigma_{ijd} \tag{II.7.72}$$

$$\sigma_{ijd} = \exp \tilde{\sigma}_{ijd} \tag{II.7.73}$$

Using a batch estimation mode, the CHMM parameters are iteratively updated using

$$\tilde{\Lambda}(\tau + 1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda) \Big|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}. \tag{II.7.74}$$

It can be shown [32] that the parameters $\tilde{a}_{ij}^{(c)}$, $\tilde{u}_{jk}^{(c)}$, $\tilde{\mu}_{ijd}^{(c)}$ and $\tilde{\sigma}_{ijd}^{(c)}$ need to be updated using:

$$\tilde{a}_{ij}^{(c)}(\tau + 1) = \tilde{a}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}, \tag{II.7.75}$$

$$\tilde{u}_{ij}^{(c)}(\tau+1) = \tilde{u}_{ij}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{v}_{ij}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{II.7.76}$$

$$\tilde{\mu}_{ijd}^{(c)}(\tau+1) = \tilde{\mu}_{ijd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijd}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{II.7.77}$$

and

$$\tilde{\sigma}_{ijd}^{(c)}(\tau+1) = \tilde{\sigma}_{ijd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijd}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{II.7.78}$$

where

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{a}_{ij}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))\delta(q_t^r = i, q_{t+1}^r = j)(1-a_{ij}^{(c)})\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)},$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{u}_{ij}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))u_{ij}^{(c)}(1-u_{ij}^{(c)})\delta(q_t,i)\frac{b_{q_t j}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)}\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)},$$

$$\begin{aligned}\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijd}^{(c)}} &= \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda)) \times \\ &\quad \sigma_{ijd}^{(c)}\delta(q_t,i)v_{ij}^{(c)}(o_{td}^{(r)} - \mu_{ijd}^{(c)})\frac{b_{q_t j}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)}\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)},\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijd}^{(c)}} &= \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda)) \times \\ &\quad \left[\sigma_{ijd}^{(c)}\right]^{-1}\delta(q_t,i)v_{ij}^{(c)}\left[(\frac{o_{td}^{(r)} - \mu_{ijd}^{(c)}}{\sigma_{ijd}^{(c)}})^2 - 1\right] \times \\ &\quad \frac{b_{q_t j}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)}\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)},\end{aligned}$$

In the above,

$$\frac{\partial d_c(O)}{\partial g_m(O,\Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}. \tag{II.7.79}$$

II.8 Initial estimates of HMM parameters

A key question in HMM is how to choose initial estimates of the HMM parameters to avoid local maxima of the likelihood function. Many ways of initialization have been proposed [6]. Examples include:

- Random: the HMM parameters are generated randomly from an uniform distribution.
- Manual segmentation: when the hidden states have a physical meaning, manual partitioning could be performed to split the data into the different states of the HMM and then the remaining parameters could be derived [35].
- Segmental k-means: starting form a random guess of the HMM parameters, and using the Viterbi algorithm to label the observation sequences, the segmental k-means clusters the sequences to learn the HMM parameters [6].

II.9 Chapter Summary

In this chapter, the general form of the HMM is introduced for both discrete and continuous probability distributions. The basic assumptions, as well the most general HMM topologies were described. Then we studied the three basic problems involved with any HMM. In particular, the classification problem is studied in details and the maximum likelihood and the discriminative learning algorithm were outlined.

CHAPTER III

Related work

We must learn our limits. We are all something, but none of us are everything.

Blaise Pascal

This chapter starts by introducing the problem of multi-modal temporal data analysis. It illustrates the importance of assigning relevance weights to the multiple sources of information. Then it sets the general hypothesis and assumptions of the present dissertation. Afterwards, a classification of the types of modalities is presented. The subsequent section surveys the existing approaches to combine multiple modalities/sources/streams for sequences in the context of hidden Markov models. We discuss and compare the hypotheses and assumptions of these methods and we highlight their limitations.

III.1 Introduction

For complex classification systems, data is usually gathered from multiple sources of information that have varying degrees of reliability. In fact, assuming that the different sources have the same relevance in describing all the data might lead to a suboptimal solution. The classification error accumulates and can be more severe for temporal data. In fact, in the context of hidden Markov models, and for most real applications, different modalities could contribute to the generation of the sequence.

In order to emphasize the importance of combining the outcome of multiple streams, we perform the following experiment. First, a 3-dimensional data set is generated. We assume that the data comes from two different classes and we use two normal distributions with means $\mu_1 = [2\ 2\ 2]$ and $\mu_2 = [4\ 4\ 4]$ and identity covariances Σ_1 and Σ_2. Let x, y, z denote the 3 dimensions of the generated data. This data set is displayed in figure 8 where points belonging to class 1 are displayed as red dots and class 2 are displayed as blue dots. To simulate the scenario where all features are not equally important in characterizing both classes, we corrupt the y feature of class 1

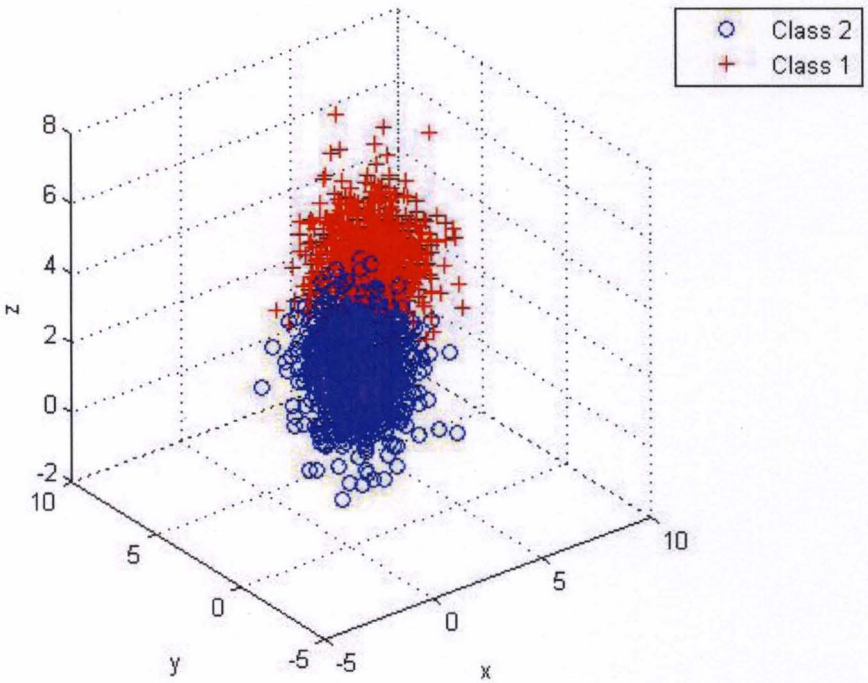
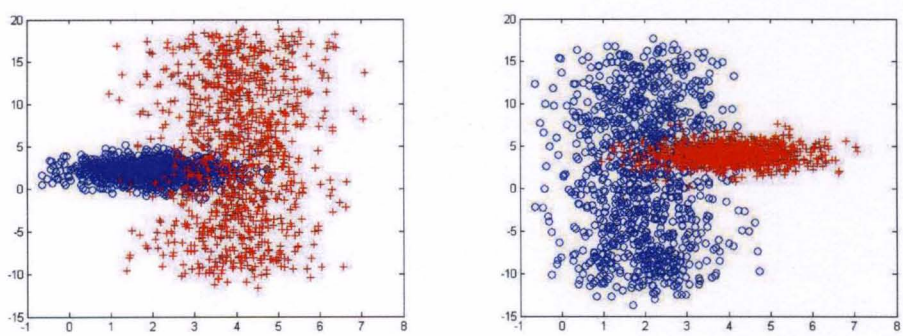


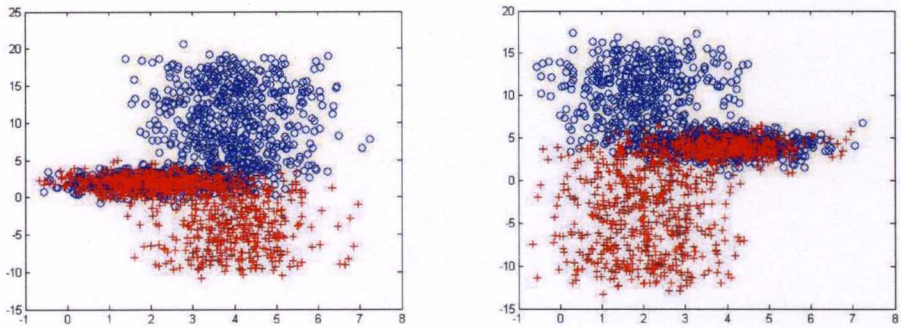
Figure 8. A two-class data set in the 3-dimensional feature space.



(a) (b)

Figure 9. Projection of the data in figure 8, corrupted by additive noise, on the x-y and x-z planes.

by adding random noise (uniformly distributed over the interval $[-14\ 14]$), thus, making this feature less relevant to this class. Similarly, we corrupt the z feature of class 2 by adding random noise (uniformly distributed over the interval $[-14\ 14]$). Figure 9 displays the corrupted data on the x-y and x-z planes. As it can be seen, the y feature is relevant for class 2 but not for class 1. Similarly, the z feature is relevant for class 1 but not class 2.



(a) (b)

Figure 10. Projection of the data in figure 9 partitioned by the EM algorithm on the x-y and x-z planes. Points assigned to cluster 1 are shown by '+' signs and points assigned to cluster 2 are shown by 'o' signs.

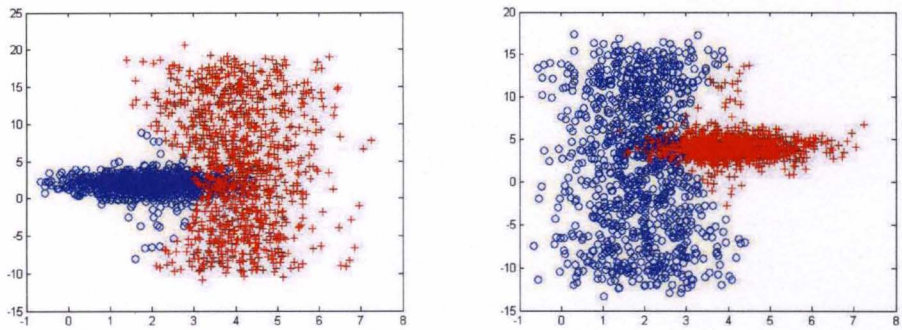
In the following experiment we ignore the ground truth of both classes and attempt to cluster them using the Expectation Maximization (EM) algorithm [30]. Like most clustering algorithms, the EM treats the three sources of information (features x, y, and z) equally important. Consequently, the EM cannot partition the data correctly. Figure 10 displays 2 projections of the clustered data where data assigned to different cluster are displayed with different symbols. As it can be seen, the EM fails to group sample from each class in a different cluster. This is mainly due to the fact that the x, y, and z features were assumed to have the same degree of relevance in both classes.

Ideally, if during the clustering process, the algorithm can learn that feature z is irrelevant to one of the clusters and that feature y is irrelevant to the other one, a better partition can be obtained. To illustrate this, we use an algorithm that can perform simultaneous clustering and feature weighting (SCAD) [36]. The partition obtained by SCAD is shown in Fig. 11. As it can be seen, SCAD achieves a clustering that is very close to the true distribution of the data in figure 9. This is mainly due to the cluster dependent feature relevance weights learned by SCAD . In fact, as we can see in table 1, the x and y features are given higher relevance weights for the cluster 1, and the x and z are given higher relevance weights for the other cluster.

TABLE 1

Feature relevance weights assigned to the two clusters

	X	Y	Z
cluster 1 ('o')	**0.299**	**0.5281**	0.1729
cluster 2 ('+')	**0.2977**	0.1745	**0.5279**



(a) (b)

Figure 11. Projection of the data in figure 9 partitioned by the SCAD algorithm on the x-y and x-z planes. Points assigned to cluster 1 are shown by 'o' signs and points assigned to cluster 2 are shown by '+' signs.

From the previous example, we can conclude that varying reliability of different attributes should be taken into account to achieve higher performance. Otherwise, assuming equal relevance for the different sources of information might lead to unreliable results. The degradation in performance can be more severe for sequential data. This is because the classification error can accumulate over the observations that form the sequence.

III.2 Information sources

In this work, we are interested in classifying sequential data that is gathered from multiple sources (or modalities or streams) that are synchronous and independent using an HMM classifier. Synchronicity means that at each time slot, we have access to the interpretation of each stream. The independence of the streams means that their interpretations of the original data and their generations of the sequences are performed independently.

The multiple sources of information usually represent heterogeneous types of data. Multi-modalities appear in several applications and could be broadly categorized into two groups. The first category consists of naturally available modalities that are intrinsical characteristics or interpretation of the raw data. An example of such modalities is the audio and video descriptors, used for automatic audio-video speech recognition (AAVSR) systems [20]. In fact, both speech and lips movement (possibly captured as video) are available when someone speaks. Natural modalities also appear in sign language recognition applications where multi-stream HMM, based on the hand positions and movements, has been used [37]. In fact, the position and the movement information are always available whenever the signer signs. In the second category, the modalities are synthe-

sized by several feature extraction techniques with different characteristics and expressiveness. They represent different (possibly independent) interpretations of the raw data. Such modalities include the Mel-frequency cepstral coefficients (MFCC) and formant-like features used to form automatic speech recognition (ASR) [38]. Synthesized modalities have also been used to combine upper contour features and lower contour features as two streams for off-line handwritten word recognition [39]. For both classes, the modalities could be synchronous or asynchronous. They can also represent independent interpretations of the raw data, or correlated ones.

III.3 Related work

III.3.1 Multi-modality information fusion using HMM

Approaches toward the combination of different modalities can be divided into three main categories: feature level fusion or direct identification, decision level fusion or separate identification (also known as late integration) and model level fusion (intermediate integration) [14].

III.3.1.1 Feature level fusion

In feature level fusion, a single HMM model is trained on the concatenated vector of the multiple features generated by different modalities [15]. In practice, the resulting feature vector can be large, causing inadequate modeling due to the curse of dimensionality and insufficient data. An appropriate transformation can remedy this, such as the projection of the concatenated vector to a lower dimensional vector while seeking the best discrimination among the different classes [40]. Figure 12 displays a diagram outlining the steps of the feature level fusion. This type of fusion has

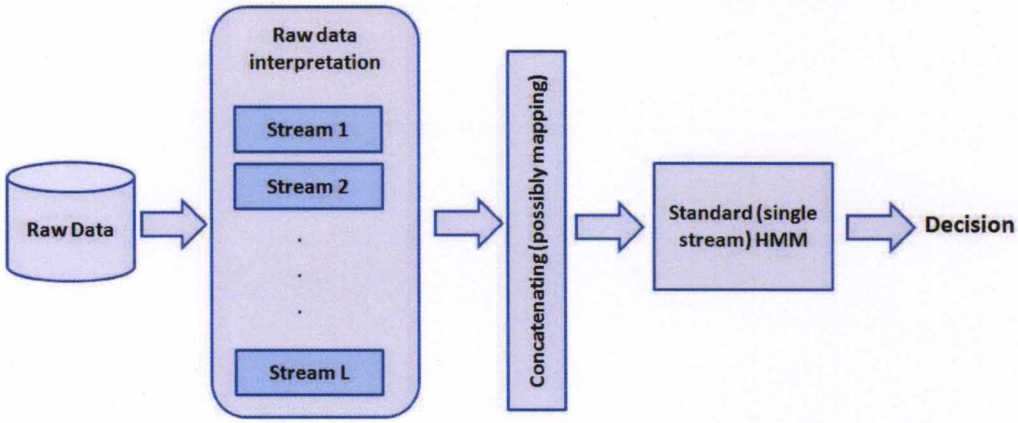


Figure 12. Diagram of the feature level fusion steps.

the drawback of treating heterogeneous features equally important. It also cannot easily represent the loose timing synchronicity between different modalities.

III.3.1.2 Decision level fusion

In decision level fusion, the modalities are processed separately to build independent models [16]. Learning the models corresponding to the different modalities is followed by an additional layer that combines the multiple decisions into a final one. The combination may apply to classification labels only, or to the class-specific continuous valued outputs of the individual experts [41]. In the latter case, classifier outputs are often normalized to the $[0, 1]$ interval, and these values are interpreted as the support given by the classifier to each class, or even as class-conditional posterior probabilities [41]. Such interpretation allows forming an ensemble through algebraic combination rules (majority voting, maximum/minimum/sum/product or other combinations of posterior probabilities) [42], fuzzy integral [43], Dempster-Shafer based classifier fusion [44], and more recently, decision templates [41]. Figure 13 displays a diagram outlining the steps of the decision level fusion. This approach assumes that the streams are completely independent and evolve asynchronously. In particular, it completely ignores the correlation between features and allows complete asynchrony between the streams. Also, it is computationally heavy since it involves two layers of decision.

III.3.1.3 Model level fusion

In model level fusion, an HMM model that is more complex than a standard one is sought. This additional complexity is needed to handle the correlation between modalities, and the loose synchronicity between sequences. Several HMM structures have been proposed for this purpose. Examples include factorial HMM [17], coupled HMM [18] and Multi-stream HMM [19].

Figure 14 displays a diagram outlining the main steps of the model level fusion. Figure 15 illustrates the factorial HMM as a graphical model [45]. This model has two streams having three states each. The states of each stream emit altogether one observation. This architecture allows for asynchrony between sequences since the different streams are assigned separate state sequences. This is performed at the expense of an approximate parameter estimation. In fact, the parameters of factorial and coupled HMMs could be estimated via the EM (Baum-Welch) algorithm [6]. However, the E- step is computationally intractable and approximation approaches such as Gibbs sampling, variational methods (mean field approximation) [17] are used instead. In addition, for each time slot, multiple states contribute to the generation of the observation vector. However, the contribution of

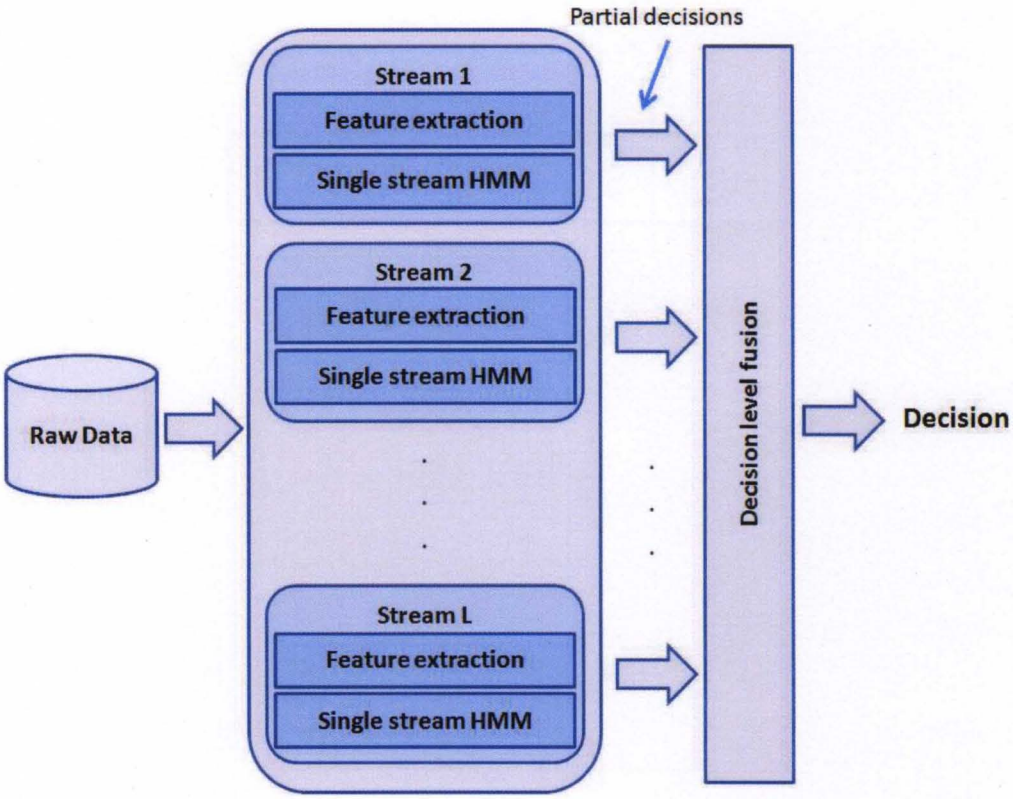


Figure 13. Diagram of the decision level fusion steps.

each stream's state is not explicit and is an absent information.

Figure 16 shows the graphical model representation of a coupled HMM with two streams having three states each. Each state of each stream emits one observation. Similarly to the factorial HMM, this architecture allows for asynchrony between sequences since the different streams are assigned separate state sequences. The complexity of this architecture increases as the number of chains in the coupled HMM increases. In particular, for a large number of chains, the E-step becomes intractable and approximation for inference, such as the N-heads algorithm [46], may be needed.

Multi-stream HMM (MSHMM) is an HMM based structure that handles multiple modalities for temporal data. It is used when the modalities (streams) are synchronous and independent. Since the streams are supposed to be synchronous, MSHMM assumes that for each time slot, there is a single hidden state, from which different streams interpret different observations. The independence of the streams means that their interpretation of the hidden state and their generation of the observations is performed independently.

Figure (17) shows the graphical model representation of a multi-stream HMM with three

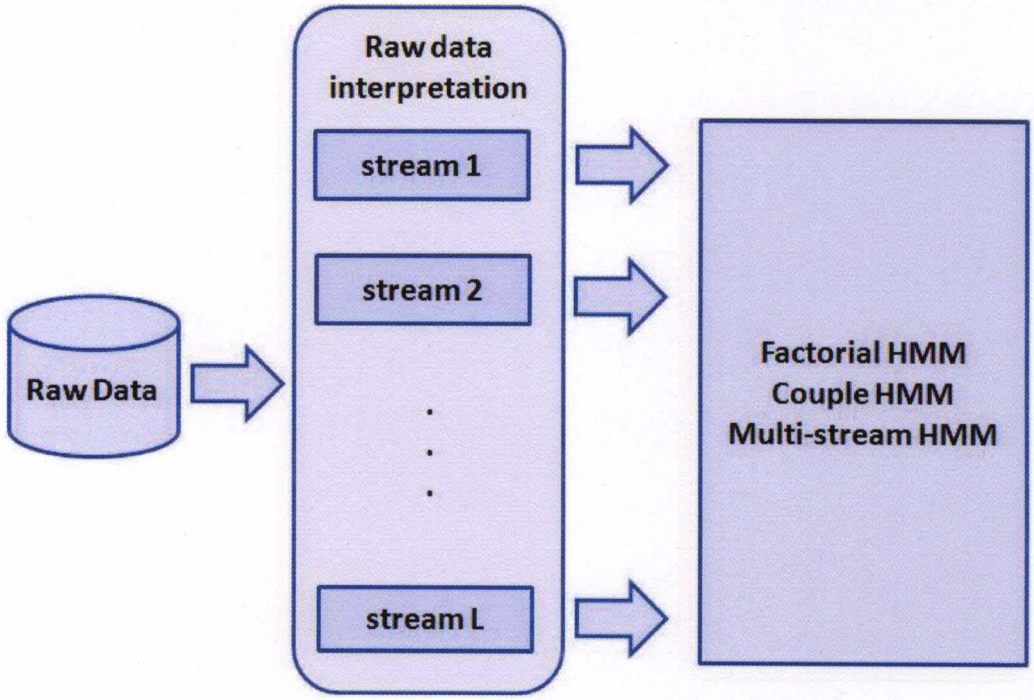


Figure 14. Diagram of the model level fusion steps.

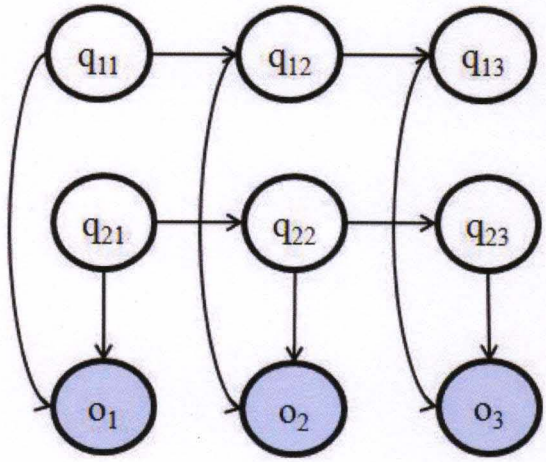


Figure 15. A graphical representation of a Factorial HMM with 2 streams, having 3 states each, and the states of each stream emit one observation.

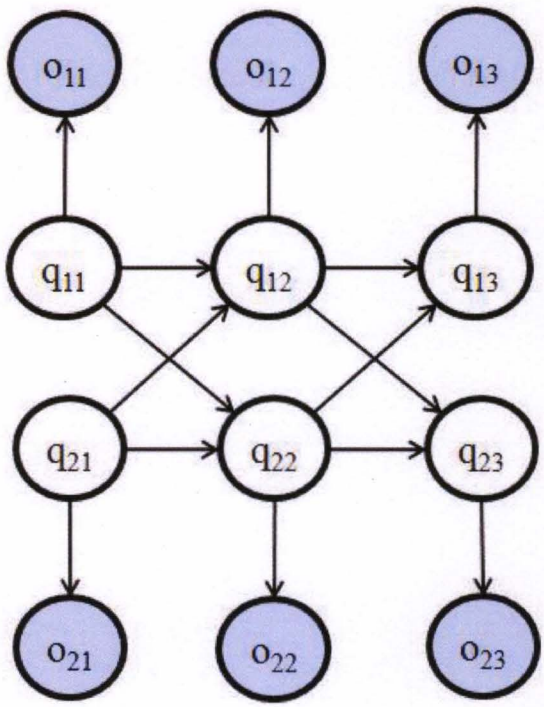


Figure 16. A graphical representation of a coupled HMM with 2 streams, having 3 states each, and each state emit one observation.

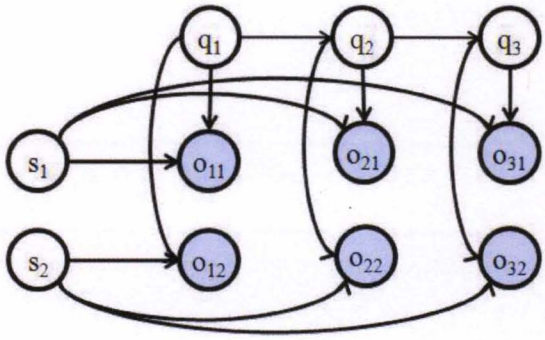


Figure 17. A graphical representation of a multi-stream HMM with 3 states and 2 streams, each stream generates an observation vector within each state.

states and two streams. Each stream generates an observation vector. More generally, multi-stream HMM (MSHMM) is considered as an HMM based structure that handles multiple modalities for temporal data. It is used when the modalities (streams) are synchronous and independent. Since the streams are supposed to be synchronous, MSHMM assumes that for each time slot, there is a single hidden state, from which different streams interpret different observations. The independence of the streams means that their interpretation of the hidden state and their generation of the observations is performed independently.

III.3.2 Multi-stream HMM

Few varieties of MSHMM have been proposed in the literature to address stream relevance weighting to discriminate between the audio and visual streams in speech recognition using continuous HMM [27, 24]. In these methods, the feature space is partitioned into different subspaces generated by different streams, and different probability density functions (pdf) are learned for the different spaces. The relevance weights for each subspace or stream could be fixed a priori by an expert [19], or learned via Minimum Classification Error/Generalized Probabilistic Descent (MCE/GPD) [24]. In [27], the authors have adapted the Baum-Welch algorithm [33] to learn the stream relevance weights. However, to derive the maximum likelihood equations, the model was restricted to include only one Gaussian component per state. The stream relevance weighting has been introduced within the pdf formula characterizing the continuous HMM.

III.3.2.1 Architecture of existing MSHMM

Two approaches have been proposed for the MSHMM: the mixture level weighting, and state level weighting.

III.3.2.1.1 Mixture level weighting This approach consists of factorizing each mixture into the product of weighted streams related pdf(s) [24]. In particular, the probability density of an observation o_t with respect to a state j is defined as:

$$b_j(o_t) = \sum_{k=1}^{M} u_{jk} \prod_{l=1}^{L} \left[\phi(o_t^{(l)}, \mu_{jkl}, \Sigma_{jkl})\right]^{w_{jkl}}, \tag{III.3.1}$$

subject to:

$$\sum_{k=1}^{M} u_{jk} = 1 \text{ and } \sum_{l=1}^{L} w_{jkl} = 1. \tag{III.3.2}$$

In (III.3.1), u_{jk} is the mixing coefficient of the kth component of state j, w_{jkl} is the exponent stream weight of stream l, in the kth component and ith state. The function $\phi(o_t^{(l)}, \mu_{jkl}, \Sigma_{jkl})$ is a probability density function describing the lth stream with mean μ_{jkl} and covariance Σ_{jkl}.

The geometric form in (III.3.1) is motivated by the following probabilistic reasoning:

$$\begin{aligned}
b_i(o_t) &= Pr(o_t|q_t = i; \lambda) \\
&= \sum_{j=1}^{M} Pr(o_t|q_t = i, e_t = j; \lambda) Pr(e_t = j|q_t = i; \lambda) \\
&= \sum_{j=1}^{M} Pr(e_t = j|q_t = i; \lambda) Pr(o_t^{(1)}, \cdots, o_t^{(1)}|q_t = i, e_t = j; \lambda) \\
&= \sum_{j=1}^{M} Pr(e_t = j|q_t = i; \lambda) \prod_{k=1}^{L} Pr(o_t^{(k)}|q_t = i, e_t = j; \lambda) \\
&\approx \sum_{j=1}^{M} Pr(e_t = j|q_t = i; \lambda) \prod_{k=1}^{L} \left[Pr(o_t^{(k)}|q_t = i, e_t = j; \lambda) \right]^{w_{ijk}}
\end{aligned}$$

where e_t is a random variable that represents the index of the stream that occurs in time t. Notice that (III.3.1) does not represent a probability distribution in general, and was therefore referred to as "score".

III.3.2.1.2 State level weighting This formulation considers the pdf as a product of exponent weighted mixture of Gaussians [25]. In this case, the pdf is a product of summation of Gaussians, whereas in the former case, the pdf is the summation of a product of Gaussians. In particular, the probability density of an observation o_t in a state j is defined as:

$$b_j(o_t) = \prod_{l=1}^{L} \left[\sum_{k=1}^{M} u_{jlk} \phi(o_t^{(l)}, \mu_{jlk}, \Sigma_{jlk}) \right]^{w_{jl}}, \tag{III.3.3}$$

subject to:

$$\sum_{k=1}^{M} u_{jlk} = 1 \text{ and } \sum_{l=1}^{L} w_{jl} = 1. \tag{III.3.4}$$

As in (III.3.1), we note that (III.3.3) does not represent a probability distribution in general, and is also referred to as "score".

The geometric form in (III.3.3) is motivated by the following probabilistic reasoning:

$$\begin{aligned}
b_i(o_t) &= Pr(o_t|q_t = i;\lambda) \\
&= Pr(o_t^{(1)}, \cdots, o_t^{(L)}|q_t = i;\lambda) \\
&= \prod_{k=1}^{L} Pr(o_t^{(k)}|q_t = i;\lambda) \\
&\approx \prod_{k=1}^{L} \left[Pr(o_t^{(k)}|q_t = i;\lambda)\right]^{w_{ik}} \\
&= \prod_{k=1}^{L} \left[\sum_{j=1}^{M} Pr(o_t^{(k)}|q_t = i, e_t = j;\lambda)\right]^{w_{ik}}
\end{aligned}$$

where e_t is a random variable that represents the index of the component that occurs in time t.

III.3.2.2 Parameter estimation of the MSHMM parameters

Due to the form of the emission scores in (III.3.1) and (III.3.3), the stream exponents cannot be obtained by maximum likelihood estimation [25, 47]. In this case, It was shown in [24] that it is not possible to derive the maximum likelihood learning equation for the exponent weights. Thus, the exponent weights are learnt via MCE/GPD approach as explained in the previous section, and the remaining HMM parameters are estimated independently by means of traditional maximum likelihood techniques [26]. The only attempt for exponent weights equation learning within Baum-Welch was reported in [27]. However, this alternative solution restricts the HMM structure to only one Gaussian component per state. In particular, the authors in [27] have used a pdf to model two streams (audio, visual) through the following form:

$$b_j(o_t) = \prod_{s=1}^{2} \left(b_{js}(o_s^t)\right)^{w_{js}}, \tag{III.3.5}$$

subject to the constraint:

$$\sum_{s=1}^{2} (w_{js})^m = K, \tag{III.3.6}$$

where m and K are constants.

It can be shown that using (III.3.5) within the Baum-Welch algorithm leads to the following equation to update the feature relevance weights:

$$w_{js} = \left\{ K \frac{\left(\sum_{t=1}^{T} \alpha_j(o_t)\beta_j(o_t)\ln(b_{js}(o_t^s))\right)^{\frac{m}{m-1}}}{\sum_{s=1}^{2}\left(\sum_{t=1}^{T} \alpha_j(o_t)\beta_j(o_t)\ln(b_{js}(o_t^s))\right)^{\frac{m}{m-1}}} \right\}^{\frac{1}{m}} \tag{III.3.7}$$

An alternative constraint to the one in (III.3.6) is to use:

$$\sum_{s=1}^{2} m^{w_{js}} = K. \tag{III.3.8}$$

In this case, it can be shown that the feature relevance weights need to be updated using:

$$w_{js} = \frac{1}{\ln m} \ln \left(K \frac{\left(\sum_{t=1}^{T} \alpha_j(o_t)\beta_j(o_t) \ln(b_{js}(o_t^s))\right)^{\frac{m}{m-1}}}{\sum_{s=1}^{2} \left(\sum_{t=1}^{T} \alpha_j(o_t)\beta_j(o_t) \ln(b_{js}(o_t^s))\right)^{\frac{m}{m-1}}} \right) \tag{III.3.9}$$

For the general structure where more than one component per state is needed, two sequential learning steps are needed. The first step consists of a maximum likelihood based learning using the standard Baum-Welch algorithm in order to learn the parameters π, $\mathbf{A}$, and $\mathbf{B}$. These parameters could be updated by running the standard Baum-Welch on the concatenation of the observations generated by the different streams. Alternatively, different sets of parameters could be learned from the different streams via the standard Baum-Welch, and then averaged to form values of the parameters.

The second learning step consists of the estimation of the stream relevance weights. These weights could be fixed using a priori knowledge. Alternatively, they could be learned using discriminative training techniques. Some of these methods seek to minimize a smooth function of the resulting multi-stream HMM on the data, and employ the generalized probabilistic descent (GPD) algorithm [32] for stream exponent estimation [25]. In fact, each parameter ϕ_s that we wish to optimize is iteratively re-estimated in order to minimize a cost function $\mathcal{L}$ representing the classification error. At iteration k, ϕ_s is updated by gradient descent of the cost function, i.e.,

$$\phi_{s,k} = \phi_{s,k-1} - \eta \left. \frac{\partial \mathcal{L}}{\partial \phi_s} \right]_{\Phi_{k-1}}, \tag{III.3.10}$$

where η is the learning rate. If we assume that we have a set of training samples $\{O_1, \ldots, O_S\}$, and there is a set of classes $\{\lambda_1, \ldots, \lambda_C\}$, the cost function can be defined as:

$$\mathcal{L} = \frac{1}{S} \sum_{m=1}^{S} l_m(O_m), \tag{III.3.11}$$

where $l_m(O_m)$ is the cost function for the event O_m. Typically, it is defined as a sigmoid function of an error measure $d_m(O_m)$,

$$l_m(O_m) = \frac{1}{1 + \exp[-\alpha d_m(O_m)]}, \tag{III.3.12}$$

where α is the transition parameter from correct to incorrect classification. The error measure $d_m(O_m)$ is defined as

$$d_m(O_m) = -g_k(m) + \frac{1}{\beta} \ln \left[\frac{1}{C-1} \sum_{j \neq k(m)} \exp(\beta g_j) \right], \tag{III.3.13}$$

where $g_i(m) = g_i(O_m, \lambda_i)$ is the discriminant functions and $\lambda_{k(m)}$ is the correct class for sample O_m. It can be shown that the discriminant is given by,

$$g_i(O_m, \lambda_i) = \frac{1}{T}\sum_{t=1}^{T} \ln\left[\sum_k v_{jk} \times \prod_{l=1}^{L}[\phi(o_t^{(l)}, \mu_{jkl}, \Sigma_{jkl})]^{w_{jkl}}\right] \tag{III.3.14}$$

Other techniques use maximum mutual information (MMI) training [34, 48], or the maximum entropy criterion [49].

III.3.3 Limitations of existing methods

Even though existing MSCHMM structures can outperform the baseline HMM, they are not general enough and they have several limitations. In particular, there is no solution for the discrete case. In addition, existing multi-stream continuous HMMs have the following limitations:

1. They do not provide an optimization framework that learns all the HMM parameters simultaneously. In general, a two step training approach is needed. First, the Baum-Welch learning algorithm is used to learn the parameters of the HMM relative to each subspace. Then, the MCE/GPD algorithm is used to learn the relevance weights. Typically, this is not due to the desirable minimization of the classification error, but rather to the difficulty that arises when using the proposed pdf within the Baum-Welch learning algorithm.

2. The only approach that extends the Baum-Welch learning was derived for the special case that restricts the number of components per state to one. This can be too restrictive for most real applications.

3. Since the MCE/GPD learning algorithm usually comes after a layer of ML learning (e.g., Baum-Welch) to minimize the miss-classified cases, the feature relevance weights trained with the MCE/GPD approach only may not correspond to local minima of the ML optimization. Thus, the learned feature relevance weights may not achieve their objective.

To overcome the above limitations, we propose a generic approach that integrates stream discrimination within the HMM classifier. Our proposed solution can be used for both the continuous and discrete cases. All the parameters of the proposed model could be optimized simultaneously.

III.4 Chapter Summary

In this chapter, multi-modal temporal data is introduced. Synchronicity and independence are set as the underlying assumptions on the nature of the modalities generating the temporal data

studied in this dissertation. The state of the art is surveyed for techniques based on hidden Markov models that tackles the classification of such temporal data. In particular, multi-stream hidden Markov models (MSHMMs) are the underlying machine used for temporal data generated from synchronous and independent streams. A sketch of the MSHMM structures has been presented as well as their limitations.

CHAPTER IV

Generalized Multi-stream Discrete Hidden Markov Models

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

One of the limitations of the state of the art in discrete HMM is the absence of any treatment for the multi-stream case. In this chapter, we propose various multi-stream Discrete HMM (MSDHMM) structures that integrate stream relevance weights. For each structure, we generalize the Baum-Welch and the MCE/GPD training algorithms. In particular, we generalize the objective function to include the stream relevance weights and derive the necessary conditions to update the parameters. We assume that we have L streams of information. These streams could have been generated by different sensors and/or different feature extraction algorithms. Each stream is thus represented by a different subset of features. Instead of treating the streams equally important or using user-specified weights, the proposed MSDHMM structure integrates an additional component to learn a relevance weight for each stream. We propose two different data driven methods to learn the relevance weights. The first one is based on distance weighting and the second one is based on probability weighting. In the distance based approach, a weight is assigned to each feature subset (i.e., each stream), and the distance computation between samples becomes a weighted aggregation of the partial distances from the different streams. In the probability based approach, a partial probability is assigned to each stream of each symbol and the overall observation probability of each symbol is computed as an aggregation between the stream relevance weights and the partial probabilities. For the probability based approach, we propose linear and geometric aggregations methods.

IV.1 A Distance-based approach to learn multi-stream relevance weights

The proposed distance based MSDHMM (referred to as MSDHMM^{D}) structure is defined as:

$$\lambda = (\pi, \mathbf{A}, \mathbf{B}, \mathbf{W}) \tag{IV.1.1}$$

where π, $\mathbf{A}$ and $\mathbf{B}$ are the state prior probabilities, the transition probabilities and the observation probabilities respectively. These are the same parameters used in the baseline discrete HMM structure. The additional parameter $\mathbf{W} = [w_{jk}]$ is an $M \times L$ matrix that represents the relevance weight of each symbol with respect to each stream. In particular, a stream relevance weight w_{jk} is assigned to each symbol j to indicate the relevance of stream k for this symbol. The proposed structure assumes a dependency between the streams and the states.

From a graphical model perspective, a MSDHMM^{D} could be represented by a graph as shown in figure 18. This figure displays a MSDHMM^{D} with 3 states and 2 streams. As illustrated, the streams (in red and green) generate observations independently. For instance, for state 1, the generated observations o_{11} and o_{12} are generated by stream 1 (red) and stream 2 (green) respectively and are two different interpretations of the hidden state q_1. Moreover, the two observations are available at the same time. This makes the two streams synchronous.

Optimization of MSDHMM^{D} parameters can be achieved in two steps. The first step combines the initialization and the learning of $\mathbf{W}$. The second step uses the standard Baum-Welch algorithm [6] to learn the $\mathbf{A}$ and $\mathbf{B}$ parameters.

For each MSDHMM^{D} model, λ_c, the initialization step consists of learning the N_s states, learning the codebook, and assigning initial probabilities to each symbol. The states and the codebook could be obtained by partitioning and quantizing the training data. Any clustering algorithm, such as the k-means [50] or the fuzzy c-means [51] could be used for this task. In our application we use the Simultaneous Clustering and Attribute Discrimination (SCAD) [36]. SCAD can perform clustering and feature weighting simultaneously and in an unsupervised manner. It learns a feature relevance weight for each feature subset in each cluster. More details of the SCAD algorithm are given in Appendix (B). The feature relevance weights learned by SCAD have two main advantages. First, they guide the clustering process in identifying more meaningful clusters by identifying clusters in subspaces of the original high dimensional feature space. Second, the learned feature weights could be used subsequently as the relevance weights of the symbols with respect to the different streams.

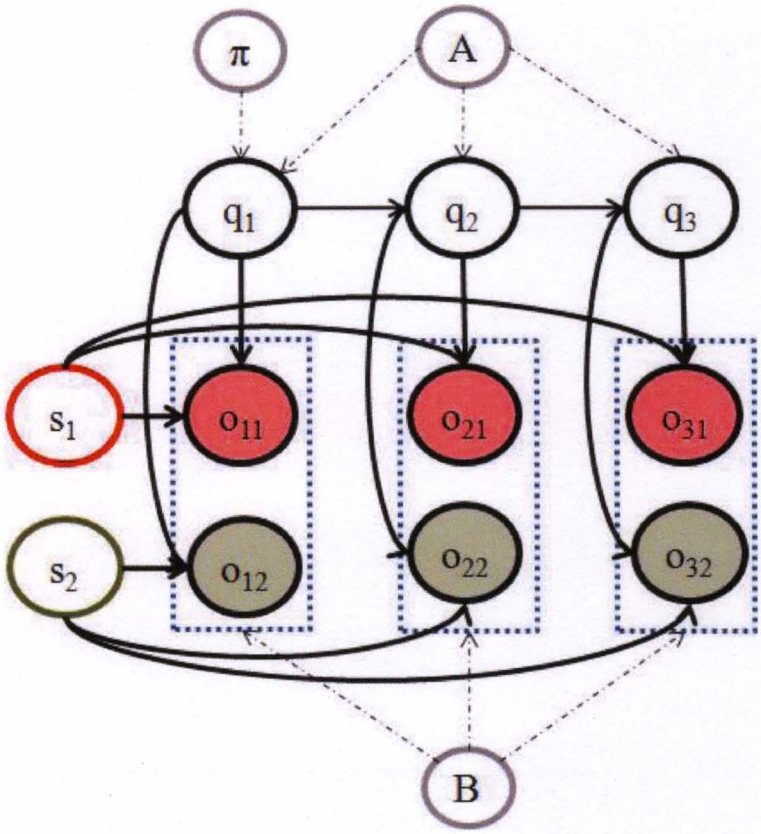


Figure 18. A Multi stream DHMM with 3 states and 2 streams.

First, using SCAD, we partition the training data into N_s clusters that correspond to the N_s states. A representative vector, s_j (centroid of cluster j) is selected as the mean vector of each state. In this step, we use our prior knowledge about the features and the expected HMM structure to fix the number of clusters and initialize them. Second, we use SCAD to cluster the training data into a larger number of clusters (M) and learn the codebook. In other words, we used SCAD to initialize the codebook $\mathbf{V}$ and the feature relevance weights $\mathbf{W}$ associated with each symbol.

Let d_{ij}^k be the partial distance between data vector x_j and cluster i with respect to the kth stream. Note that the distance d_{ij}^k is not required to be the Euclidean distance. Moreover, different distance measures could be used for different streams. We only require the different measures to be normalized to yield values within the same dynamic range. The total distance, d_{ij}, between x_j and cluster i is then computed by aggregating the partial degrees of similarities and their weights. That is, we let

$$d_{ij}^2 = \sum_{k=1}^{L} w_{ik}(d_{ij}^k)^2. \tag{IV.1.2}$$

SCAD is an iterative algorithm. It starts with an initial set of centers $v_j^{(0)}$ and an initial set of equal weights $w_{jk}^{(0)} = \frac{1}{L}$. In each iteration, SCAD would first compute a partial degree of similarity of each subset of features, and update the degree of relevance of each subset in each center. After few iterations, SCAD would converge to the optimal clusters' prototypes and weights that minimize the sum of intra-cluster distances. Let v_j represent the center of each of the M clusters, and let w_{jk} represent the learned stream relevance of each cluster. After learning the codebook, the DHMM requires associating a probability value with each symbol in each state. The probability of v_j in state i represents its likelihood in that state. We use an FCM-type [51] membership function to initialize these probabilities, i.e., we let:

$$b_{ij} = \frac{1/d^2(v_j, s_i)}{\sum_{k=1}^{N_s} 1/d^2(v_j, s_k)}, \tag{IV.1.3}$$

where $d(v_j, s_i)$ is the distance d_{ij} defined in (IV.1.2), and the correspondent partial distance, $d^k(v_j, s_i)$ is the L^2 norm. The closer v_j is to s_i, the higher its likelihood is in state i, which explain the usage of the inverse of the distance in the numerator of (IV.1.3). The denominator in (IV.1.3) is a normalizing factor. Expanding (IV.1.3) to include the partial distances and their relevance weights, we obtain:

$$b_{ij} = \left[\sum_{m=1}^{N_s} \frac{\sum_{k=1}^{L} w_{jk} \|v_j^k - s_j^k\|^2}{\sum_{k=1}^{L} w_{jk} \|v_j^k - s_m^k\|^2} \right]^{-1}. \tag{IV.1.4}$$

To satisfy the requirement that $\sum_{j=1}^{M} b_{ij} = 1$, we scale b_{ij} using:

$$b_{ij} \longleftarrow \frac{b_{ij}}{\sum_{k=1}^{M} b_{ik}}. \tag{IV.1.5}$$

After the initialization step, the DHMM model parameters $\mathbf{A}$, $\mathbf{B}$ and π are then estimated using the standard Baum-Welch algorithm [6] as outlined in chapter II with a minor modification. Recall that the learning equation of b_{ij} in the discrete Baum-Welch is:

$$b_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i) \delta(Q_V(o_t), j)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{IV.1.6}$$

where $\gamma_t(i)$ and $\delta(.,.)$ are as defined in (II.7.6) and (II.7.63) respectively.

In (IV.1.6), Q_V is the quantization operation defined on an observation vector o_t as the index of its closest symbol. In our case, to identify the closest symbol to an observation, we take advantage of the stream relevance weights associated with each symbol. That is, the closest symbol to o_t is the symbol which index $Q_V(o_t)$ satisfies:

$$Q_V(o_t) = \underset{1 \le j \le M}{\operatorname{argmin}} \sum_{k=1}^{L} w_{jk} \|o_t^k - v_j^k\|^2. \tag{IV.1.7}$$

In (IV.1.7), w_{jk} emphasizes the contribution of the stream k in the decision of the closest symbol to o_t. Thus, these learned weights affect the b_{ij} update equation in the Baum-Welch algorithm.

The steps of the resulting training procedure for the parameters of λ are outlined in Algorithm (1). The stopping criterions are either the likelihood $Pr(O|\lambda)$ becomes less than a threshold, or the number of iterations exceeds a predefined limit. In this version of the MSDHMM^{D}, the stream

Algorithm 1 Baum-Welch Training of the Distance-based MSDHMM

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L.
Ensure:
1: Cluster training data into N_s clusters using SCAD, and let s_i, the center of each cluster, be the representative of state i
2: Cluster the training data (using SCAD) to quantize it into M symbols and learn the stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: **while** stopping criteria not satisfied **do**
4: Compute the closest observation to o_t using (IV.1.7);
5: update $\mathbf{A}$ using (II.7.9);
6: update $\mathbf{B}$ using (II.7.10);
7: **end while**

relevance weights are learned during the initial clustering step and are not updated in the HMM parameter learning. In addition, the discriminative training version of this MSDHMM is carried out in the same way as the baseline DHMM, using the quantization operation in (IV.1.7). Algorithm (2) outlines the steps needed to learn the parameters of all the models λ_c using the MCE/GPD framework. The distance weighting approach provides a simple structure of the multi-stream dis-

Algorithm 2 MCE/GPD Training of the Distance-based MSDHMM

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L for each model λ_c.
Ensure:
1: For each λ_c, cluster training data into N_s clusters using SCAD, and let s_i, the center of each cluster, be the representative of state i.
2: For each λ_c, cluster the training data (using SCAD) to quantize it into M symbols and learn the stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: **while** stopping criteria not satisfied **do**
4: Compute the closest observation to each o_t using (IV.1.7);
5: Compute the loss function of each sequence O using (II.7.52);
6: update $\mathbf{A}$ of each λ_c using (II.7.60);
7: update $\mathbf{B}$ of each λ_c using (II.7.61);
8: **end while**

crete HMM. However, it has two main limitations. First, the stream weights are independent of the states. Second, the weights are learned independently from the rest of the DHMM parameters and do not necessarily maximize the Likelihood estimates. To overcome these limitations, we propose an alternative approach that is based on assigning partial probabilities to the different streams.

IV.2 A Probability based approach to learn multi-stream relevance weights

Similarly to the MSDHMM^{D}, the probability based MSDHMM has the following compact representation:

$$\lambda = (\pi, \mathbf{A}, \mathbf{B}, \mathbf{W}) \tag{IV.2.1}$$

However, in this case, $\mathbf{W}$ is an $N_s \times M \times L$ stream relevance weight matrix. In particular, we assume that a stream relevance weight w_{ijk} is assigned to each symbol j of each stream k within each state i. This choice takes into account the additional dependency between the streams and the states. We refer to this new structure of MSDHMM as MSDHMM^{P}. The graphical representation of this model is shown in figure 19 where an MSDHMM^{P} with 3 states and 2 streams is illustrated. This diagram is similar to the one in figure 19. The only difference here is that streams 1 and 2 depend also on the hidden states.

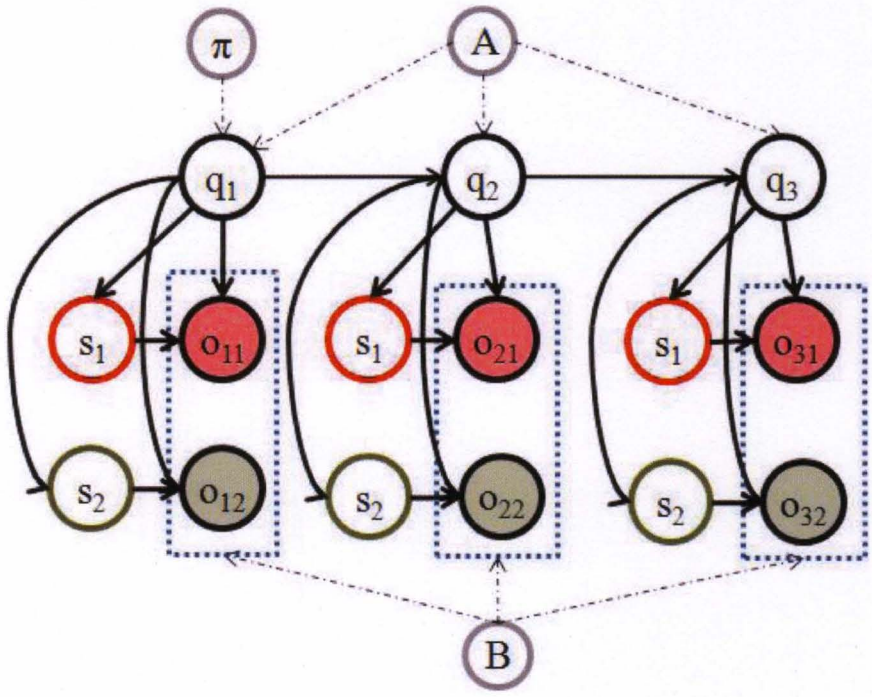


Figure 19. A Multi stream DHMM with 3 states and 2 streams.

In the proposed MSDHMM^{P} structure, each symbol j of each stream k, i.e. v_j^k, is assigned a partial probability b_{ijk} in each state i. The partial probability b_{ijk} measures the likelihood of v_j^k in state i. The stream relevance weights w_{ijk} and the probabilities b_{ijk} are combined to form the observation state probabilities b_{ij}. Two different combination methods are proposed. The first one

uses a linear combination of the weights and the partial probabilities, while the second one uses a geometric combination.

IV.2.1 Linear aggregation

In this method, the observation state probabilities are computed using:

$$b_{ij} = \sum_{k=1}^{L} w_{ijk} b_{ijk}, \tag{IV.2.2}$$

subject to

$$\sum_{k=1}^{L} w_{ijk} = 1, \text{ and } \sum_{j=1}^{M} b_{ijk} = 1. \tag{IV.2.3}$$

This linear form of the observation probability in (IV.2.2) is motivated by the following probabilistic reasoning:

$$\begin{aligned} b_{ij} &= Pr(o_t|q_t = i; \lambda) \\ &= \sum_{k=1}^{L} Pr(v_j|q_t = i, f_t = k; \lambda) Pr(f_t = k|q_t = i; \lambda) \\ &= \sum_{k=1}^{L} Pr(v_j^{(1)}, \cdots, v_j^{(L)}|q_t = i, f_t = k; \lambda) Pr(f_t = k|q_t = i; \lambda) \\ &\approx \sum_{k=1}^{L} Pr(v_j^{(k)}|q_t = i, f_t = k; \lambda) Pr(f_t = k|v_j, q_t = i; \lambda) \end{aligned}$$

where f_t is a random variable representing the index of the stream that occurs in time t. It follows then that:

$$b_{ijk} = Pr(v_j|q_t = i, f_t = k; \lambda),$$

and

$$w_{ijk} = Pr(f_t = k|v_j, q_t = i; \lambda).$$

We will refer to this probability based MSDHMM with linear aggregation as $\text{MSDHMM}^{\text{P}_1}$. In (IV.2.2), the higher the value of w_{ijk}, the more the kth stream contributes to the overall probability of v_j in the state i.

For a C-class classification problem, each random sequence O is to be classified into one of the C classes. Each class, c, is modeled by a DHMM λ_c. Let $\mathbb{O} = [O^{(1)}, \ldots, O^{(R)}]$ be a set of R sequences drawn from these C different classes and let $g_c(O)$ be a discriminant function associated with classifier c that indicates the degree to which O belongs to class c. The classifier $\Gamma(O)$ defines

a mapping from the sample space $O \in \mathbb{O}$ to the discrete categorical set $\{1, 2, \ldots, C\}$. That is,

$$\Gamma(O) = I \quad \text{iff} \quad I = \underset{c=1,\cdots,C}{\operatorname{argmax}}\, g_c(O). \tag{IV.2.4}$$

We propose two main approaches for learning the MSDHMM$^{\mathrm{P}_1}$ parameters. The first one is based on maximizing the likelihood of the data correspondent to each model. The second approach is based on discriminative training and aims at minimizing the classification error over all classes.

IV.2.1.1 Parameters initialization

As in the distance based approach, the learning starts by clustering the training data into N_s clusters using SCAD [36]. The center s_i of each cluster is used as the state's representative. Next, SCAD is used to partition the data into a larger number of clusters and build the codebook $\mathbf{V} = [v_1, \ldots, v_M]$. SCAD also learns an initial stream relevance weight, w_{jk}, for each symbol. Since the MSDHMM$^{\mathrm{P}_1}$ structure requires a weight in each state, initially we duplicate the weights computed via SCAD, i.e., we let $w_{ijk} = w_{jk}$ for $i = 1 \cdots N_s$. The probability of each symbol v_j^k in each stream k and within each state i can be represented by the fuzzy membership degree of v_j^k in this state. That is, we use :

$$b_{ijk} = \frac{1/d^k(v_j^k, s_i^k)}{\sum_{l=1}^{N_s} 1/d^k(v_j^k, s_l^k)}, \tag{IV.2.5}$$

where $d^k(v_j^k, s_i^k)$ is the partial distance between symbol v_j and state s_i taking into account only features from stream k. To satisfy the requirement that $\sum_{j=1}^{M} b_{ijk} = 1$, we scale the values using:

$$b_{ijk} \longleftarrow \frac{b_{ijk}}{\sum_{m=1}^{M} b_{imk}}. \tag{IV.2.6}$$

The overall probability of v_j in state i is then computed using (IV.2.2).

IV.2.1.2 Generalized Baum-Welch learning algorithm for MSDHMM$^{\mathrm{P}_1}$

After initialization, the model parameters can be tuned using the maximum Likelihood approach. Given a sequence of training observations $O = [o_1, \ldots, o_T]$, the parameters of λ_c could be learned by maximizing the likelihood of the observation sequence O, i.e., $Pr(O|\lambda)$. We achieve this by generalizing the Baum-Welch algorithm to include a stream relevance weight component. In particular, we define the generalized Baum-Welch algorithm by extending the auxiliary function in (II.7.1) to

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \sum_{Q} \sum_{F} \ln Pr(O, Q, F|\bar{\lambda}) Pr(Q, F|O, \lambda), \tag{IV.2.7}$$

where $F = [f_1, \ldots, f_T]$ is a sequence of random variables representing the stream indices for each time step. It can be shown that a critical point of $Pr(O|\lambda)$, with respect to λ, is a critical point of the new auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ with respect to $\bar{\lambda}$ when $\bar{\lambda} = \lambda$, that is:

$$\frac{\partial Pr(O|\lambda)}{\partial \lambda} = \frac{\partial \mathbb{Q}(\lambda, \bar{\lambda})}{\partial \bar{\lambda}}|_{\bar{\lambda}=\lambda}. \tag{IV.2.8}$$

Equation (IV.2.8) could be proved by using the same steps needed to prove propositions (II.7.1) and (II.7.2).

Maximizing the likelihood $Pr(O|\lambda)$ is equivalent to maximizing the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ which could be rewritten as:

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \mathcal{E}_{\sim(Q,F)|O,\lambda} \left[\log Pr(O, Q, F|\bar{\lambda})\right]. \tag{IV.2.9}$$

The above formulation of the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ could be interpreted as the conditional expectation of the log likelihood of the complete data (observed sequence and hidden parameters: O,Q,F) using the model $\bar{\lambda}$, with respect to the distribution of the hidden data (Q and F) conditioned to the observed sequence O and using the initial guess λ. More explicitly, the $\mathbb{Q}(\lambda, \bar{\lambda})$ function has the following integral form:

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \int_{\mathfrak{Q},\mathfrak{F}} \log Pr(O, Q, F|\bar{\lambda}) Pr(Q, F|O, \lambda) dQdF \tag{IV.2.10}$$

where Q and F belong to the spaces $\mathfrak{Q}$ and $\mathfrak{F}$ respectively. Since $\mathfrak{Q}$ and $\mathfrak{F}$ are discrete, the integral in (IV.2.10) form above is equivalent to the form in (IV.2.7). It follows that the formulation of the maximization of the likelihood $Pr(O|\lambda)$ through maximizing the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ is an EM [30] type optimization that is performed in two steps: the estimation step and the maximization step.

The estimation step consists of computing the conditional expectation in (IV.2.7) and write it in an analytical form if possible. The objective function in (IV.2.7) involves the quantity $Pr(O, Q, F|\bar{\lambda})$ which could be expressed analytically as:

$$Pr(O, Q, F|\bar{\lambda}) = \bar{\pi}_{q_1} \prod_{t=1}^{T-1} \bar{a}_{q_t q_{t+1}} \prod_{t=1}^{T} \bar{w}_{q_t Q_V(o_t) f_t} \bar{b}_{q_t Q_V(o_t) f_t} \tag{IV.2.11}$$

where Q_V is the quantization operation defined on an observation vector o_t as:

$$Q_V(o_t) = \underset{1 \le j \le M}{\operatorname{argmin}} \frac{1}{N_s} \sum_{i=1}^{N_s} \sum_{k=1}^{L} w_{ijk} \|o_t^k - v_j^k\|. \tag{IV.2.12}$$

In particular, Q_V maps each observation o_t to the index of its closest symbol. Thus, the objective

function in (IV.2.7) can be expanded as:

$$\begin{aligned}\mathbb{Q}(\lambda,\bar{\lambda}) &= \sum_{Q}\sum_{F}\log\bar{\pi}_{q_1}Pr(Q,F|O,\lambda)+\\ &\sum_{t=1}^{T-1}\sum_{Q}\sum_{F}\log\bar{a}_{q_tq_{t+1}}Pr(Q,F|O,\lambda)+\\ &\sum_{t=1}^{T}\sum_{Q}\sum_{F}\log\bar{w}_{q_tQ_V(o_t)f_t}Pr(Q,F|O,\lambda)+\\ &\sum_{t=1}^{T}\sum_{Q}\sum_{F}\log\bar{b}_{q_tQ_V(o_t)f_t}Pr(Q,F|O,\lambda). \qquad \text{(IV.2.13)}\end{aligned}$$

After the estimation step, the maximization step consists of finding the parameters of $\bar{\lambda}$ that maximize the function in (IV.2.13). The expanded form of the function $\mathbb{Q}(\lambda,\bar{\lambda})$ in (IV.2.13) has 4 terms involving $\overline{\pi}$, $\overline{a}$, $\overline{w}$ and $\overline{b}$. To find the values of $\overline{\pi}_i$, $\overline{a}_{ij}$, $\overline{w}_{ijk}$, and $\overline{b}_{ijk}$ that maximize $\mathbb{Q}(\lambda,\bar{\lambda})$, we consider the terms in (IV.2.13) that depend respectively on $\overline{\pi}$, $\overline{a}$, $\overline{w}$, and $\overline{b}$. In particular, the first and second terms in (IV.2.13) depend respectively on $\overline{\pi}$ and $\overline{a}$, and they have the same analytical expressions sketched in the case of the baseline DHMM in (II.7.4). It follows that the update equations for $\overline{\pi}_i$, and $\overline{a}_{ij}$ are the same as in the DHMM, that is:

$$\begin{aligned}\pi_i &= \gamma_1(i),\\ a_{ij} &= \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{t=1}^{T}\gamma_t(i)}.\end{aligned}$$

To find the value of $\overline{w}_{ijk}$ that maximizes the auxiliary function $\mathbb{Q}(.,.)$, only the third term of the expression in (IV.2.13) is considered since it is the only part of $\mathbb{Q}(.,.)$ that depends on $\overline{w}_{ijk}$. This term can be expressed as:

$$\begin{aligned}&\sum_{t=1}^{T}\sum_{Q}\sum_{F}Pr(Q,F|O,\lambda)\log\bar{w}_{q_tQ_V(o_t)f_t} =\\ &\sum_{t=1}^{T}\sum_{i}\sum_{j}\sum_{k}\log\bar{w}_{ijk}\sum_{Q}\sum_{F}Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t), \qquad \text{(IV.2.14)}\end{aligned}$$

where $\delta(i,q_t)\delta(j,o_t)\delta(k,f_t)$ keeps only those cases for which $q_t=i$, $Q_V(o_t)=j$ and $f_t=k$. That is,

$$\sum_{Q}\sum_{F}Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t)=Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t)), \qquad \text{(IV.2.15)}$$

therefore:

$$\begin{aligned}&\sum_{t=1}^{T}\sum_{Q}\sum_{F}Pr(Q,F|O,\lambda)\log(\bar{w}_{q_tQ_V(o_t)f_t}) =\\ &\sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L}Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t))\ln(\bar{w}_{ijk}) \qquad \text{(IV.2.16)}\end{aligned}$$

To find the update equation of $\overline{w}_{ijk}$, we use the Lagrange multipliers optimization (see Appendix C) with the constraint in (IV.2.3), and obtain:

$$\overline{w}_{ijk} = \frac{\sum_{t=1}^{T} Pr(q_t = i, f_t = k|O, \lambda)\delta(Q_V(o_t), j)}{\sum_{t=1}^{T} Pr(q_t = i|O, \lambda)\delta(Q_V(o_t), j)}, \tag{IV.2.17}$$

where

$$Pr(q_t = i, f_t = k|O, \lambda) = Pr(q_t = i|O, \lambda)\frac{w_{iQ_V(o_t)k} b_{iQ_V(o_t)k}}{b_{iQ_V(o_t)}}. \tag{IV.2.18}$$

Let, $\gamma_t(i,k) = Pr(q_t = i, f_t = k|O, \lambda)$. Since $Pr(q_t = i|O, \lambda) = \gamma_t(i)$, it follows that,

$$\gamma_t(i,k) = \gamma_t(i)\frac{w_{iQ_V(o_t)k} b_{iQ_V(o_t)k}}{b_{iQ_V(o_t)}}. \tag{IV.2.19}$$

Thus, the update equation for $\overline{w}_{ijk}$ becomes:

$$\overline{w}_{ijk} = \frac{\sum_{t=1}^{T} \gamma_t(i,k)\delta(o_t, j)}{\sum_{t=1}^{T} \gamma_t(i)\delta(o_t, j)}. \tag{IV.2.20}$$

The numerator in (IV.2.20) reflects the quantity of information provided by stream k while the denominator is used for normalization. It is possible that none of the closest symbols to o_t, $1 \leq t \leq T$, is v_j. If this situation occurs, the expression in (IV.2.20) becomes undefined. To avoid this case, we generalize the update equation in (IV.2.20) to the following:

$$\overline{w}_{ijk} = \begin{cases} \frac{\sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t),j)}{\sum_{t=1}^{T} \gamma_t(i)\delta(Q_V(o_t),j)} & \text{if } \exists t, \delta(Q_V(o_t), j) = 1 \\ \frac{1}{L} & \text{otherwise.} \end{cases} \tag{IV.2.21}$$

Similarly, it can be shown that the partial probabilities need to be updated using:

$$\overline{b}_{ijk} = \begin{cases} \frac{\sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t),j))}{\sum_{t=1}^{T} \gamma_t(i,k)} & \text{if } \exists t, \delta(Q_V(o_t), j) = 1 \\ \frac{1}{M} & \text{otherwise} \end{cases} \tag{IV.2.22}$$

In (IV.2.22), the numerator represents the contribution of each stream k for each code j within state i, and the denominator is a normalization factor. The details of the derivations of the above equations can be found in appendix D.

In the case of multiple observations $[O^{(1)}, \ldots, O^{(R)}]$, it can be easily shown that the learning equations need to be updated using:

$$\overline{w}_{ijk} = \begin{cases} \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(i,k)\delta(Q_V(o_t^r),j)}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(i)\delta(Q_V(o_t^r),j)} & \text{if } \exists t, \delta(Q_V(o_t^r), j) = 1 \\ \frac{1}{L} & \text{otherwise} \end{cases}, \tag{IV.2.23}$$

and

$$\overline{b}_{ijk} = \begin{cases} \frac{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(i,k)\delta(Q_V(o_t^r),j))}{\sum_{r=1}^{R}\sum_{t=1}^{T} \gamma_t^r(i,k)} & \text{if } \exists t, \delta(Q_V(o_t^r), j) = 1 \\ \frac{1}{M} & \text{otherwise} \end{cases} \tag{IV.2.24}$$

The variables $\gamma_t^{(r)}(i)$ and $\gamma_t^{(r)}(i,k)$ in (VI.1.11) and (IV.2.24) are the same variables as $\gamma_t(i)$ and $\gamma_t(i,k)$ for each sequence $O^{(r)}$. Algorithm (3) outlines the steps of the MLE training procedure of the different parameters of the MSDHMM$^{\mathbf{P}_l}$.

Algorithm 3 Generalized BW training for the probability based MSDHMM with linear aggregation

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L.
Ensure:
1: Cluster training data into N_s clusters, and let, the center of each cluster, s_i, be the representative of state i.
2: Quantize the training data into M symbols and learn initial stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: Let $w_{ijk} = w_{jk}$, for $1 \leq i \leq N_s$
4: **while** stopping criteria not satisfied **do**
5: Compute the closest observation to o_t using (IV.2.12);
6: update **A** using (II.7.9);
7: update **W** using (VI.1.11);
8: update **B** using (IV.2.24);
9: **end while**

IV.2.1.3 Generalized MCE/GPD learning algorithm for the MSDHMM$^{\mathbf{P}_l}$

The minimization of the classification error via a gradient descent scheme is the most common discriminative training method for HMMs. We generalize this approach to the MSDHMM$^{\mathbf{P}_l}$ structure. In particular, we let,

$$g_c(O,\Lambda) = \log[\max_Q g_c(O,Q,\Lambda)] \tag{IV.2.25}$$

be the discriminant function, associated with classifier λ_c, that indicates the degree to which O belongs to class c. In (IV.2.25), Q is a state sequence correspondent to the observation sequence O, Λ includes the models parameters, and

$$\begin{aligned} g_c(O,Q,\Lambda) &= Pr(O,Q|\lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a_{q_t q_{t+1}}^{(c)} \prod_{t=1}^{T} \left[\sum_{k=1}^{L} w_{q_t k}^{(c)}(o_t) b_{q_t k}^{(c)}(o_t) \right], \end{aligned} \tag{IV.2.26}$$

where $b_{q_t k}^{(c)}(o_t) = b_{q_t Q_V(o_t) k}^{(c)}$, $w_{q_t k}^{(c)}(o_t) = w_{q_t Q_V(o_t) k}^{(c)}$, and Q_V is defined in (IV.2.12).

The misclassification measure of the sequence O is defined by:

$$d_c(O) = -g_c(O,\Lambda) + \log \left[\frac{1}{C-1} \sum_{j, j \neq c} \exp[\eta \, g_j(O,\Lambda)] \right]^{\frac{1}{\eta}} \tag{IV.2.27}$$

where η is a positive number. A positive $d_c(O)$ implies misclassification while a negative $d_c(O)$ indicates a correct decision. The misclassification measure is embedded in a smoothed zero-one loss

function, defined as:

$$l_c(O, \Lambda) = l(d_c(O)), \tag{IV.2.28}$$

where l is the sigmoid function in (II.7.51). Finally, for any unknown sequence O, the classifier performance is measured by:

$$l(O; \Lambda) = \sum_{c=1}^{C} l_c(O; \Lambda)\mathbb{I}(O \in C_c) \tag{IV.2.29}$$

where $\mathbb{I}(.)$ is the indicator function.

Given a set of training observation sequences $O^{(r)}$, $r = 1, 2, \ldots, R$, an empirical loss function on the training data set is defined as:

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R}\sum_{c=1}^{C} l_c(O^{(r)}; \Lambda)\mathbb{I}(O^{(r)} \in C_c). \tag{IV.2.30}$$

The empirical loss above approximates the true Bayes risk [52]. The MSDHMM$^{\mathrm{P}_1}$ parameters are therefore estimated by minimizing $\mathbf{L}(\Lambda)$ using a gradient descent algorithm. In order to ensure that the estimated MSDHMM$^{\mathrm{P}_1}$ parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$, $w_{ijk} \geq 0$, $\sum_{k=1}^{L} w_{ijk} = 1$, $b_{ijk} \geq 0$, and $\sum_{j=1}^{M} b_{ijk} = 1$, we map these parameters using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij} \tag{IV.2.31}$$

$$w_{ijk} \rightarrow \tilde{w}_{ijk} = \log w_{ijk} \tag{IV.2.32}$$

$$b_{ijk} \rightarrow \tilde{b}_{ijk} = \log b_{ijk} \tag{IV.2.33}$$

Then, the parameters are updated with respect to $\tilde{\Lambda}$. After updating, we map them back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}} \tag{IV.2.34}$$

$$w_{ijk} = \frac{\exp \tilde{w}_{ijk}}{\sum_{k'=1}^{L} \exp \tilde{w}_{ijk'}} \tag{IV.2.35}$$

$$b_{ijk} = \frac{\exp \tilde{b}_{ijk}}{\sum_{j'=1}^{M} \exp \tilde{b}_{ij'k}}. \tag{IV.2.36}$$

Using a steepest descent batch estimation mode, the MSDHMM$^{\mathrm{P}_1}$ parameters are iteratively updated using:

$$\tilde{\Lambda}(\tau + 1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda)\Big|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}, \tag{IV.2.37}$$

where ϵ is the learning rate, and ∇ is the gradient operator.

The updating mechanism in (IV.2.37) applies for the variables $\bar{\pi}$, $\bar{a}$, $\bar{w}$, and $\bar{b}$. However, it could be shown that $\bar{\pi}$, $\bar{a}$ could be updated similarly to the standard DHMM as in (II.7.59) and (II.7.60).

It can be shown that $\tilde{w}_{ijk}^{(c)}$ and $\tilde{b}_{ijk}^{(c)}$ need to be updated using:

$$\tilde{w}_{ijk}^{(c)}(\tau+1) = \tilde{w}_{ijk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)} \tag{IV.2.38}$$

and

$$\tilde{b}_{ijk}^{(c)}(\tau+1) = \tilde{b}_{ijk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}. \tag{IV.2.39}$$

For the parameters $\bar{w}$, and $\bar{b}$, the derivatives $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$ and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}}$ in (IV.2.38) and (IV.2.39) respectively could be expanded using the chain rule as follows:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial w_{ijk}^{(c)}} \times \frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} \mathbb{I}(O \in C_c), \tag{IV.2.40}$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial b_{ijk}^{(c)}} \times \frac{\partial b_{ijk}^{(c)}}{\partial \tilde{b}_{ijk}^{(c)}} \mathbb{I}(O \in C_c), \tag{IV.2.41}$$

where

$$\frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} = \zeta l_m(O,\Lambda)\left[1 - l_m(O,\Lambda)\right],$$

$$\frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}, \tag{IV.2.42}$$

$$\frac{\partial g_c(O,\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{t=1}^{T} \delta(q_t = i, Q_V(o_t) = j)\frac{b_{ijk}^{(c)}}{b_{ij}^{(c)}},$$

$$\frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} = w_{ijk}^{(c)}\left[1 - w_{ijk}^{(c)}\right].$$

$$\frac{\partial g_c(O,\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} = \sum_{t=1}^{T} \delta(q_t = i, Q_V(o_t) = j)\frac{w_{ijk}^{(c)}}{b_{ij}^{(c)}},$$

and

$$\frac{\partial b_{ijk}^{(c)}}{\partial \tilde{b}_{ijk}^{(c)}} = b_{ijk}^{(c)}\left[1 - b_{ijk}^{(c)}\right].$$

A closed form of $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$ and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}}$ could be then inferred:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1 - l_m(O_r,\Lambda))w_{ijk}^{(c)}(1 - w_{ijk}^{(c)})\frac{b_{ijk}^{(c)}}{b_{ij}^{(c)}}\delta(q_t^r = i, Q_V(o_t^r) = j)\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)},$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1 - l_m(O_r,\Lambda))b_{ijk}^{(c)}(1 - b_{ijk}^{(c)})\frac{w_{ijk}^{(c)}}{b_{ij}^{(c)}}\delta(q_t^r = i, Q_V(o_t^r) = j)\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}.$$

Algorithm (4) outlines the steps needed to learn the parameters of all the models λ_c in the MCE/GPD framework.

Algorithm 4 Generalized MCE/GPD training for the probability-based MSDHMM with linear aggregation

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$.. Fix the variables N_s, M, and L for each model λ_c.

Ensure:

1: For each λ_c, cluster training data into N_s clusters and let the center of each cluster, s_i, be the representative of state i.
2: For each λ_c, quantize the training data into M symbols and learn initial stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: Let $w_{ijk} = w_{jk}$, for $1 \leq i \leq N_s$
4: **while** stopping criteria not satisfied **do**
5: Compute the closest observation to each o_t using (IV.1.7).
6: Compute the loss function of each sequence O using (V.1.24);
7: update $\mathbf{A}$ of each λ_c using (II.7.60);
8: update $\mathbf{B}$ of each λ_c using (IV.2.39);
9: update $\mathbf{W}$ of each λ_c using (IV.2.38);
10: **end while**

IV.2.2 Geometric aggregation

In this method, the partial probabilities are combined using:

$$b_{ij} = \prod_{k=1}^{L} [b_{ijk}]^{w_{ijk}}, \tag{IV.2.43}$$

subject to

$$\sum_{k=1}^{L} w_{ijs}^{\nu} = \kappa, \quad \text{and} \quad \sum_{j=1}^{M} b_{ijk} = 1. \tag{IV.2.44}$$

The geometric form of the observation probability in (IV.2.43) is motivated by the following probabilistic reasoning:

$$\begin{aligned} b_{ij} &= Pr(v_j|q_t = i; \lambda) \\ &= Pr(v_j^{(1)}, \cdots, v_j^{(L)}|q_t = i; \lambda) \\ &= \prod_{k=1}^{L} Pr(v_j^{(k)}|q_t = i; \lambda) \\ &\approx \prod_{k=1}^{L} \left[Pr(v_j^{(k)}|q_t = i; \lambda)\right]^{w_{ijk}} \end{aligned}$$

It follows that:

$$b_{ijk} = Pr(v_j^{(k)}|q_t = i; \lambda).$$

We will refer to this probability based MSDHMM with geometric aggregation as $\text{MSDHMM}^{\text{P}_g}$. The exponential weight w_{ijk} weighs the contribution of each stream to state i. In (IV.2.44), κ is a constant, usually set to one and $\nu \in (1, \infty)$ is an exponent that controls the discrimination between the different streams.

Similarly to the linear aggregation case, two main approaches are considered for the training of the MSDHMM^{P_g} parameters. The first one consists of maximizing the likelihood of the data correspondent to each model. The second approach consists of a discriminative training that aims at minimizing the classification error over all classes.

IV.2.2.1 Parameter initialization

For the initialization of the model parameters, we follow the same steps as those used for the linear combination. The only difference resides in using (IV.2.43) instead of the linear combination in (IV.2.2).

IV.2.2.2 Generalized Baum-Welch learning algorithm for MSDHMM^{P_g}

After initialization, the model parameters can be tuned using the maximum Likelihood approach. Given a sequence of training observation $O = [o_1, \ldots, o_T]$, the parameters of λ_c could be learned by maximizing the likelihood of the observation sequence O, i.e., $Pr(O|\lambda)$. We achieve this by generalizing the Baum-Welch algorithm to include a stream relevance weight component. We define the generalized Baum-Welch algorithm by extending the auxiliary function in (II.7.1) to

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \sum_{Q} \sum_{F} \ln Pr(O, Q, F|\bar{\lambda}) Pr(Q, F|O, \lambda), \tag{IV.2.45}$$

where $F = [f_1, \ldots, f_T]$ is a sequence of random variables representing the stream indices for each time step. It can be shown that a critical point of $Pr(O|\lambda)$, with respect to λ, is a critical point of the new auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ with respect to $\bar{\lambda}$ when $\bar{\lambda} = \lambda$, that is:

$$\frac{\partial Pr(O|\lambda)}{\partial \lambda} = \frac{\partial \mathbb{Q}(\lambda, \bar{\lambda})}{\partial \bar{\lambda}}|_{\bar{\lambda}=\lambda}. \tag{IV.2.46}$$

The proof of (IV.2.46) could be achieved using the same steps needed to prove propositions (II.7.1) and (II.7.2).

Similar to the linear aggregation case, it could be shown that the formulation of the maximization of the likelihood $Pr(O|\lambda)$ through maximizing the the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ is an EM [30] type optimization that is performed in two steps: the estimation step and the maximization step. The estimation step consists of computing the conditional expectation in (IV.2.45) and write it in an analytical form. The objective function in (IV.2.45) involves the quantity $Pr(O, Q, F|\bar{\lambda})$ which could be expressed analytically as:

$$Pr(O, Q, F|\bar{\lambda}) = \bar{\pi}_{q_1} \prod_{t=1}^{T-1} \bar{a}_{q_t q_{t+1}} \prod_{t=1}^{T} \left[\bar{b}_{q_t Q_V(o_t) f_t}\right]^{\bar{w}_{q_t Q_V(o_t) f_t}} \tag{IV.2.47}$$

Thus, the objective function in (IV.2.45) can be expanded as follows:

$$\begin{aligned}\mathbb{Q}(\lambda,\bar{\lambda}) &= \sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log\bar{\pi}_{q_1} + \\ &\sum_{t=1}^{T-1}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log\bar{a}_{q_t q_{t+1}} + \\ &\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\bar{w}_{q_t Q_V(o_t) f_t}\log\bar{b}_{q_t Q_V(o_t) f_t}. \end{aligned} \tag{IV.2.48}$$

After the estimation step, the maximization step consists of finding the parameters of $\bar{\lambda}$ that maximize the function in (IV.2.48). The expanded form of the function $\mathbb{Q}(\lambda,\bar{\lambda})$ in (IV.2.48) has 3 terms involving $\overline{\pi}$, $\overline{a}$, and ($\overline{w}$, $\overline{b}$) independently. To find the values of $\overline{\pi}_i$, $\overline{a}_{ij}$, $\overline{w}_{ijk}$, and $\overline{b}_{ijk}$ that maximize $\mathbb{Q}(\lambda,\bar{\lambda})$, we consider the terms in (IV.2.48) that depend on $\overline{\pi}$, $\overline{a}$, $\overline{w}$, and $\overline{b}$. In particular, the first and second terms in (IV.2.48) depend on $\overline{\pi}$ and $\overline{a}$, and they have the same analytical expressions sketched in the case of the baseline DHMM in (II.7.4). Thus, the update equations for $\overline{\pi}_i$, and $\overline{a}_{ij}$ are the same as in the DHMM, that is:

$$\begin{aligned}\pi_i &= \gamma_1(i), \\ a_{ij} &= \frac{\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{t=1}^{T}\gamma_t(i)}.\end{aligned}$$

To find the value of $\overline{w}_{ijk}$ that maximizes the auxiliary function $\mathbb{Q}(.,.)$, only the third term of the expression in (IV.2.48) is considered since it is the only part of $\mathbb{Q}(.,.)$ that depends on $\overline{w}_{ijk}$. This term can be expressed as follows:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\bar{w}_{q_t Q_V(o_t) f_t}\log(\bar{b}_{q_t Q_V(o_t) f_t}) = \sum_{t=1}^{T}\sum_{i}\sum_{j}\sum_{k}\log(\bar{w}_{ijk}) \times \\ \sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t),\end{aligned} \tag{IV.2.49}$$

where $\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t)$ keeps only those cases for which $q_t = i$, $Q_V(o_t) = j$ and $f_t = k$. That is,

$$\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t) = Pr(q_t = i, f_t = k|O,\lambda)\delta(Q_V(o_t),j). \tag{IV.2.50}$$

Therefore:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\bar{w}_{q_t Q_V(o_t) f_t}\log(\bar{b}_{q_t Q_V(o_t) f_t}) = \\ \sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t = i, f_t = k|O,\lambda)\delta(o_t,j)\bar{w}_{q_t Q_V(o_t) f_t}\ln(\bar{b}_{ijk})\end{aligned} \tag{IV.2.51}$$

To find the update equation of $\overline{w}_{ijk}$ we use the Lagrange multipliers optimization with the constraint in (IV.2.44), and obtain

$$\overline{w}_{ijk} = \left[\kappa \frac{[D_{ijk}]^{\frac{\nu}{\nu-1}}}{\sum_{k'=1}^{L} [D_{ijk'}]^{\frac{\nu}{\nu-1}}} \right]^{\frac{1}{\nu}}, \tag{IV.2.52}$$

where

$$D_{ijk} = \sum_{t=1}^{T} \gamma_t(i,k)\delta(o_t, j) \log b_{ijk}. \tag{IV.2.53}$$

The numerator in (IV.2.52) reflects the quantity of information provided by stream k while the denominator is used for normalization. It is possible that none of the closest symbols to o_t, $1 \leq t \leq T$, is v_j. If this situation occurs, the expression in (IV.2.52) becomes undefined. To avoid this case, we generalize the update equation in (IV.2.52) to:

$$\overline{w}_{ijk} = \begin{cases} \left[\kappa \frac{[D_{ijk}]^{\frac{\nu}{\nu-1}}}{\sum_{k'=1}^{L} [D_{ijk'}]^{\frac{\nu}{\nu-1}}} \right]^{\frac{1}{\nu}} & \text{if } \exists t, \delta(o_t, j) = 1 \\ \left(\frac{\kappa}{L}\right)^{\frac{1}{\nu}} & \text{otherwise} \end{cases} \tag{IV.2.54}$$

Similarly, it can be shown that the update equation for the partial probabilities is:

$$\overline{b}_{ijk} = \begin{cases} \frac{\sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t^r),j) w_{ijk}}{\sum_{j'=1}^{M} \sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t^r),k) w_{ij'k}} & \text{if } \exists t, \delta(Q_V(o_t^r), j) = 1 \\ \frac{1}{M} & \text{otherwise} \end{cases} \tag{IV.2.55}$$

In (IV.2.55), the numerator represents the contribution of each stream k for each code j within state i, and the denominator is a normalization factor. The detailed derivation of the above equations could be found in appendix D.

In the case of multiple observations $[O^{(1)}, \ldots, O^{(R)}]$, it can be easily shown that the update equations become:

$$\overline{w}_{ijk} = \begin{cases} \left[\kappa \frac{\left[\sum_{r=1}^{R} D_{ijk}^{(r)}\right]^{\frac{\nu}{\nu-1}}}{\sum_{r=1}^{R} \sum_{k=1}^{L} \left[D_{ijk}^{(r)}\right]^{\frac{\nu}{\nu-1}}} \right]^{\frac{1}{\nu}} & \text{if } \exists t, \delta(Q_V(o_t^r), j) = 1 \\ \left(\frac{\kappa}{L}\right)^{\frac{1}{\nu}} & \text{otherwise} \end{cases} \tag{IV.2.56}$$

and

$$\overline{b}_{ijk} = \begin{cases} \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t^r),j) w_{ijk}}{\sum_{r=1}^{R} \sum_{j'=1}^{M} \sum_{t=1}^{T} \gamma_t(i,k)\delta(Q_V(o_t^r),k) w_{ij'k}} & \text{if } \exists t, \delta(Q_V(o_t^r), j) = 1 \\ \frac{1}{M} & \text{otherwise} \end{cases} \tag{IV.2.57}$$

where

$$D_{ijk}^{(r)} = \sum_{t=1}^{T} \gamma_t^{(r)}(i,k)\delta(Q_V(o_t^{(r)}), j) \log b_{ijk}. \tag{IV.2.58}$$

Algorithm (5) outlines the steps of the generalized MLE training procedure of the different parameters of the MSDHMM$^{\mathrm{P}_g}$.

Algorithm 5 Generalized BW training for the probability-based MSDHMM with geometric aggregation

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L.
Ensure:
1: Cluster training data into N_s clusters and let the center of each cluster, s_i, be the representative of state i.
2: Quantize the training data into M symbols and learn initial stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: Let $w_{ijk} = w_{jk}$, for $1 \leq i \leq N_s$
4: **while** stopping criteria not satisfied **do**
5: Compute the closest observation to o_t using (IV.2.12);
6: update **A** using (II.7.9);
7: update **W** using (IV.2.56);
8: update **B** using (IV.2.57);
9: **end while**

IV.2.2.3 Generalized MCE/GPD learning algorithm for the MSDHMM$^{\mathbf{P}_g}$

Let,

$$g_c(O, \Lambda) = \log[\max_Q g_c(O, Q, \Lambda)] \tag{IV.2.59}$$

be the discriminant function, associated with classifier λ, that indicates the degree to which O belongs to class c. In (IV.2.59), Q is a state sequence correspondent to the observation sequence O, Λ includes the models parameters, and

$$\begin{aligned} g_c(O, Q, \Lambda) &= Pr(O, Q; \lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} \prod_{k=1}^{L} \left[b^{(c)}_{q_t k}(o_t) \right]^{w^{(c)}_{q_t k}(o_t)}, \end{aligned} \tag{IV.2.60}$$

where $b^{(c)}_{q_t k}(o_t) = b^{(c)}_{q_t Q_V(o_t) k}$, $w^{(c)}_{q_t k}(o_t) = w^{(c)}_{q_t Q_V(o_t) k}$, and Q_V is defined in (IV.2.12).

The misclassification measure of the sequence O is defined by:

$$d_c(O) = -g_c(O, \Lambda) + \log \left[\frac{1}{C-1} \sum_{j, j \neq c} \exp[\eta \, g_j(O, \Lambda)] \right]^{\frac{1}{\eta}} \tag{IV.2.61}$$

where η is a positive number. The misclassification measure is embedded in a smoothed zero-one loss function, defined as:

$$l_c(O, \Lambda) = l(d_c(O)), \tag{IV.2.62}$$

where l is the sigmoid function in (II.7.51). For any unknown sequence O, the classifier performance is measured by:

$$l(O; \Lambda) = \sum_{c=1}^{C} l_c(O; \Lambda) \mathbb{I}(O \in C_c) \tag{IV.2.63}$$

where $\mathbb{I}(.)$ is the indicator function.

Given a set of training observation sequences $O^{(r)}$, $r = 1, 2, \ldots, R$, an empirical loss function on the training data set is defined as

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R} \sum_{c=1}^{C} l_c(O^{(r)}; \Lambda) \mathbb{I}(O^{(r)} \in C_c). \tag{IV.2.64}$$

The MSDHMM$^{\mathrm{P_g}}$ parameters are estimated by minimizing $\mathbf{L}(\Lambda)$ using a gradient descent algorithm. In order to ensure that the estimated MSDHMM$^{\mathrm{P_g}}$ parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$, $w_{ijk} \geq 0$, $\sum_{k=1}^{L} w_{ijk}^{\nu} = \kappa$, $b_{ijk} \geq 0$, and $\sum_{j=1}^{M} b_{ijk} = 1$, we map these parameters using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij} \tag{IV.2.65}$$

$$w_{ijk} \rightarrow \tilde{w}_{ijk} = \log w_{ijk} \tag{IV.2.66}$$

$$b_{ijk} \rightarrow \tilde{b}_{ijk} = \log b_{ijk} \tag{IV.2.67}$$

Then, the parameters are updated with respect to $\tilde{\Lambda}$. After updating, we map them back using:

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}} \tag{IV.2.68}$$

$$w_{ijk}^{\nu} = \kappa \frac{\exp \tilde{w}_{ijk}^{\nu}}{\sum_{k'=1}^{L} \exp \tilde{w}_{ijk'}^{\nu}} \tag{IV.2.69}$$

$$b_{ijk} = \frac{\exp \tilde{b}_{ijk}}{\sum_{j'=1}^{M} \exp \tilde{b}_{ij'k}}. \tag{IV.2.70}$$

Using a steepest descent batch estimation mode, the MSDHMM$^{\mathrm{P_g}}$ parameters are iteratively updated using:

$$\tilde{\Lambda}(\tau + 1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda) \Big|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)}. \tag{IV.2.71}$$

where ϵ is the learning rate, and ∇ is the gradient operator. It can be shown that $\tilde{w}_{ijk}^{(c)}$ and $\tilde{b}_{ijk}^{(c)}$ need to be updated using:

$$\tilde{w}_{ijk}^{(c)}(\tau + 1) = \tilde{w}_{ijk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)} \tag{IV.2.72}$$

and

$$\tilde{b}_{ijk}^{(c)}(\tau + 1) = \tilde{b}_{ijk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} \Bigg|_{\tilde{\Lambda} = \tilde{\Lambda}(\tau)} \tag{IV.2.73}$$

The derivatives $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$ and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}}$ in (IV.2.38) and (IV.2.38) respectively could be expanded using the chain rule as follows:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O, \Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O, \Lambda)} \times \frac{\partial g_c(O, \Lambda)}{\partial w_{ijk}^{(c)}} \times \frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} \mathbb{I}(O \in C_c), \tag{IV.2.74}$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial b_{ijk}^{(c)}} \times \frac{\partial b_{ijk}^{(c)}}{\partial \tilde{b}_{ijk}^{(c)}} \mathbb{I}(O \in C_c), \tag{IV.2.75}$$

where

$$\frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} = \zeta l_m(O,\Lambda)\left[1 - l_m(O,\Lambda)\right],$$

$$\frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases},$$

$$\frac{\partial g_c(O,\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{t=1}^{T} \delta(q_t = i, Q_V(o_t) = j) \log b_{ijk}^{(c)},$$

$$\frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} = \sqrt[\nu]{\kappa}[\tilde{w}_{ijk}^{(c)}]^{\nu-1} w_{ijk}^{(c)} \left[1 - [w_{ijk}^{(c)}]^{\nu}\right],$$

$$\frac{\partial g_c(O,\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} = \sum_{t=1}^{T} \delta(q_t = i, Q_V(o_t) = j) \frac{w_{ijk}^{(c)}}{b_{ijk}^{(c)}},$$

and

$$\frac{\partial b_{ijk}^{(c)}}{\partial \tilde{b}_{ijk}^{(c)}} = b_{ijk}^{(c)}\left[1 - b_{ijk}^{(c)}\right].$$

A closed form of $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$ and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}}$ could be then inferred:

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} &= \sqrt[\nu]{\kappa} \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1 - l_m(O_r,\Lambda)) \left[\tilde{w}_{ijk}^{(c)}\right]^{\nu-1} \\ &\quad \times w_{ijk}^{(c)}(1 - \left[w_{ijk}^{(c)}\right]^{\nu}) \log b_{ijk}^{(c)} \delta(q_t^r = i, Q_V(o_t^r) = j) \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \end{aligned} \tag{IV.2.76}$$

and

$$\begin{aligned} \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{b}_{ijk}^{(c)}} &= \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1 - l_m(O_r,\Lambda)) \times b_{ijk}^{(c)}(1 - b_{ijk}^{(c)}) \times \\ &\quad \frac{w_{ijk}^{(c)}}{b_{ijk}^{(c)}} \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)} \delta(q_t^r = i, Q_V(o_t^r) = j). \end{aligned} \tag{IV.2.77}$$

Algorithm (6) outlines the steps needed to learn the parameters of all the models λ_c in the MCE/GPD framework.

IV.3 Inference

To test a new observation sequence O=[o_1,...,o_T], we need to compute $Pr(O|\lambda_c)$, with respect to each model λ_c. This computation can be performed efficiently using the Viterbi algorithm

Algorithm 6 Generalized MCE/GPD training of the probability-based MSDHMM with geometric aggregation

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L for each model λ_c.

Ensure:

1: For each λ_c, cluster training data into N_s clusters and let the center of each cluster be, s_i, the representative of state i.
2: For each λ_c, quantize the training data into M symbols and learn initial stream relevance weights w_{jk}. The center of each cluster v_j is a symbol.
3: Let $w_{ijk} = w_{jk}$, for $1 \leq i \leq N_s$
4: **while** stopping criteria not satisfied **do**
5: Compute the closest observation to each o_t using (IV.1.7).
6: Compute the loss function of each sequence O using (V.1.24);
7: update $\mathbf{A}$ of each λ_c using (II.7.60);
8: update $\mathbf{B}$ of each λ_c using (IV.2.73);
9: update $\mathbf{W}$ of each λ_c using (IV.2.72);
10: **end while**

[29]. The Viterbi algorithm computes also the correspondent optimal state sequence $[q_1, \ldots, q_T]$ to O. This in turn requires the computation of $b_i(o_t)$. For the MSDHMM, this can be computed using:

$$b_i(o_t) = b_{ij}, \tag{IV.3.1}$$

where $j = Q_V(o_t)$ and is computed using (IV.2.12).

IV.4 Convergence properties

The aim of the Baum-Welch algorithm is to find estimates of the parameters of the HMM that maximizes the likelihood $Pr(O|\lambda)$. It is well known that the maximum likelihood estimator (MLE) have the following properties [53]:

- **Unbiasedness**:The MLE could be biased or unbiased. However, when the MLE is a biased estimator, its bias tends to 0 as $n \longrightarrow \infty$.
- **Consistency**: Subject to fairly weak regularity conditions, ML estimators are consistent.
- **Efficiency**:Since ML estimators may be biased we can only talk in general about asymptotic efficiency. However, it has been shown that MLE is asymptotically efficient. It has also the asymptotic normality. It is then called best asymptotically normal (BAN).
- **Sufficiency**: If $\hat{\theta}$ is the unique MLE of a parameter θ, then $\hat{\theta}$ must be a function of the minimal sufficient statistic for θ. This does not mean that $\hat{\theta}$ is necessarily sufficient, although it often is.

Appendix (E) gives the technical definition of the unbiasedness, consistency, efficiency, and sufficiency of general estimators.

It could be inferred then that in the presence of "enough" data, the MLE is almost optimal. This however is not possible in all cases. In addition, we wish to have the global maximum of the likelihood objective function. The Baum-Welch algorithm carries out an EM like optimization of the likelihood function. As stated previously, the estimation step consists of writing an analytical form of the auxiliary functions in (IV.2.7), and (IV.2.45), which have the form of a conditional expectation. The maximization step consists of finding a maximum (local at least, global if possible) of the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$. However, it is shown that the solutions found by the algorithms (3), and (5) are proven to be critical points of the likelihood function $Pr(O|\lambda)$. Therefore, it is of interest to see if these solutions are (local) maximum of their correspondent objective functions. This is given by the following theorem:

Theorem IV.4.1. *The generalized Baum-Welch ensures a convergence to a* ***local maximum*** *for* $MSDHMM^{P_l}$, *and* $MSDHMM^{P_g}$.

Proof. It could be shown that the computed critical points are local maximum since the second derivative of each objective function is negative when evaluated on the correspondent found critical points. In the following we show that the objective function $\mathbb{Q}(\lambda, \bar{\lambda})$ in (IV.2.7) is locally maximized when evaluated in the points computed in (II.7.9), (IV.2.21), and (IV.2.22). In fact, for $\tilde{w}_{ijk}$,

$$\frac{\partial^2 \mathbb{Q}(\lambda, \bar{\lambda})}{\partial \tilde{w}_{ijk}^2} = -\sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t = i, f_t = k|O, \lambda)\delta(j, Q_V(o_t))\left[\bar{w}_{ijk}\right]^{-2} \leq 0 \qquad \text{(IV.4.1)}$$

The same result could be found for the rest of the parameters. Thus, it could be concluded that the solutions in (II.7.9), (IV.2.21), and (IV.2.22) represent a local maxima of the objective $\mathbb{Q}(\lambda, \bar{\lambda})$ in the case of $\mathrm{MSDHMM}^{\mathrm{P_l}}$. Similar steps lead to the same conclusion for the $\mathrm{MSDHMM}^{\mathrm{P_g}}$. □

For the discriminative training, it has been proven in [52] that the MCE empirical cost measured on a finite training set approximates the theoretical classification risk. As the training data set grows larger, the MCE estimates have the property to minimize the Bayes risk. In addition, reducing the MCE empirical loss can always be achieved by the steepest descent mechanism if a sufficiently small learning rate is chosen. However, this almost guaranteed convergence does not always imply a fast convergence rate [54].

IV.5 Experimental Evaluation

IV.5.1 Data generation

To validate the proposed MSDHMM structures, we generate two synthetic data sets. The first set is a single stream sequential data, and the second one is a multi-stream one. Both sets are generated using two discrete DHMMs to simulate a two class problem. We follow a similar approach to the one used in [55] to generate sequential data using a discrete HMM with $N_s = 4$ and $M = 120$ symbols with 4 dimensions. We start by fixing N_s different vectors $\mu_i \in \mathbb{R}^4$, $i = 1, \cdots, N_s$ to represent the different states. Then, we randomly generate $\frac{M}{N_s}$ vectors from each normal distribution with mean μ_i and identity covariance matrix. A codebook with M symbols is then formed. For each symbol, the membership in each state is computed using

$$b_{ij} = \left[\sum_{m=1}^{N_s} \frac{\|v_j - \mu_i\|}{\|v_j - \mu_m\|} \right]^{-1} \tag{IV.5.1}$$

and then scaled using:

$$b_{ij} \longleftarrow \frac{b_{ij}}{\sum_{l=1}^{M} b_{il}}. \tag{IV.5.2}$$

In (IV.5.1), v_j denotes the jth symbol. The initial state probability distribution and the state transition probability distribution are generated randomly from a uniform distribution in the interval $[0, 1]$. The randomly generated values are then scaled to satisfy the stochastic constraints.

For the single stream sequential data, we generate R sequences of length $T = 15$ vectors with dimension p for each of the two classes. We start by generating a discrete HMM with N_s states and M symbols as described above. Then, we generate the single stream sequences using Algorithm (12).

Algorithm 7 Single stream sequential data generation for each class.

```
for r = 1 to R do
  Select the initial state according to the initial states probability distribution π
  Randomly pick a vector v from the M symbols among those representing the selected state
  Sample an observation from a normal distribution with mean v and covariance σI
  for t = 2 to T do
    Select next state according to the probabilities transition matrix A,
    Randomly pick a symbol v among those representing the selected state,
    Sample an observation o_t from the normal distribution which mean v and covariance σI.
  end for
end for
```

For the multi-stream case, we assume that the sequential data is synthesized by L=2 streams, and that each stream k is described by N_s states, where each state is represented by vector μ_i^k of

dimension p_k=4. To construct a set of M symbols based on the L streams, for each state i, 30 symbols are generated from each stream k, and concatenated to form a double-stream set of symbols. To simulate streams with various relevance wights, we create 3 groups of symbols in each state. The first group is formed by concatenating 10 symbols from each stream by just appending the features (i.e., both streams are relevant). The second group of symbols are formed by concatenating noise (instead of stream 2 features) to stream 1 features (i.e., stream 1 is relevant and stream 2 is irrelevant). The last group of symbols are formed by concatenating noise (instead of stream 1 features) to stream 2 features (i.e., stream 1 is irrelevant and stream 2 is relevant). Thus, for each state i we have a set of double-stream symbols where the streams have different degrees of relevance. Once the set of double-stream symbols is generated, a state transition probability distribution is generated, and the double-stream sequential data is generated using Algorithm (7).

IV.5.2 Results

In the first experiment, we apply the baseline DHMM and the proposed multi-stream DHMM structures to the single stream sequential data where the features are generated from one homogeneous source of information. The MSDHMM architectures treat the single stream sequential data as a double-stream one (each stream is assumed to have 2-dimensional observation vectors). In this experiment all models are trained using standard Baum-Welch (for the baseline DHMM and distance based MSDHMM), the generalized Baum-Welch (for the probability based MSDHMM), the standard and generalized MCE/GPD algorithms, or a combination of the two (Baum-Welch followed by MCE/GPD). The results of this experiment are reported in table 2. As it can be seen, the performance of the proposed MSDHMM structures and the baseline DHMM are comparable for most training methods. This is because when both streams are equally relevant for the entire data the different streams receive nearly equal weights in all states and the MSDHMM reduces to baseline DHMM. Fig. 20 displays the weights for all symbols learned by the MSDHMM^{D}. As it can be seen, most weights are clustered around 0.5 (between 0.35 and 0.7). Since the weights of both streams must sum to 1, both weights are considered equally important for all symbols.

Fig. 21 and 22 display the weights of stream 1 in all 4 states learned by the $\text{MSDHMM}^{\text{P}_l}$ and $\text{MSDHMM}^{\text{P}_g}$ methods. As it can be seen, most weights are clustered around 0.5. Thus, as for the MSDHMM^{D}, the weights are treated equally important for all symbols.

The second experiment involves applying both the baseline DHMM and the proposed MSDHMM to the double stream sequential data where the features are generated from two different

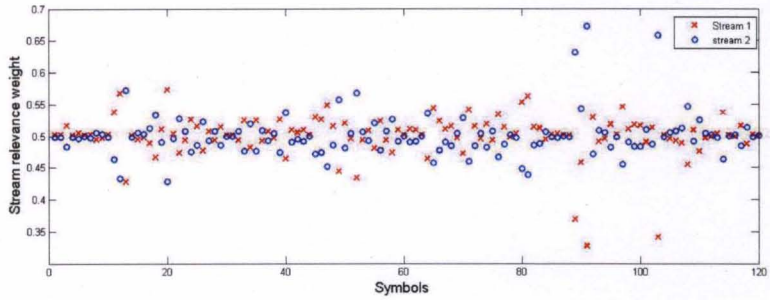


Figure 20. Stream relevance weights of the symbols learned by the MSDHMM^{D} model for the single-stream sequential data.

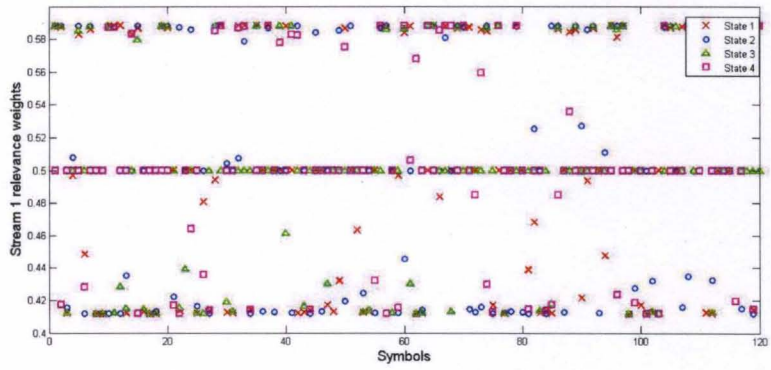


Figure 21. Stream 1 relevance weights of the symbols in all 4 states, learned by the $\text{MSDHMM}^{\text{P}_\text{l}}$ model for the single-stream sequential data.

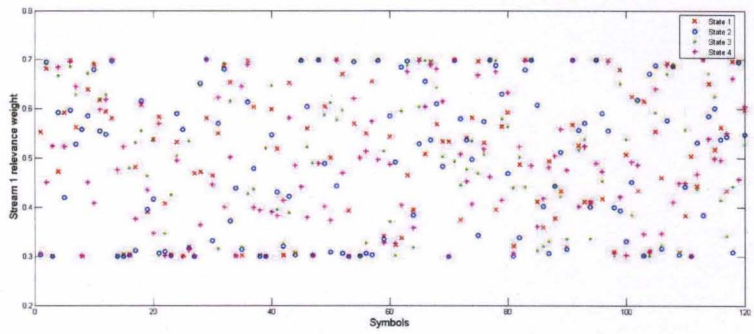


Figure 22. Stream 1 relevance weights of the symbols in all 4 states, learned by the $\text{MSDHMM}^{\text{P}_\text{g}}$ model for the single-stream sequential data.

TABLE 2

Classification rates of the different DHMM structures of the single stream data

Classifier	Baum-Welch	MCE	BW and MCE
Baseline DHMM	90.08 %	90.5%	91.25%
MSDHMM^{D}	91.075 %	92.00%	93.75%
$\text{MSDHMM}^{\text{P}_a}$	91.25 %	92.25 %	98.75%
$\text{MSDHMM}^{\text{P}_g}$	90.25 %	92.50%	95.75%

streams. In this experiment the various models are trained using Baum-Welch, MCE, and Baum-Welch followed by MCE training algorithms. First, we note that using stream relevance weights, the generalized Baum-Welch and MCE training algorithms converge faster and result in a small error. Fig. 23 displays the number of misclassified samples versus the number of iterations for the baseline DHMM and the proposed MSDHMM using MCE/GPD training. As it can be seen, learning stream relevance weights causes the error to drop faster. In fact, at each iteration, the classification error for the MSDHMM structure is lower than the baseline DHMM. In particular, for the probability based linear MSDHMM, the error reaches the minimum after only two iterations.

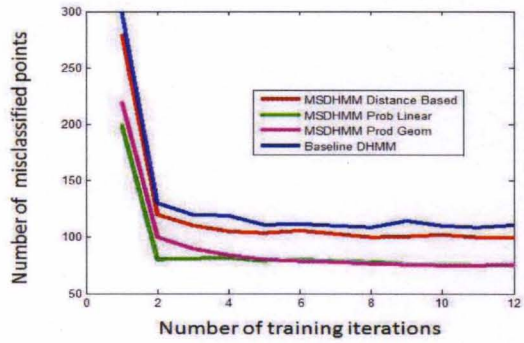


Figure 23. Number of misclassified points versus the training iteration number for the standard and the multistream DHMMs.

The testing results of the second experiment are reported in table 3. First, we note that all proposed multi-stream DHMMs outperform the baseline DHMM for all training methods. This is because the data set used for this experiment was generated from two streams with different degrees of relevancy and the baseline DHMM treats both streams equally important. The proposed MSDHMMs on the other hand, learn optimal relevance weights for each symbol within each state.

The learned weights for streams 1 and 2 by the MSDHMM^{D} are displayed in Fig. 24. As it can be seen, some symbols are highly relevant (weight close to 1), while others are completely

irrelevant (weight close to 0). The latter ones correspond to symbols where the stream features were replaced by noise in the data generation.

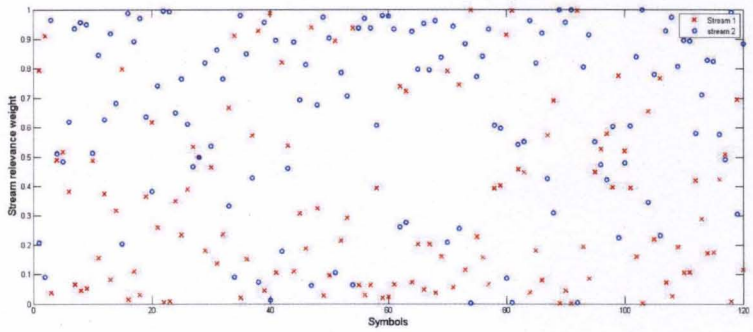


Figure 24. Stream 1 relevance weights of the symbols learned by the MSDHMM^{D} model for the double-stream sequential data

The learned weights for streams 1 by the MSDHMM^{P_l} and MSDHMM^{P_g} are displayed in Fig. 25 and 26. As it can be seen, some symbols are highly relevant (weight close to 1) in some states, while others are completely irrelevant (weight close to 0). The latter ones correspond to symbols where stream 1 features were replaced by noise in the data generation.

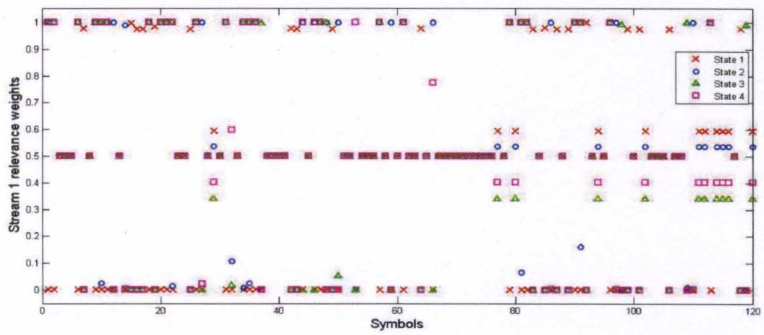


Figure 25. Stream 1 relevance weights of the symbols in all 4 states learned by the MSDHMM^{P_l} model for the double-stream sequential data

From table 3, we also notice that the probability based MSDHMMs outperform the distance-based MSDHMM. This can be attributed to two main factors. First, the MSDHMM^{D} learns an initial set of relevance weights and does not optimize these weights in the subsequent learning phase. Second, these weights are not state-dependent. The results also indicate that using the generalized Baum-Welch followed by the MCE to learn the model parameters is a better strategy. This is consistent with what have been reported for the baseline HMM [32].

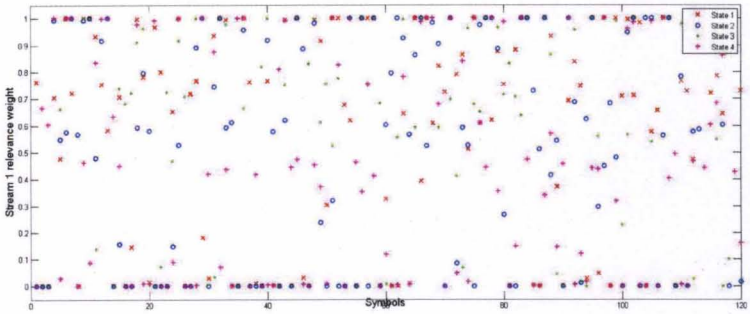
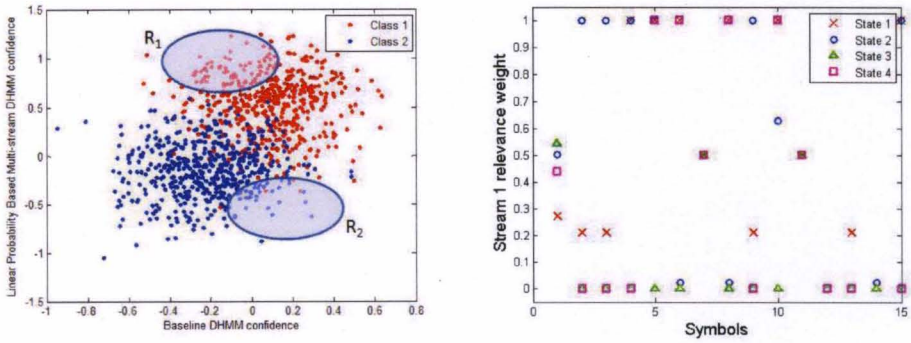


Figure 26. Stream 1 relevance weights of the symbols in all 4 states learned by the MSDHMM^{P_g} model for the double-stream sequential data



(a) (b)

Figure 27. (a) Scatter plot of the MSDHMM^{P_1} confidence values versus the baseline DHMM confidence values. (b) Stream 1 relevance weights of the closest symbols associated with 15 observation of a sequence extracted from R_1.

To illustrate the advantages of the MSDHMM further, in Fig. 27, we display a scatter plot of the baseline DHMM vs. the MSDHMM^{P_1} confidence values. As it can be seen, the confidence values are highly correlated. However, for few sequences (e.g. highlighted regions R_1 for class 1 and R_2 for class 2) the MSDHMM^{P_1} outperforms the baseline DHMM. To verify that this difference is attributed to the learned relevance weights, in Fig. 27 we display the learned stream 1 relevance weights for the symbols associated with the 15 observations for one of the sequences in region R_1. As it can be seen, only 4 symbols have equal relevance weights in all 4 states.

IV.6 Chapter summary

In this chapter, we have presented the details of the generalized multi-stream discrete HMM (GMSDHMM) structures. These models are proposed to take into account the different degree of

TABLE 3

Comparison of BW and MCE algorithms for the different DHMM structures

Classifier	Baum-Welch	MCE	BW and MCE
Baseline DHMM	54.075 %	59.075 %	60.025%
$MSDHMM^{D}$	62.075 %	64.075%	71.25%
$MSDHMM^{P_a}$	60.25%	70.25 %	72.65%
$MSDHMM^{P_g}$	58.25 %	65.25 %	75.00 %

relevancy of different streams. Our approach is data driven. It relies on training data to associate feature relevance weights to each symbol in the codebook. Two approaches have been proposed: distance based and probability based. In both cases, the Baum-Welch and MCE/GPG learning algorithms have been generalized to allow for simultaneous learning of all the model parameters. We derive the necessary conditions to update the different model parameters. The proposed structures have been evaluated using synthetic data sets. Results show that the $MSDHMM^{D}$ and $MSDHMM^{P}$ structures outperform the baseline DHMM.

CHAPTER V

Generalized Multi-stream Continuous Hidden Markov Models

It is not knowledge, but the act of learning,
not possession but the act of getting there,
which grants the greatest enjoyment.

Carl Friedrich Gauss

In this chapter, we propose two multi-stream Continuous HMM (MSCHMM) structures that integrate stream relevance weights and alleviate the limitations of existing multi-stream continuous HMM structures by linearizing the observation probability density function. This linearization allows the generalization of the Baum-Welch and the MCE/GPD training algorithms. In particular, we generalize the objective function to include stream relevance weights and derive the necessary conditions to update the parameters of both algorithms.

We assume that we have L streams of information. These streams could have been generated by different sensors and/or different feature extraction algorithms. Each stream is thus represented by a different subset of features. Instead of treating the streams equally important or using user-specified weights, the proposed MSCHMM structure introduces a built-in component to learn a relevance weight to each stream. Two forms of pdfs are proposed. A mixture level streaming pdf, and a state level streaming pdf. The former method models local stream relevance that depends on states and components. The latter method models a less local stream relevance that depends only on the states. We refer to the proposed MSCHMM structures with linear pdfs as MSCHMM^{L}.

V.1 Multi-stream CHMM with mixture level streaming

Let $b_{ijk}(o_t^{(k)})$ be the jth component in state i using only the feature subset coming from stream k, and let w_{ijk} be the stream relevance weight of this component. To cover the entire feature space (i.e. the L streams), we use a mixture of L components, i.e.,

$$b_{ij}(o_t) = \sum_{k=1}^{L} w_{ijk} b_{ijk}(o_t^{(k)}), \tag{V.1.1}$$

where $o_t^{(k)}$ is the kth stream contribution to the observation vector o_t. Then, to model each state by a mixture of M components, let

$$b_i(o_t) = \sum_{j=1}^{M} u_{ij} \sum_{k=1}^{L} w_{ijk} b_{ijk}(o_t^{(k)}), \tag{V.1.2}$$

subject to

$$\sum_{k=1}^{L} w_{ijk} = 1, \text{ and } \sum_{j=1}^{M} u_{ij} = 1. \tag{V.1.3}$$

The component $b_{ijk}(.)$ is assumed to be a normal distribution $\mathcal{N}(., \mu_{ijk}, \Sigma_{ijk})$ where μ_{ijk} is its mean and Σ_{ijk} is its diagonal covariance matrix. The distribution $\mathcal{N}(., \mu_{ijk}, \Sigma_{ijk})$ applies to the kth stream contribution $o_t^{(k)}$ of each observation vector o_t. The parameter u_{ij} is similar to the mixing coefficient in the standard HMM. In this case, it is a weight assigned to the mixture of L components that cover the entire feature space and not to a single component. We will refer to this mixture level MSCHMM^{L} as MSCHMM$^{\mathrm{L_m}}$.

The linearization of the pdf in (V.1.2) could be inferred from the following:

$$\begin{aligned}
b_i(o_t) &= Pr(o_t|q_t = i; \lambda) \\
&= \sum_{j=1}^{M} Pr(o_t|q_t = i, e_t = j; \lambda) Pr(e_t = j|q_t = i; \lambda) \\
&= \sum_{j=1}^{M} \sum_{k=1}^{L} Pr(o_t|q_t = i, e_t = j, f_t = k; \lambda) Pr(e_t = j|q_t = i; \lambda) Pr(f_t = k|q_t = i, e_t = j; \lambda) \\
&\approx \sum_{j=1}^{M} Pr(e_t = j|q_t = i; \lambda) \sum_{k=1}^{L} Pr(f_t = k|q_t = i, e_t = j; \lambda) Pr(o_t^{(k)}|q_t = i, e_t = j, f_t = k; \lambda)
\end{aligned}$$

where e_t and f_t are two random variables that represent the indices of the component and stream that occur in time t. It follows then that:

$$\begin{aligned}
b_{ijk}(o_t^{(k)}) &= Pr(o_t^{(k)}|q_t = i, e_t = j, f_t = k; \lambda), \\
w_{ijk} &= Pr(f_t = k|q_t = i, e_t = j; \lambda), \\
u_{ij} &= Pr(e_t = j|q_t = i; \lambda).
\end{aligned}$$

V.1.1 Generalized Baum-Welch learning algorithm for MSCHMM$^{\mathrm{L_m}}$

The MSCHMM$^{\mathrm{L_m}}$ parameters can be learned using the maximum Likelihood approach. Given a sequence of training observation $O = [o_1, \cdots, o_T]$, the parameters of λ could be learned by maximizing the likelihood of the observation sequence O, i.e., $Pr(O|\lambda)$. We achieve this by generalizing the Baum-Welch algorithm to include stream relevance weights. We define the generalized

Baum-Welch algorithm by extending the auxiliary function in (II.7.1) to

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \ln Pr(O, Q, E, F|\bar{\lambda}) \tag{V.1.4}$$

where $E = [e_1, \cdots, e_T]$ and $F = [f_1, \cdots, f_T]$ are two sequences of random variables representing respectively the component and stream indices at each time step. It can be shown that a critical point of $Pr(O|\lambda)$, with respect to λ, is a critical point of the new auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ with respect to $\bar{\lambda}$ when $\bar{\lambda} = \lambda$, that is,

$$\frac{\partial Pr(O|\lambda)}{\partial \lambda} = \frac{\partial \mathbb{Q}(\lambda, \bar{\lambda})}{\partial \bar{\lambda}}|_{\bar{\lambda}=\lambda}. \tag{V.1.5}$$

The proof of (V.1.5) could be achieved using the same steps needed to prove propositions (II.7.1) and (II.7.2).

Similar to the discrete case, it could be shown that the formulation of the maximization of the likelihood $Pr(O|\lambda)$ through maximizing the the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$ is an EM [30] type optimization that can be performed by an estimation step and a maximization step. The estimation step consists of computing the conditional expectation in (V.1.4) and writing it in an analytical form. The objective function in (V.1.4) involves the quantity $Pr(O, Q, E, F|\bar{\lambda})$ which could be expressed analytically as:

$$Pr(O, Q, E, F|\bar{\lambda}_c) = \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} u^{(c)}_{q_t e_t} w^{(c)}_{q_t e_t f_t} b^{(c)}_{q_t e_t f_t}(o_t) \tag{V.1.6}$$

Thus, the objective function in (V.1.4) can be expanded as follows:

$$\begin{aligned}
\mathbb{Q}(\lambda, \bar{\lambda}) \quad = \quad & \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \log \bar{\pi}_{q_1} + \\
& \sum_{t=1}^{T-1} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \log \bar{a}_{q_t q_{t+1}} + \\
& \sum_{t=1}^{T-1} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \log \bar{u}_{q_t e_t} + \\
& \sum_{t=1}^{T-1} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \log \bar{w}_{q_t e_t f_t} + \\
& \sum_{t=1}^{T-1} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F|O, \lambda) \log \mathcal{N}(o_t^{(f_t)}, \bar{\mu}_{q_t e_t f_t}, \bar{\Sigma}_{q_t e_t f_t})
\end{aligned} \tag{V.1.7}$$

After the estimation step, the maximization step consists of finding the parameters of $\bar{\lambda}$ that maximize the function in (V.1.7). The expanded form of the function $\mathbb{Q}(\lambda, \bar{\lambda})$ in (V.1.7) has 5 terms involving $\overline{\pi}$, $\overline{a}$,and ($\overline{w}$, $\overline{b}$) independently. To find the values of $\overline{\pi}_i$, $\overline{a}_{ij}$, $\overline{w}_{ijk}$, and $\overline{b}_{ijk}$ that maximize

$\mathbb{Q}(\lambda, \bar{\lambda})$, we consider the terms in (V.1.7) that depend on $\overline{\pi}$, $\overline{a}$, $\overline{w}$, and $\bar{b}$. In particular, the first and second terms in (V.1.7) depend on $\overline{\pi}$ and $\overline{a}$, and they have the same analytical expressions sketched in the case of the baseline CHMM (refer to (II.7.4)). It follows that the update equations for $\overline{\pi}_i$, $\overline{a}_{ij}$, and $\overline{u}_{ij}$ are the same as in the standard CHMM. That is,

$$\begin{aligned} \overline{\pi}_i &= \gamma_1(i), \\ \overline{a}_{ij} &= \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}, \end{aligned}$$

and

$$\overline{u}_{ij} = \frac{\sum_{t=1}^{T} Pr(q_t = i, e_t = j | o, \lambda)}{\sum_{t=1}^{T} Pr(q_t = i | o, \lambda)}.$$

To find the value of $\overline{w}_{ijk}$ that maximizes the auxiliary function $\mathbb{Q}(.,.)$, only the fourth term of the expression in (V.1.7) is considered since it is the only part of $\mathbb{Q}(.,.)$ that depends on $\overline{w}_{ijk}$. This term can be expressed as:

$$\sum_{t=1}^{T} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F | O, \lambda) \log \bar{w}_{q_t e_t f_t} = \sum_{t=1}^{T} \sum_{i} \sum_{j} \sum_{k} \log(\bar{w}_{ijk}) \times \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F | O, \lambda) \delta(i, q_t) \delta(j, e_t) \delta(k, f_t), \tag{V.1.8}$$

where $\delta(i, q_t)\delta(j, e_t)\delta(k, f_t)$ keeps only those cases for which $q_t = i$, $e_t = j$ and $f_t = k$. That is,

$$\sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F | O, \lambda) \delta(i, q_t) \delta(j, e_t) \delta(k, f_t) = Pr(q_t = i, e_t = j, f_t = k | o_t, \lambda), \tag{V.1.9}$$

therefore:

$$\sum_{t=1}^{T} \sum_{Q} \sum_{E} \sum_{F} Pr(Q, E, F | O, \lambda) \log \bar{w}_{q_t e_t f_t} = \sum_{t=1}^{T} \sum_{i=1}^{N_s} \sum_{j=1}^{M} \sum_{k=1}^{L} Pr(q_t = i, e_t = j, f_t = k | o_t, \lambda) \log \bar{w}_{q_t e_t f_t} \tag{V.1.10}$$

To find the update equation of $\overline{w}_{ijk}$ we use the Lagrange multipliers optimization with the constraint in (V.1.3), and obtain

$$w_{ijk} = \frac{\sum_{t=1}^{T} \gamma_t(i,j,k)}{\sum_{t=1}^{T} \gamma_t(i,j)}, \tag{V.1.11}$$

where

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N_s} \alpha_t(j)\beta_t(j)}, \tag{V.1.12}$$

$$\gamma_t(i,j) = \gamma_t(i) \frac{u_{ij} b_{ij}(o_t)}{b_i(o_t)}, \tag{V.1.13}$$

and

$$\gamma_t(i,j,k) = \gamma_t(i)\frac{u_{ij}w_{ijk}\mathcal{N}(o_t^{(k)},\mu_{ijk},\Sigma_{ijk})}{b_j(o_t)}. \tag{V.1.14}$$

Similarly, it can be shown that the update equations for the rest of the parameters are:

$$\mu_{ijkd} = \frac{\sum_{t=1}^{T}\gamma_t(i,j,k)o_{td}^{(k)}}{\sum_{t=1}^{T}\gamma_t(i,j,k)}, \tag{V.1.15}$$

and

$$\sigma_{ijkd} = \frac{\sum_{t=1}^{T}\gamma_t(i,j,k)(o_{td}^{(k)}-\mu_{ijkd})^2}{\sum_{t=1}^{T}\gamma_t(i,j,k)}. \tag{V.1.16}$$

The details of deriving the above update equations can be found in appendix D.

In the case of multiple observations $[O^{(1)},\ldots,O^{(R)}]$, it can be shown that the update equations become:

$$w_{ijk} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t^{(r)}(i,j,k)}{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t(i,j)}, \tag{V.1.17}$$

$$\mu_{ijkd} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t^{(r)}(i,j,k)o_{td}^{(k)}}{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t(i,j,k)}, \tag{V.1.18}$$

and

$$\sigma_{ijkd} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t^{(r)}(i,j,k)(o_{td}^{(k)}-\mu_{ijkd})^2}{\sum_{r=1}^{R}\sum_{t=1}^{T}\gamma_t(i,j,k)}. \tag{V.1.19}$$

The parameters $\gamma_t^{(r)}(i)$, $\gamma_t^{(r)}(i,j)$, and $\gamma_t^{(r)}(i,j,k)$ are the same as those for $\gamma_t(i)$, $\gamma_t(i,j)$, and $\gamma_t(i,j,k)$ when observation sequence $O^{(r)}$ is used.

Algorithm (8) outlines the steps of the Generalized Baum-Welch training algorithm for the parameters of the MSCHMM$^{\mathrm{L_m}}$.

Algorithm 8 Generalized BW training for the mixture level MSCHMM

Require: Training data $[O^{(1)},\cdots,O^{(R)}]$, $O^{(r)}=[o_1,\cdots,o_T]$. Fix the variables N_s, M, and L.
Ensure:

```
Cluster training data into N_s subsets and identify the N_s states.
Cluster each subset into M clusters and initialize the coefficients u_ij, stream relevance weights
  w_ijk, the centers and the matrices.
while stopping criteria not satisfied do
    Compute the probability density b_i(o_t) for each observation vector o_t using (IV.2.12);
    update A using (II.7.9);
    update u_ij using (II.7.23);
    update w_ijk using (V.1.17);
    update μ_ijkd using (V.1.18);
    update σ_ijkd using (V.1.19);
end while
```

V.1.2 Generalized MCE/GPD learning algorithm for MSCHMM^{L_m}

The minimization of the classification error via a gradient descent scheme is the most common discriminative training method for HMMs. In this section, we propose our generalized version for the MSCHMM^{L_m}. Let

$$g_c(O,\Lambda) = \log[\max_Q g_c(O,Q,\Lambda)] \tag{V.1.20}$$

be the discriminant function, associated with classifier λ, that indicates the degree to which O belongs to class c. In (V.1.20), Q is a state sequence corresponding to the observation sequence O, Λ includes the models parameters, and

$$\begin{aligned} g_c(O,Q,\Lambda) &= Pr(O,Q;\lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} b^{(c)}_{q_t}(o_t) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} \sum_{j=1}^{M} u^{(c)}_{q_t j} \sum_{k=1}^{L} w^{(c)}_{q_t jk} b^{(c)}_{q_t jk}(o_t) \end{aligned} \tag{V.1.21}$$

Thus, $g_c(O,\Lambda) = \log[g_c(O,\bar{Q},\Lambda)]$, where $\bar{Q} = [\bar{q}_1,\ldots,\bar{q}_T]$ is the optimal state sequence that achieves $\max_q g_c(O,q,\Lambda)$, which could be computed using the Viterbi algorithm [29].

The misclassification measure of sequence O is defined by:

$$d_c(O) = -g_c(O,\Lambda) + \log\left[\frac{1}{C-1}\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]\right]^{\frac{1}{\eta}} \tag{V.1.22}$$

where η is a positive number. The misclassification measure is embedded in a smoothed zero-one function, referred to as loss function, defined as:

$$l_c(O,\Lambda) = l(d_c(O)), \tag{V.1.23}$$

where l is the sigmoid function in (II.7.51). For an unknown sequence O, the classifier performance is measured by:

$$l(O;\Lambda) = \sum_{c=1}^{C} l_c(O;\Lambda)\mathbb{I}(O\in C_c) \tag{V.1.24}$$

where $\mathbb{I}(.)$ is the indicator function.

Given a set of training observation sequences $O^{(r)}$, $r = 1,2,\ldots,R$, an empirical loss function on the training data set is defined as

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R}\sum_{c=1}^{C} l_c(O;\Lambda)\mathbb{I}(O\in C_c). \tag{V.1.25}$$

Minimizing the empirical loss is equivalent to minimizing the total misclassification error. The MSCHMM^{L_m} parameters estimated by carrying out a gradient descent on $\mathbf{L}(\Lambda)$. In order to ensure

that the estimated MSCHMM^{Lm} parameters satisfy the stochastic constraints of $a_{ij} \geq 0$, $\sum_{j=1}^{N_s} a_{ij} = 1$, $u_{ij} \geq 0$, $\sum_{j=1}^{M} u_{ij} = 1$, $w_{ijk} \geq 0$, $\sum_{k=1}^{L} w_{ijk} = 1$, $\mu_{ijkd} \geq 0$, and $\sigma_{ijkd} \geq 0$, we map these parameters using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij}, \tag{V.1.26}$$

$$u_{ij} \rightarrow \tilde{u}_{ij} = \log u_{ij}, \tag{V.1.27}$$

$$w_{ijk} \rightarrow \tilde{w}_{ijk} = \log w_{ijk}, \tag{V.1.28}$$

$$\mu_{ijkd} \rightarrow \tilde{\mu}_{ijk} = \frac{\mu_{ijkd}}{\sigma_{ijkd}}, \tag{V.1.29}$$

$$\sigma_{ijkd} \rightarrow \tilde{\sigma}_{ijkd} = \log \sigma_{ijkd}. \tag{V.1.30}$$

Then, the parameters are updated with respect to $\tilde{\Lambda}$. After updating, we map them back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}}, \tag{V.1.31}$$

$$u_{ij} = \frac{\exp \tilde{u}_{ij}}{\sum_{j'=1}^{M} \exp \tilde{u}_{ij'}}, \tag{V.1.32}$$

$$w_{ijk} = \frac{\exp \tilde{w}_{ijk}}{\sum_{k'=1}^{L} \exp \tilde{w}_{ijk'}}, \tag{V.1.33}$$

$$\mu_{ijkd} = \tilde{\mu}_{ijkd}\sigma_{ijkd}, \tag{V.1.34}$$

$$\sigma_{ijkd} = \exp \tilde{\sigma}_{ijkd}. \tag{V.1.35}$$

Using a steepest descent batch estimation mode, the MSCHMM^{Lm} parameters are iteratively updated using:

$$\tilde{\Lambda}(\tau+1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda)\Big|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.1.36}$$

where ϵ is the learning rate, and ∇ is the gradient operator.

It can be shown that $\tilde{w}_{ij}^{(c)}$, $\tilde{\mu}_{ijkd}^{(c)}$, and $\tilde{\sigma}_{ijkd}^{(c)}$ need to be updated using:

$$\tilde{w}_{ijk}^{(c)}(\tau+1) = \tilde{w}_{ijk}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.1.37}$$

$$\tilde{\mu}_{ijkd}^{(c)}(\tau+1) = \tilde{\mu}_{ijkd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijkd}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.1.38}$$

and

$$\tilde{\sigma}_{ijkd}^{(c)}(\tau+1) = \tilde{\sigma}_{ijkd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}. \tag{V.1.39}$$

The derivatives $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$, $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijkd}^{(c)}}$, and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}}$ in (V.1.37), (V.1.38), and (V.1.39) could be expanded using the chain rule as follows:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{r=1}^{R} \sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial w_{ijk}^{(c)}} \times \frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.1.40}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijkd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial \mu_{ijkd}^{(c)}} \times \frac{\partial \mu_{ijkd}^{(c)}}{\partial \tilde{\mu}_{ijkd}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.1.41}$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial \sigma_{ijkd}^{(c)}} \times \frac{\partial \sigma_{ijkd}^{(c)}}{\partial \tilde{\sigma}_{ijkd}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.1.42}$$

where

$$\frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} = \zeta l_m(O,\Lambda)\left[1 - l_m(O,\Lambda)\right],$$

$$\frac{\partial d_c(O)}{\partial g_m(O,\Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}. \tag{V.1.43}$$

$$\frac{\partial g_m(O,\Lambda)}{\partial u_{ij}^{(c)}} = \sum_{t=1}^{T} \frac{b_{q_t j^{(c)}}}{b_{q_t(o_t)}} \delta(q_t, i), \tag{V.1.44}$$

$$\frac{\partial u_{ij}^{(c)}}{\partial \tilde{u}_{ij}^{(c)}} = u_{ij}^{(c)}\left[1 - u_{ij}^{(c)}\right].$$

$$\frac{\partial g_m(O,\Lambda)}{\partial w_{ijk}^{(c)}} = \sum_{t=1}^{T} \frac{u_{q_t j}^{(c)} b_{q_t jk}^{(c)}}{b_{q_t(o_t)}} \delta(q_t, i), \tag{V.1.45}$$

$$\frac{\partial w_{ijk}^{(c)}}{\partial \tilde{w}_{ijk}^{(c)}} = w_{ijk}^{(c)}\left[1 - w_{ijk}^{(c)}\right].$$

$$\frac{\partial g_m(O,\Lambda)}{\partial \mu_{ijkd}^{(c)}} = \sum_{t=1}^{T} \frac{u_{q_t j}^{(c)} w_{q_t jk}^{(c)} (o_{td}^{(k)} - \mu_{ijkd}^{(c)}) b_{q_t jk}^{(c)}}{(\sigma_{ijkd}^{(c)})^2 b_{q_t(o_t)}} \delta(q_t, i), \tag{V.1.46}$$

$$\frac{\partial \mu_{ijkd}^{(c)}}{\partial \tilde{\mu}_{ijkd}^{(c)}} = \sigma_{ijkd}^{(c)}.$$

$$\frac{\partial g_m(O,\Lambda)}{\partial \sigma_{ijkd}^{(c)}} = \sum_{t=1}^{T} \frac{u_{q_t j}^{(c)} w_{q_t jk}^{(c)} b_{q_t jk}^{(c)}}{b_{q_t(o_t)}} (\sigma_{ijkd}^{(c)})^{-1} ((\frac{o_{td}^{(k)} - \mu_{ijkd}^{(c)}}{\sigma_{ijkd}^{(c)}})^2 - 1) \delta(q_t, i), \tag{V.1.47}$$

and

$$\frac{\partial \sigma_{ijkd}^{(c)}}{\partial \tilde{\sigma}_{ijkd}^{(c)}} = \sigma_{ijkd}^{(c)}.$$

A closed form of $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}}$, $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijkd}^{(c)}}$, and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}}$ could be then inferred:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ijk}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1 - l_m(O_r,\Lambda)) w_{ijk}^{(c)}(1 - w_{ijk}^{(c)}) \delta(q_t, i) \frac{b_{q_t jk}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)} u_{q_t j}^{(c)} \times \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \tag{V.1.48}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ijkd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda)) \left[\sigma_{ijkd}^{(c)}\right]^{-1} \delta(q_t,i) u_{ij}^{(c)} (o_{td}^{(r)} - \mu_{ijkd}^{(c)}) \times \frac{b_{q_t jk}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)} w_{q_t jk}^{(c)} \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \quad \text{(V.1.49)}$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda)) \delta(q_t,i) u_{ij}^{(c)} w_{q_t jk}^{(c)} \times \left[\left(\frac{o_{td}^{(r)} - \mu_{ijkd}^{(c)}}{\sigma_{ijkd}^{(c)}}\right)^2 - 1 \right] \frac{b_{q_t jk}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)} \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}. \quad \text{(V.1.50)}$$

Algorithm (9) outlines the steps needed to learn the parameters of all the models λ_c in the MCE/GPD framework.

Algorithm 9 Generalized MCE/GPD training of the MSCHMM$^{\mathrm{L_m}}$

Require: Training data $[O^{(1)},\cdots,O^{(R)}]$, $O^{(r)}=[o_1,\cdots,o_T]$. Fix the variables N_s, M, and L for each model λ_c.

Ensure:

1: For each λ_c, cluster training data into N_s subsets and identify the N_s states.
2: For each λ_c, Cluster each subset into M clusters and initialize the coefficients u_{ij}, stream relevance weights w_{ijk}, the centers and the matrices.
3: **while** stopping criteria not satisfied **do**
4: Compute the probability density $b_i(o_t)$ of each observation vector o_t using (IV.1.7).
5: Compute the loss function of each sequence O using (V.1.24);
6: update $\mathbf{A}$ of each λ_c using (II.7.60);
7: update u_{ij} of each λ_c using (IV.2.39);
8: update w_{ijk} of each λ_c using (IV.2.38);
9: update μ_{ijkd} of each λ_c using (IV.2.38);
10: update σ_{ijkd} of each λ_c using (IV.2.38);
11: **end while**

V.2 Multi-stream CHMM with state level streaming

In this case, we assume that the streaming of data is performed at the state level, i.e., each state is generated by L different streams, and each stream embodies M Gaussian components. Let b_{ik} be the probability density function of state i within stream k. Since stream k is modeled by a mixture of M components, b_{ik} can be written as:

$$b_{ik}(o_t^{(k)}) = \sum_{j=1}^{M} u_{ijk} b_{ijk}(o_t^{(k)}), \quad \text{(V.2.1)}$$

where u_{ijk} represent the mixing coefficient of the jth component in each state i and generated by the kth stream, and $b_{ijk}(.)$ is a normal distribution $\mathcal{N}(.,\mu_{ijk},\Sigma_{ijk})$ with mean μ_{ijk} and diagonal

covariance matrix that applies to the kth stream contribution of observation vector o_t. Let w_{ik} be the relevance weight of stream k. The probability density function covering the entire feature space is then approximated by:

$$b_i(o_t) = \sum_{k=1}^{L} w_{ik} \sum_{j=1}^{M} u_{ikj} \mathcal{N}(o_t^{(k)}, \mu_{ikj}, \Sigma_{ikj}) \tag{V.2.2}$$

$$b_i(o_t) = \sum_{k=1}^{L} w_{ik} \sum_{j=1}^{M} u_{ijk} b_{ijk}(o_t^{(k)}), \tag{V.2.3}$$

subject to:

$$\sum_{k=1}^{L} w_{ik} = 1, \text{ and } \sum_{j=1}^{M} u_{ijk} = 1 \tag{V.2.4}$$

We will refer to this state level MSCHMM^{L} as MSCHMM$^{\mathrm{L_s}}$.

The linearization of the pdf in (V.2.3) could be inferred from:

$$\begin{aligned}
b_i(o_t) &= Pr(o_t|q_t = i; \lambda) \\
&= \sum_{k=1}^{L} Pr(o_t|q_t = i, f_t = k; \lambda) Pr(f_t = k|q_t = i; \lambda) \\
&\approx \sum_{k=1}^{L} Pr(o_t^{(k)}|q_t = i, f_t = k; \lambda) Pr(f_t = k|q_t = i; \lambda) \\
&= \sum_{k=1}^{L} Pr(f_t = k|q_t = i; \lambda) \sum_{j=1}^{M} Pr(e_t = j|q_t = i, f_t = k; \lambda) Pr(o_t^{(k)}|q_t = i, f_t = k, e_t = j; \lambda)
\end{aligned}$$

where e_t and f_t are two random variables that represent the indices of the component and stream that occur at time t. It follows then that

$$\begin{aligned}
\mathcal{N}(o_t^{(k)}, \mu_{ikj}, \Sigma_{ikj}) &= Pr(o_t^{(k)}|q_t = i, f_t = k, e_t = j; \lambda), \\
u_{ikj} &= Pr(e_t = j|q_t = i, f_t = k; \lambda), \\
w_{ik} &= Pr(f_t = k|q_t = i; \lambda).
\end{aligned}$$

V.2.1 Generalized Baum-Welch learning algorithm for MSCHMM$^{\mathrm{L_s}}$

The MSCHMM$^{\mathrm{L_s}}$ model parameters can be learned using a maximum Likelihood approach. Given a sequence of training observation $O = [o_1, \ldots, o_T]$, the parameters of λ could be learned by maximizing the likelihood of the observation sequence O, i.e., $Pr(O|\lambda)$. We achieve this by generalizing the Baum-Welch algorithm to include a stream relevance weight component. We define

5 terms involving $\overline{\pi}$, $\overline{a}$, $\overline{w}$, $\overline{u}$, and (μ, Σ). To find the values of $\overline{\pi}_i$, $\overline{a}_{ij}$, $\overline{w}_{ik}$, $\overline{u}_{ikj}$, $\overline{\mu}_{ikjd}$, and $\overline{\sigma}_{ikjd}$ that maximize $\mathbb{Q}(\lambda, \bar{\lambda})$, we consider the terms in (V.2.8) that depend on $\overline{\pi}$, $\overline{a}$, $\overline{w}$, $\overline{u}$, and (μ, Σ). In particular, the first and second terms in (V.1.7) depend on $\overline{\pi}$ and $\overline{a}$, and they have the same analytical expressions sketched in the case of the baseline CHMM in (II.7.4). It follows that the update equations for $\overline{\pi}_i$, and $\overline{a}_{ij}$ are the same as in the standard CHMM. That is,

$$\overline{\pi}_i = \gamma_1(i),$$

and

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}.$$

To find the value of $\overline{w}_{ik}$ that maximizes the auxiliary function $\mathbb{Q}(.,.)$, only the third term of the expression in (V.2.8) is considered since it is the only part of $\mathbb{Q}(.,.)$ that depends on $\overline{w}_{ik}$. This term can be expressed as:

$$\sum_{t=1}^{T}\sum_{Q}\sum_{F}\sum_{E} Pr(Q,E,F|O,\lambda)\log \bar{w}_{q_t f_t} = \sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log \bar{w}_{q_t f_t} =$$
$$\sum_{t=1}^{T}\sum_{i}\sum_{k} \log(\bar{w}_{ik}) \times \sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(k,f_t), \qquad \text{(V.2.9)}$$

where $\delta(i,q_t)\delta(k,f_t)$ keeps only those cases for which $q_t = i$, and $f_t = k$. That is,

$$\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(k,f_t) = Pr(q_t = i, f_t = k|o_t,\lambda), \qquad \text{(V.2.10)}$$

therefore:

$$\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log \bar{w}_{q_t f_t} =$$
$$\sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{k=1}^{L} Pr(q_t = i, f_t = k|o_t,\lambda)\log \bar{w}_{q_t f_t} \qquad \text{(V.2.11)}$$

To find the update equation of $\overline{w}_{ik}$ we use the Lagrange multipliers optimization with the constraint in (V.1.3), and obtain

$$\overline{w}_{ik} = \frac{\sum_{t=1}^{T} \gamma_t(i,k)}{\sum_{t=1}^{T} \gamma_t(i)}, \qquad \text{(V.2.12)}$$

where,

$$\gamma_t(i) = Pr(q_t = i|O,\lambda),$$

and

$$\gamma_t(i,k) = \gamma_t(i)\frac{w_{ik} b_{ik}(o_t)}{b_i(o_t)}.$$

Similarly, it can be shown that the update equations for the rest of the parameters are:

$$\overline{u}_{ikj} = \frac{\sum_{t=1}^{T} \gamma_t(i,k,j)}{\sum_{t=1}^{T} \gamma_t(i,k)}, \tag{V.2.13}$$

$$\overline{\mu}_{ikjd} = \frac{\sum_{t=1}^{T} \gamma_t(i,k,j) o_{td}^{(l)}}{\sum_{t=1}^{T} \gamma_t(i,k,j)}, \tag{V.2.14}$$

$$\overline{\sigma}_{ikjd} = \frac{\sum_{t=1}^{T} \gamma_t(i,k,j)(o_{td}^{(k)} - \mu_{ijd}^{(k)})^2}{\sum_{t=1}^{T} \gamma_t(i,k,j)}, \tag{V.2.15}$$

where

$$\gamma_t(i,k,j) = \gamma_t(i) \frac{w_{ik} u_{ijk} \mathcal{N}(o_t^{(k)}, \mu_{ijk}, \Sigma_{ijk})}{b_i(o_t)}$$

The details of deriving the above update equations can be found in appendix D.

For the case of multiple observations $[O^{(1)}, \ldots, O^{(R)}]$, it can be shown that the learning equations need to be updated using:

$$\overline{w}_{ik} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i,k)}{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i)}, \tag{V.2.16}$$

$$\overline{u}_{ijk} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i,k,j)}{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i,k)}, \tag{V.2.17}$$

$$\overline{\mu}_{ijkd} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t(i,k,j) o_{td}^{(k)(l)}}{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t(i,k,j)}, \tag{V.2.18}$$

and

$$\overline{\sigma}_{ijkd} = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i,k,j)(o_{td}^{(k)(r)} - \mu_{ijkd})^2}{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(i,k,j)}. \tag{V.2.19}$$

Algorithm (11) outlines the steps of the MLE learning algorithm for the different parameters of the MSCHMM^{L_s}.

Algorithm 10 Generalized BW training for the state level MSCHMM

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the parameters N_s, M and L.
Ensure:
Cluster training data into N_s clusters and initialize stream relevance weights w_{ik}.
Cluster each subset into M clusters and initialize the coefficients u_{ikj} , the centers and the matrices.
while stopping criteria not satisfied **do**
Compute the probability density $b_i(o_t)$ of each observation vector o_t using (V.2.3).
Update **A** using (II.7.9)
Update w_{ik} using (V.2.16)
Update u_{ikj} using (V.2.17)
Update μ_{ikjd} using (V.2.18)
Update σ_{ikjd} using (V.2.19)
end while

V.2.2 Generalized MCE/GPD learning algorithm for the MSCHMM^{L_s}

We generalize the MCE/GPD training approach to the case of MSCHMM^{L_s}. Let

$$g_c(O,\Lambda) = \log[\max_Q g_c(O,Q,\Lambda)] \tag{V.2.20}$$

be the discriminant function, associated with classifier λ, that indicates the degree to which O belongs to class c. In (V.2.20), Q is a state sequence correspondent to the observation sequence O, Λ includes the models parameters, and

$$\begin{aligned} g_c(O,Q,\Lambda) &= P(O,Q;\lambda_c) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} b^{(c)}_{q_t}(o_t) \\ &= \pi_{q_0^{(c)}} \prod_{t=1}^{T-1} a^{(c)}_{q_t q_{t+1}} \prod_{t=1}^{T} \sum_{k=1}^{L} w^{(c)}_{q_t k} \sum_{j=1}^{M} u^{(c)}_{q_t jk} b^{(c)}_{q_t jk}(o_t). \end{aligned} \tag{V.2.21}$$

Thus, $g_c(O,\Lambda) = \log[g_c(O,\bar{Q},\Lambda)]$, where $\bar{Q} = (\bar{q}_0, \bar{q}_1, \ldots, \bar{q}_T)$ is the optimal state sequence that achieves $\max_q g_c(O,q,\Lambda)$, which could be computed using the Viterbi algorithm [29].

The misclassification measure of the sequence O is defined by

$$d_c(O) = -g_c(O,\Lambda) + \log\left[\frac{1}{C-1}\sum_{j,j\neq c} \exp[\eta g_j(O,\Lambda)]\right]^{\frac{1}{\eta}} \tag{V.2.22}$$

where η is a positive number. The misclassification measure is first embedded in a smoothed zero-one function, referred to as loss function, defined as:

$$l_c(O,\Lambda) = l(d_c(O)), \tag{V.2.23}$$

where l is the sigmoid function in (II.7.51). Then, for any unknown sequence O, the classifier performance is measured by:

$$l(O;\Lambda) = \sum_{c=1}^{C} l_c(O;\Lambda)\mathbb{I}(O \in C_c) \tag{V.2.24}$$

where $\mathbb{I}(.)$ is the indicator function. For a set of training observation sequences $O^{(r)}$, $r = 1, 2, \ldots, R$, an empirical loss function on the training data set is defined as

$$\mathbf{L}(\Lambda) = \sum_{r=1}^{R}\sum_{c=1}^{C} l_c(O;\Lambda)\mathbb{I}(O \in C_c). \tag{V.2.25}$$

The MSCHMM^{L_s} parameters can be estimated by carrying out a gradient descent on $\mathbf{L}(\Lambda)$. In order to ensure that the estimated MSCHMM^{L_s} parameters satisfy the stochastic constraints of $a_{ij} \geq 0$,

$\sum_{j=1}^{N_s} a_{ij} = 1$, $w_{ik} \geq 0$, $\sum_{k=1}^{L} w_{ik} = 1$, $u_{ikj} \geq 0$, $\sum_{j=1}^{M} u_{ikj} = 1$, $\mu_{ikjd} \geq 0$, and $\sigma_{ikjd} \geq 0$, we map these parameters using

$$a_{ij} \rightarrow \tilde{a}_{ij} = \log a_{ij}, \tag{V.2.26}$$

$$w_{ik} \rightarrow \tilde{w}_{ik} = \log w_{ik}, \tag{V.2.27}$$

$$u_{ikj} \rightarrow \tilde{u}_{ikj} = \log u_{ikj}, \tag{V.2.28}$$

$$\mu_{ikjd} \rightarrow \tilde{\mu}_{ikjd} = \frac{\mu_{ikjd}}{\sigma_{ikjd}}, \tag{V.2.29}$$

and

$$\sigma_{ikjd} \rightarrow \tilde{\sigma}_{ikjd} = \log \sigma_{ikjd}. \tag{V.2.30}$$

Then, the parameters are updated w.r.t to $\tilde{\Lambda}$. After updating, we map them back using

$$a_{ij} = \frac{\exp \tilde{a}_{ij}}{\sum_{j'=1}^{N_s} \exp \tilde{a}_{ij'}}, \tag{V.2.31}$$

$$w_{ik} = \frac{\exp \tilde{w}_{ik}}{\sum_{k'=1}^{L} \exp \tilde{w}_{ik'}}, \tag{V.2.32}$$

$$u_{ikj} = \frac{\exp \tilde{u}_{ikj}}{\sum_{j'=1}^{M} \exp \tilde{u}_{ikj'}}, \tag{V.2.33}$$

$$\mu_{ikjd} = \tilde{\mu}_{ikjd}\sigma_{ikjd}, \tag{V.2.34}$$

and

$$\sigma_{ikjd} = \exp \tilde{\sigma}_{ikjd}. \tag{V.2.35}$$

Using a batch estimation mode, the the MSCHMM$^{\mathrm{L_s}}$ parameters are iteratively updated using:

$$\tilde{\Lambda}(\tau+1) = \tilde{\Lambda}(\tau) - \epsilon \nabla_{\tilde{\Lambda}} \mathbf{L}(\Lambda)\Big|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}. \tag{V.2.36}$$

where ϵ is the learning rate, and ∇ is the gradient operator.

It can be shown that $\tilde{w}_{ij}^{(c)}$, $\tilde{u}_{ikj}^{(c)}$, $\tilde{\mu}_{ijkd}^{(c)}$, and $\tilde{\sigma}_{ijkd}^{(c)}$ need to be updated using:

$$\tilde{w}_{ik}^{(c)}(\tau+1) = \tilde{w}_{ik}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ik}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.2.37}$$

$$\tilde{u}_{ikj}^{(c)}(\tau+1) = \tilde{u}_{ikj}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{u}_{ikj}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.2.38}$$

$$\tilde{\mu}_{ikjd}^{(c)}(\tau+1) = \tilde{\mu}_{ikjd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ikjd}^{(c)}}\Bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}, \tag{V.2.39}$$

and

$$\tilde{\sigma}_{ikjd}^{(c)}(\tau+1) = \tilde{\sigma}_{ikjd}^{(c)}(\tau) - \epsilon \frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ikjd}^{(c)}}\bigg|_{\tilde{\Lambda}=\tilde{\Lambda}(\tau)}. \tag{V.2.40}$$

The above partial derivatives could be expanded using chain rule as follows:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ik}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial w_{ik}^{(c)}} \times \frac{\partial w_{ik}^{(c)}}{\partial \tilde{w}_{ik}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.2.41}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{u}_{ikj}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial u_{ikj}^{(c)}} \times \frac{\partial u_{ikj}^{(c)}}{\partial \tilde{u}_{ikj}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.2.42}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ikjd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial \mu_{ikjd}^{(c)}} \times \frac{\partial \mu_{ikjd}^{(c)}}{\partial \tilde{\mu}_{ikjd}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.2.43}$$

and

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ijkd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C} \frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} \times \frac{\partial d_m(O)}{\partial g_c(O,\Lambda)} \times \frac{\partial g_c(O,\Lambda)}{\partial \sigma_{ijkd}^{(c)}} \times \frac{\partial \sigma_{ijkd}^{(c)}}{\partial \tilde{\sigma}_{ijkd}^{(c)}} \mathbb{I}(O \in C_c), \tag{V.2.44}$$

where

$$\frac{\partial l_m(O,\Lambda)}{\partial d_m(O)} = \zeta l_m(O,\Lambda)\left[1 - l_m(O,\Lambda)\right],$$

$$\frac{\partial d_c(O)}{\partial g_m(O,\Lambda)} = \begin{cases} -1 & \text{if } c = m \\ \frac{\exp[\eta g_c(O,\Lambda)]}{\sum_{j,j\neq c}\exp[\eta g_j(O,\Lambda)]} & \text{if } c \neq m \end{cases}, \tag{V.2.45}$$

$$\frac{\partial g_m(O,\Lambda)}{\partial w_{ik}^{(c)}} = \sum_{t=1}^{T} \frac{b_{q_t k}^{(c)}}{b_{q_t(o_t)}} \delta(q_t, i), \tag{V.2.46}$$

$$\frac{\partial w_{ik}^{(c)}}{\partial \tilde{w}_{ik}^{(c)}} = w_{ik}^{(c)}\left[1 - w_{ik}^{(c)}\right],$$

$$\frac{\partial g_m(O,\Lambda)}{\partial u_{ikj}^{(c)}} = \sum_{t=1}^{T} \frac{w_{q_t k}^{(c)} b_{q_t jk^{(c)}}}{b_{q_t(o_t)}^{(c)}} \delta(q_t, i), \tag{V.2.47}$$

$$\frac{\partial u_{ij}^{(c)}}{\partial \tilde{u}_{ikj}^{(c)}} = u_{ikj}^{(c)}\left[1 - u_{ikj}^{(c)}\right].$$

$$\frac{\partial g_m(O,\Lambda)}{\partial \mu_{ikjd}^{(c)}} = \sum_{t=1}^{T} \frac{w_{q_t k}^{(c)} u_{q_t kj}^{(c)} (o_{td}^{(k)} - \mu_{ikjd}^{(c)}) b_{q_t jk}^{(c)}}{(\sigma_{ikjd}^{(c)})^2 b_{q_t(o_t)}} \delta(q_t, i), \tag{V.2.48}$$

$$\frac{\partial \mu_{ikjd}^{(c)}}{\partial \tilde{\mu}_{ikjd}^{(c)}} = \sigma_{ikjd}^{(c)}.$$

$$\frac{\partial g_m(O,\Lambda)}{\partial \sigma_{ikjd}^{(c)}} = \sum_{t=1}^{T} \frac{w_{q_t k}^{(c)} u_{q_t kj}^{(c)} b_{q_t kj}^{(c)}}{b_{q_t(o_t)}} (\sigma_{ikjd}^{(c)})^{-1} ((\frac{o_{td}^{(k)} - \mu_{ijkd}^{(c)}}{\sigma_{ijkd}^{(c)}})^2 - 1) \delta(q_t, i), \tag{V.2.49}$$

and

$$\frac{\partial \sigma_{ikjd}^{(c)}}{\partial \tilde{\sigma}_{ikjd}^{(c)}} = \sigma_{ikjd}^{(c)}.$$

A closed form of $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ik}^{(c)}}$, $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{u}_{ikj}^{(c)}}$, $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ikjd}^{(c)}}$, and $\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ikjd}^{(c)}}$ could then be inferred:

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{w}_{ik}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))w_{ik}^{(c)}(1-w_{ik}^{(c)})\delta(q_t,i)\frac{b_{q_tk}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)} \times \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \quad \text{(V.2.50)}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{u}_{ikj}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))u_{ikj}^{(c)}(1-u_{ikj}^{(c)})w_{ik}^{(c)}\delta(q_t,i)\frac{b_{q_tkj}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)} \times \frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \quad \text{(V.2.51)}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\mu}_{ikjd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))\left[\sigma_{ikjd}^{(c)}\right]^{-1}\delta(q_t,i)w_{ik}^{(c)}u_{ikj}^{(c)}(o_{td}^{(r)}-\mu_{ikjd}^{(c)}) \times \frac{b_{q_tkj}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)}\frac{\partial d_c(O_r)}{\partial_m(O_r,\Lambda)}, \quad \text{(V.2.52)}$$

$$\frac{\partial \mathbf{L}(\Lambda)}{\partial \tilde{\sigma}_{ikjd}^{(c)}} = \sum_{r=1}^{R}\sum_{m=1}^{C}\sum_{t=1}^{T} \zeta l_m(O_r,\Lambda)(1-l_m(O_r,\Lambda))\delta(q_t,i)w_{ik}^{(c)}u_{ikj}^{(c)} \times \left[\left(\frac{o_{td}^{(r)}-\mu_{ikjd}^{(c)}}{\sigma_{ikjd}^{(c)}}\right)^2-1\right]\frac{b_{q_tkj}^{(c)}(o_t)}{b_{q_t}^{(c)}(o_t)}\frac{\partial d_c(O_r)}{\partial g_m(O_r,\Lambda)}, \quad \text{(V.2.53)}$$

Algorithm (11) outlines the generalized MCE/GPD training procedure for the different parameters of the MSCHMM^{L_s}.

V.3 Inference

To test a new observation sequence O=[o_1,...,o_T], we need to compute $Pr(O|\lambda_c)$ with respect to each model λ_c. This computation can be performed efficiently using the Viterbi algorithm [29]. The Viterbi algorithm also computes also the corresponding optimal state sequence [q_1,..., q_T] of O. This in turn requires the computation of $b_i(o_t)$. For the MSCHMM, it could be computed using (V.1.2) in the case of mixture level MSCHMM, and (V.2.1) in the case of state level MSCHMM.

Algorithm 11 Generalized MCE/GPD training for the state level MSCHMM

Require: Training data $[O^{(1)}, \cdots, O^{(R)}]$, $O^{(r)}=[o_1, \cdots, o_T]$. Fix the variables N_s, M, and L for each model λ_c.

Ensure:

For each λ_c, cluster training data into N_s clusters and initialize stream relevance weights w_{ik}.
For each λ_c, cluster each subset into M clusters and initialize the coefficients u_{ikj} , the centers and the matrices.
while stopping criteria not satisfied **do**
  Compute the probability density $b_i(o_t)$ of each observation vector o_t using (V.2.3).
  Compute the loss function of each sequence O using (V.2.24);
  Update $\mathbf{A}$ of each λ_c using (II.7.60)
  Update w_{ik} of each λ_c using (V.2.37)
  Update u_{ikj} of each λ_c using (V.2.38)
  Update μ_{ikjd} of each λ_c using (V.2.39)
  Update σ_{ikjd} of each λ_c using (V.2.40)
end while

V.4 Convergence properties

V.4.1 On the convergence properties of the Generalized Baum-Welch algorithm

The aim of the Baum-Welch algorithm is to find estimates of the HMM parameters that maximize the likelihood $Pr(O|\lambda)$. As mentioned in the previous chapter, it is well known that the maximum likelihood estimator (MLE) is asymptotically (in the presence of infinite data collection) optimal [53]. This however may not be possible for most applications.

In addition, it is desirable to reach the global maximum of the likelihood objective function. The Baum-Welch algorithm carries out an EM like optimization of the likelihood function. As stated previously, the estimation step consists of writing an analytical form of the auxiliary functions in (V.1.4), and (V.2.5) which have the form of a conditional expectation. The maximization step consists on finding a maximum (local at least, global if possible) of the auxiliary function $\mathbb{Q}(\lambda, \bar{\lambda})$. However, it was shown that the solutions found by algorithms 8 and 11 are proven to be critical points of the likelihood function $Pr(O|\lambda)$. Therefore, it is of interest to ensure that these solutions are (local) maximum of their correspondent objective functions. This is given by the following theorem:

Theorem V.4.1. *The generalized Baum-Welch ensures convergence to a* ***local maximum*** *for the* $MSCHMM^{L_m}$ *and the* $MSCHMM^{L_s}$.

Proof. It could be shown that the computed critical points are local maximum since the second derivative of each objective function is negative when evaluated on the obtained critical points.
In the following we show that the objective function $\mathbb{Q}(\lambda, \bar{\lambda})$ in (V.1.4) is locally maximized when

evaluated in the points computed in (II.7.9), (V.1.17), (V.1.18), and (V.1.19). In fact, for $\tilde{w}_{ijk}$,

$$\frac{\partial^2 \mathbb{Q}(\lambda, \bar{\lambda})}{\partial \tilde{w}_{ijk}^2} = -\sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t = i, e_t = j, f_t = k|O, \lambda)\,[\bar{w}_{ijk}]^{-2} \leq 0 \tag{V.4.1}$$

The same result could be found for the rest of the parameters. Thus, it could be concluded that the solutions in (II.7.9), (V.1.17), (V.1.18), and (V.1.19) represent a local maxima of the objective $\mathbb{Q}(\lambda, \bar{\lambda})$ in the case of MSCHMM$^{\mathrm{L_m}}$. Similar steps lead to the same conclusion for the MSCHMM$^{\mathrm{L_s}}$.

□

V.4.2 On the convergence properties of the Generalized MCE/GPD algorithm

It has been proven in [52] that the MCE empirical cost measured on a finite training set approximates the theoretical classification risk. As the training data grows larger, the MCE estimates have the property to minimize Bayes risk. In addition, reducing the MCE empirical loss can always be achieved by the steepest descent mechanism if a sufficiently small learning rate is chosen. However, this almost guaranteed convergence does not always imply a fast convergence rate [54].

V.4.3 Evaluation on a synthetic data

V.4.4 Data generation

To validate the proposed MSCHMM structures, we generate two synthetic data sets. The first set is a single stream sequential data, and the second one is a multi-stream one. Both sets are generated using two continuous HMMs to simulate a two class problem. We follow a similar approach to the one used in [55] to generate sequential data using a continuous HMM with $N_s = 4$ states and $M = 4$ components with 4 dimensions. We start by fixing N_s different vectors $\mu_i \in \mathbb{R}^4$, $i = 1, \cdots, N_s$ to represent the different states. Then, we randomly generate M vectors from each normal distribution with mean μ_i and identity covariance matrix to form the mixture components of each state. The mixture weights of the components of each state are randomly generated and then normalized. The covariance matrix of each mixture component is set to identity. The initial state probability distribution and the state transition probability distribution are generated randomly from a uniform distribution in the interval $[0, 1]$. The randomly generated values are then scaled to satisfy the stochastic constraints.

For the single stream sequential data, we generate R sequences of length $T = 15$ vectors with dimension $p = 4$ for each of the two classes. We start by generating a continuous HMM with

N_s states and M components as described above. Then, we generate the single stream sequences using Algorithm (12).

Algorithm 12 Single stream sequential data generation for each class.

for $r = 1$ to R **do**
  Select the initial state according to the initial states probability distribution π
  Randomly pick a component v from the M components representing the selected state according to its mixture weights
  Sample an observation from a normal distribution with mean v and covariance σI
  for $t = 2$ to T **do**
    Select next state according to the probabilities transition matrix A,
    Randomly pick a component v among those representing the selected state,
    Sample an observation o_t from the normal distribution which mean v and covariance σI.
  end for
end for

For the multi-stream case, we assume that the sequential data is synthesized by L=2 streams, and that each stream k is described by N_s states, where each state is represented by vector μ_i^k of dimension p_k=4. To construct a set of M components based on the L streams, for each state i, three components are generated from each stream k, and concatenated to form a double-stream components. To simulate components with various relevance wights, we create 3 combinations of components in each state. The first combination consists of concatenating a component from each stream by just appending the features (i.e., both streams are relevant). The second combination consists on concatenating noise (instead of stream 2 features) to stream 1 features (i.e., stream 1 is relevant and stream 2 is irrelevant). The last combination consists on concatenating noise (instead of stream 1 features) to stream 2 features (i.e., stream 1 is irrelevant and stream 2 is relevant). Thus, for each state i we have a set of double-stream components where the streams have different degrees of relevance. Once the set of double-stream components is generated, a state transition probability distribution is generated, and the double-stream sequential data is generated using Algorithm (12).

V.4.5 Results

In the first experiment, we apply the baseline CHMM and the proposed multi-stream CHMM structures to the single stream sequential data where the features are generated from one homogeneous source of information. The MSCHMM architectures treat the single stream sequential data as a double-stream one (each stream is assumed to have 2-dimensional observation vectors). In this experiment all models are trained using standard Baum-Welch (for the baseline CHMM), the generalized Baum-Welch (for the MSCHMM), the standard and generalized MCE/GPD algorithms, or a combination of the two (Baum-Welch followed by MCE/GPD). The results of this experiment

are reported in table 5. As it can be seen, the performance of the proposed MSCHMM structures and the baseline CHMM are comparable for most training methods. This is because when both streams are equally relevant for the entire data, the different streams receive nearly equal weights in all states' components and the MSCHMM reduces to baseline CHMM.

Fig. 28 displays stream 1 relevance weights for components of the 4 states learned by the MSCHMM^{L_m}. As it can be seen, most weights are clustered around 0.5 (maximum weight is less than 0.6 and minimum weight is more than 0.4). Since weights of both streams must sum to 1, both weights use equally important for all symbols. The stream relevance weights learned by the MSCHMM^{L_s} are shown in table 4 . Similar results are obtained for the MSCHMM^{L_m}.

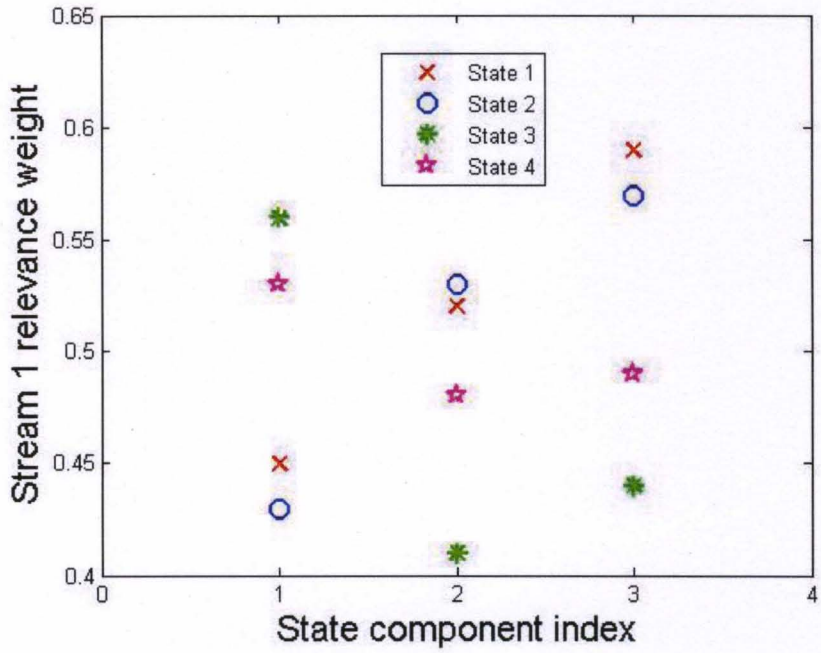


Figure 28. Stream 1 relevance weights of the mixture components in all 4 states, learned by the MSCHMM^{L_m} model for the single-stream sequential data.

TABLE 4

Stream relevance weights of the MSCHMM^{L_s} learned from the single stream data

state \ stream	k=1	k=2
i=1	0.4770	0.5230
i=2	0.5889	0.4111
i=3	0.4950	0.5050
i=4	0.5022	0.4978

TABLE 5

Classification rates of the different CHMM structures for the single stream data

Classifier	Baum-Welch	MCE	BW and MCE
Baseline CHMM	89.00 %	91.25%	93.15%
MSCHMM^{L_m}	93.25 %	94.00%	95.00%
MSCHMM^{L_s}	92.75 %	95.25 %	97.45%

The second experiment involves applying the baseline CHMM and the MSCHMM^{L} structures to the double stream sequential data where the features are generated from two different streams. In this experiment the various models are trained using Baum-Welch, MCE, and Baum-Welch followed by MCE training algorithms. First, we note that using stream relevance weights, the generalized Baum-Welch and MCE training algorithms converge faster and result in a small error. Fig. 29 displays the number of misclassified samples versus the number of iterations for the baseline CHMM and the MSCHMM^{L} structures using MCE/GPD training. As it can be seen, learning stream relevance weights causes the error to drop faster. In fact, at each iteration, the classification error for the MSCHMM^{L} structure is lower than the baseline CHMM.

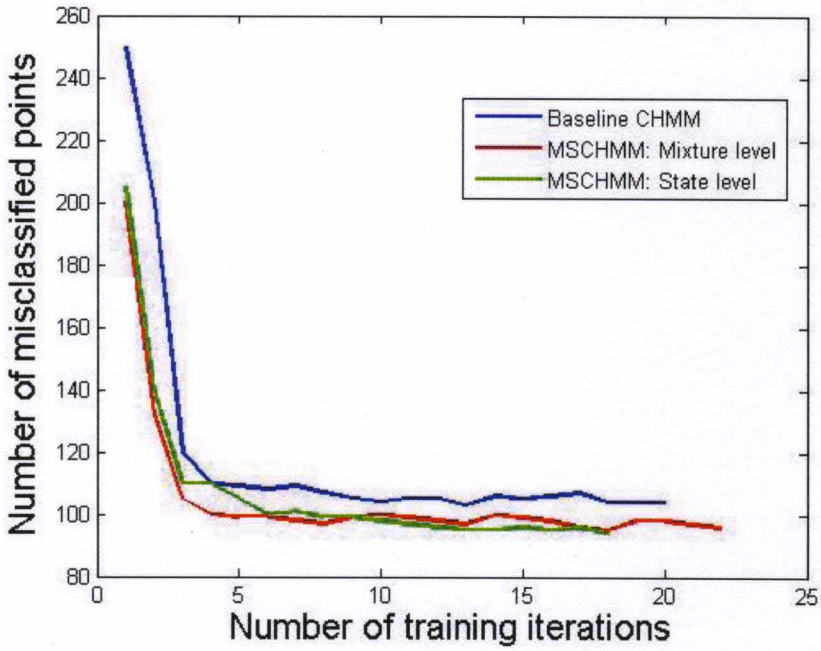


Figure 29. Number of misclassified samples versus the number of iterations for the standard and MSCHMM^{L}

The testing results are reported in table 7. First, we note that all MSCHMM^{L} structures outperform the baseline CHMM for all training methods. This is because the data set used for this experiment was generated from two streams with different degrees of relevancy and the baseline CHMM treats both streams equally important. The MSCHMM^{L} structures on the other hand, learn the optimal relevance weights for each symbol within each state. The learned weights for stream 1 by the $\text{MSCHMM}^{\text{L}_{\text{m}}}$ are displayed in Fig. 31. As it can be seen, some components are highly relevant (weight close to 1) in some states, while others are completely irrelevant (weights close to 0). The latter ones correspond to the components where stream 1 features were replaced by noise in the data generation.

Table 6 shows the stream relevance weights learned by the $\text{MSCHMM}^{\text{L}_{\text{s}}}$. As it can be seen, the learned relevance vary. This is consistent with the fact that the two streams of data do not have the same relevance. Also, all the proposed MSCHMM^{L} outperform the existing MSCHMM^{G} structures [25, 24]. This is mainly due to the fact that the parameters of MSCHMM^{L} structures are updated simultaneously by both Baum-Welch and MCE/GPD training. However, for the MSCHMM^{G} parameters are learned separately.

From table 7, we also notice that using the generalized Baum-Welch followed by the MCE to learn the model parameters is a better strategy. This is consistent with what have been reported for the baseline HMM [32].

To illustrate the advantages of the MSCHMM^{L} further, in Fig. 30, we display a scatter plot of the baseline CHMM vs. the $\text{MSCHMM}^{\text{L}_{\text{m}}}$ confidence values. As it can be seen, the confidence values are highly correlated. However, for few sequences (e.g. highlighted regions R_1 for class 1 and R_2 for class 2) the $\text{MSCHMM}^{\text{L}_{\text{m}}}$ outperforms the baseline CHMM. To verify that this difference is attributed to the learned relevance weights, we consider one of the sequences in R_1. For all the 15 observations, the learned stream 1 relevance weights for the components of the most likely state of $\text{MSCHMM}^{\text{L}_{\text{m}}}$ are displayed in Fig. 32 . As it can be seen, none of the components have equal stream relevance weights in all 4 states.

V.5 Chapter summary

In this chapter, we have presented the details of the generalized multi-stream continuous HMM (GMSCHMM) structures. These models are proposed to take into account the different degree of relevancy of different streams. Our approach is data driven. It relies on training data to associate feature relevance weights to each state and/or mixture component. The proposed GM-

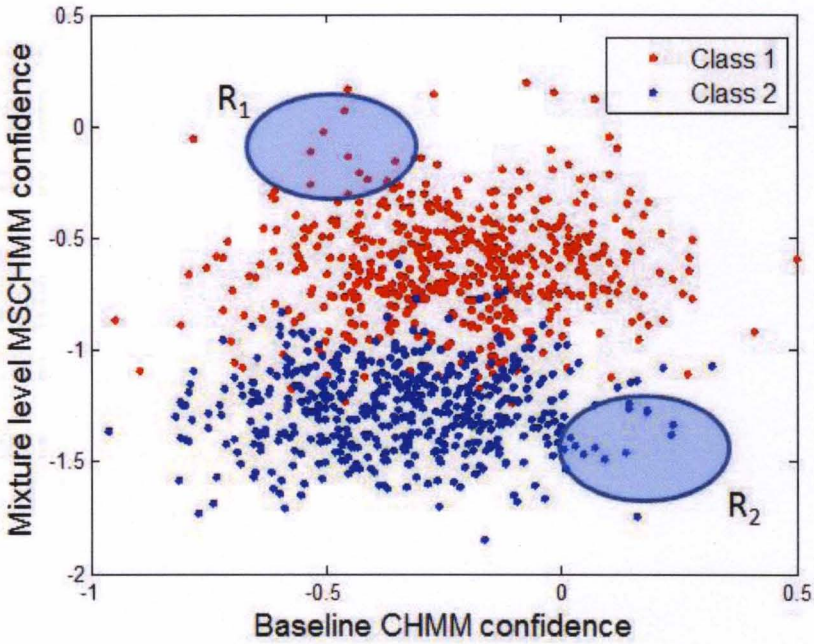


Figure 30. Scatter plot of the confidences of the two-class data in the baseline CHMM and the $\text{MSCHMM}^{\text{Lm}}$.

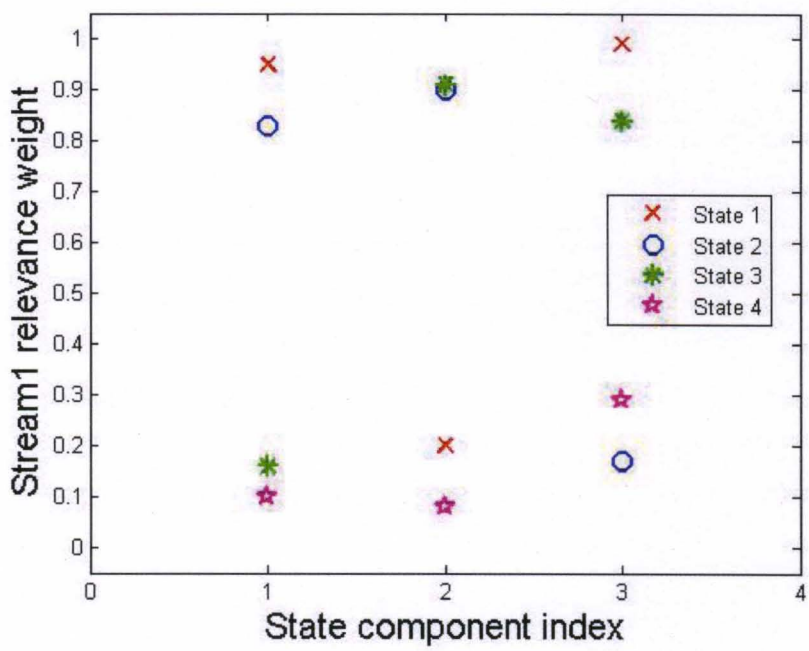


Figure 31. Stream 1 relevance weights of the mixture components in all 4 states learned by the $\text{MSCHMM}^{\text{Lm}}$ model for the double-stream sequential data

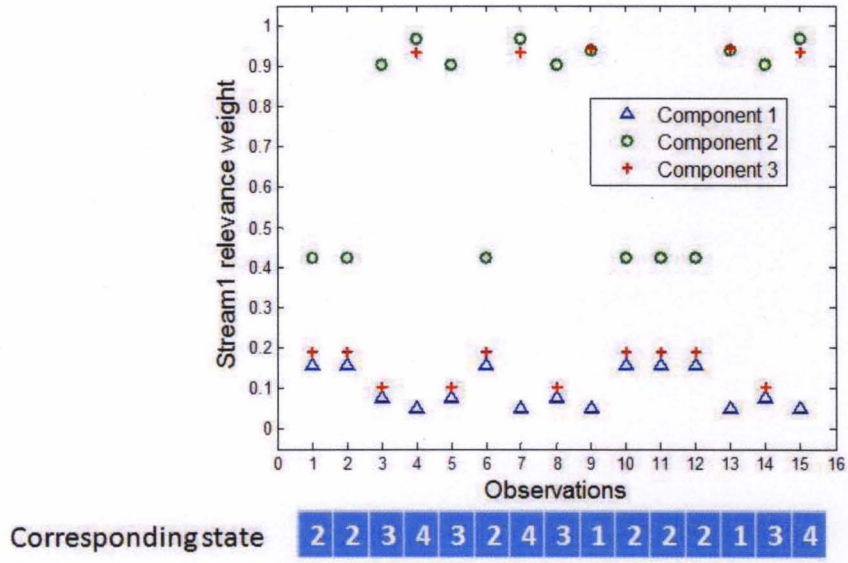


Figure 32. Stream 1 relevance weights of the mixture components in most likely state corresponding to the observation of a sequence from R_1 in Fig.fig:SynthScatterMissedPointCHMM, learned by the MSCHMM^{L_m}

TABLE 6

Stream relevance weights of the MSCHMM^{L_s} learned from the double stream data

state \ stream	k=1	k=2
i=1	0.2857	0.7143
i=2	0.3636	0.6364
i=3	0.4000	0.6000
i=4	0.1429	0.8571

SCHMM architectures include stream relevance component via the linearization of the observation pdf. The pdf linearization meets the independence assumption between streams of data. Two form of pdf have been proposed: mixture and state level pdfs. In both cases, the Baum-Welch and MCE/GPG learning algorithms have been generalized to allow for simultaneous learning of all the model parameters. We derive the necessary conditions to update the different model parameters. The proposed structures have been evaluated using synthetic data sets. Results show that the MSCHMM^{L_m} and MSCHMM^{L_s} structures outperform the baseline CHMM.

TABLE 7

Comparison of BW and MCE algorithms for the different CHMM structures

Classifier	Baum-Welch	MCE	BW and MCE
Baseline CHMM	63.25 %	65.75 %	68.85%
$MSCHMM^{L_m}$	70.35 %	72.75%	79.65%
$MSCHMM^{L_s}$	71.65%	71.25 %	80.00%
$MSCHMM^{G_m}$	-	-	70.65%
$MSCHMM^{G_s}$	-	-	72.00%

CHAPTER VI

Applications

Experience does not ever err; it is only your judgment that errs in promising itself results which are not caused by your experiments

Leonardo da Vinci

In this chapter, the proposed multi-stream Hidden Markov models structures are evaluated using real data sets for landmine detection, Australian sign language classification, audio classification, and face classification. We show that the proposed MSHMM structures outperform the standard HMM as well as existing multi-stream HMM.

VI.1 Landmine detection using ground penetrating radar

VI.1.1 Introduction

Detection, localization and subsequent neutralization of buried antipersonnel (AP) and anti-tank (AT) landmines is a worldwide humanitarian and military problem. The latest statistics show that in 2006, a total of 5,751 casualties from mines were recorded in 68 countries and areas, including 1,367 people killed and 4,296 injured. In fact, the number of mine survivors in the world continue to grow and reached over 473,000 in 2006, many needing life-long care. Detection and removal of landmines is therefore a significant problem, and has attracted several researchers in recent years. One challenge in landmine detection lies in plastic or low metal mines that cannot or are difficult to detect by traditional metal detectors. Varieties of sensors have been proposed or are under investigation for landmine detection. The research problem for sensor data analysis is to determine how well signatures of landmines can be characterized and distinguished from other objects under the ground using returns from one or more sensors. Ground Penetrating Radar (GPR) offers the promise of detecting landmines with little or no metal content. Unfortunately, landmine detection via GPR has been a difficult problem [56, 57]. Although systems can achieve high detection rates,

they have done so at the expense of high false alarm rates. The key challenge to mine detection technology lies in achieving a high rate of mine detection while maintaining low level of false alarms. The performance of a mine detection system is therefore commonly measured by a receiver operating characteristics (ROC) curve that jointly specifies rate of mine detection and level of false alarm.

Automated detection algorithms can generally be broken down into four phases: pre-processing, feature extraction, confidence assignment, and decision-making. Pre-processing algorithms perform tasks such as normalization of the data, corrections for variations in height and speed, removal of stationary effects due to the system response, etc. Methods that have been used to perform this task include wavelets and Kalman filters [58], subspace methods and matching to polynomials [59], and subtracting optimally shifted and scaled reference vectors[60]. Feature extraction algorithms reduce the pre-processed raw data to form a lower-dimensional, salient set of measures that represent the data. Principal component (PC) transforms are a common tool to achieve this task [61, 62]. Other feature analysis approaches include wavelets [63] image processing methods of derivative feature extraction [64], curve analysis using Hough and Radon transforms [65], as well as model-based methods. Confidence assignment algorithms can use methods such as Bayesian [65], hidden Markov Models [64, 66, 35], fuzzy logic [67], rules and order statistics[68], neural networks, or nearest neighbor classifiers [69, 70], to assign a confidence that a mine is present at a point. Decision-making algorithms often post-process the data to remove spurious responses and use a set of confidence values produced by the confidence assignment algorithm to make a final mine/no-mine decision.

In [64, 66], hidden Markov modeling was proposed for detecting both metal and nonmetal mine types using data collected by a moving-vehicle-mounted GPR system and has proved that HMM techniques are feasible and effective for landmine detection. This (baseline) system uses observation vectors that encode the degree to which edges occur in the diagonal and anti-diagonal directions. It assumes that mine signatures have a rising edge (with an orientation close to 45°) and a falling edge (with an orientation close to 135°). This assumption may be too restrictive for some signature and may degrade the performance of the HMM detector. In this dissertation, we propose an alternative approach to extract features for the HMM detector that does not impose an explicit structure on the signature. This approach is based on Gabor filters and encodes the signature by its response to multiple filters at different scales and orientations [35]. Moreover, the edge histogram descriptors (EHD) [70], an MPEG7 based feature extraction mechanism, is also used in this application. Since the different features are not equally important in characterizing different mine types in different

Figure 33. NIITEK vehicle mounted GPR system

environments, we will use the proposed multi-stream HMM structures to assign different weights to different features.

VI.1.2 Data Preprocessing and Pre-screening

VI.1.2.1 GPR Data

The input data consists of a sequence of raw GPR signatures collected by a NIITEK Inc. landmine detection system comprising a vehicle-mounted 51-channel GPR array [71] (see Fig. 33). The NIITEK GPR collects 51 channels of data. Adjacent channels are spaced approximately 5 centimeters apart in the cross-track direction, and sequences (or scans) are taken at approximately 6 centimeter down-track intervals. The system uses a V-dipole antenna that generates a wide-band pulse ranging from 200 MHz to 7 GHz. Each A-scan, that is, the measured waveform that is collected in one channel at one downtrack position, contains 416 time samples at which the GPR signal return is recorded. Each sample corresponds to roughly 8 picoseconds. We often refer to the time index as depth although, since the radar wave is traveling through different media, this index does not represent a uniform sampling of depth. Thus, we model an entire collection of input data as a three-dimensional matrix of sample values, $S(z, x, y), z = 1, \cdots, 416; x = 1, \cdots, 51; y = 1, \cdots, N_S$, where N_S is the total number of collected scans, and the indices z, x,and y represent depth, cross-track position, and down-track positions respectively. A collection of scans, forming a volume of data, is illustrated in Fig. 34.

Fig. 35 displays several B-scans (sequences of A-scans) both downtrack (formed from a

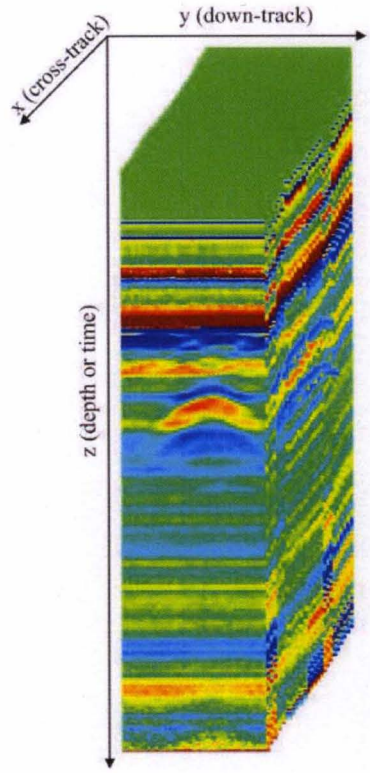
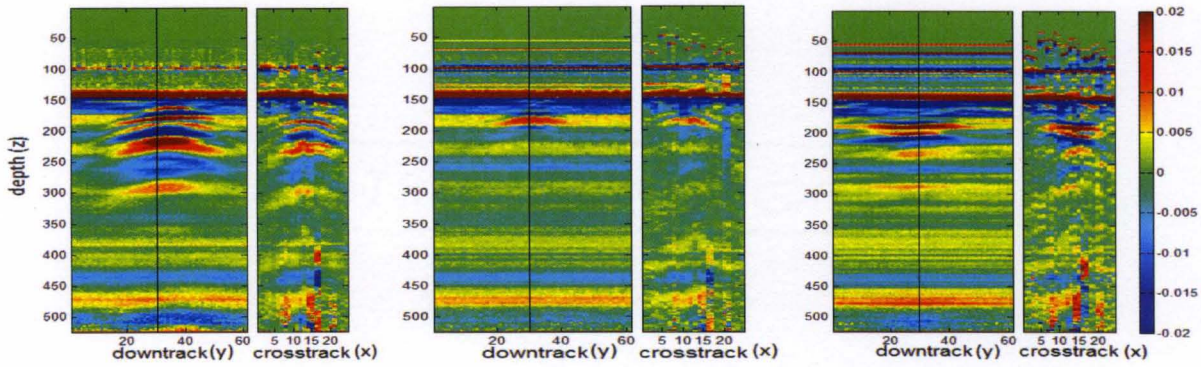


Figure 34. a collection of few GPR scans



(a) AT High-metal mine (b) AP High-metal mine (c) Non-metal Clutter

Figure 35. NIITEK Radar down-track and cross-track (at position indicated by a line in the down-track) B-scans pairs for (a) an Anti-Tank (AT) mine, (b) an Anti-Personnel (AP) mine, and (c) a non-metal clutter alarm.

time sequence of A-scans from a single sensor channel) and crosstrack (formed from each channels response in a single sample). The surveyed object position is highlighted in each figure. The objects scanned are (a) a high-metal content antitank mine, (b) a low-metal antitank mine, and (c) a wood block.

VI.1.2.2 Data preprocessing

Preprocessing is an important step to enhance the mine signatures for detection. In general, preprocessing includes ground-level alignment and signal and noise background removal. First, we identify the location of the ground bounce as the signals peak and align the multiple signals with respect to their peaks. This alignment is necessary because the vehicle-mounted system cannot maintain the radar antenna at a fixed distance above the ground. The early time samples of each signal, up to few samples beyond the ground bounce are discarded. The remaining signal samples are divided into N depth bins, and each bin would be processed independently. The reason for this segmentation is to compensate for the high contrast between the responses from deeply buried and shallow anomalies. Next, the adaptive least mean squares (LMS) pre-screener proposed by Torrione et al. [72] is used to focus attention and identify regions with subsurface anomalies. The goal of a pre-screener algorithm in the framework of vehicle-mounted realtime landmine detection is to flag locations of interest utilizing a computationally inexpensive algorithm so that more advanced feature-processing approaches are applied only to the small subsets of data flagged by the pre-screener. The LMS is applied to the energy at each depth bin and assigns a confidence value to each point in the cross-track, down-track plane based on its contrast with a neighboring region. The components that satisfy empirically pre-determined conditions are considered as potential targets. Their cross-track x_s, and down-track y_s positions of the connected component center are reported as alarm positions for further processing by the feature-based discrimination algorithm to attempt to separate mine targets from naturally occurring clutter.

VI.1.3 Feature Extraction

VI.1.3.1 Gradient based features

Landmines (and other buried objects) appear in time domain GPR as shapes that are similar to hyperbolas corrupted by noise. Thus, the feature representation adopted by the HMM-based system is based on the degree to which edges occur in the diagonal and anti-diagonal directions, and the features were extracted to accentuate these edges. Figure 36 displays a hyperbolic curve superimposed on a preprocessed metal mine signature to illustrate the features of a typical mine signature. First, we compute the first and second derivative of the signal $S(x, y, z)$ along the down-

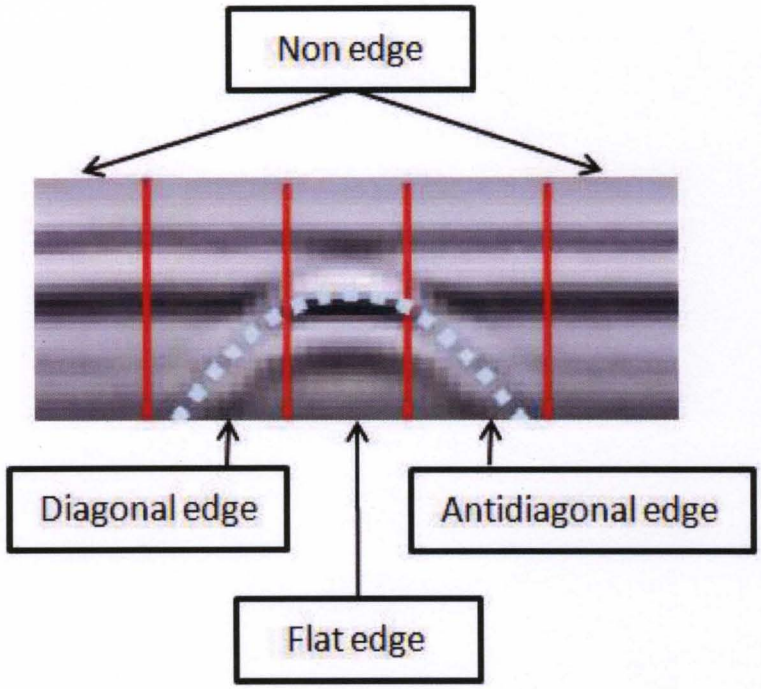


Figure 36. Shape of a typical mine signature and the interpretation of the 4 states of the HMM stucture.

track (y) direction using:

$$\begin{aligned} D_y(x,y,z) &= \frac{[S(x,y+2,z)+2S(x,y,z)-2S(x,y-1,z)-S(x,y-2,z)]}{3}, \\ D_{yy}(x,y,z) &= \frac{[D_y(x,y+2,z)+2D_y(x,y-1,z)-D_y(x,y-2,z)]}{3} \end{aligned} \tag{VI.1.1}$$

Then, the derivative values are normalized using

$$N(x,y,z) = \frac{D_{yy}(x,y,z)-\mu(x,z)}{\sigma(x,z)}, \tag{VI.1.2}$$

where $\mu(x,z)$ and $\sigma(x,z)$ are the running mean and standard deviation updated using a small background area around the target flagged by the prescreener.

The down-track dimension is taken as the time variable in the HMM model. The goal is to produce a confidence that a mine is present at various positions, (x,y), on the surface being traversed. To fit into the HMM context, a sequence of observation vectors must be produced for each point. These observation vectors encode the degree to which edges occur in the diagonal and antidiagonal directions. The observation vector at a point (x_s, y_s) consists of a set of 15 features that are computed on a normalized array of GPR data of size 32×8. Let x_s and y_s be given and let A denote the array

$$A = A(y,z) = N(x,y,z), \tag{VI.1.3}$$

where $x = x_s, y = y_s - 3, \ldots, y_s + 4$, and $z = 1, 2, \ldots, 32$. The array A is then broken into positive and negative parts according to the formulas

$$A^+(y,z) = \begin{cases} A(y,z) & \text{if } A(y,z) \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{VI.1.4}$$

$$A^-(y,z) = \begin{cases} -A(y,z) & \text{if } A(y,z) \leq -1 \\ 0 & \text{otherwise} \end{cases} \tag{VI.1.5}$$

Next, for each point in the positive and negative parts of A, the strengths of the diagonal and anti-diagonal edges are estimated. The strengths are measured by taking the local minimum in either the 45° or 135° direction around the column $y_s + 1$. Four types of edges that correspond to the, positive anti-diagonal (PA), negative anti-diagonal (NA), positive diagonal (PD), and negative diagonal (ND) edges are defined. These edges are computed using

$$\begin{aligned}
PA(z) &= \min A^+(y_s, z-1), A^+(y_s+1, z), A^+(y_s+2, z+1), A^+(y_s+3, z+2) \\
NA(z) &= \min A^-(y_s, z-1), A^-(y_s+1, z), A^-(y_s+2, z+1), A^-(y_s+3, z+2) \\
PD(z) &= \min A^+(y_s, z+2), A^+(y_s+1, z+1), A^+(y_s+2, z), A^+(y_s+3, z-1) \\
ND(z) &= \min A^-(y_s, z+2), A^-(y_s+1, z+2), A^-(y_s+2, z), A^-(y_s+3, z-1)
\end{aligned}$$

For each edge type, we find the position of the maximum value over a neighborhood of 32 depth values. For example, in the array PA we compute

$$m_{pa} = \text{argmax}\{PA(z) : z = 1, 2, \cdots, 32\} \tag{VI.1.6}$$

where m_{pa} denotes "maximum of the positive anti-diagonal". The variables m_{pd}, m_{na}, and m_{nd} are defined similarly. The values of the positive and negative diagonal and anti-diagonal arrays are used to define the 4-dimensional (4-D) observation vector associated with the point (x_s, y_s), $O(x_s, y_s) = [PD(m_{pd}), PA(m_{pa}), ND(m_{nd}), NA(m_{na})]$. Observation sequences of length 15 are formed at point (x, y) by extracting the observation sequence:

$$O(x, y-7), O(x, y-6), \cdots, O(x, y-1), O(x, y), O(x, y+1), \cdots, O(x, y+7).$$

VI.1.3.2 Gabor based features

The edge features assume that mine signatures have a diagonal (45°) rising edge and an anti-diagonal (135°) falling edge. However, this assumption may be too restrictive and may not be satisfied for some mine signatures. In fact, a rising edge could follow other orientations such as (30°)

or (60°) depending on the radar resolution and the sampling rate. In addition, the gradient edge features are extracted locally and thus, they do not consider the global variability or the frequencies of the signature.

In this section, we adopt the Homogeneous Texture Descriptor [73] to capture the spatial distribution of the edges within the 3-D GPR alarms. In particular, we propose extracting features by expanding the signature's B-scan using a bank of scale and orientation selective Gabor filters. We fix the number of scales to four and the number of orientations to four at a 45° intervals.

Let $S(x, y, z)$ denote the 3-D GPR data volume of an alarm. To keep the computation simple, we use 2-D filters (in the $y - z$ plane) and average the response over the third dimension. Let $S_x(y, z)$ be the xth plane of the 3-D signature $S(x, y, z)$. Let $SG_x^{(k)}(y, z)$, $k = 1, \cdots, 16$ denote the response of $S_x(y, z)$ to the 16 Gabor filters. Fig. 37(a) displays a strong signature of a typical metal mine and its response to the 16 Gabor filters. As it can be seen, the signature has a strong response to the θ_2 (45°) filters (especially scale 1 and scale 2 to a lesser degree) on the left part of the signature (rising edge), and a strong response to the θ_4 (135°) filters on the right part of the signature (falling edge). Similarly, the middle of the signature has a strong response to the θ_3 (horizontal) filters (flat edge). Fig. 37(b) displays a weak mine signature and its response to the Gabor filters. For this signature, the edges are not as strong as those in Fig. 37(a). As a result, it has a weaker response at all scales (scale 2 has the strongest response), especially for the falling edge. Fig. 37(c) displays a clutter signature (with high energy) and its response. As it can be seen, this signature has strong response to the θ_4 (135°) degree filters. However, this response is not localized on the right side of the signature as it is the case for most mine signatures.

In our HMM models, we take the down-track dimension as the time variable (i.e., y corresponds to time in the HMM model). Our goal is to produce a confidence that a mine is present at various positions, (x, y), on the surface being traversed. To fit into the HMM context, a sequence of observation vectors must be produced at each point. The observation sequence of $S_x(y, z)$ at a fixed depth z, is the sequence of 15 observation vectors

$$\mathbf{O}(x, y-7, z), \mathbf{O}(x, y-6, z), \cdots, \mathbf{O}(x, y-1, z), \mathbf{O}(x, y, z), \mathbf{O}(x, y+1, z), \cdots, \mathbf{O}(x, y+7, z),$$

where

$$\mathbf{O}(x, y, z) = [O^1(x, y, z), \cdots, O^{16}(x, y, z)], \tag{VI.1.7}$$

and

$$O^k(x, y, z) = \frac{1}{45} \sum_{z=1}^{45} SG_x^{(k)}(y, z), \tag{VI.1.8}$$

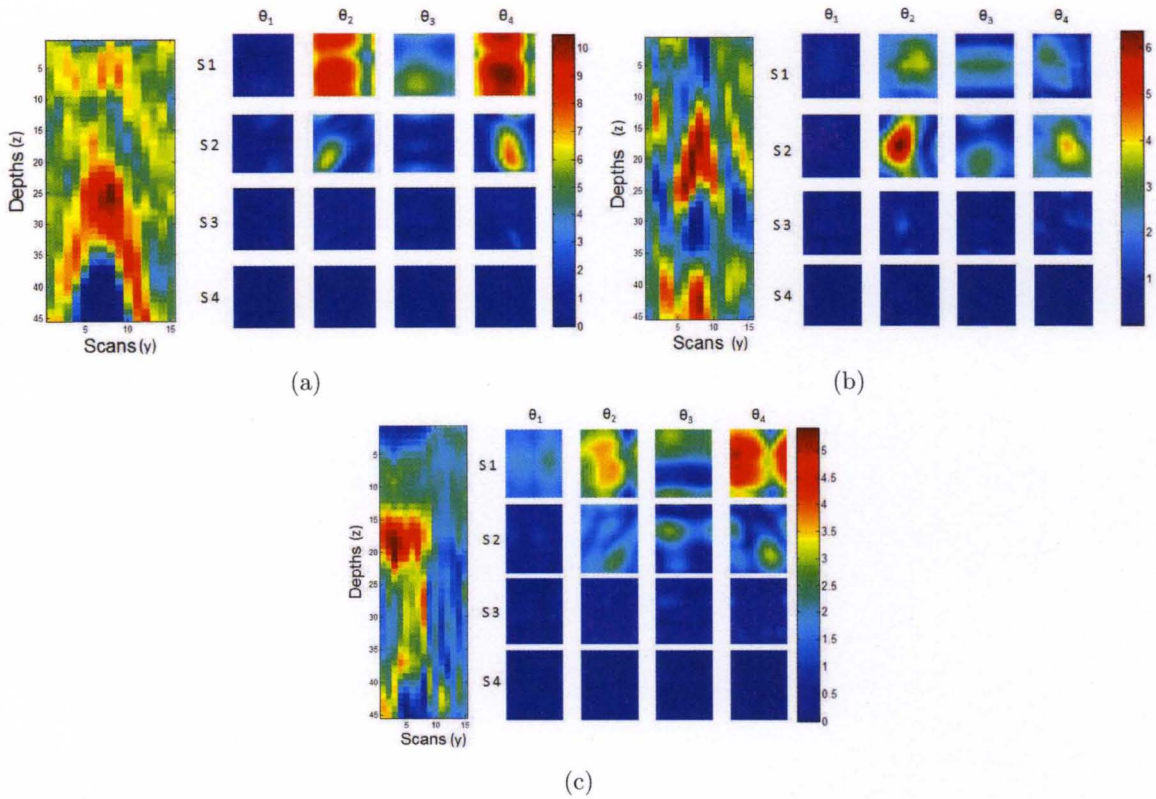


Figure 37. Response of 3 alarms to the 16 Gabor filters at different scales and orientations. (a) Strong mine signature, (b) Weak mine signature, and (c) clutter signature with high energy.

encodes the response of $S(x, y, z)$ to the kth Gabor filters.

VI.1.3.3 Edge histogram descriptors

The Edge Histogram Descriptors (EHD) [74] captures the salient properties of the 3-D alarms in a compact and translation-invariant representation. This approach, inspired by the MPEG-7 EHD [75], extracts edge histograms capturing the frequency of occurrence of edge orientations in the data associated with a ground position. The basic MPEG-7 EHD has undergone rigorous testing and development, and thus, represents one of the mature, generic, and efficient texture descriptors. For a generic image, the EHD represents the frequency and the directionality of the brightness changes in the image. Simple edge detector operators are used to identify edges and group them into five categories: vertical, horizontal, 45° diagonal, 135° antidiagonal, and isotropic (nonedges). The EHD would include five bins corresponding to the aforementioned categories. For our application, we adapt the EHD to capture the spatial distribution of the edges within a 3-D GPR data volume. To keep the computation simple, we still use 2-D edge operators. In particular, we fix the cross-

track dimension and extract edges in the (depth, down-track) plane. The overall edge histogram is obtained by averaging the output of the individual (depth, down-track) planes. Also, since vertical, horizontal, diagonal, and antidiagonal edges are the main orientations present in the mine signatures, we keep the five edge categories of the MPEG-7 EHD.

Let $S_{zy}^{(x)}$ be the xth plane of the 3-D signature $S(x, y, z)$. First, for each $S_{zy}^{(x)}$, we compute four categories of edge strengths: vertical, horizontal, 45° diagonal, 135° antidiagonal. If the maximum of the edge strengths exceeds a certain preset threshold θ_G, the corresponding pixel is considered to be an edge pixel. Otherwise, it is considered a nonedge pixel.

In our HMM models, we take the down-track dimension as the time variable (i.e., y corresponds to time in the HMM model). Our goal is to produce a confidence that a mine is present at various positions, (x, y), on the surface being traversed. To fit into the HMM context, a sequence of observation vectors must be produced for each point. The observation sequence of $S_{zy}^{(x)}$ at a fixed depth z, is the sequence of 15 observation vectors $H_{zy_i}^{(x)}$, $i = 1, \cdots, 15$, each represents a five-bin edge histogram correspondent to $S_{zy_i}^{(x)}$.

The overall sequence of observation vectors computer from the 3-D signature $S(x, y, z)$ is then:

$$O(x, y, z) = [\overline{H}_{zy_1}, \overline{H}_{zy_2}, \cdots, \overline{H}_{zy_{15}}], \tag{VI.1.9}$$

where $\overline{H}_{zy_i}$ is the cross-track average of the edge histograms of subimage $S_{zy_i}^{(x)}$ over N_C channels, i.e.

$$\overline{H}_{zy_i} = \frac{1}{N_C} \sum_{x=1}^{N_C} \overline{H}_{zy_i}. \tag{VI.1.10}$$

The extraction of the EHD is illustrated in Fig. 38.

Figs. 39 and 40 display the edge histogram feature for a strong mine and a false alarm identified by the prescreener due to its high-energy contrast. As can be seen, the EHD of the mine signature can be characterized by a stronger response to the diagonal and antidiagonal edges. Moreover, the frequency of the diagonal edges is higher than the frequency of the antidiagonal edges on the left of the image (rising edge of the signature) and lower on the right part (falling edge). This feature is typical in mine signatures. The EHD of the false alarm, on the other hand, does not follow this pattern. The edges do not follow a specific structure, and the diagonal and antidiagonal edges are usually weaker.

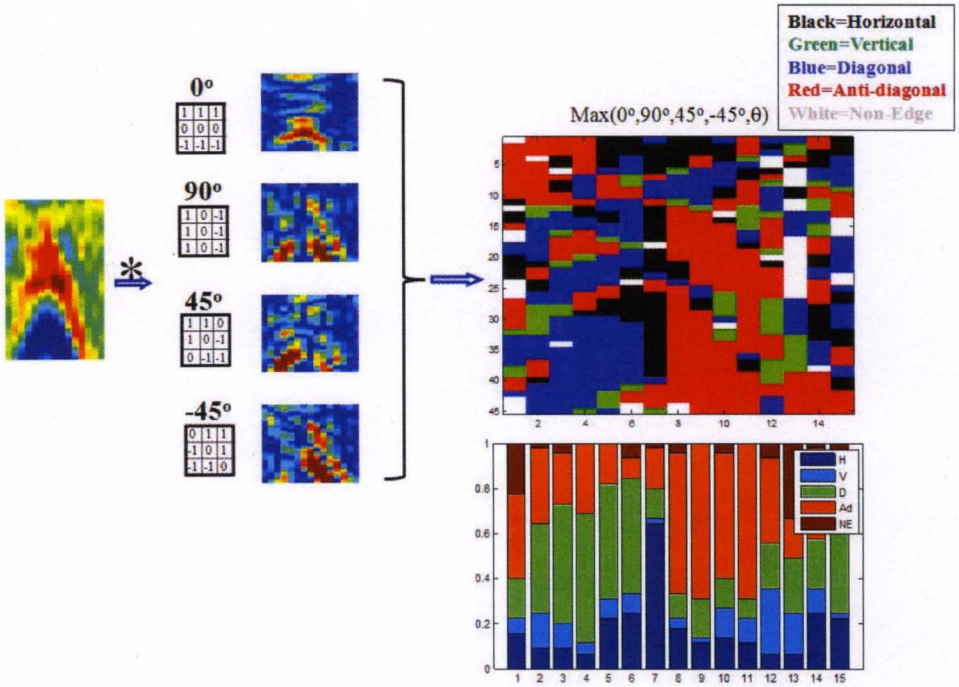


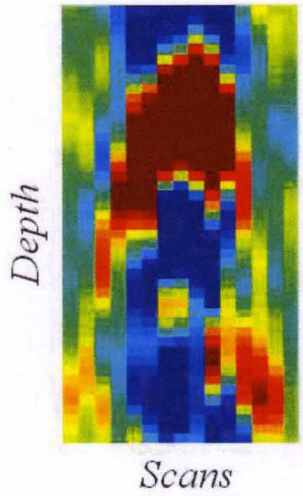
Figure 38. Illustration of the EHD feature extraction process.

VI.1.4 HMM parameters learning

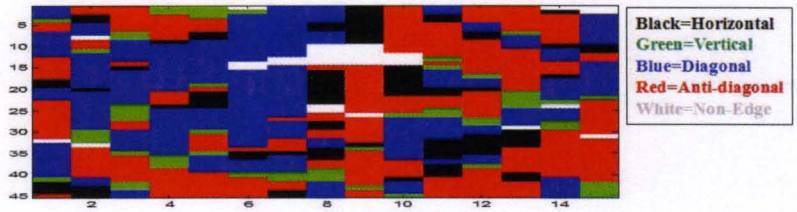
VI.1.4.1 Baseline (single stream) HMM

The baseline HMM classifier for landmine detection consists of two HMM models, one for mine and one for background. Each model has three or four states and produces a probability value by backtracking through model states using the Viterbi algorithm [29]. The mine model, λ^m, is designed to capture the hyperbolic spatial distribution of the features. Typically, λ^m has 3 states, they correspond to the rising edge, flat, and decreasing edge. The mine model is left to right model in that states are ordered and the transition probabilities for moving to a lower numbered state are zero.

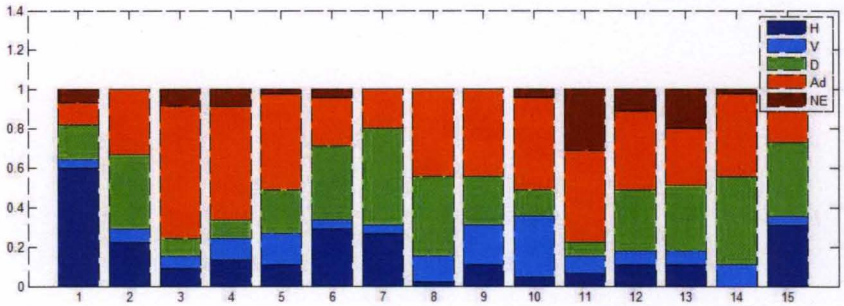
Another architecture is to have λ^m with four states. These states correspond to the non-edge, the rising edge, flat, and decreasing edge. The mine model is illustrated in Fig. 41. In addition to the mine model λ^m, a clutter model λ^b is needed to capture the background characteristics and to reject clutter. The clutter model, have three or four states depending on the corresponding mine model. The probability value produced by the mine (clutter) model can be thought of as an estimate of the probability of the observation sequence given that there is a mine (clutter) present.



(a)

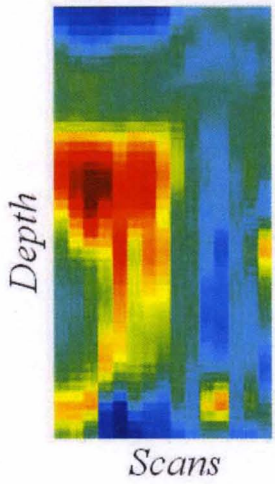


(b)

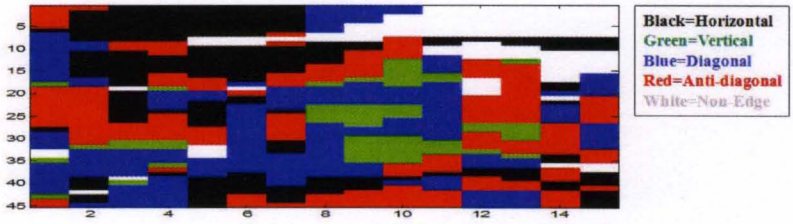


(c)

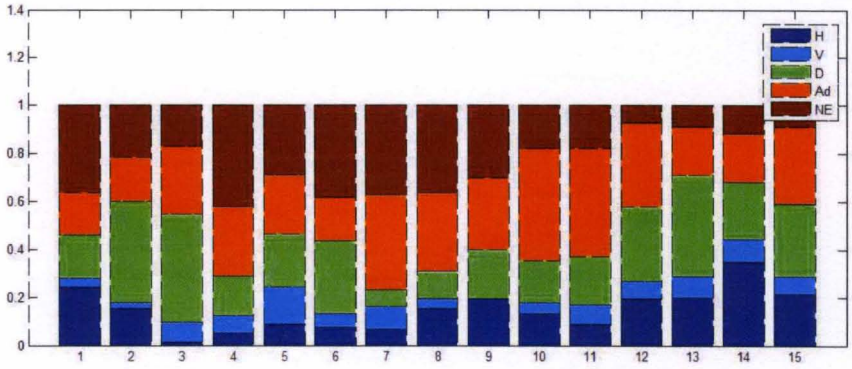
Figure 39. EHD feature of a strong mine signature. (a) Mine signature in the (depth, down-track) plane. (b) Pixels classified to the closest edges (c) EHD features for the 15 observations



(a)



(b)



(c)

Figure 40. EHD feature of a false alarm signature. (a) False alarm in the (depth, down-track) plane. (b) Pixels classified to the closest edges (c) EHD features for the 15 observations

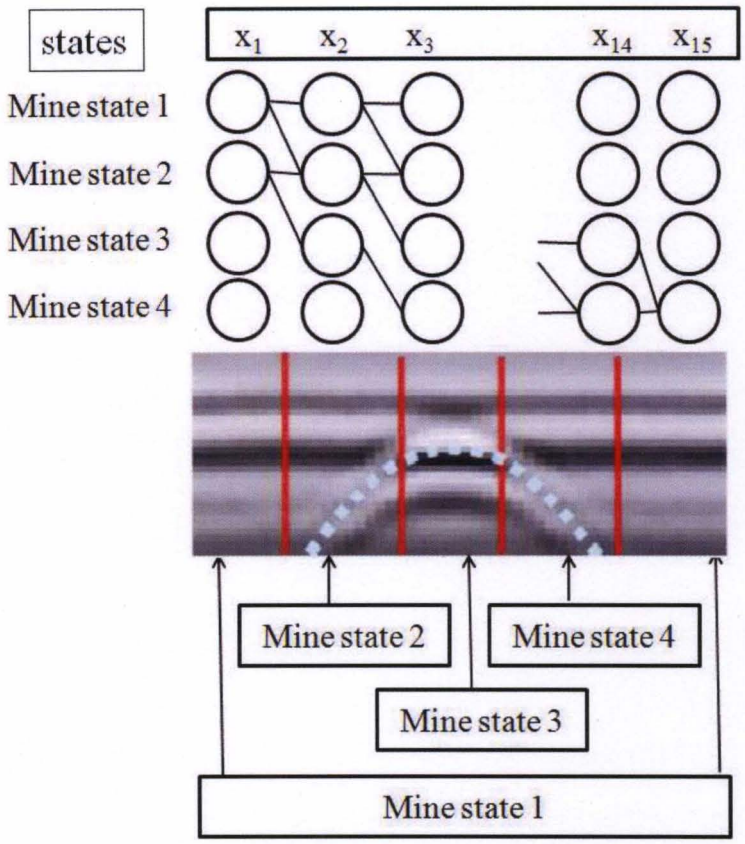


Figure 41. Illustration of the HMM mine model with four states.

The architecture of the HMM mine detector is illustrated in Fig.42.

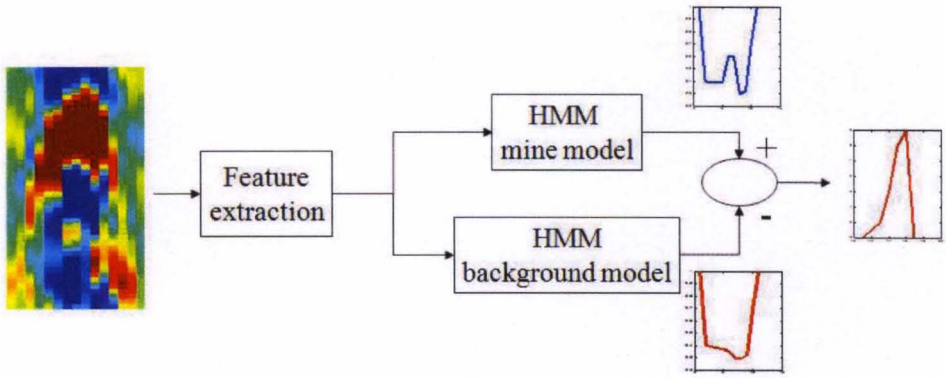


Figure 42. Illustration of the baseline HMM mine detector

For the baseline HMM, we treat all feature sets (Gradient, Gabor, and EHD) equally important. For the discrete case, to generate the codebook, we cluster the training data into M clusters using the FCM algorithm [51]. To generate the state components for the continuous HMM, we

cluster the training data relative to each state into M clusters also using the FCM algorithm [51]. For both DHMM and CHMM, the parameters are then estimated using the Baum-Welch algorithm [6], the MCE/GPD algorithm [32], or a combination of the two.

VI.1.4.2 Multi-stream HMM

In the baseline HMM, the different features are assumed to be equally important in characterizing alarm signatures. However, this assumption may not be valid for most cases. For instance, some alarms may be better characterized with the gradient features, while others may be better characterized with Gabor or EHD features. Also, even within the same features set, components may not be equally important. For instance, within the Gabor features, some alarms may be better characterized at lower scales, while others may be better characterized at higher scales. The different feature sets could then be treated as different sources of information, i.e., different streams. Since it is not possible to know a priori which feature is more discriminative, we propose considering the different features as different streams of information and use the training data to learn Multi-Stream HMMs (discrete and continuous).

The MSHMM based landmine detector's architecture is illustrated in Fig.43. We use L

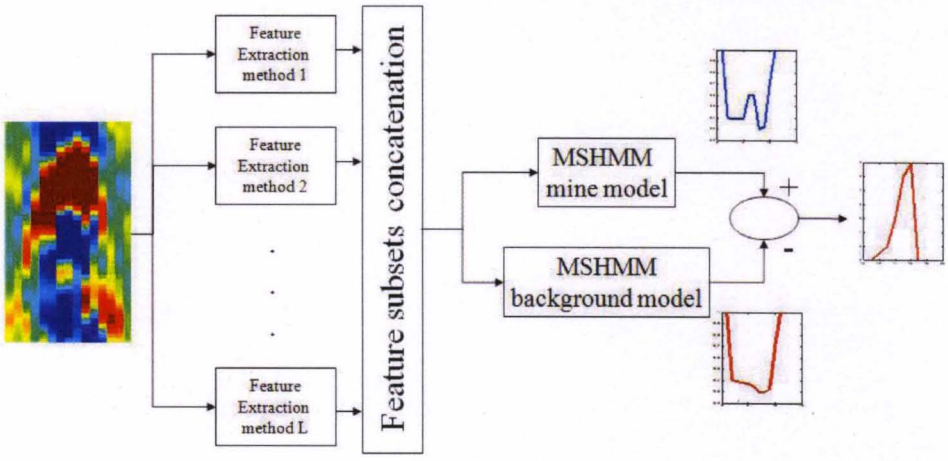


Figure 43. Illustration of the multi-stream HMM mine detector

streams where each stream (Gradient, EHD, Gabor, or Gabor response at a fixed scale) produces a p_k-dimensional feature vectors. For the discrete case (MSDHMM), to generate the codebook, we cluster the training data in M clusters using SCAD [36] and learn initial stream relevance weights for each symbol. The state transition probabilities $\mathbf{A}$ and the observation probabilities $\mathbf{B}$ are learned using the generalized Baum-Welch (see section IV.2), the generalized MCE/GPD (section IV.2.2.3

), or a combination of the two.

To generate the state components for the continuous case (MSCHMM), we cluster the training data relative to each state in M clusters using SCAD [36] and learn initial stream relevance weights for each state and component. The state transition probabilities $\mathbf{A}$, the mixing coefficients $\mathbf{U}$, and the component parameters and the observation probabilities $\mathbf{B}$ are learned using the generalized Baum-Welch (see section IV.2), the generalized MCE/GPD (section IV.2.2.3), or a combination of the two.

VI.1.4.2.1 Confidence value assignment The confidence value assigned to each observation sequence, Conf(O), depends on: (1) the probability assigned by the mine model, $Pr(O|\lambda^m)$; (2) the probability assigned by the clutter model, $Pr(O|\lambda^c)$; and (3) the optimal state sequence. In particular, we use:

$$\text{Conf}(O) = \begin{cases} \max(\log \frac{Pr(O|\lambda^m)}{Pr(O|\lambda^c)}, 0) & \text{if } \#\{s_t = 1, t = 1, \cdots, T\} \leq T_{max} \\ 0 & \text{otherwise} \end{cases} \tag{VI.1.11}$$

Since each alarm has over 500 depth values and only 45 depths are processed at a time, we divide the test alarm into 10 overlapping sub-alarms and test each one independently to obtain 10 partial confidence values. These values could be combined using various fusion methods such as averaging, artificial neural networks [76], or an order-weighted average (OWA) [77]. In our work, we use the average of the top 3 confidences. This simple approach has been successfully used in [78].

VI.1.5 Evaluation Measure: Receiver Operating Characteristic Curve

The Receiver Operating Characteristic Curve (ROC) curve is a graphical plot of the sensitivity vs. specificity for a binary classifier system as its discrimination threshold is varied. The ROC can also be represented equivalently by plotting the fraction of true positives (TPR = true positive rate) vs. the fraction of false positives (FPR = false positive rate). Consider a two-class prediction problem (binary classification), in which the outcomes are labeled either as positive (p) or negative (n) class. There are four possible outcomes from a binary classifier. If the outcome from a prediction is p and the actual value is also p, then it is called a true positive (TP); however if the actual value is n then it is said a false positive (FP). Conversely, a true negative occurs when both the prediction outcome and the actual value are n, and false negative is when the prediction outcome is n while the actual value is p. Let us define an experiment from P positive instances and N negative instances. The four outcomes can be formulated in a 2×2 contingency table or a

confusion matrix, (refer to Table.8). To draw an ROC curve, only the true positive rate (TPR) and false positive rate (FPR) are needed. TPR determines a classifier or a diagnostic test performance on classifying positive instances correctly among all positive samples available during the test. FPR, on the other hand, defines how many incorrect positive results while they are actually negative among all negative samples available during the test.

An ROC space is defined by FPR and TPR as x and y axes respectively, which depicts relative trade-offs between true positive (benefits) and false positive (costs). Since TPR is equivalent with sensitivity and FPR is equal to 1-specificity, the ROC graph is sometimes called the sensitivity vs (1-specificity) plot. Each prediction result or one instance of a confusion matrix represents one point in the ROC space.

TABLE 8

Contingency Table

	p	n	total
p'	True Positive	False Positive	P'
n'	False Negative	True Negative	N'
total	P	N	P+N

VI.1.6 Experimental results

VI.1.6.1 MSHMM with four Gabor scales

In this experiment we use only Gabor features to illustrate the need for treating features at multiple scales differently. In particular, each Gabor scale is considered as a separate stream. Thus, we use our MSHMM with $L = 4$ streams.

The data collection used in this experiment includes 600 mine and 600 clutter signatures. We use a 5-fold cross validation scheme to evaluate the proposed MSHMM structures and compare them to the baseline HMM as well as the existing MSHMM structures. For each cross-validation, we use a different subset of the data that has 80% of the alarms for training and test on the remaining 20% of the alarms. As mentioned earlier, the evaluation is performed in terms of the receiver operating characteristics (ROC) curve. For the probability based DHMM with geometric aggregation, we set the values of ν and κ in (IV.2.44) to 1.25 and 1 respectively.

For the MCE/GPD training, the parameter of the sigmoid loss function was empirically chosen as $\zeta = 1$, $\theta = 0$. In general, in MCE training the step size parameter ϵ needs to be carefully chosen to balance learning rate and convergence behavior. A large ϵ leads to fast learning but may

cause divergence, while a small ϵ leads to slow learning but is safe in convergence. Our experiments revealed that the best step size ϵ was often data dependent, and it also depended on how well the baseline models fit the data. In this dissertation we report the results when, the step-size is set to 10^{-3} for the first iteration, and is increased by a step of 10^{-3}. It has been noticed that the number of iterations required for convergence is around 50.

VI.1.6.1.1 Discrete case Fig. 44 compares the ROC curves generated using each of the four streams (Gabor features at each scale). All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. We note that the DHMM with Gabor features at scale 2 outperforms all other features (for $FAR \leq 40$). In this figure, the individual scales (with the baseline DHMM) are also compared to the case where all scales are concatenated (with the baseline DHMM). We note that the baseline DHMM with all 4 scales is not much better than the DHMM at scale 2. In fact, for some FAR, the performance can be worse. This is due mainly to the way the four scales are combined equally.

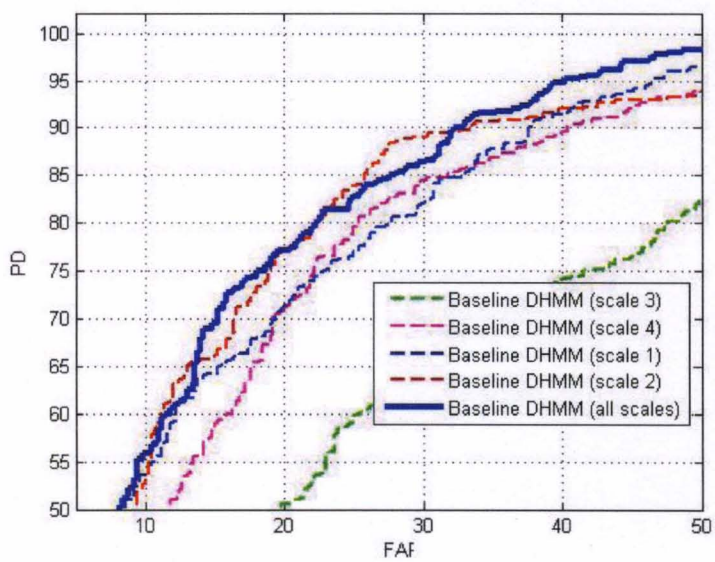


Figure 44. Comparison of the baseline DHMM with the individual Gabor scales and when all scales are concatenated.

To illustrate the complementary information provided by the different scales, in Fig. 45 we display a scatter plot of the confidence values generated by the baseline DHMM that uses Gabor features at scale 1 and scale 2. As it can be seen, for many alarms, the confidence values generated by both DHMMs are correlated. However, there are few alarms, e.g., those highlighted in region R_3,

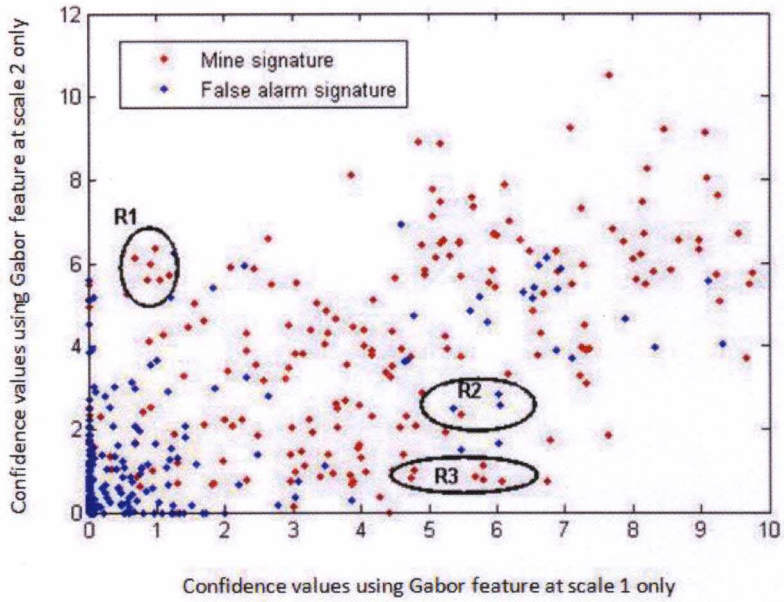
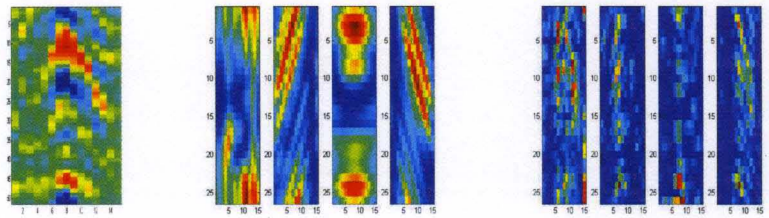


Figure 45. Scatter plot of the confidence values generated using 2 baseline DHMM that use Gabor features at scales 1 and 2.

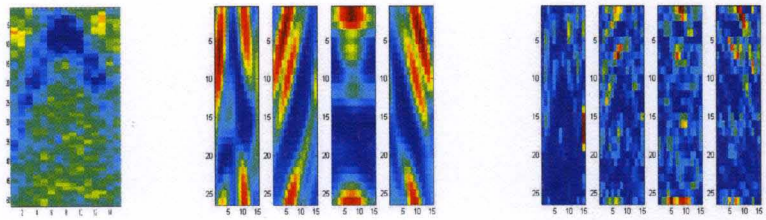


(a) GPR data (b) Response to scale 1 Gabor filters at 4 orientations (c) Response to scale 2 Gabor filters at 4 orientations

Figure 46. A sample mine signature (from region R_1 in Fig.45) where the DHMM with scale 2 outperforms the DHMM with scale 1

where the DHMM with scale 1 features is more reliable than the DHMM with scale 2. The alarm shown in Fig. 46 is one of those alarms, and as it can be seen, the alarm's response to scale 1 Gabor filters is more dominant. Similarly, there are few alarms, e.g., those highlighted in region R_1, where the DHMM with scale 2 features is more reliable than the DHMM with scale 1. The alarm shown in Fig. 47 is one of those alarms. For this alarm, its response to scale 2 is more noticeable. This difference in behavior exists for clutter alarms too as highlighted in R_2. The proposed MSDHMM is designed to identify the different types of alarms and construct a codebook where the symbols have stream dependent relevance weights in each state.

Fig. 48 compares the ROC curves generated using each of the four streams (Gabor features



(a) GPR data (b) Response to scale 1 Gabor filters at 4 orientations (c) Response to scale 2 Gabor filters at 4 orientations

Figure 47. A sample mine signature (from region R_3 in Fig.45) where the DHMM with scale 1 outperforms the DHMM with scale 2

at each scale) and their combination using simple concatenation (Baseline DHMM) and using the different variations of the multi-stream DHMM. As it can be seen, all MSDHMM structures outperform the baseline DHMM. Moreover, the MSDHMM with linear aggregation outperforms the other structures. These results are consistent with those obtained with the synthetic data.

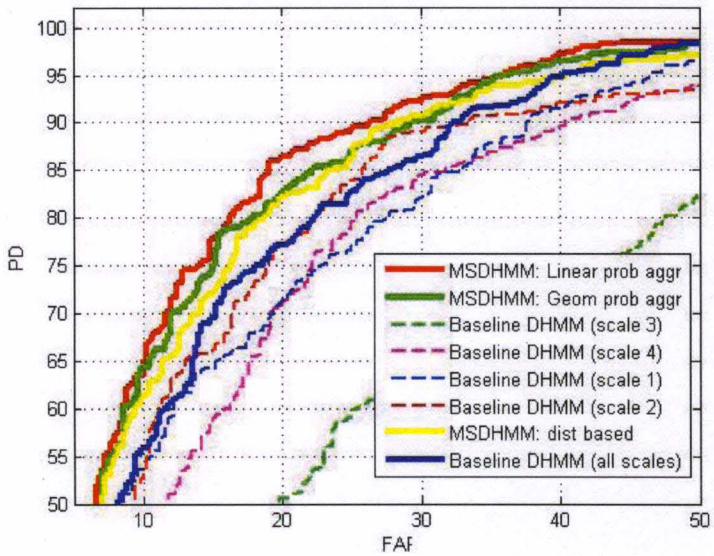


Figure 48. Comparison of the different variations of the proposed multi-stream DHMM to the baseline DHMM.

Fig. 49 displays the number of misclassified samples versus the number of iterations for the baseline DHMM and the proposed MSDHMM using MCE/GPD training. As it can be seen, learning stream relevance weights causes the error to drop faster. In fact, at each iteration, the classification error for the MSDHMM structure is lower than the baseline DHMM.

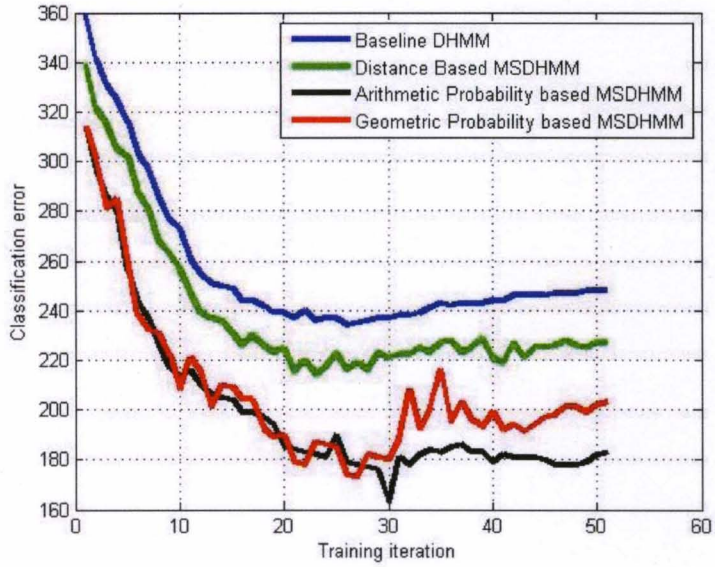


Figure 49. Number of misclassified samples versus the number of iterations for the standard and MSDHMM.

VI.1.6.1.2 Continuous case Fig. 50 compares the ROC curves generated using each of the four streams (Gabor features at each scale). All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. We note that the CHMM with Gabor features at scale 2 and 4 are very comparable and outperform all other features (for $FAR \leq 40$). In this figure, the individual scales (with the baseline CHMM) are compared to the case where all scales are concatenated (with the baseline CHMM). As it can be seen, the baseline CHMM with all 4 scales is not much better than the CHMM at scale 2 and 4 especially for $FAR \leq 30$. In fact, for some FAR, the performance can be worse. This is due mainly to the way the four scales are combined equally.

As in the discrete case, in Fig. 51 we display a scatter plot of the confidence values generated by baseline CHMM that use Gabor features at scale 1 and scale 2. As it can be seen, for most alarms, the confidence values generated by both CHMMs are correlated. However, there are few alarms, e.g., those highlighted in region R_3, where the CHMM with scale 1 features is more reliable than the CHMM with scale 2. The alarm shown in Fig. 37 (a) is one of those alarms, and as it can be seen, the alarm's response using scale 1 Gabor filters is more reliable. Similarly, there are few alarms, e.g., those highlighted in region R_1, where the DHMM with scale 2 features is more reliable than the DHMM with scale 1. The alarm shown in Fig. 37 (b)is one of those alarms. For this alarm,

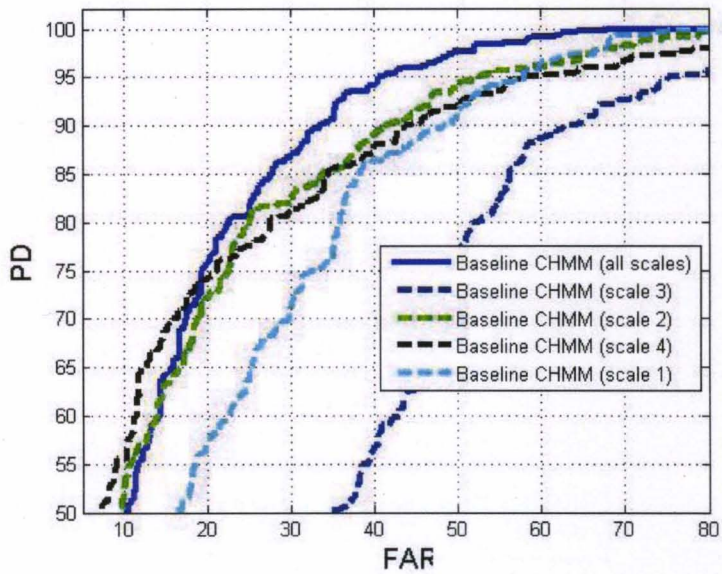


Figure 50. Comparison of the baseline CHMM with the individual Gabor and all scales concatenated.

its response to scale 2 is more reliable. This difference in behavior exists for clutter alarms too as highlighted in R_2.

Fig. 52 compares the ROC curves generated using each of the four streams (Gabor features at each scale) and their combination using simple concatenation (Baseline CHMM), the proposed MSCHMM variations, and the existing MSCHMM. All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. As it can be seen, all MSCHMM structures outperform the baseline CHMM. Moreover, the MSCHMM with mixture level streaming outperforms the other structures. The proposed MSCHMM structures also outperform the MSCHMM^{G} [25, 24] approach (outlined in section III.3.2.1). This is due to the fact that the stream relevance weights are learned separately from the rest of the model parameters. These results are consistent with those obtained with the synthetic data in sections IV.5 and V.4.3.

VI.1.6.2 MSHMM with Gradient, Gabor and EHD feature sets

VI.1.6.2.1 Discrete case In this experiment, we apply the proposed MSHMM structures to a bigger collection of data that contains 5215 alarms. The number of mine alarms is 1554, and the number of 3878 clutter alarms. We use the same settings as in the previous experiment. However, we consider the three feature collections (Gradient, Gabor, and EHD) as three separate streams.

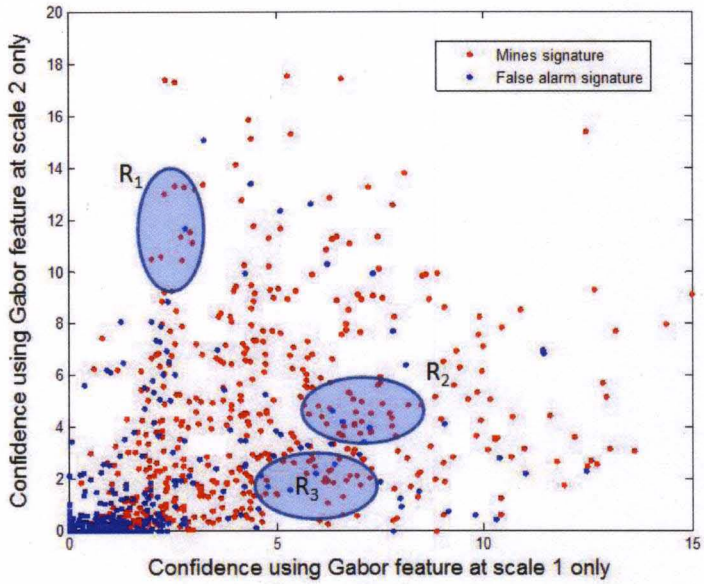


Figure 51. Scatter plot of the confidence values generated using 2 baseline CHMM that use Gabor features at scales 1 and 2.

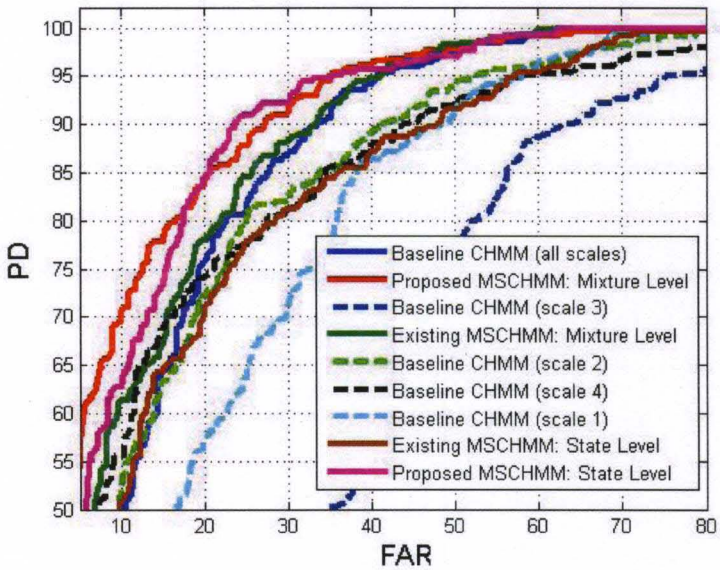


Figure 52. Comparison of the different variations of the proposed multi-stream CHMM to the baseline CHMM and the existing MSCHMM^{G}.

Fig. 53 compares the ROC curves generated using each of the three streams (Gradient, Gabor, and EHD). All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. We note that the DHMM with EHD features outperforms all other features (for $FAR \geq 25$). The baseline DHMM with gradient features has the lowest performance. The individual features (with the baseline CHMM) are also compared to the case where all features are concatenated (with the baseline DHMM). We note that the baseline DHMM with all 3 features is not better than the DHMM with EHD and Gabor. In fact, for some regions of the ROC, the performance can be even worse. This is due mainly to the way the three features are treated equally important for all alarms combined equally.

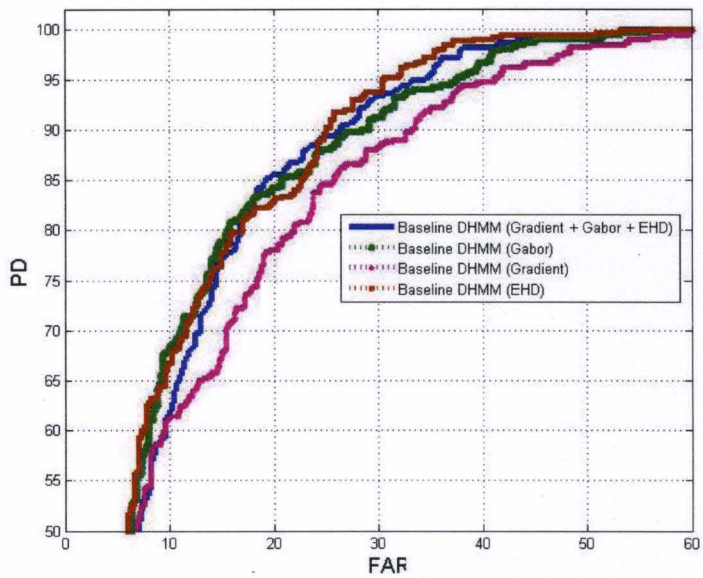


Figure 53. Comparison of the baseline DHMM with the individual features (Gradient, Gabor, and EHD) and all features concatenated.

In Fig. 54, we display a scatter plot of the confidence values generated by the baseline DHMM that uses the EHD and Gabor features. As it can be seen, for most alarms, the confidence values generated by both DHMMs are correlated. However, there are few alarms, e.g., those highlighted in region R_3, where the DHMM with EHD features is more reliable than the DHMM with Gabor features.

Fig. 55 compares the ROC curves generated using each of the three streams (Gradient, Gabor, and EHD features) and their combination using simple concatenation (Baseline DHMM) and using multi-stream DHMM. All results were obtained when the model parameters are learned

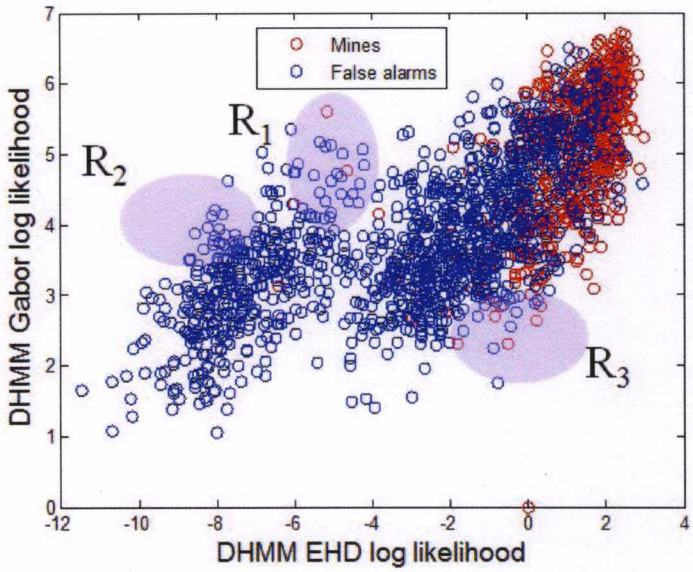


Figure 54. Scatter plot of the confidence values generated using 2 baseline DHMM that use EHD and Gabor features.

using Baum-Welch followed by the MCE/GPD training method. We note that MSDHMM structures outperform the baseline DHMM. Moreover, the MSDHMM with linear aggregation outperforms the other structures. These results are consistent with those obtained with the third experiment.

VI.1.6.2.2 Continuous case Fig. 56 compares the ROC curves generated using each of the three streams (Gradient, Gabor, and EHD) with the baseline CHMM. All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. We note that the three modalities have comparable performance. The individual features (with the baseline CHMM) are also compared to the case where all scales are concatenated (with the baseline CHMM). We note that the baseline CHMM with all 3 features is not better than the CHMM with individual features. As in the discrete case, this is due mainly to the way the different sets of features are treated equally important for all alarms.

Fig. 57 compares the ROC curves generated when all the streams are combined using simple concatenation (Baseline CHMM), the proposed MSCHMM variations, and the existing MSCHMM^G algorithms [25, 24]. All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method. We note that all MSCHMM structures outperform the baseline CHMM. Moreover, the MSCHMM with mixture level streaming outperforms the other structures. Also, the proposed MSCHMM structures outperform the MSCHMM^G. This

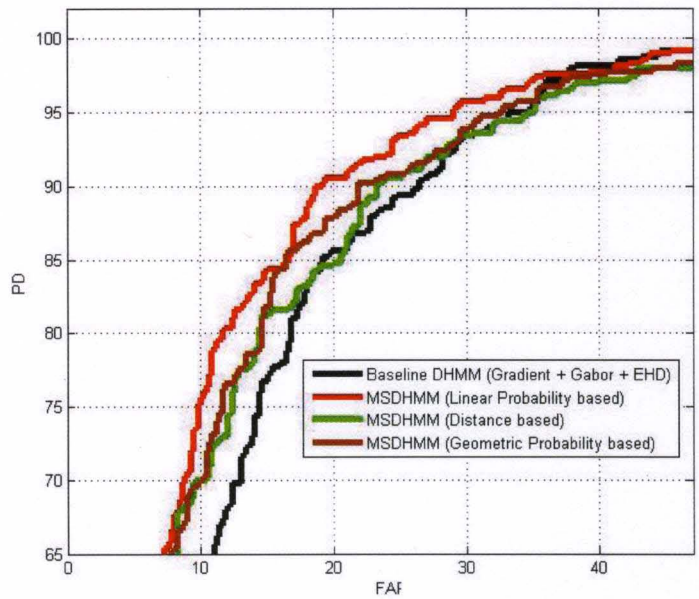


Figure 55. Comparison of the different variations of the proposed multi-stream DHMM to the baseline DHMM.

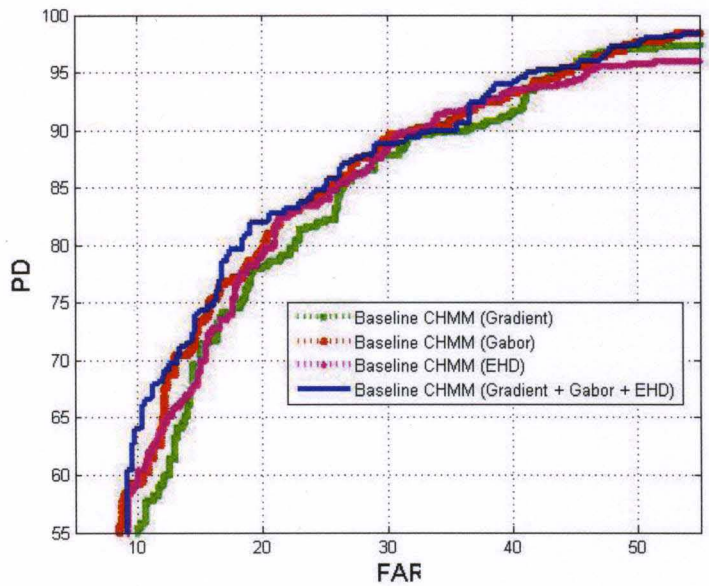


Figure 56. Comparison of the baseline CHMM with the individual features (Gradient, Gabor, and EHD) and all features concatenated.

is due to the fact that the stream relevance weights in MSCHMM^{G} are learned separately from the rest of the model parameters.

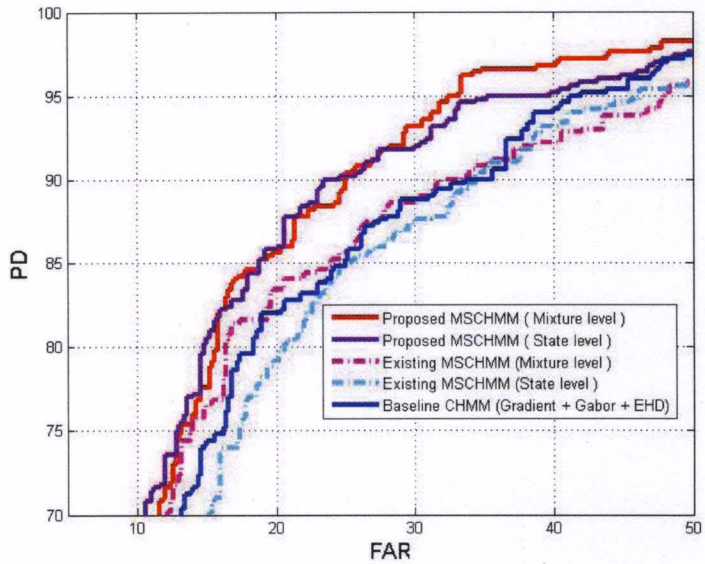


Figure 57. Comparison of the different variations of the proposed multi-stream CHMM to the baseline CHMM..

VI.2 Australian sign language classification

VI.2.1 Data collection

This dataset (from the University of California-Irvine [1]) consists of multiple Australian sign-language gestures, each represented by 27 instances of 22-dimensional time-series sequences.

VI.2.2 Feature extraction

The 22-dimensional vectors encode information gathered from the movement of both hands while signing [79]. Figure 58 shows the glove based system used for gathering this information. In particular, each hand is represented by the following 11 attributes:

- x, y, and z: encode the position of the hand relative to a zero point set slightly below the chin. These attributes are real numbered expressed in meters.

[1] http://www.cse.unsw.edu.au/~waleed/tml/data

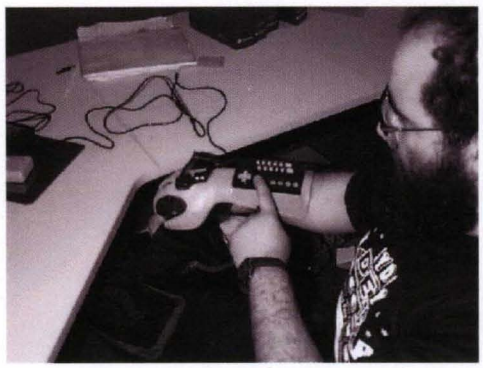


Figure 58. The user starting to sign.

- roll: 0 being palm down. Positive means the palm is rolled clockwise from the perspective of the signer. To get degrees, multiply by 180. The range of this attribute is $[-0.5\ 0.5]$.
- pitch: 0 being palm flat (horizontal). Positive means the palm is pointing up. To get degrees, multiply by 180. The range of this attribute is $[-0.5\ 0.5]$.
- yaw: 0 being palm straight ahead from the perspective of the signer. Positive means clockwise from the perspective above the signer. To get degrees, multiply by 180. The range of this attribute is $[-1\ 1]$.
- thumb, forefinger, middle finger, ring finger, little finger: real attributes in the range of $[0\ 1]$. They encode the position correspondent to each finger. A value of zero means totally flat, and a value of one means totally bent.

VI.2.2.1 Results

Among the 95 classes (words), we consider binary classification of semantically-related expressions such as *write* and *draw* or antonyms such as *give* and *take*. These expressions were assumed to have similar real-world symbols and formed the basis of the experiment with this dataset. To fit this data set into the multi-stream context, we assume that the attributes correspondent to each hand represent a separate interpretation of the original "signal" and thus a separate stream. Both discrete and continuous HMM models in this experiment have $N_s = 5$ states. In the discrete case, the training data is summarized into $M = 100$ symbols. In the continuous case, each state is represented by a mixture of $M = 4$ Gaussian components.

Table 9 shows a comparison of the accuracy of the baseline DHMM, MSDHMM^{D},

$MSDHMM^{P_l}$, and $MSDHMM^{P_g}$ over the classification of 10 such pairs of sequences. It could be seen from table 9 that the proposed MSDHMM are outperforming both the baseline DHMM.

Table 10 shows a comparison of the accuracy of the baseline CHMM, $MSCHMM^{G_m}$, $MSCHMM^{G_s}$, $MSCHMM^{L_m}$, and $MSCHMM^{L_s}$ over the classification of 10 such pairs of sequences. It could be seen from table 10 that the proposed $MSCHMM^{L}$ are outperforming both the baseline CHMM and the state of the art $MSCHMM^{G}$.

All results were obtained when the model parameters are learned using Baum-Welch followed by the MCE/GPD training method.

TABLE 9

Comparison of the performance of the different DHMM structures over the AUSLAN data

Simple pairs	Baseline DHMM	$MSDHMM^{D}$	$MSDHMM^{P_l}$	$MSDHMM^{P_g}$
'hot' vs 'cold'	53.025 %	56.55%	58.075 %	58.25%
'eat' vs 'drink'	60.075 %	60.00%	68.00%	70.00%
'happy' vs 'sad'	58.25%	67.35%	70.25 %	72.65%
'yes' vs 'no'	55.00 %	59.50 %	67.00 %	73.25 %
'give' vs 'take'	70.00 %	75.00 %	79.45 %	80.00%
'paper' vs 'pen'	75.25 %	75.00 %	78.00 %	78.00%
'science' vs 'research'	79.00 %	81.00 %	82.50 %	84.25%
'soon' vs 'hurry'	62.45 %	64.00 %	66.00 %	70.00%
'spend' vs 'cost'	66.00 %	68.00 %	75.00 %	72.00 %
'write' vs 'read'	95.25 %	96.00 %	98.00 %	99.00%

To illustrate the advantages of combining the different features coming from the two hands into a MSHMM structure and learning stream dependent relevance weights, in Fig. 59 and 60 we display a scatter plot of the confidence values generated by the baseline DHMM and CHMM that use the feature relative to left and right hand. As it can be seen, for most alarms, the confidence values generated by both CHMMs are correlated. However, there are few points, where the DHMM with the left hand features is more reliable than the DHMM with the right hand. The same observation could be noticed with the CHMM.

In Fig. 61, we display a point from the class of "YES" words that has been missed by both standard discrete and continuous HMMs, and was correctly classified by all the multi-stream structures. Fig. 62 shows stream 1 (right hand) relevance weight of the closest symbols to the sequence in Fig. 61, learned by the model $MSDHMM^{P_l}$. It could be inferred that both streams (hands) do not have similar relevance in a considerable number of symbols. Also Fig. 63 display

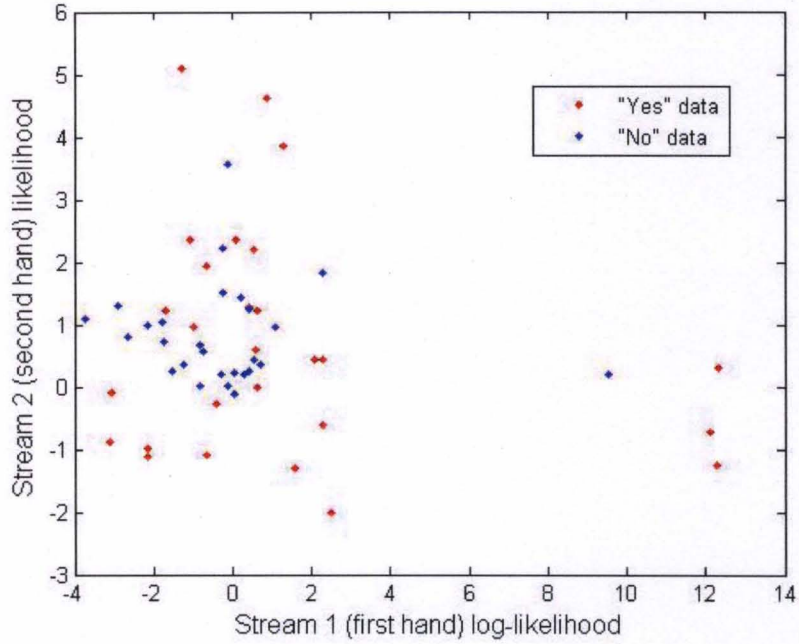


Figure 59. Scatter plot of the confidences of the two-word data ("YES" vs "NO") in the baseline DHMM using both streams.

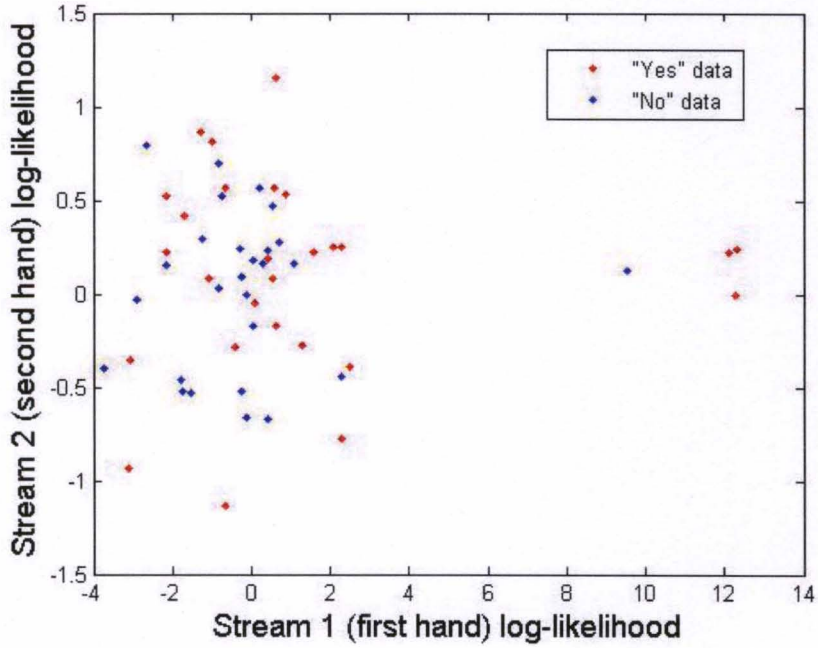


Figure 60. Scatter plot of the confidences of the two-word data ("YES" vs "NO") in the baseline CHMM using both streams.

the stream 1 (right hand) relevance weight in all the components of each state learned by the MSCHMM^{L_m}. As it can be seen, most of the components do not have similar relevance weights in all 5 states.

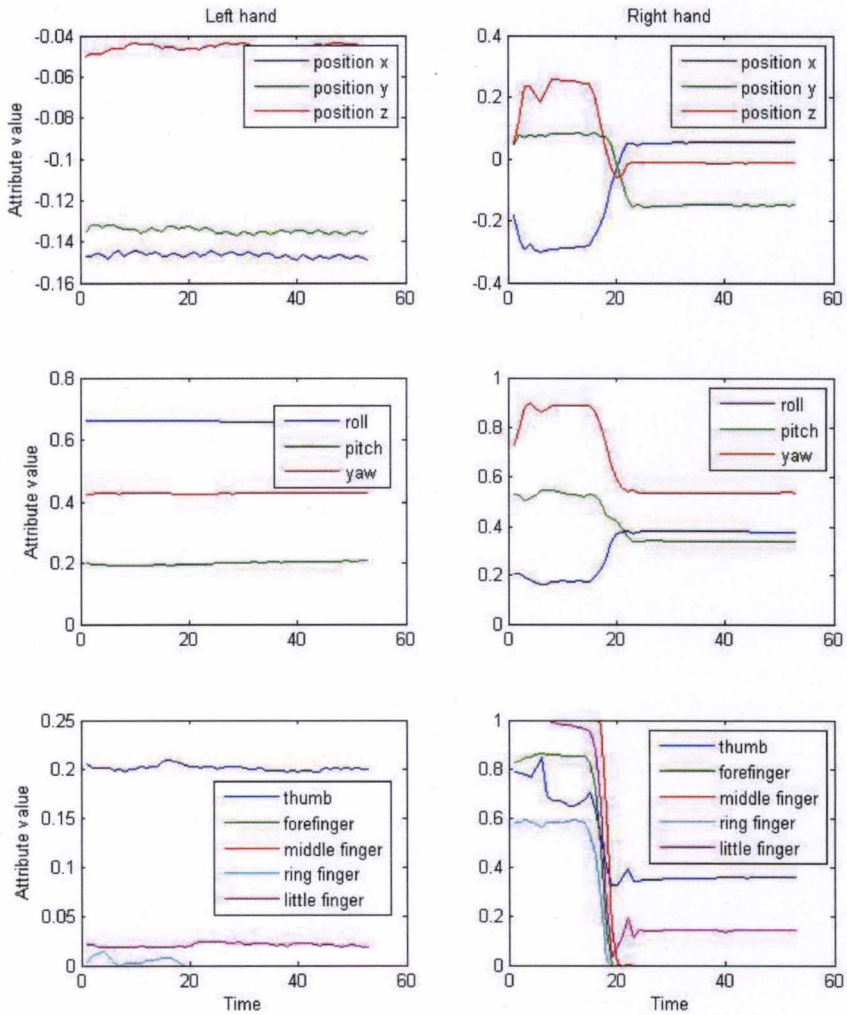


Figure 61. Features of a sample from "YES" class misclassified by standard DHMM and CHMM, and correctly classified by the MSHMM structures.

VI.3 Audio classification

In this experiment we apply the proposed MSCHMM structures to the problem of music classification. We exploit several feature extraction mechanisms that we assume are different inter-

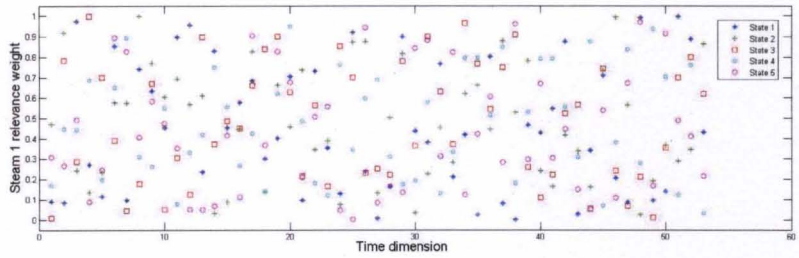


Figure 62. The stream 1 relevance weight of the closest symbols to the point in Fig. 61, using the MSDHMM^{P_1}

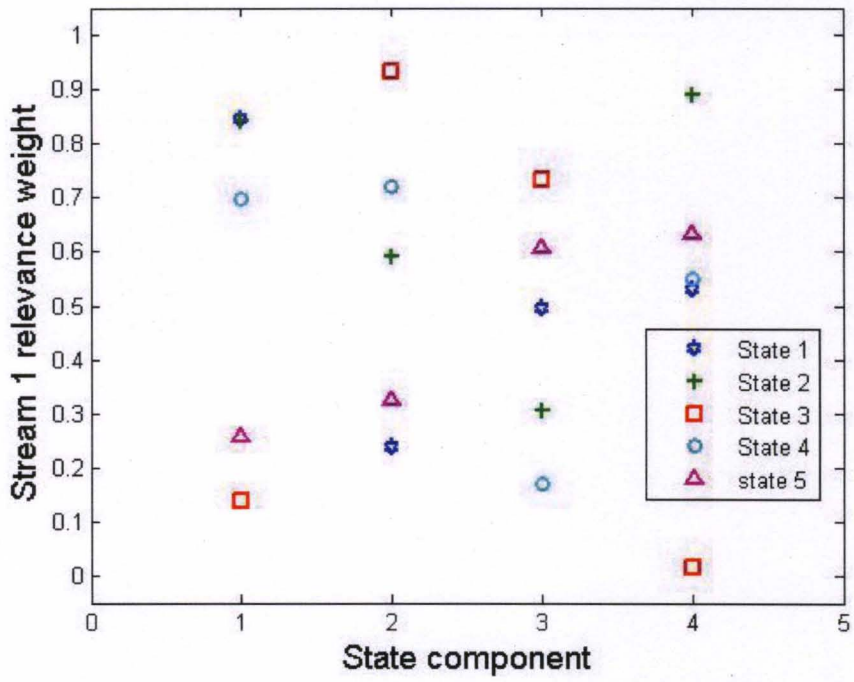


Figure 63. The stream 1 relevance weight within the components of each state of the model MSCHMM^{L_m}

pretations (streams) of the underlying characteristics of the image.

VI.3.1 Data collection

We use the benchmark data set for audio classification and clustering proposed in [80]. This data consists of 10 seconds samples of 1886 songs from Garageband website. Each song is encoded using mp3 with a sampling rate of 44.1kHz and a bitrate of 128kbit/s. The songs belong to 9 different genres as shown in table 11.

VI.3.2 Feature extraction

Two different set of features are considered in this experiment. The first one is the Mel-frequency cepstral coefficient (MFCC) [81] and the second one is the Linear predictive coding (LPC) [82]

VI.3.2.1 Mel-frequency cepstral coefficient (MFCC)

The MFCC is a low-level audio feature representation that gained popularity within the research community [83, 84]. The human ear resolves frequencies non-linearly across the audio spectrum, and the log-scale bands introduced in the AudioSpectrumEnvelope Descriptor are used to address this problem. However, experiments have showed that simple rectangular form filters, placed on log-scale in the AudioSpectrumEnvelope Descriptor, do not match the human perception accurately. In [84], a Mel frequency scale that takes into account how humans perceive the difference between sounds of different frequencies was introduced. The input signal is divided into overlapping frames typically 20ms to 40ms with 50% overlap. To minimize signal discontinuities at the borders, a Humming windowing function is used. A Fast Fourier Transform (FFT) is then applied to each frame and the absolute value is taken to obtain the magnitude spectrum. The spectrum is then processed with a Mel-filter bank, a set of triangular shape filters, whose center frequencies are spaced according to the mel scale [83]. The response of each filter is log transformed, thus, resulting in a reduced representation of the spectrum. The cepstral coefficients are finally obtained through a Discrete Cosine Transform of the reduced log-energy spectrum.

VI.3.2.2 Linear predictive coding (LPC)

This feature is extracted using an approach that performs spectral analysis with an all-pole modeling constraint. It is fast and provides accurate estimates of speech parameters. The basic idea behind linear predictive analysis is that a speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of the squared differences (over a finite interval) between the actual speech samples and the linearly predicted ones, a unique set of predictor coefficients can be determined. (The predictor coefficients are the weighting coefficients used in the linear combination.) [?].

VI.3.3 Results

In this experiment we compare the performance of the proposed multi-stream HMM structures to the standard HMM as well as the existing MSHMM. Cross validation with 10 folds is performed where 10% of the data is set for testing and the remaining 90% for training. For all the HMM structures, the number of states is set to N_s=5. For the discrete case, $M = 100$ symbols are generated. For the continuous case, each state has M=5 Gaussian mixtures. For the multi-stream structure, the number of streams is set to $L = 2$, and the streams are the MFCC and LPC feature sets. Table 12 displays the performance for the various DHMM structures. As it can be seen, the proposed MSDHMM^{P} structures outperform the baseline DHMM. Table 13 displays the performance of the various CHMM structures. As it can be seen, the proposed MSCHMM^{L} structures outperform both the state of the art MSCHMM^{G} structures and the baseline CHMM.

To illustrate the advantages of combining the different features (MFCC and LPC) into a MSHMM structure and learning stream dependent relevance weights, in Fig. 64 and 65 we display a scatter plot of the log-likelihood values generated by the baseline DHMM and CHMM that use MFCC and LPC features individually for the Rock vs Non-Rock subset. As it can be seen, for most alarms, the confidence values generated by both CHMMs are correlated. However, there are few points, where the DHMM with the MFCC features is more reliable than the DHMM with the right hand. The same observation could be noticed for the CHMM.

To highlight the advantage of the proposed multi-stream strcuture, we consider a point from the class of Rock songs that has been missed by both standard discrete and continuous HMMs, and was correctly classified by all the multi-stream structures. Fig. 66 shows the stream relevance weights of the closest symbols to the sequence correspondent to this point, learned by the model MSDHMM^{D}. As it can be seen, both streams (MFCC and LPC) have different relevance weights in a considerable number of symbols. Fig. 67 displays stream 1 (MFCC) relevance weights in all states of the closest symbols to the sequence correspondent to this point, learned by the model $\text{MSDHMM}^{\text{P}_1}$. Fig. 68 displays stream 1 (MFCC) relevance weight in all the states' components learned by the $\text{MSDHMM}^{\text{P}_1}$ for the same sequence. As it can be noticed, the stream relevance weights relative to both MFCC and LPC features can vary significantly. In fact, for the first component of state 2, the LPC features are of negligible relevance compared to the relevance of the MFCC.

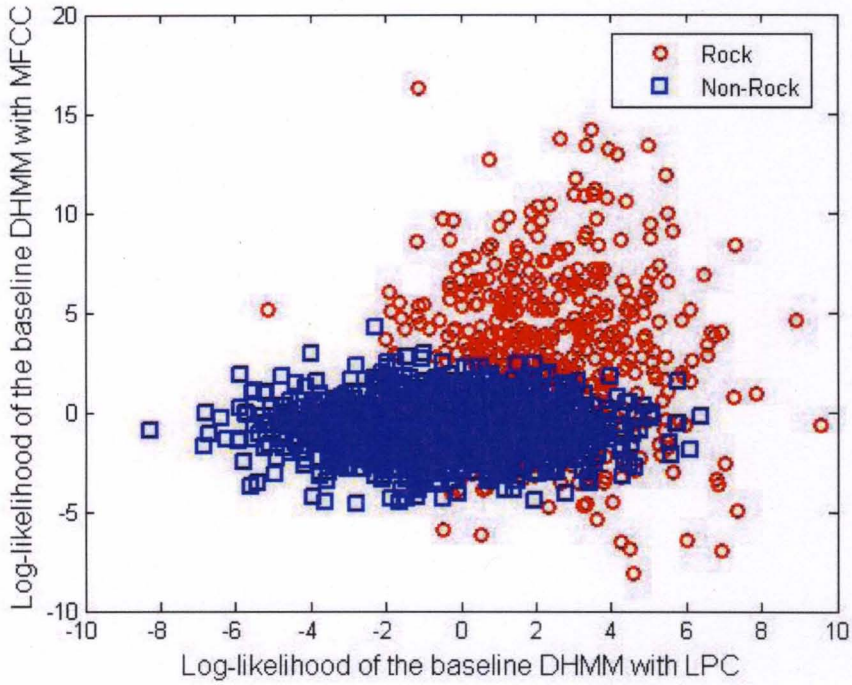


Figure 64. Scatter plot of the log-likelihood generated by the baseline CHMM using each feature set independently.

VI.4 Face vs non-face image classification

In this experiment, we apply the MSDHMM structures to the problem of binary classification of face images versus non-face images. We exploit several feature extraction mechanisms that we assume are different interpretations (streams) of the underlying characteristics of the face image.

VI.4.1 Data collection

In this application, we consider a subset of the data set available on the CBCL webpage [**?**]. The CBCL data set consists of 2901 images for face, and 28121 images for non face. All the images are of size 19×19. Figure 69 displays samples of face images and figure 70 displays samples of the non-face images.

VI.4.2 Feature extraction

For frontal face images, the significant facial regions (hair, forehead, eyes, nose, mouth) come in a natural order from top to bottom, even if the images are taken under small rotations in the

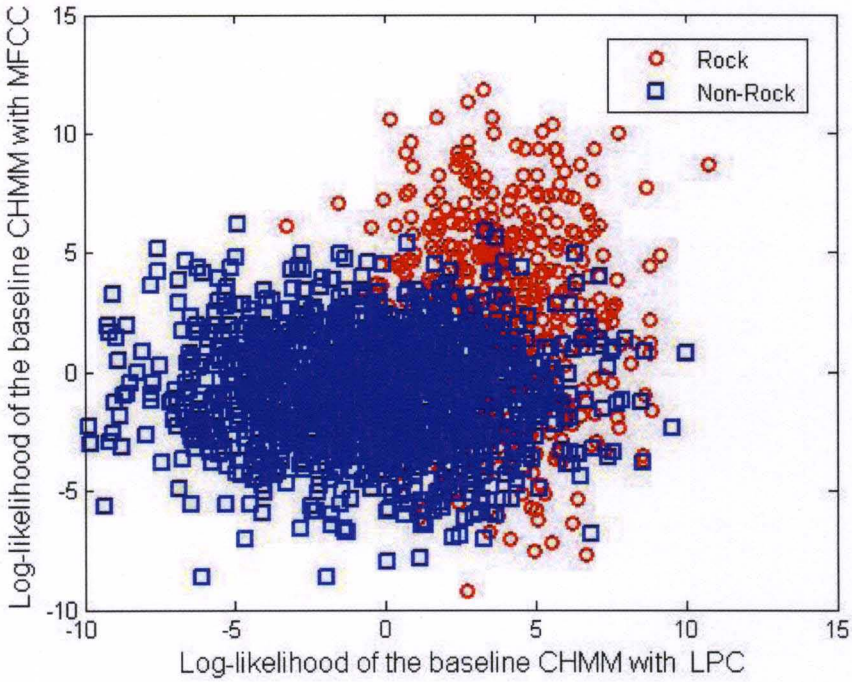


Figure 65. Scatter plot of the log-likelihood generated by the baseline CHMM using each feature set independently.

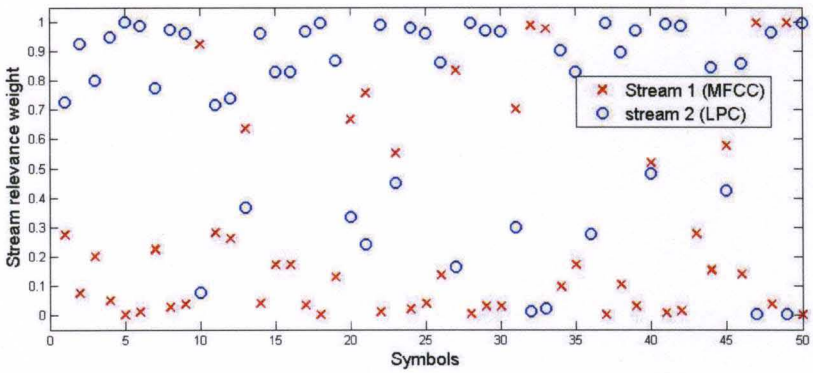


Figure 66. Stream relevance weights of the closest symbols to a sequence from the Rock class that is missed by the standard HMM but correctly classified by the MSDHMM^{D}

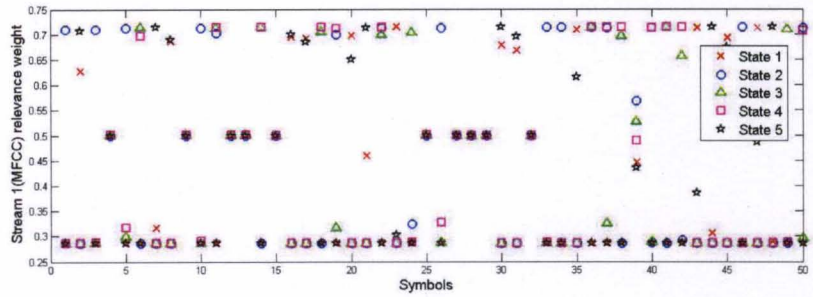


Figure 67. Stream 1 (MFCC) relevance weights of the closest symbols to a sequence from the Rock class that is misclassified by the standard HMM but correctly classified by the MSDHMM^{P_1}

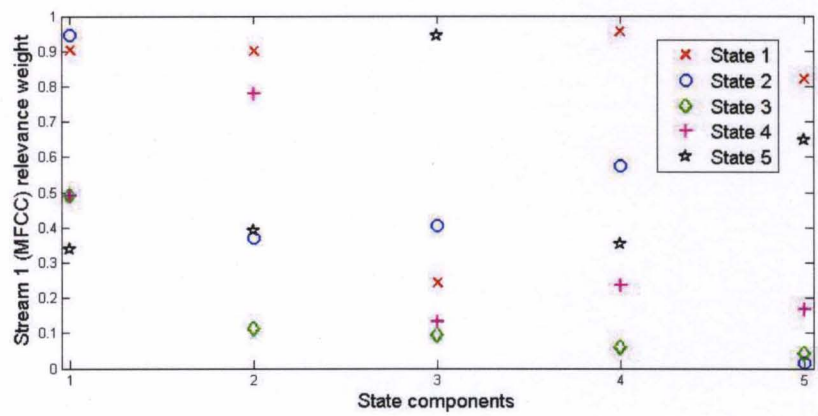


Figure 68. Stream 1 (MFCC) relevance weights of the state components learned by the MSCHMM^{L_m}

image plane and/or rotations in the plane perpendicular to the image plane. In particular, the face images available within the CBCL data set include only the eyes, nose and mouth as in figure 71. To fit this within the HMM context, each of these facial regions is assigned to a state in a left to right discrete HMM. The state transition structure of the face model are shown in figure 72 . Each face image of width W and height D is divided into overlapping blocks of height Z and width W. The amount of overlap between consecutive blocks is P (figure 71).

The number of observation vectors T, that is the number of blocks extracted from each face image is given by:

$$T = \frac{D - Z}{Z - P} + 1. \tag{VI.4.1}$$

The choice of the parameters P and Z can affect the system recognition rate. A high amount of overlap P can increase the recognition rate because it allows the features to be captured in a manner that is independent of the vertical position. The choice of the parameters Z is not trivial. A small value of Z can bring insufficient discriminant information to the observation vector, while a large

Figure 69. Sample of 100 face images

value can increase the probability of cutting across the features. However, the system recognition rate is not very sensitive to variations in Z, as long as P is sufficiently large ($P \leq Z - 1$).

The use of the pixel values as observation vectors has two important disadvantages: first, pixel values do not represent robust features, are sensitive to image noise as well as image rotation, shift or changes in illumination. Second, the large dimension of the observation vector leads to high computational complexity of the system. This can be a major problem for face recognition over large databases or when the recognition system is used for real time applications. In this work, each block is interpreted through four transformations:

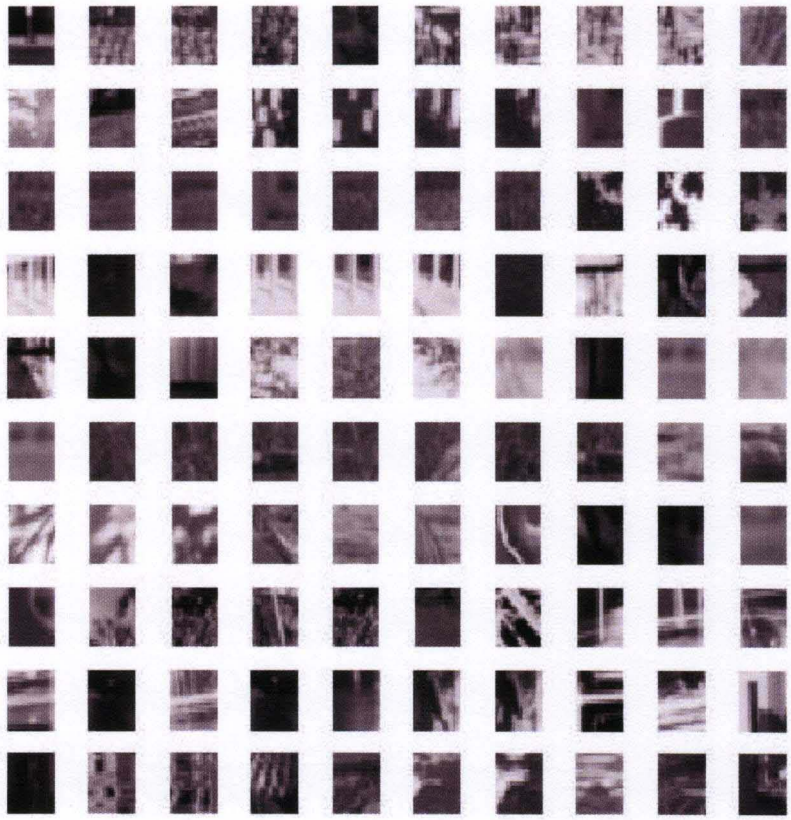


Figure 70. Sample of 100 non face images

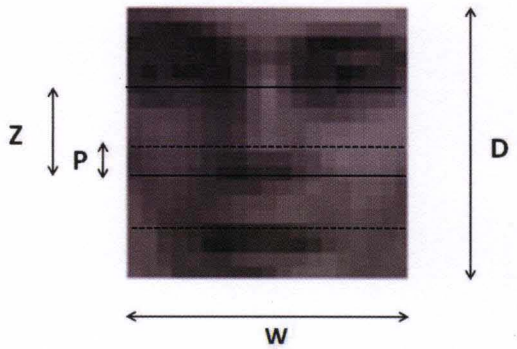


Figure 71. Face image parametrization and blocks extraction

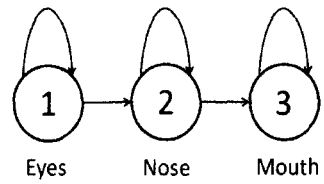


Figure 72. Left to right HMM for face recognition

VI.4.2.1 Discrete Cosine Transform (DCT)

Like other transforms, the Discrete Cosine Transform (DCT) attempts to decorrelate the image data [85]. After decorrelation each transform coefficient can be encoded independently without losing compression efficiency. The DCT possess some desirable properties, i.e., de-correlation, energy compaction, separability, symmetry and orthogonality. These attributes led to widespread deployment of the DCT in virtually every image/video processing standard of the last decade.
For an $M \times N$ image, we have an $M \times N$ DCT coefficient matrix covering all the spatial frequency components of the image. The DCT coefficients with large magnitude are mainly located in the upper-left corner of the DCT matrix. Accordingly, we scan the DCT coefficient matrix in a s zig-zag manner starting from the upper-left corner and subsequently convert it to a one-dimensional vector. In our application, we keep the largest 9 coefficients. Figure 73 displays a sample face image with 9 DCT coefficient of the 15 subimages.

VI.4.2.2 Fast Fourier Transform (FFT)

We use the FFT to extract the important frequencies (in magnitude), that encodes the important activity within each sub-image. For an $M \times N$ image, we have an $M \times N$ FFT coefficient matrix covering all the spatial frequency components of the image. In our application, we keep the largest 9 coefficients. Figure 74 displays a sample face image with 9 DCT coefficient of the 15 subimages.

VI.4.2.3 Edge Histogram Descriptor (EHD)

The EHD feature encodes important information about the signature of each block of each image in a compact form. Each block is transformed to a feature vector that encodes the response of edge detection filters [74]. The edges considered are the horizontal, vertical, diagonal (45°), antidiagonal (135°). A non-edge dimension is also considered to capture the non-well defined edges. Hence, a 5-dimensional observation vector is formed in each level as it is shown in figure 75. More

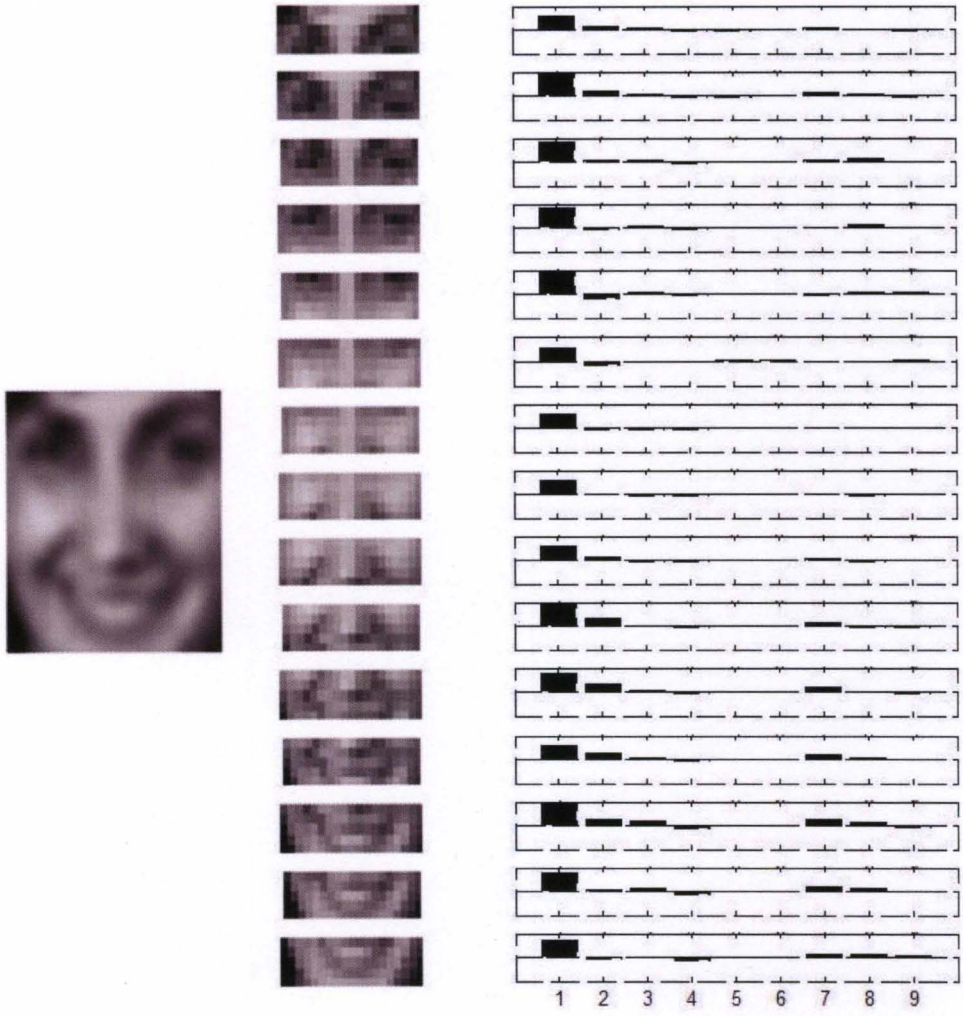


Figure 73. A face image with the correspondent DCT feature of each block.

details about this feature were given in section VI.1.3.3.

VI.4.2.4 Gabor feature extraction

The Gabor feature is based on a bank of Gabor filters or kernels. They are similar to the receptive field profiles in cortical simple cells, which are characterized as localized, orientation selective, and frequency selective. A family of Gabor kernels is the product of a Gaussian envelope and a plane wave. These kernel are available at different scales and different orientations. More details about these features are presented in appendix F. We extract Gabor features from each block of each image (face or non-face). We choose 3 scales and 6 orientations, resulting in a total of 18

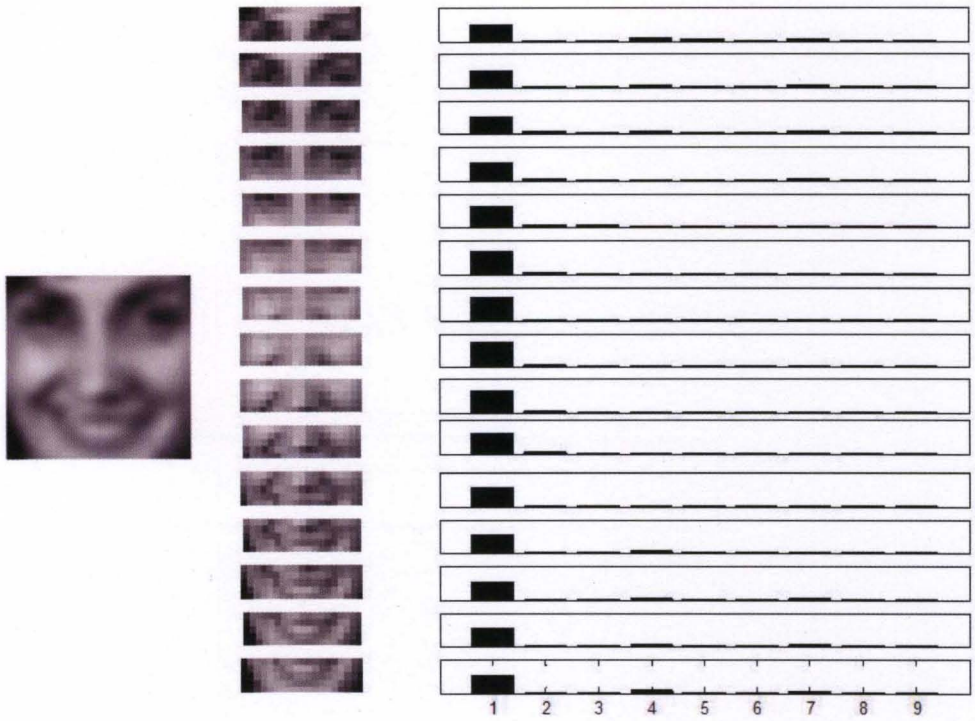


Figure 74. A face image with the correspondent FFT feature of each block.

Gabor functions in our study. We take the average of the largest 10 values of each filter response, so that we get each block represented by an 18 dimensional vector as it is shown in figure 76.

VI.4.2.5 Results

For our experiments we take at random 1000 face images and 1000 non face images from the CBCL data set. We perform 10-fold cross validation. We set $Z = 10$, and $P = 9$. Since the image size is 19×19, each image is transformed to a sequence of 15 observation vectors. We train HMMs with $N_s = 3$ states. For the discrete case, we generate $M = 80$ symbols as codebook, and for the continuous case, each state is represented by $M = 5$ components. The number of streams is set for L=4. We use the baseline HMM with the concatenation of all the streams to learn a model for face images and a model for non-face images. In addition, for each stream, a face and non-face models are learned. The proposed MSHMM structures: MSDHMM^{D}, $\text{MSDHMM}^{\text{P}_l}$, $\text{MSDHMM}^{\text{P}_g}$, $\text{MSCHMM}^{\text{L}_\text{m}}$, and $\text{MSCHMM}^{\text{L}_\text{s}}$ are also used to learn a face model and a non-face model.

Table 14 summarizes the result of the experiments performed using single and multi-stream DHMMs . As it can be seen, the multi-stream DHMM structures outperform the baseline DHMM with all the stream concatenated and with the individual streams. We also notice that the

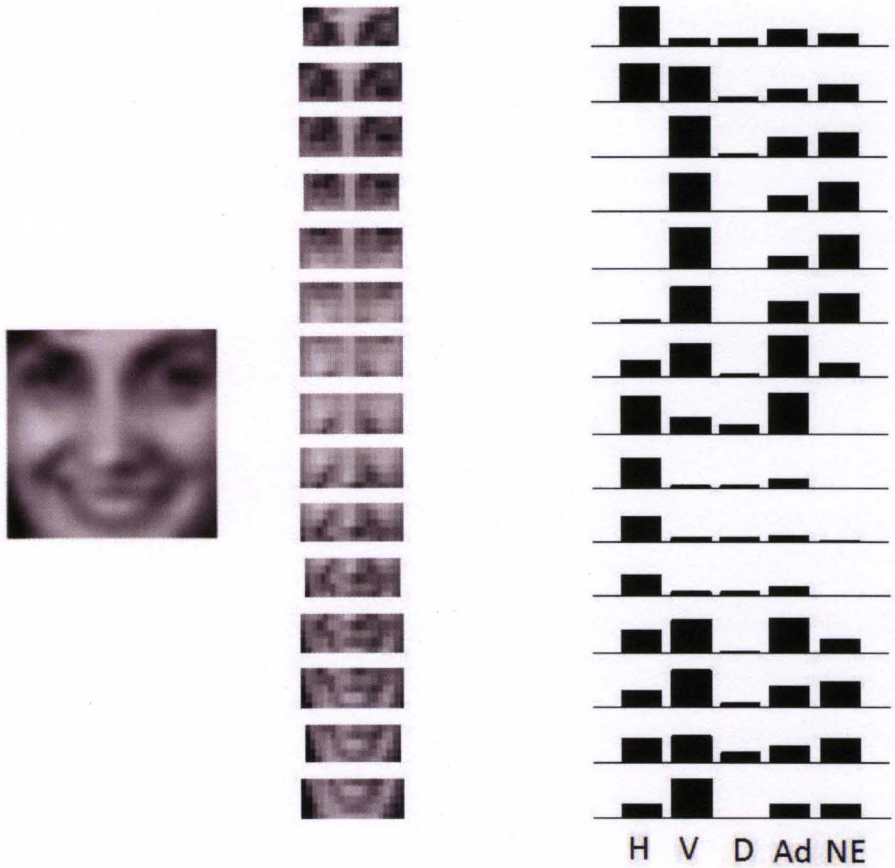


Figure 75. A face image with the correspondent EHD feature of each block.

$MSDHMM^{P_l}$ has a slight advantage over the other two structures: $MSDHMM^{D}$ and $MSDHMM^{P_g}$.

Table 15 summarizes the result of the experiments performed using single and multi-stream CHMM. As it can be seen, the $MSCHMM^{L}$ structures outperform the baseline CHMM with all the stream concatenated and with the individual streams, as well as the existing $MSCHMM^{G}$ structures.

To confirm that the increase in performance for the proposed multi-stream HMM structures is due to the stream weighting component, we consider a sequence that has been misclassified by the standard HMM and correctly classified by the proposed MSHMM structures. The face image in figure 73 is one of those samples. Figure 77 displays the stream relevance weights of the sequence closest symbols learned by the $MSDHMM^{P_1}$. As it can be noticed, only very few symbols have comparable stream relevance weights.

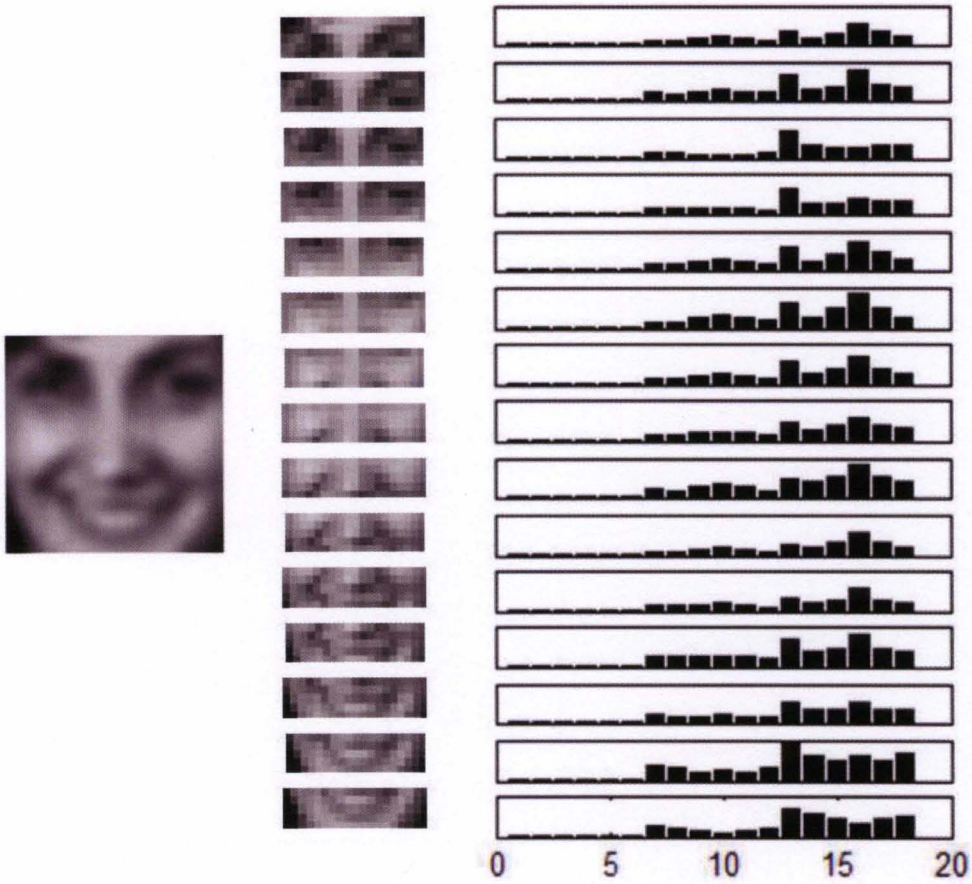


Figure 76. A face image with the correspondent Gabor feature of each block.

Fig. 78 display the learned stream relevance weights for the components of the states of $\text{MSDHMM}^{\text{Lm}}$. As it can be seen, none of the components have equal stream relevance weights in all 3 states.

VI.5 Chapter summary

In this chapter, the proposed multi-stream Hidden Markov models structures have been applied to the problems of Landmine detection, Australian sign language classification, audio classification, and face classification. For the landmine application, several experiments performed on various data collections have shown that the proposed MSHMM structures outperform the standard HMM as well as the multi-stream HMM available in the literature. The same observation is noticed with the other applications. In particular, the MSDHMM structures proposed outperform the baseline DHMM in a setting of multi-modal temporal data. This is mainly due to the stream weight-

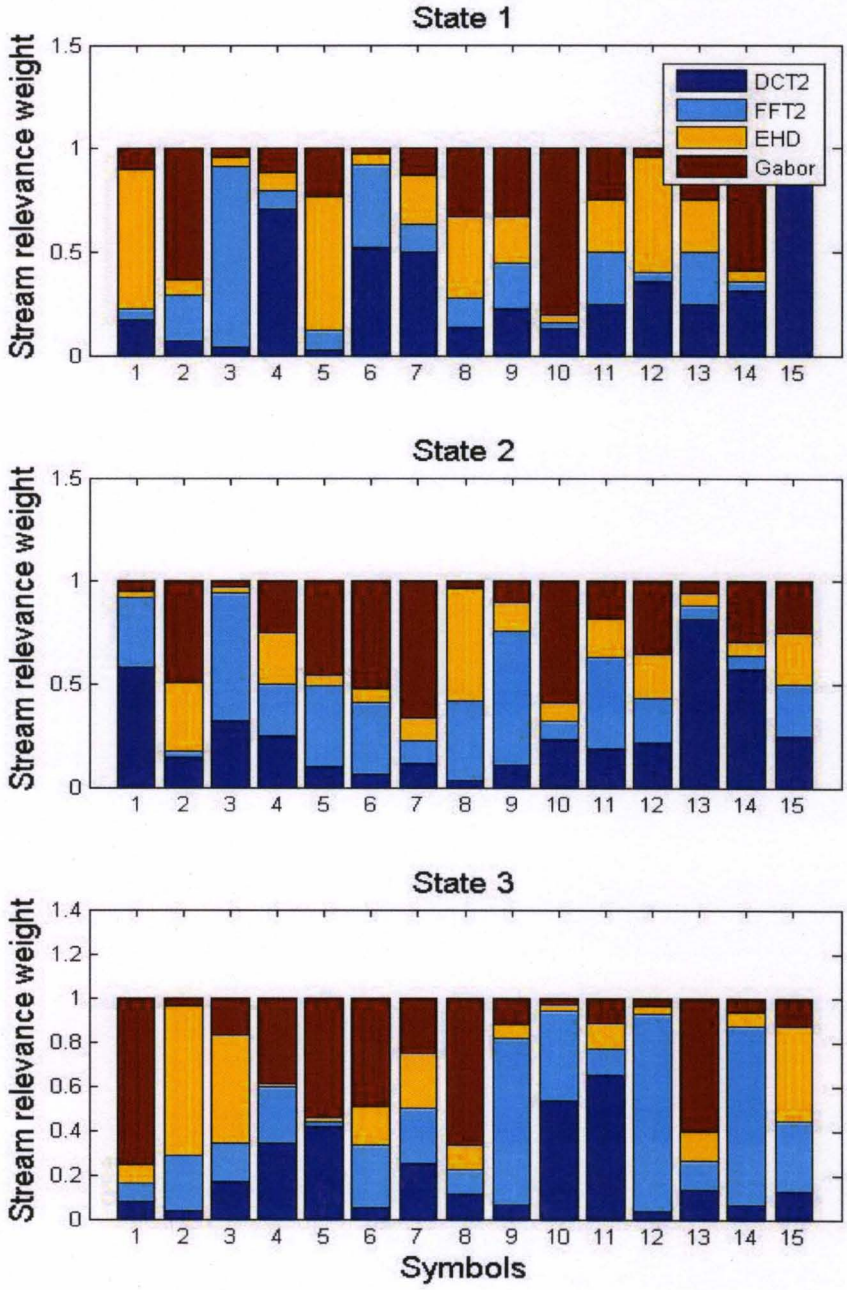


Figure 77. The stream relevance weights of the closest symbols to a missed face image standard HMMs but correctly classified by the MSHMM, learned by the MSDHMM^{P_1}

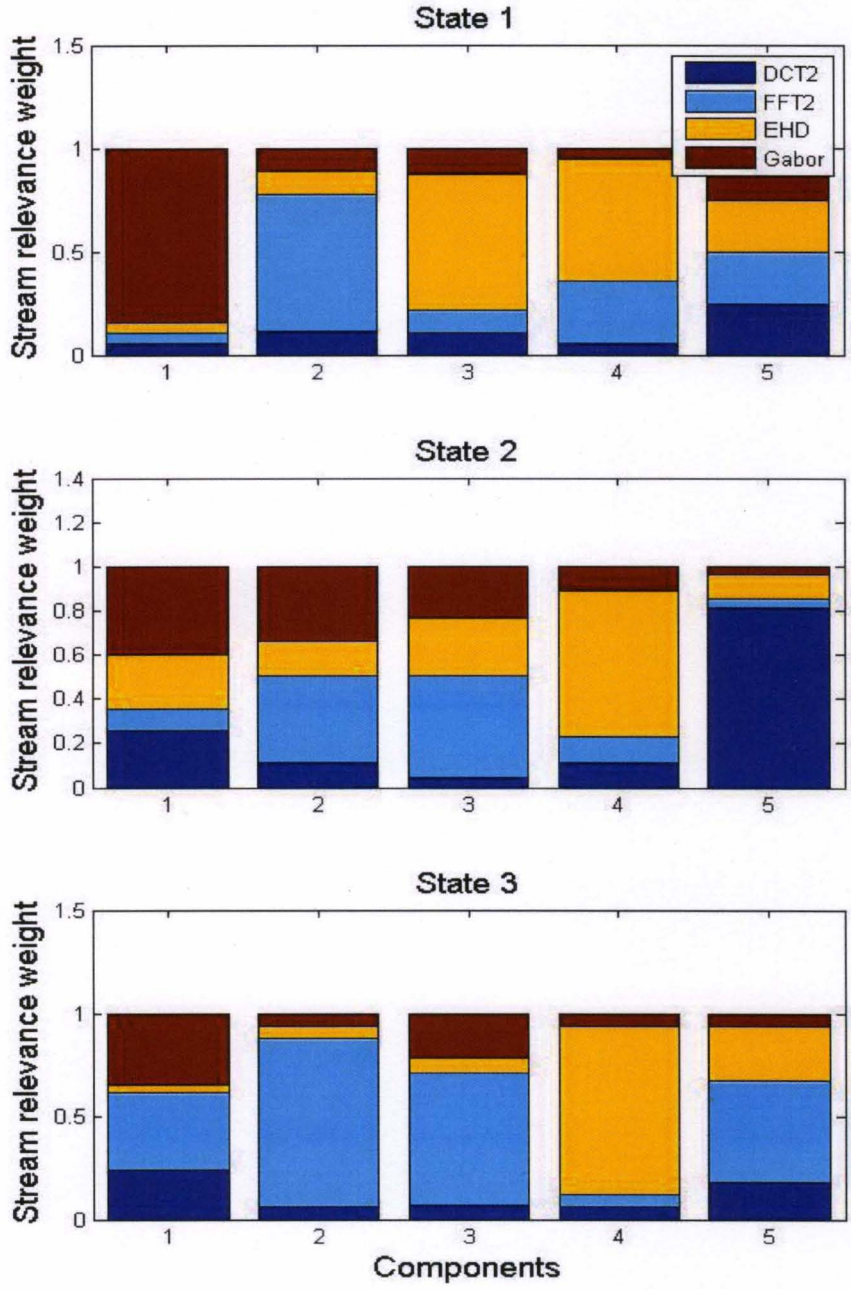


Figure 78. The stream relevance weights of the closest symbols to a missed face image standard HMMs but correctly classified by the MSHMM, learned by the MSDHMM^{L_m}

ing component the MSDHMM includes. The proposed MSCHMM structures outperform existing MSCHMM. This is due mainly to the simultaneous parameter optimization that is made possible within the newly proposed structures.

TABLE 10. Comparison of the performance of the different CHMM structures over the AUSLAN data

Simple pairs	Baseline CHMM	MSCHMM^{G_m}	MSCHMM^{G_s}	MSCHMM^{L_m}	MSCHMM^{L_s}
'hot' vs 'cold'	54.075 %	52.00 %	55.5%	59.075 %	60.025%
'eat' vs 'drink'	62.075 %	61.00%	60.00%	64.075%	71.25%
'happy' vs 'sad'	60.25%	63.86 %	67.35%	70.25 %	72.65%
'yes' vs 'no'	58.25 %	57.25 %	58.00 %	65.25 %	75.00 %
'give' vs 'take'	70.15 %	72.15 %	75.00 %	79.45 %	80.00%
'paper' vs 'pen'	75.25 %	75.00 %	75.00 %	78.00 %	78.00%
'science' vs 'research'	80.25 %	79.00 %	81.00 %	83.00 %	86.00%
'soon' vs 'hurry'	65.35 %	65.25 %	66.00 %	70.00 %	69.00%
'spend' vs 'cost'	75.65 %	75.00 %	74.00 %	75.00 %	77 %
'write' vs 'read'	99.25 %	100 %	100 %	100 %	100%

TABLE 11

Music data statistics

Genre	Pop	Rock	Folk/Country	Alternative	Jazz	Electronic	Blues	Rap/HipHop	Funk/Soul
Size	116	504	222	145	319	113	120	300	47

TABLE 12

Comparison of the performance of the different DHMM structures over the music data

Classifier	Recognition rate
Baseline DHMM (MFCC)	26.00 %
Baseline DHMM (LPC)	24.00 %
Baseline DHMM (MFCC + LPC)	27.00 %
MSDHMM^{D}	29.00 %
$\text{MSDHMM}^{\text{P}_l}$	33.00 %
$\text{MSDHMM}^{\text{P}_g}$	31.00 %

TABLE 13

Comparison of the performance of the different CHMM structures over the music data

Classifier	Recognition rate
Baseline CHMM (MFCC)	25.23 %
Baseline CHMM (LPC)	26.56 %
Baseline CHMM (MFCC + LPC)	28.3 %
$\text{MSCHMM}^{\text{G}_m}$	30.35 %
$\text{MSCHMM}^{\text{G}_s}$	31.65 %
$\text{MSCHMM}^{\text{L}_m}$	35.88 %
$\text{MSCHMM}^{\text{L}_s}$	38.95 %

TABLE 14

Comparison of standard DHMM and MSDHMMs on the face data base

Classifier	Recognition rate
Baseline DHMM (all streams)	64.025%
Baseline DHMM (2D-DCT)	68.5%
Baseline DHMM (2D-FFT)	62.65%
Baseline DHMM (Gabor)	60.00%
Baseline DHMM (EHD)	65.85%
MSDHMM^{D}	78.25%
$\text{MSDHMM}^{\text{P}_l}$	84.65%
$\text{MSDHMM}^{\text{P}_g}$	82.00 %

TABLE 15

Comparison of standard CHMM, $MSCHMM^G$, and $MSCHMM^L$ on the face data base

Classifier	Recognition rate
Baseline CHMM (all streams)	67.35%
Baseline CHMM (2D-DCT)	63.45%
Baseline CHMM (2D-FFT)	65.25%
Baseline CHMM (Gabor)	61.50%
Baseline CHMM (EHD)	63.00%
$MSCHMM^{G_m}$	68.00%
$MSCHMM^{G_s}$	70.00%
$MSCHMM^{L_m}$	73.00%
$MSCHMM^{L_s}$	75.00 %

CHAPTER VII

Conclusions and future directions

An expert is a man who has made all the mistakes, which can be made, in a very narrow field.

Niels Bohr

VII.1 Conclusions

This dissertation addressed the problems associated with modeling multi-modal temporal data with Hidden Markov Models. We have assumed that the original (raw data) can be characterized better by various sources of information (modalities/streams/views) that do not necessarily share the same relevance or power of expressiveness. We also assumed that these views represent separate interpretations of the raw data, and generate synchronous sequences. Given these assumptions, we have proposed Generalized Multi-stream HMM structures for both discrete and continuous distributions. We argued that the proposed structures alleviate the limitations of the existing multi-stream HMM structures.

We have proposed multi-stream HMM for the discrete case. This problem has not been addressed in the literature. We have proposed two different approaches: the first one is distance based and the second is probability based. The distance based approach consists of a two step learning method. The first step aims at initializing the model parameters and learning the stream relevance weights that are symbol dependent. The second step uses the standard Baum-Welch algorithm to learn the rest of the model parameters.

The probability based approach consists of a novel DHMM structure asserting that each symbol of the codebook is assigned a set of partial probabilities and relevance weights relative to each stream. Combining both partial probabilities and stream relevance weights had lead to the linear and geometric probability based MSDHMM. For these structures, we have generalized the Baum-Welch and the MCE/GPD learning algorithms to allow for the simultaneous learning of all model parameters including the stream relevance weights.

For the continuous case, our approach consists of approximating the pdfs by a linear combination of pdfs representing the individual streams. This approximation is motivated by the assumption that in the presence of the information about the most relevant stream occurring at time t, the likelihood of an observation vector o_t reduces to the likelihood of the contribution of stream k, say, $o_t^{(k)}$. This is performed at the state, and the mixture levels. This linearization allows for a maximum likelihood based learning. In fact, the standard Baum-Welch algorithm is generalized to allow for simultaneous learning of all of the model parameters. A discriminative training is also proposed by generalizing of the MCE/GPD algorithm. The necessary conditions are then derived to learn the different parameters.

For both the discrete and continuous multi-stream HMM, we have proven that the generalized Baum-Welch algorithm guarantees a convergence toward a local maximum of the likelihood function. For the discriminative training part, the MCE/GPD algorithm was selected since its objective function approximates the true Bayes risk when large amount of training data is available.

Evaluation of the proposed models on several applications shows that the GMSHMM outperform the baseline HMM as well as the existing HMM. Furthermore, extensive experiments with various landmine data collections show that the GMSHMM based landmine detector is more accurate than the standard HMM based landmine detector. Also, for all the MSHMM variations, the generalized Baum-Welch algorithm combined with the generalized MCE has been shown to perform better than the individual Baum-Welch and MCE. This mainly due to the discriminative component embedded within the MCE algorithm, that guarantees maximum separation between the models learned by the Baum-Welch, and hence better generalization.

In the discrete case, even though all the variations of the proposed MSDHMM outperform the baseline DHMM, the $\text{MSDHMM}^{\text{P}_\text{l}}$ has a superior performance. This is mainly due to the linear form of the observation probability distribution that has a less sensitivity than the geometric form in the $\text{MSDHMM}^{\text{P}_\text{g}}$.

In the continuous case, the $\text{MSCHMM}^{\text{L}_\text{m}}$ has the most superior performance, especially for the landmine data. This is basically due to the fact that $\text{MSCHMM}^{\text{L}_\text{m}}$ captures deep stream relevance: in the mixture level of each state. However, we notice a comparable performance with the $\text{MSCHMM}^{\text{L}_\text{s}}$ when the streams relevance variability is not high within the feature space.

VII.2 Future directions

The proposed GMSHMM, by the addition of the stream relevance weights, have more parameters than the standard HMM. Thus, it has higher complexity. Accordingly, in the presence of not enough data, the GMSHMM tends to overfit more than the baseline HMM. One approach to alleviate this limitation is to use regularization theory to control the complexity of the proposed models.

The proposed models have been studied under the frequentist probabilistic approach, and no prior knowledge have been used. The full Bayesian approach allows the use of prior knowledge. This approach could be adapted to the proposed MSHMM to alleviate the overfitting problem in the presence of limited data.

REFERENCES

[1] Laxman Srivatsan and Sastry P. S., "A survey of temporal data mining.," *Sadhana Vol. 31, Part 2, pp. 173-198*, April 2006.

[2] Skjellaug Bjorn, "Temporal Data: Time and Object Databases.," *Research Report 245, ISBN 82-7368-160-2*, April 1997.

[3] F. Roddick John and Spiliopoulou Myra, "A Bibliography of Temporal, Spatial and Spatio-Temporal Data Mining Research.," *ACM SIGKDD, Volume 1, Issue 1, page 34*, June 1999.

[4] Roddick Jhon F. and Spiliopoulou Myra, "A survey of Temporal Knowledge Discovery Paradigms and Methods.," *IEEE Transactions on Knowledge and Data Engineering, Vol. 14, No. 4*, August, 2002.

[5] M. Antunes Claudia and L. Oliveira Arlindo, "Temporal Data Mining: an overview.," *Artificial Intelligence for Financial Times Series Analysis. A workshop of the 10 th Portuguese Conference on Artificial Intelligence EPIA 01*, December 2001.

[6] Rabiner Lawrence, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.," *Proceedings of the IEEE, VOL. 77, NO.2,*, February, 1989.

[7] Stefan Eickeler, Andreas Kosmala, and Gerhard Rigoll, "Hidden markov model based continuous online gesture recognition," *Int. Conference on Pattern Recognition (ICPR)*, pp. 1206–1208, 1998.

[8] Freitas C.O.D.A. De Oliveira Junior J.J., De Carvalho J.M. and Sabourin R., "Evaluating nn and hmm classifiers for handwritten word recognition," *Computer Graphics and Image Processing. Proceedings. XV Brazilian Symposium on*, pp. 210 – 217, 7-10 Oct. 2002.

[9] Rejean Plamondon Xiaolin Li, Marc Parizeau, "Training hidden markov models with multiple observations: A combinatorial method.," *IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 22, No. 4*, April, 2000.

[10] Hunkapillier T McClure M Baldi P, Chauvin Y, "Hidden markov models of biological primary sequence information.," *Proc. Nat. Acad. Sci. USA 91: 1059-1063*, 1994.

[11] Grant G R Ewens W J, *Statistical methods in bioinformatics: An introduction.*, Springer-Verlag, New York, 2001.

[12] Dorffner G Tino P, Schittenkopf C, "Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading.," 2000.

[13] Timo Koski, *Hidden Markov Models for Bioinformatics.*, Kluwer Academic Publishers, Netherlands, 2001.

[14] Wu Zhiyong, Cai Lianhong, and Meng Helen, "Multi-level fusion of audio and visual features for speaker identification," *International Conference on Biometrics (ICB)*, 2005.

[15] C Chibelishi, J Mason, and F Deravi, "Feature Level Data Fusion For Bimodal Person Recognition," *International Conference on Image Processing and Its Applications (ICIP)*, 1997.

[16] V Chatziz, A Bors, and I Pitas, "Multimodal decision level fusion for person authentication," *IEEE Trans. Syst. Man Cybern. A 29*, 1999.

[17] Zoubin Ghahramani and Michael Jordan, "Factorial Hidden Markov Models," *Neural Information Processing Systems (NIPS)*, 1995.

[18] A Nefian, L Liang, T Fu, and X Liu, "A Bayesian approach to audio-visual speaker identification," *Fourth International Conf. Audio- and Video-based Biometric Person Authentication*, 2003.

[19] S. Dupont and J. Luettin, "Audio-visual speech modeling for continuous speech recognition.," *IEEE Transactions on Multimedia, Vol. 2, No. 3, September.*

[20] Potamianos Gerasimos, Neti Chalapathy, Luettin Juergen, and Matthews Iain, "Audio-Visual Automatic Speech Recognition: An Overview," *Audio-Visual Speech Processing, E. Vatikiotis-Bateson, G. Bailly, and P. Perrier (Eds.), MIT Press, ISBN: 0-26-222078-4*, 2009.

[21] Petar S. Aleksic and Aggelos K. Katsaggelos, "Product hmms for audio-visual continuous speech recognition using facial animation parameters," *Proceedings of the International Conference on Multimedia and Expo - Volume 1, pages: 481 - 484*, 2003.

[22] Koji Iwano Taichi Asami and Sadaoki Furui, "A stream-weight and threshold estimation method using adaboost for multi-stream speaker verification," *Acoustics, Speech and Signal Processing. ICASSP Proceedings.*, 2006.

[23] Chuan Wang Dongxin Xu, Craig Fancourt, "Multi-channel hmm.," *Proceedings of the Acoustics, Speech, and Signal Processing. on Conference Proceedings., IEEE International Conference - Volume 02, pages: 841-844*, 1996.

[24] Antonio J. Rubio Jose C. Segura Carmen Benitez Angel de la Torre, Antonio M. Peinado, "Discriminative feature weighting for hmm-based continuous speech recognizers.," *Speech Communication 38, on pages: 267-286.*

[25] Gerasimos Potamianos and Hans Peter Graf, "Discriminative training of hmm stream exponents for audio-visual seech recognition.," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Seattle, Vol. 6, pp. 3733-3736*, 1998.

[26] Gerasimos Potamianos and Alexandros Potamianos, "Speaker adpatation for audio-visual speech recognition.," *Proceedings EUROSPEECH, Budapest, Vol. 3, pp. 1291-1294.*

[27] Javier Hernando, "Maximum likelihood weighting of dynamic speech features for cdhmm speech recognition," *Acoustics, Speech, and Signal Processing,ICASSP.,IEEE, on pages: 1267-1270 vol.2.*

[28] H. Frigui and O Nasraoui, "Simulatneous clustering and attribute discrimination.," *Fuzzy systems, The Ninth IEEE International Conference, Vol.1, pp.: 158-163*, 2000.

[29] G. D. Forney, "The viterbi algorithm.," *Proc. IEEE, vol. 61,no. 3, pp. 268-278*, 1973.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM algorithm,," *Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No.1., pp. 1-38.*, 1977.

[31] A. Nadas, "A decision theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional vesus conditional maximum likelihood,," *IEEE Trans. Acoust., Speech, Signal Processing, vol. 31, no. 4, pp. 814817*, 1983.

[32] Biing-Hwang Juang, Wu Chou, and Chin-Hui Lee, "Minimum Classification Error Rate Methods for Speech Recognition," *Transactions on Speech and Audio Processing, VOL. 5, No.3*, May 1997.

[33] S Kapadia, "Discriminative training of hidden markov models.," *PhD thesis, University of Cambridge.*, March 18, 1998.

[34] L Bahl, P Brown, P De Souza, and R Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition,," *in Proc. ICASSP 86,pp. 4952.*, 1986.

[35] Missaoui O. Frigui H. and Gader P. D., "Landmine detection using discrete hidden markov models with gabor features," *in SPIE Conf. Detection and Remediation Technologies for Mines and Minelike Targets XII, April*, 2007.

[36] Hichem Frigui and Salem Salem, "Fuzzy clustering and subset feature weighting.," *The IEEE International Conference on Fuzzy Systems,*, 2003.

[37] Maebatake Masaru, Suzuki Iori, Nishida Masafumi, Horiuchi Yasuo, and Kuroiwa Shingo, "Sign language recognition based on position and movement using multi-stream hmm," *Second International Symposium on Universal Communication*, 2008.

[38] Norouzian Atta, Selouani Sid-Ahmed, Tolba Hesham, and O'Shaughnessy Douglas, "Incorporating phonetic knowledge into a multi-stream HMM framework," *Canadian Conference on Electrical and Computer Engineering*, 2008.

[39] Kessentini Yousri, Paquet Thierry, and BenHamadou AbdelMajid, "A multi-stream approach to off-line handwritten word recognition," *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 2007.

[40] Guillaume Gravier Ashutosh Garg Gerasimos Potamianos, Chalapathy Neti and Andrew W. Senior, "Recent advances in the automatic recognition of audio-visual speech," *Proceedings of the IEEE, Volume: 91, Issue: 9, On pages: 1306- 1326*, 2003.

[41] J.C. Bezdek L.I. Kuncheva and R. Duin, "Decision templates for multiple classifier fusion: An experimental comparison," *Pattern Recognition, vol. 34, no. 2, pp. 299314*, 2001.

[42] J.J. Hull T.K. Ho and S.N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. on Pattern Analy. Machine Intel.,vol. 16, no. 1, pp. 6675*, 1994.

[43] S.B. Cho and J.H. Kim, "Combining multiple neural networks by fuzzy integral for robust classification," *IEEE Transactions on Systems, Man and Cybernetics, vol. 25, no. 2, pp. 380384*, 1995.

[44] Y. Lu, "Knowledge integration in a multiple classifier system,," *Applied Intelligence, vol. 6, no. 2, pp. 7586,*, 1996.

[45] M Jordan, "Graphical models," *Statistical Science*, 2004, Vol. 19, No. 1, 140-155.

[46] Matthew Brand, "Coupled hidden Markov models for modeling interacting processes," *MIT Media Lab Perceptual Computing, Learning and Common Sense Technical Report 405*, Juin 1997.

[47] S. Nakamura, "Fusion of audio-visual information for integrated speech processing," *in Audio- and Video-Based Biometric Person Authentication, J. Bigun and F. Smeraldi, Eds. Berlin, Germany: Springer-Verlag, pp. 127143.*, 2001.

[48] L. R. Bahl, P. F. Brown, P. V. De Souza, and R. L. Mercer, "Maximum mutual information estimation of hidden markov model parameters for speech recognition,," *in Proc. ICASSP 86,pp. 4952.*, Apr, 1986.

[49] Gerasimos Potamianos Chalapathy Neti Guillaume Gravier, Scott Axelrod, "Maximum entropy and mce based hmm stram weight estimation for audio-visual asr," *ICASSP*, 2002.

[50] J.A. Hartigan, *Clustering Algorithms*, Wiley, 1975.

[51] J.C. Bezdec, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, New York, 1981.

[52] Erik McDermott and Shigeru katagiri, "A derivation of minimum classification error from the theoretical classification risk using parzen estimation," *Computer Speech and Language 18 (2004) 107,122*, 2003.

[53] Ian Jolliffe Paul Garthwaite and Byron Jones, *Statistical Inference*, Oxford University Press, 2002.

[54] Adriaan van den Bos, *Parameter estimation for scientists and engineers*, John Wiley and Sons, Inc., 2007.

[55] Z Ghahramani and M Jordan, "Factorial hidden Markov models," *Machine Learning*, 1997.

[56] T.R. Witten, "Present state of the art in ground-penetrating radars for mine detection," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets III, orlando, FL, pp. 576-586*, 1998.

[57] B. Nelson G. Vaillette P.D. Gader, H. Frigui and J.M. Kelle, "New results in fuzzy set based detection of landmines with gpr," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 1075-1084*, 1999.

[58] D. Carevic, "Kalman filter-based approach to target detection and target-background separation in ground-penetrating radar data," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 1284-1288*, 1999.

[59] A. Gunatilaka and B.A. Baertlein, "Subspace decomposition technique to improve gpr imaging of anti-personal mines," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets V, orlando, FL*, 2000.

[60] H. Brunzell, "Detection of shallowly burried objects using impulse radar," *IEEE Trans. Geoscience and Remote Sensing, vol. 37, pp. 875-886*, 1999.

[61] R.K. Mehra S. Yu and T.R. Witten, "Automatic mine detection based on ground penetrating radar.," *in SPIE Conf Detection and Remediation Technologies for Mines and Minelike Targets IV, orlando, FL, pp. 961-972*, 1999.

[62] K. Satyanarayana H. Frigui and P. Gader, "Detection of landmines using fuzzy and possibilistic membership functions.," *in proceedings of the IEEE Conference on Fuzzy Systems, Saint Louis, Missouri*, 2003.

[63] D. Carevic, "Clutter reduction and target detection in ground penetrating radar using wavelets.," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IV, Orlando, FL, USA*, 1999.

[64] M. Mystkowski P. Gader and Y. Zhao, "Landmine detection with ground penetrating radar using hidden markov models.," *IEEE Trans, Geoscience and Remote Sensing, vol.39, pp. 1231-1244*, 2001.

[65] V. S. Munshi S. L. Tantum, Y. Wei and L. M. Collins, "A comparison of algorithms for landmine detection and discrimination using ground penetrating radar.," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Mineslike Targets, Orlando, FL, USA*, 2002.

[66] Paul Gader Hichem Frigui, K.C.Ho, "Real-time landmine detection with ground-penetrating

radar using discriminative and adaptive hidden markov models.," *EURASIP Journal on Applied Signal Processing 2005:12, 1867-1885.*

[67] H. Frigui G. Vaillette P. Gader, B. Nelson and J. Keller, "Fuzzy logic detection of landmines with ground penetrating radar.," *Signal Processing, special issue on fuzzy logic in signal processing, vol. 80, pp. 1069-1084,*, 2000.

[68] W. H. Lee P. Gader and J. N. Wilson, "Detecting landmines with ground penetrating radar using feature-based rules, order statistics, and adaptive whitening,," *IEEE Trans, Geoscience and Remote Sensing, vol. 42, no. 11, pp.2522-2534,*, 2004.

[69] Gader P. Frigui H. and Kotturu S., "Detection and discrimination of landmines in ground penetrating radar using an eigenmine and fuzzy membership function approach," *in SPIE Conf. Detection and Remediation Technologies for Mines and Minelike Targets, April*, 2004.

[70] H. Frigui and P. D. Gader, "Detection and discrimination of land mines based on edge histogram descriptors and fuzzy k-nearest neighbors," *in Proceedings of the IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, July*, 2006.

[71] K. J. Hintz, "Snr improvements in niitek ground penetrating radar," *in Proceedings of the SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IX, Orlando, FL, USA, April*, 2004.

[72] C. S. Throckmorton P. A. Torrione and L. M. Collins, "Performance of an adaptive feature-based processor for a wideband ground penetrating radar system,," *IEEE Trans. Aerospace and Electronic Systems (in press).*

[73] Hichem Frigui, Oualid Missaoui, and Paul Gader, "Landmine detection using discrete hidden Markov models with Gabor features.," *Proc. SPIE Vol. 6553 , Apr. 27*, 2007.

[74] H. Frigui and P. Gader, "Detection and discrimination of landmines in ground-penetrating radar based on edge histogram descriptors," *in Proc.SPIE Conf. Detect. Remediation Technol. Mines Minelike Targets, pp. 62176233.*, 2006.

[75] B. S. Manjunath, P. Salembier, and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface.*, WILEY, July ,2002.

[76] R Duda and P Hart, *Pattern Classification and Scene Analysis.*, John Wiley and Sons, 1973.

[77] Paul Gader, R. Grandhi, W. Lee, J. Wislon, and D. Ho, "Feature analysis for the NIITEK ground-penetrating radar using order weighted averaging operators for landmine detection,," *in SPIE Conf. Detect. Remediation Technol. Mines Minelike Targets, Orlando, FL, vol. 5415, pp. 953962.*, Apr. 2004,.

[78] Hichem Frigui and Paul Gader, "Detection and Discrimination of Land Mines in Ground-Penetrating Radar Based on Edge Histogram Descriptors and a Possibilistic K-Nearest Neighbor Classifier,," *IEEE Transactions on Fuzzy Systems, Vol. 17, No.1,*, February 2009.

[79] Mohammed Waleed Kadous, "Temporal classification: Extending the classification paradigm to multivariate time series," *PhD thesis, School of Computer Science and Engineering, University of New South Wales*, 2002.

[80] H. Homburg, I. Mierswa, B. Mo"ller, K. Morik, and M. Wurst, "A benchmark dataset for audio classification and clustering," 2005.

[81] Davis S.B. and Mermelstein P., "Comparison of parametric representations for monosyllabic word recognition with continuously spoken sentences,," *IEEE Transactions on Acoustics Speech and Signal Processing, vol. 28, no. 4, pp. 357-366*, 1980.

[82] Rabiner L. and Juang B-H, "Fundamentals of speech recognition," 1993.

[83] Slaney M., "Auditory toolbox version 2.," *Technical report, Interval Research Corporation*, 1998.

[84] Stevens S. S., Volkmann J., and B. Newman. E., "A scale for the measurement of the psychological magnitude pitch.," *The Journal of the Acoustical Society of America, 8(3):155210*, 1937.

[85] Britanak Vladimir, C. Yip Patrick, and Rao K. R, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*, Academic Press, 2006.

[86] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. PAMI, vol. 18, pp. 837842*, 1996.

APPENDIX A

Proof of propositions (II.7.1) and (II.7.2)

A.1 Proof of the proposition (II.7.1)

Proposition A.1.1. *If the value of $\mathbb{Q}(\lambda, \overline{\lambda})$ increases, then the value of $Pr(O|\lambda)$ also increases, i.e.,*

$$\mathbb{Q}(\lambda, \overline{\lambda}) \geq \mathbb{Q}(\lambda, \lambda) \Longrightarrow Pr(O|\overline{\lambda}) \geq Pr(O|\lambda)$$

Proof. We have $\mathbb{Q}(\lambda, \overline{\lambda}) = \sum_Q Pr(Q|O, \lambda) \ln(Pr(O, Q|\overline{\lambda}))$.

In one hand, we can write:

$$\begin{aligned}
\mathbb{Q}(\lambda, \overline{\lambda}) - \mathbb{Q}(\lambda, \lambda) &= \sum_Q Pr(Q|O, \lambda) \ln(Pr(O, Q|\overline{\lambda})) - \sum_Q Pr(Q|O, \lambda) \ln(Pr(O, Q|\overline{\lambda})) \\
&= \sum_Q Pr(Q|O, \lambda) \ln \frac{Pr(O, Q|\overline{\lambda})}{Pr(O, Q|\overline{\lambda})}
\end{aligned}$$

In another hand,

$$\begin{aligned}
\ln \frac{Pr(O|\overline{\lambda})}{Pr(O|\lambda)} &= \ln \sum_Q \frac{Pr(O, Q|\overline{\lambda})}{Pr(O|\lambda)} \\
&= \ln \sum_Q \frac{Pr(O, Q|\lambda)}{Pr(O|\lambda)} \frac{Pr(O, Q|\overline{\lambda})}{Pr(O, Q|\lambda)} \\
&= \ln \sum_Q Pr(Q|O, \lambda) \frac{Pr(O, Q|\overline{\lambda})}{Pr(O, Q|\lambda)}
\end{aligned}$$

Using the Jensen's inequality due to the convexity of the logarithm function, we can get:

$$\ln \sum_Q Pr(Q|O, \lambda) \frac{Pr(O, Q|\overline{\lambda})}{Pr(O, Q|\lambda)} \geq \sum_Q Pr(Q|O, \lambda) \ln \frac{Pr(O, Q|\overline{\lambda})}{Pr(O, Q|\lambda)} \tag{A.1.1}$$

Thus,

$$\ln \frac{Pr(O|\overline{\lambda})}{Pr(O|\lambda)} \geq \mathbb{Q}(\lambda, \overline{\lambda}) - \mathbb{Q}(\lambda, \lambda)$$

We conclude then that,

$$\mathbb{Q}(\lambda, \overline{\lambda}) \geq \mathbb{Q}(\lambda, \lambda) \Longrightarrow Pr(O|\overline{\lambda}) \geq Pr(O|\lambda)$$

□

A.2 Proof of the proposition (II.7.2)

Proposition A.2.1. *λ is a critical point of $Pr(O|\lambda)$ if and only if it is a critical point of $\mathbb{Q}(\lambda, \overline{\lambda})$, i.e.,*

$$\frac{\partial Pr(O|\lambda)}{\partial \theta_p} = \left.\frac{\partial \mathbb{Q}(\lambda, \overline{\lambda})}{\partial \overline{\theta}_p}\right|_{\overline{\lambda}=\lambda},$$

Proof.

$$\begin{aligned}
\left.\frac{\partial \mathbb{Q}(\lambda, \overline{\lambda})}{\partial \overline{\theta}_p}\right|_{\overline{\lambda}=\lambda} &= \left.\frac{\partial \sum_Q Pr(Q|O,\lambda) \ln Pr(O,Q|\overline{\lambda})}{\partial \overline{\theta}_p}\right|_{\overline{\lambda}=\lambda} \\
&= \sum_Q Pr(Q|O,\lambda) \left.\frac{\partial \ln Pr(O,Q|\overline{\lambda})}{\partial \overline{\theta}_p}\right|_{\overline{\lambda}=\lambda} \\
&= \sum_Q Pr(Q|O,\lambda) \left.\frac{\frac{\partial Pr(O,Q|\overline{\lambda})}{\partial \overline{\theta}_p}}{Pr(O,Q|\overline{\lambda})}\right|_{\overline{\lambda}=\lambda} \\
&= \sum_Q Pr(Q|O,\lambda) \frac{\frac{\partial Pr(O,Q|\lambda)}{\partial \theta_p}}{Pr(O,Q|\lambda)} \\
&= \sum_Q \frac{1}{Pr(O|\lambda)} \frac{\partial Pr(O,Q|\lambda)}{\partial \theta_p} \\
&= \frac{1}{Pr(O|\lambda)} \frac{\partial \sum_Q Pr(O,Q|\lambda)}{\partial \theta_p} \\
&= \frac{1}{Pr(O|\lambda)} \frac{\partial Pr(O|\lambda)}{\partial \theta_p} \\
&= \frac{\partial \ln Pr(O|\lambda)}{\partial \theta_p} \\
&= \frac{\partial Pr(O|\lambda)}{\partial \theta_p}
\end{aligned}$$

□

APPENDIX B

Simultaneous Clustering and Attribute Discrimination: SCAD

The initial SCAD algorithm [28], was designed to search for the optimal clusters' prototypes and the optimal relevance weight for each feature of each cluster. However, for high dimensional data, learning a relevance weight for each feature may lead to overfitting. To avoid this situation, a coarse version of SCAD [36] (called $SCAD_c$) was proposed. Instead of learning a weight for each feature, the set of features is divided into logical subsets, and a weight is learned for each feature subset.

Let $\chi = \{x_j \in \Re^p | j = 1, ..., N\}$ be a set of N feature vectors. Let $\mathbf{B} = (\beta_1, ..., \beta_c)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. Each β_i consists of a set of parameters. Let u_{ij} represent the membership of x_j in cluster β_i. The $C \times N$ fuzzy C-partition $U = [u_{ij}]$ statisfies [16]:

$$u_{ij} \in [0, 1] \ \forall i, \text{and} \sum_{i=1}^{C} u_{ij} = 1 \forall j. \tag{B.0.1}$$

Assume that the p features have been partitioned into K subsets: FS^1, FS^2,...,FS^K, and that each subset FS^s, includes k^s features. Let d_{ij}^s be the partial distance between x_j and cluster i using the sth feature subset. Let $V = [v_{is}]$ be the relevance weight for FS^s with respect to cluster i. The total distance, D_{ij}, between x_j and the cluster i is then computed by aggregating the partial distances and their weights. Typically $D_{ij}^2 = \sum_{s=1}^{K} v_{is}(d_{ij}^s)^2$.

SCADc minimizes

$$J = \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^m \sum_{s=1}^{K} v_{is}(d_{ij}^s)^2 + \sum_{i=1}^{C} \delta_i \sum_{s=1}^{K} v_{is}^2, \tag{B.0.2}$$

subject to (B.0.1) and

$$v_{is} \in [0, 1] \ \forall i, s; \text{ and } \sum_{s=1}^{K} v_{is}, \forall i, \tag{B.0.3}$$

To optimize J, with respect to V, we use the Lagrange multiplier technique, and obtain

$$v_{is} = \frac{1}{K} + \frac{1}{2\delta_i} \sum_{j=1}^{N} (u_{ij})^m [D_{ij}^2/K - (d_{ij}^s)^2]. \tag{B.0.4}$$

The first term in (B.0.4), $(1/K)$, is the default value if all K subsets are treated equally and no discrimination is performed. The second term is a bias that can be either positive or negative. It

is positive for compact feature subsets where the partial distance is, on the average, less than the total distance. If a feature subset is more compact for the most of the points that belong to a given cluster (high u_{ij}), then it would be very relevant for that cluster.

Minimization of J with respect to $\mathbf{U}$ yields

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}(D_{ij}^2/D_{kj}^2)^{\frac{1}{m-1}}}. \tag{B.0.5}$$

Minimization of J with respect to the prototype parameters depends on the choice of d_{ij}^s. Since the partial distances are treated independent of each other (i.e., disjoint feature subsets), and since the second term in (B.0.2) does not depend on the prototype parameters explicitly, the objective function in (B.0.2) can be decomposed into K independent problems:

$$J_s = \sum_{i=1}^{C}\sum_{j=1}^{N} u_{ij}^m v_{is}(d_{ij}^s)^2, \text{ for } s = 1, ..., K. \tag{B.0.6}$$

Each J_s would be optimized with respect to a different set of prototype parameters. For instance, if d_{ij}^s is the Euclidean distance, minimization of J_s would yield the following update equation for the centers of subset s

$$c_i^s = \frac{\sum_{j=1}^{N} u_{ij}^m x_j^s}{\sum_{j=1}^{N} u_{ij}^m} \tag{B.0.7}$$

SCADc is an iterative algorithm that starts with an initial partition and alternates between the update equations of u_{ij} , v_{is}, and c_i^s.

APPENDIX C

Lagrange multipliers optimization

Suppose we seek the position x_0 of an extremum of a scalar-valued function $f(x)$, subject to some constraint. If a constraint can be expressed in the form $g(x) = 0$, then we can find the extremum of $f(x)$ as follows. First we form the Langrangian function:

$$L(x, \rho) = f(x) + \underbrace{\rho g(x)}_{=0}, \tag{C.0.8}$$

where ρ is a scalar called the Lagrange *undetermined multiplier*. We convert this constraint optimization problem into an unconstrained problem by taking the derivative,

$$\frac{\partial L(x, \rho)}{\partial x} = \frac{\partial f(x)}{\partial x} + \rho \frac{\partial g(x)}{\partial x} \tag{C.0.9}$$

and using standards methods from calculus to solve the resulting equations for ρ and the extremizing value of x. (Note that the term $\rho \frac{\partial g}{\partial x}$ does not vanish, in general.) The solution gives x position of the extremum, and it is a simple matter of substitution to find the extreme value of $f(.)$ under the constraints.

APPENDIX D

Generalized Baum-Welch for the proposed Multi-stream HMM strucutres

In this section, we outline the step by step approach followed to generalize the Baum-Welch algorithm in order to learn the parameters of the MSDHMM^{P_1} structure.

The objective function to optimize is the following:

$$\mathbb{Q}(\lambda, \bar{\lambda}) = \sum_{Q} \sum_{F} \ln Pr(O, Q, F|\bar{\lambda}) Pr(Q, F|O, \lambda),$$

where $F = [f_1, \ldots, f_T]$ is a sequence of random variables representing the stream indices for each time step.

This objective function involves the quantity $Pr(O, Q, F|\bar{\lambda})$ which could be expressed analytically as:

$$Pr(O, Q, F|\bar{\lambda}) = \bar{\pi}_{q_1} \prod_{t=1}^{T-1} \bar{a}_{q_t q_{t+1}} \prod_{t=1}^{T} \bar{w}_{q_t Q_V(o_t) f_t} \bar{b}_{q_t Q_V(o_t) f_t}$$

where Q_V is the quantization operation defined on an observation vector o_t as:

$$Q_V(o_t) = \underset{1 \leq j \leq M}{\operatorname{argmin}} \frac{1}{N_s} \sum_{i=1}^{N_s} \sum_{k=1}^{L} w_{ijk} \|o_t^k - v_j^k\|. \tag{D.0.10}$$

Thus, the objective function expands as follows:

$$\begin{aligned}
\mathbb{Q}(\lambda, \bar{\lambda}) \quad = \quad & \sum_{Q} \sum_{F} \log \bar{\pi}_{q_1} Pr(Q, F|O, \lambda) + \\
& \sum_{t=1}^{T-1} \sum_{Q} \sum_{F} \log \bar{a}_{q_t q_{t+1}} Pr(Q, F|O, \lambda) + \\
& \sum_{t=1}^{T} \sum_{Q} \sum_{F} \log \bar{w}_{q_t Q_V(o_t) f_t} Pr(Q, F|O, \lambda) + \\
& \sum_{t=1}^{T} \sum_{Q} \sum_{F} \log \bar{b}_{q_t Q_V(o_t) f_t} Pr(Q, F|O, \lambda).
\end{aligned}$$

To find the value of $\overline{w}_{ijk}$ that maximizes the auxiliary function $\mathbb{Q}(.,.)$, only the third term of the expanded expression is considered since it is the only part of $\mathbb{Q}(.,.)$ that depends on $\overline{w}_{ijk}$.

This term can be expressed as follows:

$$\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log \bar{w}_{q_t Q_V(o_t) f_t} =$$

$$\sum_{t=1}^{T}\sum_{i}\sum_{j}\sum_{k}\log \bar{w}_{ijk}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t),$$

$$\sum_{t=1}^{T}\sum_{i}\sum_{j}\sum_{k}\log \bar{w}_{ijk}\sum_{q_1}\cdots\sum_{q_T}\sum_{f_1}\cdots\sum_{f_T}\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t)\prod_{t_1=1}^{T} Pr(q_{t_1},f_{t_1}|o_{t_1},\lambda),$$

Let $\mathcal{T}$ the expression $\sum_{q_1}\cdots\sum_{q_T}\sum_{f_1}\cdots\sum_{f_T}\delta(i,q_t)\delta(j,Q_V(o_t))\delta(k,f_t)\prod_{t_1=1}^{T} Pr(q_{t_1},f_{t_1}|o_{t_1},\lambda)$. It could be expanded to:

$$\begin{aligned}
\mathcal{T} &= \sum_{q_1}\cdots\sum_{q_{t-1}}\sum_{q_{t+1}}\cdots\sum_{q_T}\sum_{f_1}\cdots\sum_{f_{t-1}}\sum_{f_{t+1}}\cdots\sum_{f_T} \\
&\quad \delta(j,Q_V(o_t))Pr(q_t=i,f_t=k|o_t;\lambda)\prod_{t_1=1,t_1\neq t}^{T} Pr(q_{t_1},f_{t_1}|o_{t_1},\lambda) \\
&= \prod_{t_1=1,t_1\neq t}^{T}\left[\sum_{q_{t_1}}\sum_{f_{t_1}} Pr(q_{t_1},f_{t_1}|o_{t_1},\lambda)\right] Pr(q_t=i,f_t=k|o_t;\lambda)\delta(j,Q_V(o_t)) \\
&= Pr(q_t=i,f_t=k|o_t;\lambda)\delta(j,Q_V(o_t))
\end{aligned}$$

That is,

$$\sum_{t=1}^{T}\sum_{Q}\sum_{F} Pr(Q,F|O,\lambda)\log(\bar{w}_{q_t Q_V(o_t) f_t}) =$$

$$\sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t))\ln(\bar{w}_{ijk})$$

To find the update equation of $\overline{w}_{ijk}$, we use the Lagrange multipliers optimization with the constraint $\sum_{k=1}^{L}\overline{w}_{ijk}=1$. The value of $\overline{w}_{ijk}$ that maximizes the objective function $\mathbb{Q}(\lambda,\bar{\lambda})$ is exactly the same value that maximizes

$$\mathbb{Q}_w(\lambda,\bar{\lambda}) = \sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t))\ln(\bar{w}_{ijk})$$

Adding the constraint term, we obtain an extended objective:

$$\tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda}) = \sum_{t=1}^{T}\sum_{i=1}^{N_s}\sum_{j=1}^{M}\sum_{k=1}^{L} Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t))\ln(\bar{w}_{ijk}) + \sum_{i=1}^{N_s}\sum_{j=1}^{M}\rho_{ij}(1-\sum_{k=1}^{L}\overline{w}_{ijk})$$

Thus,

$$\frac{\partial\tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda})}{\partial\overline{w}_{ijk}} = -\sum_{t=1}^{T} Pr(q_t=i,f_t=k|O,\lambda)\delta(j,Q_V(o_t))\frac{1}{\overline{w}_{ijk}} - \rho_{ij}$$

Setting $\frac{\partial \tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda})}{\partial \overline{w}_{ijk}}$ to zero leads to:

$$\overline{w}_{ijk}\rho_{ij} = -\sum_{t=1}^{T} Pr(q_t = i, f_t = k|O,\lambda)\delta(j, Q_V(o_t)),$$

That is,

$$\begin{aligned}
\sum_{k=1}^{L} \overline{w}_{ijk}\rho_{ij} &= -\sum_{k=1}^{L}\sum_{t=1}^{T} Pr(q_t = i, f_t = k|O,\lambda)\delta(j, Q_V(o_t)), \\
\rho_{ij} &= -\sum_{t=1}^{T} Pr(q_t = i|O,\lambda)\delta(j, Q_V(o_t)),
\end{aligned}$$

Injecting the value of ρ_{ij} into the expression of $\frac{\partial \tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda})}{\partial \overline{w}_{ijk}}$ gives:

$$\frac{\partial \tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda})}{\partial \overline{w}_{ijk}} = -\sum_{t=1}^{T} Pr(q_t = i, f_t = k|O,\lambda)\delta(j, Q_V(o_t))\frac{1}{\overline{w}_{ijk}} + \sum_{t=1}^{T} Pr(q_t = i|O,\lambda)\delta(j, Q_V(o_t))$$

Setting the new expression of $\frac{\partial \tilde{\mathbb{Q}}_w(\lambda,\bar{\lambda})}{\partial \overline{w}_{ijk}}$ to zero gives the update equation of $\overline{w}_{ijk}$:

$$\overline{w}_{ijk} = \frac{\sum_{t=1}^{T} Pr(q_t = i, f_t = k|O,\lambda)\delta(Q_V(o_t), j)}{\sum_{t=1}^{T} Pr(q_t = i|O,\lambda)\delta(Q_V(o_t), j)}. \tag{D.0.11}$$

We recognize in $Pr(q_t = i|O,\lambda)$ the intermediate variable $\gamma_t(i)$. For $Pr(q_t = i, f_t = k|o_t,\lambda)$, it could be computed as follows:

$$\begin{aligned}
Pr(q_t = i, f_t = k|o_t,\lambda) &= Pr(q_t = i|o_t,\lambda)Pr(f_t = k|q_t = i, o_t,\lambda) \\
&= Pr(q_t = i|o_t,\lambda)\frac{Pr(o_t, f_t = k|q_t = i,\lambda)}{Pr(o_t|q_t = i,\lambda)} \\
&= Pr(q_t = i|o_t,\lambda)\frac{Pr(o_t|q_t = i, f_t = k,\lambda)Pr(f_t = k|q_t = i,\lambda)}{Pr(o_t|q_t = i,\lambda)} \\
&\approx \gamma_t(i)\frac{w_{iQ_V(o_t)k}b_{iQ_V(o_t)k}}{b_{iQ_V(o_t)}}.
\end{aligned}$$

Let, $\gamma_t(i,k) = Pr(q_t = i, f_t = k|O,\lambda)$. It follows that,

$$\overline{w}_{ijk} = \frac{\sum_{t=1}^{T} \gamma_t(i,k)\delta(o_t, j)}{\sum_{t=1}^{T} \gamma_t(i)\delta(o_t, j)}.$$

Following the same procedure, we can derive the update equations for the parameters $\overline{\pi}_i$, $\overline{a}_{ij}$, and $\overline{b}_{ijk}$.

Similarly, the necessary conditions to learn the parameters of MSDHMM$^{\mathrm{P_g}}$, MSCHMM$^{\mathrm{L_m}}$, and MSCHMM$^{\mathrm{L_s}}$ could be obtained.

APPENDIX E

Estimator properties

suppose that we have a random sample $(x_1, x_2, \cdots, x_n)$ from a probability distribution with pdf $f(x;\theta)$, and that we wish use the values, $x_1, x_2, \cdots, x_n$ to estimate θ, which is unknown. In particular, let $\hat{\theta}(x_1, x_2, \cdots, x_n)$ be a function of $x_1, x_2, \cdots, x_n$ which we use as a (point) estimate of θ; the corresponding function $\hat{\theta}(X_1, X_2, \cdots, X_n)$ of the random variables (rvs) $X_1, X_2, \cdots, X_n$, which is iteself a rv, is an **estimator** for θ.

In any situation, there will be a variety of possible estimators, though some may be more obvious than others, and we need some way of choosing between them. Here we look at a number of desirable properties which we might like estimators to possess - unbiasedness, consistency, efficiency, and sufficiency. These might be named 'classical' properties of estimators.

E.1 Unbiasedness

definition E.1.1. *$\hat{\theta}$ is an unbiased estimator for θ if $E[\hat{\theta}] = \theta$; otherwise it is biased. The bias of $\hat{\theta}$ is defined to be $bias(\hat{\theta}) = E[\hat{\theta}] - \theta$.*

Intuitively this means that the distribution of $\hat{\theta}$ is centered at θ, and there is no persistent tendency to under or overestimate θ.

E.2 Consistency

Although some bias may be acceptable in an estimator, we would like the bias to tend to 0 as the sample size, n, tends to ∞. In addition we would like the variance to tend to 0 as n tends to ∞. These requirements are related to the idea of consistency.

definition E.2.1. *An estimator $\hat{\theta}$ for θ is (weakly) consistent if $Pr[|\hat{\theta} - \theta|] \to 0$ as $n \to \infty$, that is, the pdf of $\hat{\theta}$ becomes increasingly concentrated around θ for large n.*

Strong consistency corresponds to convergence with probability 1.

E.3 Efficiency

Using unbiasedness and consistency may still leave a (possibly infinite) number of candidate estimators. How can we choose between them (if we feel that is is necessary to make a choice)? One fairly strategy is to try to minimize variance, and since it does not make sense to compare estimators with different biases with respect to variance alone, we only compare estimators with the same bias. Further, to keep things simple, the study is usually restricted to unbiased estimators, and the goal becomes looking for minimum variance unbiased estimators (MVUEs).

The words 'efficient' and 'efficiency', when applied to estimators, refer to the variances of the estimators. The lower the variance of an unbiased estimator, the more efficient it is.

definition E.3.1. *An unbiased estimator is said to be efficient if it has the minimum possible variance; the efficiency of an unbiased estimator is the ratio of the minimum possible variance to the variance of the estimator.*

The relative efficiency of two (unbiased) estimators is the reciprocal of the ratio of their variances.

Since efficiencies may vary with sample sizes, the asymptotic efficiencies and asymptotic relative efficiencies (as $n \to \infty$) are often used as one-and-for-all measures.

E.4 Sufficiency

definition E.4.1. *As usual, suppose that X_1, X_2, $\cdots$, X_n form a random sample from $f(x;\theta)$. Suppose further that $t(x_1, x_2, \cdots, x_n)$ is a function of the observation $x_1, x_2, \cdots, x_n$, and not of θ and that $T(X_1, X_2, \cdots, X_n)$ is the corresponding random variable. T is then a statistic, and T is sufficient for θ - a sufficient statistic for θ - if the conditional distribution of $X_1, X_2, \cdots, X_n$, given the value of T, does not depend on θ.*

APPENDIX F

Gabor Functions and Wavelets

A two dimensional Gabor function $g(x, y)$ and its Fourier transform $G(u, v)$ can be written :

$$g(x,y) = (\frac{1}{2\pi\sigma_x\sigma_y})\exp[-\frac{1}{2}(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}) + 2\pi jWx] \tag{F.0.1}$$

$$G(u,v) = \exp\{-\frac{1}{2}[\frac{(u-W)^2}{\sigma_u^2} + \frac{v^2}{\sigma_v^2}]\} \tag{F.0.2}$$

where $\sigma_u = \frac{1}{2\pi\sigma_x}$ and $\sigma_v = \frac{1}{2\pi\sigma_y}$. Gabor functions form a complete but nonorthogonal basis set. Expanding a signal using this basis provides a localized frequency description. A class of self-similar functions, referred to as *Gabor wavelets* in the following discussion, is now considered. Let $g(x, y)$ be the mother Gabor wavelet, then this self-similar filter dictionary can be obtained by appropriate dilations and rotations of $g(x, y)$ through the generating function:

$$g_{mn}(x,y) = a^{-m}G(x',y'), a > 1, m, n = \text{integer} \tag{F.0.3}$$

$$x' = a^{-m}(x\cos(\theta) + y\sin(\theta)), \text{and} \tag{F.0.4}$$

$$y' = a^{-m}(-x\sin(\theta) + y\cos(\theta)), \tag{F.0.5}$$

where $\theta = \frac{n\pi}{K}$ and K is the total number of orientations. The scale factor a^{-m} [86] is used to ensure that the energy is independent of m.

The nonorthogonality of the Gabor wavelets implies that there is redundant information un the filtered images, and the following strategy is used to reduce this redundancy. Let U_l and U_h denote the lower and upper center frequencies of interest. Let K be the number of orientations and S be the number of scales in the mutliresolution decomposition. Then the design strategy is to ensure that the half-peak magnitude support the filter responses in the frequency spectrum touch each other [86]. This results in the following formulas for computing the filter parameters σ_u and

σ_v (and thus σ_x and σ_y).

$$a = (\frac{U_h}{U_l})^{\frac{1}{S-1}}, \tag{F.0.6}$$

$$\sigma_u = \frac{(a-1)U_h}{(a+1)\sqrt{2\log 2}}, \tag{F.0.7}$$

$$\sigma_v = \tan(\frac{\pi}{2k})[U_h - 2\log(\frac{\sigma_u^2}{U_h})][2\log 2 - \frac{(2\log 2)^2\sigma_u^2}{U_h^2}]^{-\frac{1}{2}}, \tag{F.0.8}$$

where $W = U_h$ and $m = 0, 1, ..., S-1$.

APPENDIX G

Acronyms

HMM	Hidden Markov Model
CHMM	Continuous Hidden Markov Model
DHMM	Discrete Hidden Markov Model
FHMM	Factorial Hidden Markov Model
MSHMM	Multi-stream Hidden Markov Model
MSDHMM	Multi-stream Discrete Hidden Markov Model
MSCHMM	Multi-stream Continuous Hidden Markov Model
MSDHMM^{D}	Distance based Multi-stream Discrete Hidden Markov Model
MSDHMM^{P}	Probability based Multi-stream Discrete Hidden Markov Model
$\text{MSDHMM}^{\text{P}_\text{l}}$	Linear Probability based Multi-stream Discrete Hidden Markov Model
$\text{MSDHMM}^{\text{P}_\text{g}}$	Geometric Probability based Multi-stream Discrete Hidden Markov Model
MSCHMM^{G}	Multi-stream Geometric Continuous Density Hidden Markov Model
$\text{MSCHMM}^{\text{G}_\text{m}}$	Mixture level Multi-stream Geometric Continuous Density Hidden Markov Model
$\text{MSCHMM}^{\text{G}_\text{s}}$	State level Multi-stream Geometric Continuous Density Hidden Markov Model
MSCHMM^{L}	Multi-stream Linear Continuous Density Hidden Markov Model
$\text{MSCHMM}^{\text{L}_\text{m}}$	Mixture level Multi-stream Linear Continuous Density Hidden Markov Model
$\text{MSCHMM}^{\text{L}_\text{s}}$	State level Multi-stream Linear Continuous Density Hidden Markov Model
MFCC	Mel-frequency cepstral coefficients
LPC	Linear predictive coding
pdf	probability density function
MLE	Maximum Likelihood Estimator/Estimation
MCE	Minimum Classification Error
GPD	Gradient Probabilistic Descent
SCAD	Simultaneous Clustering and Attribute Discrimination
EHD	Edge Histogram Descriptor

APPENDIX H

Notations

Pr	the probability of an event it is also used to note a probability mass function (in the case of discrete random variable) or a probability density function (in the continuous case)
C	the number of classes
c	the index of a given class among the C classes
λ	compact notation of an HMM model
π	initial state probabilities
$\mathbf{A}$	probability transition matrix
$\mathbf{B}$	observation probability matrix
$\mathbf{W}$	stream relevance weight matrix
$\mathbf{V}$	the set of symbols or codebook in the case of DHMM
p	Dimension of the data
N_s	number of states
s_i	represents the ith state
M	number of symbols/gaussian mixtures
v_j	the jth symbol of the codebook $\mathbf{V}$
L	number of data generating streams
i	index of the state s_i
j	index of a symbol/gaussian mixture component
k	index of a generating stream
a_{ij}	the transition probability from state i to state j
b_{ij}	the probability of an observation v_j given a state i
T_r	sequence length, it might vary from a sequence to another.
t	index of time along a sequence
O	an observation sequence
o_t	an observation vector at time t in sequence O
$\mathbb{O}$	The training data consisting of a set of sequences
R	Number of sequences in the training data
r	index of an observation sequence in the training data
Q	the state sequence correspondent to each observation sequence O
q_t	the state generating the observation o_t
E	the sequence of the Gaussian mixture components correspondent to each observation sequence O
e_t	The Gaussian mixture component generating the observation o_t
F	the sequence of stream indicies correspondent to each observation sequence O
f_t	The relevant feature subset for each observation o_t
$\mathbb{Q}(.,.)$	objective function for MLE
$\mathbb{L}$	objective function for MCE training
τ	iteration number of the MCE training

CURRICULUM VITAE

NAME: Oualid Missaoui

ADDRESS: Department of Computer Engineering and Computer Science
University of Louisville
Louisville, KY 40292

EDUCATION:

Ph.D., Computer Science and Engineering,
May 2010
University of Louisville, Louisville, Kentucky

M.Sc., Applied Mathematics
with Highest Honors, November 2005
Polytechnic School of Tunisia, Ariana, Tunisia

B.S., Signals and Systems
with Highest Honors, June 2003
Polytechnic School of Tunisia, Ariana, Tunisia

PUBLICATIONS:

1. **O. Missaoui**, H. Frigui, and Paul Gader "Landmine detection with Multi-stream Discrete Hidden Markov Models" IEEE Transactions on Geoscience and Remote Sensing Submitted Jan 09.
2. A. Hamdi, **O. Missaoui** and H. Frigui, "An SVM classifier with HMM-based Kernel for Landmine detection using Ground Penetrating Radar", IEEE International Geoscience and Remote Sensing Symposium, 2010.
3. **O. Missaoui**, H. Frigui, and P. Gader "Model level fusion

of Edge Histogram descriptors and Gabor wavelets for landmine detection with Ground Penetrating Radar", IEEE International Geoscience and Remote Sensing Symposium, 2010.

4. **O. Missaoui**, H. Frigui, and P. Gader, "Discriminative Multi-Stream Discrete HMM with MCE/GPD", International Conference on Machine Learning and Applications, ICMLA, 2009.
5. A. Fadeev, **O. Missaoui**, and H. Frigui, "Dominant Audio Descriptors for Audio Classification and Retrieval", International Conference on Machine Learning and Applications, ICMLA, 2009.
6. A. Fadeev, **O. Missaoui**, and H. Frigui, "Ensemble Possibilistic K-NN for Functional Clustering of Gene Expression profiles in Human Cancers Challenge", International Conference on Machine Learning and Applications, ICMLA, 2009.
7. H. Frigui, A. Hamdi, **O. Missaoui**, and P. Gader, "Landmine Detection using Mixture of Discrete Hidden Markov Models", SPIE, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIV, 2009.
8. **O. Missaoui** and H. Frigui, "Optimal Feature Weighting for Discrete Hidden Markov Models" , International Conference of Pattern Recognition, ICPR, 2008.
9. **O. Missaoui** and H. Frigui, "Optimal Feature Weighting for Continuous Hidden Markov Models" , International Conference of Pattern Recognition, ICPR, 2008.
9. H. Frigui, **O. Missaoui** and P. Gader, "Landmine detection with ground penetrating radar using discrete hidden Markov models with symbol dependent features", SPIE, Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XIII, 2008.
10. H. Frigui, **O. Missaoui** and P. Gader, "Landmine detection using discrete hidden Markov models with Gabor features", SPIE, Detection and Remediation Technologies for Mines and Minelike Targets XII, 2007.
11. Y. Lahbib, **O. Missaoui** and al., "Verification flow optimization using an automatic coverage driven

testing policy", International Conference on Design and Test of Integrated Systems in Nanoscale Technology, DTIS 2006.

HONORS AND AWARDS:

- University of Louisville travel awards in 2008 and 2009.
- Tunisian national scholarship for engineering studies in Polytechnic School of Tunisia (2000-2003).
- Promoted to job grade 12 using the HAY evaluation method within STMicroelectronics on March 2005.

MEMBERSHIPS:

- Institute of Electrical and Electronics Engineers (2006-present)
- Society of Photographic Instrumentation Engineers. (2009-present)