

© 2014 Daniel Joseph Soberal

FACE RECOGNITION USING HIDDEN MARKOV MODEL SUPERVECTORS

BY

DANIEL JOSEPH SOBERAL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Mark Hasegawa-Johnson

ABSTRACT

This project attempts to boost the results of face recognition algorithms already established to perform face recognition by augmenting the architecture and using HMM-based supervector classification. In this thesis, the work of Tang's 2010 dissertation is used such that the HMM based classifier takes on a UBM-MAP adaptation based approach. In addition, Tang's work is extended to the case of pseudo 2-dimensional HMMs. Thus, a supervector classifier for pseudo 2DHMMs is developed and then applied to the task of face recognition. When the recognition algorithm is applied to the ORL database, the results show that the algorithm is able to either perform as well as other face recognition algorithms applied to this database, or actually outperform them.

To my parents, for their love and support.

ACKNOWLEDGMENTS

I'd like to thank my advisor, Dr. Hasegawa-Johnson, for his patience with me while I learned the skills needed to develop this algorithm, and for his invaluable advice during the course of this research project.

TABLE OF CONTENTS

LIST OF TABLES	vii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	x
LIST OF SYMBOLS	xi
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 GENERAL SYSTEM DESIGN	4
CHAPTER 3 FEATURE GENERATION AND IMAGE SEGMENTATION	6
3.1 Image Segmentation using Cascaded Classifiers	6
3.1.1 Haar-like, Integral Features	6
3.1.2 Adaboost	7
3.1.3 Cascaded Classifier Boosted Training	7
3.2 Scale Invariant Feature Transform	8
3.2.1 Construction of Scale Space	8
3.2.2 Extrema Search and Filtering	9
3.2.3 Orientation and Descriptor	10
3.3 Implementation	11
3.3.1 Facial Feature Segmentation using Cascaded Classifiers	11
3.3.2 Dense SIFT Features	11
CHAPTER 4 LEARNING THE UNIVERSAL BACKGROUND MODEL (UBM)	13
4.1 Introduction to the Universal Background Model	13
4.2 Gaussian Mixture Models	13
4.3 Hidden Markov Models	20
4.4 Pseudo Two-Dimensional Hidden Markov Models	27
4.5 Training the Universal Background Model (Implementation)	33
4.5.1 1D Hidden Markov Model	34
4.5.2 P2DHMM	34
CHAPTER 5 MAXIMUM A POSTERIORI (MAP) ADAPTATION	35
5.1 Maximum A Posteriori (MAP) Adaptation	35
5.2 MAP Adaptation Algorithm for a 1D Hidden Markov Model	36
5.2.1 MAP Estimation of State Transition Probabilities for a Hidden Markov Model	37
5.2.2 MAP Adaptation for the Emission Parameters	37

CHAPTER 6	SUPERVECTORS	39
6.1	Introduction: The Idea of Supervectors	39
6.2	Kullback-Leibler Divergence	39
6.2.1	KLD for a Multivariate Gaussian Distribution	40
6.2.2	KLD for a GMM	40
6.2.3	KLD for a 1DHMM	42
6.3	Supervectors	44
6.3.1	KLD Between Two PMFs and Resulting Supervector	44
6.3.2	GMM Supervectors	45
6.3.3	1DHMM Supervectors	47
CHAPTER 7	SEGMENTAL MAP ADAPTATION, KULLBACK-LEIBLER DIVERGENCE, AND SUPERVECTOR DERIVATIONS FOR PSEUDO-2D HIDDEN MARKOV MODELS	49
7.1	Segmental MAP Adaptation of P2DHMMs	49
7.2	Kullback Leibler Divergence Between P2DHMMs	50
7.2.1	KLD for Converted P2DHMM	50
7.2.2	KLD for Embedded P2DHMM	53
7.3	P2DHMM Supervectors	54
7.3.1	P2HMM Supervectors Approach 1 - Conversion to 1D HMM	54
7.3.2	P2HMM Supervectors Approach 2 - Embedded (Superstate) Approach	54
CHAPTER 8	FINAL RESULTS	57
8.1	Data	57
8.2	MAP Adapted HMM Maximum Likelihood Recognizer	57
8.2.1	1D HMM	57
8.2.2	P2DHMM Decomposition	58
8.2.3	P2DHMM Embedded	60
8.3	Verification of KLD Bounds	60
8.4	Supervector Recognition using Nearest-Neighbor Classifier	62
8.4.1	1D HMM	63
8.4.2	P2DHMM Decomposition	63
8.4.3	P2DHMM Embedded	66
8.5	Recognition Results when the Number of Subjects used to Train the UBM is Varied	66
8.6	Recognition Results when Number of Images per Subject used for MAP Adaptation is Varied	67
CHAPTER 9	CONCLUSIONS AND FUTURE WORK	69
REFERENCES		70

LIST OF TABLES

1.1	Summary of Face Recognition Results on ORL Database	2
8.1	ORL Database Relevant Information	57
8.2	3 Superstates, 2 Sub-States per Superstate	59
8.3	3 Superstates, 3 Sub-States per Superstate	59
8.4	3 Superstates, 4 Sub-States per Superstate	59
8.5	5 Superstates, 2 Sub-States per Superstate	59
8.6	5 Superstates, 3 Sub-States per Superstate	60
8.7	5 Superstates, 4 Sub-States per Superstate	60
8.8	3 Superstates, 2 Sub-States per Superstate	64
8.9	3 Superstates, 3 Sub-States per Superstate	64
8.10	3 Superstates, 4 Sub-States per Superstate	65
8.11	5 Superstates, 2 Sub-States per Superstate	65
8.12	5 Superstates, 3 Sub-States per Superstate	65
8.13	5 Superstates, 4 Sub-States per Superstate	65

LIST OF FIGURES

2.1	Generic Recognition Diagram	4
2.2	Raster - Scanned Uniform Block Segmentation	5
2.3	System Generation Block Diagram	5
3.1	Haar-like Image Kernels	6
3.2	Binary Cascade Classifier	8
3.3	Scale Space Visualization	9
3.4	Local Extrema Detection	9
3.5	Segmentation using Cascaded Classifier	11
3.6	Dense SIFT Segmentation	11
4.1	A Left-to-Right 5-State Hidden Markov Model	21
4.2	Sequence of Operations for Computing $\xi_t(i, j)$	24
4.3	Possible Transition Stencil	28
4.4	Example Left-to-Right Pseudo 2D HMM with $N_s = 3$ Superstates and 3 Substates per Superstate	28
4.5	Reduction of P2DHMM to 1DHMM	32
5.1	Training Data and Overall Gaussian Distribution	35
5.2	New Training Data and Adapted Gaussian Distributions, (left) Adapted to Magenta Data, (center) Adapted to Red Data, (right) Adapted to Green Data	36
6.1	$y_1 = \ln x$ vs. $y_2 = x - 1$ Near $x = 1$	45
7.1	Reduction of P2DHMM to 1DHMM	50
7.2	Normed Difference between $\pi^T A^n$ and $\pi^T A^{n-1}$ across Iterations	51
7.3	Convergence Curve for ν showing $\ \mathbf{r}^T \mathbf{A}^n - \mathbf{r}^T \mathbf{A}^{n-1}\ _2^2$	52
7.4	Relative Error between $\pi^T A^n$ and $r^T A^n$ across Iterations	52
7.5	Example Left-to-Right Pseudo 2D HMM with $N_s = 3$ Superstates and 3 Substates per Superstate	53
8.1	Accuracy for Max-LL Recognition when τ and K are Varied	58
8.2	KLD Results for 1DHMM	61
8.3	KLD Results for Converted P2DHMM	62
8.4	KLD Results for Embedded P2DHMM	62
8.5	Accuracy for Supervector Approach when τ and K are Varied	63
8.6	Conditions where Accuracy of Supervector Approach was 100% or Supervector Approach Outperformed Likelihood Approach (Red = True, Blue = False) for 1DHMM Case	64
8.7	Recognition Accuracy as a Function of Number of Components used for UBM Training (1DHMM Case)	66
8.8	Recognition Accuracy as a Function of Number of Components used for UBM Training (Converted PDHMM Case)	67

8.9	Recognition Accuracy as a Function of Number of Components used for UBM Training (1DHMM Case)	68
8.10	Recognition Accuracy as a Function of Number of Components used for UBM Training (Converted P2DHMM Case)	68

LIST OF ABBREVIATIONS

HMM	Hidden Markov Model
1DHMM	1-dimensional Hidden Markov Model
P2DHMM	Pseudo-2-dimensional Hidden Markov Model
GMM	Gaussian Mixture Model
EM	Expectation-Maximization
STM	State Transition Matrix
ML	Maximum Likelihood
MAP	Maximum a Posteriori
PCA	Principal Component Analysis

LIST OF SYMBOLS

λ	Hidden Markov Model (HMM)
θ	Gaussian Mixture Model (GMM)
$\boldsymbol{\mu}$	Mean vector
$\boldsymbol{\Sigma}$	Covariance matrix
$\boldsymbol{\pi}$	HMM initial probabilities
$\mathbf{A}$	HMM State Transition Matrix (STM)
$\mathbf{B}$	HMM Emission Matrix
$\mathbf{O}$	Observation space
$\mathbf{o}_t$	Observation vector at time t

CHAPTER 1

INTRODUCTION

Face recognition has become a prevalent task in machine learning and computer vision over the past decade. Recognition algorithms in general have become standard in many consumer electronic products, especially in cell phones. While a great deal of work has been done already on this task, this thesis attempts to bring together several existing concepts for a new application towards face recognition. There are many common approaches to face recognition widely seen in academic journals, university courses, and industry applications. A common recognition algorithm is the eigenface approach. This algorithm will be used as a baseline comparison metric in this thesis. Some approaches use clustering methods, but these will not generally be discussed in this thesis. Finally, some approaches use HMM based algorithms - either 1-dimensional HMMs (1DHMMs) or pseudo 2-dimensional HMMs (P2DHMMs). Since HMMs form the core of the set of algorithms used in this thesis, it is worth discussing previous work that has been done in face recognition using HMM-based techniques.

One of the most common face recognition algorithms is the eigenface approach [1], which can be used for both representing face images efficiently and for extracting relevant information from them. The eigenface algorithm uses principal component analysis (PCA) to create a set of eigenface vectors which can be sent to a standard classifier (nearest neighbor, neural network, etc.) to complete the recognition task. As an aside, eigenfaces can also be used for classifying whether or not a given image is a face (in other words, face detection), by detecting which image patches lie closest to the subspace spanned by the eigenfaces [2]. In any case, as a motivating example, this algorithm was used as a comparison benchmark by Samaria in his thesis [3]. Samaria claims a 90% correct classification rate (in other words, a 10% error rate) using the eigenface features with a nearest neighbor classifier on the ORL database [4]. The eigenface algorithm is relatively easy to understand and implement, and it can give good recognition results for simple databases. Often, it is used as a feature space for a more complicated classifier (such as a neural net) or for a probabilistic model (such as a Gaussian mixture model). However, other approaches are often used for more difficult tasks which require more robust recognition (the variation between the test and training images is high, there are significant lighting differences in the images, etc).

Hidden Markov models (HMMs) have also been used for the face recognition task. Nefian and Hayes [5] have used 1DHMMs with a feature space using discrete cosine transform (DCT) coefficients. Using a maximum likelihood classifier, they report a recognition rate of 84% on the ORL database. Using wavelet-based features, Bicego and Murino [6] reportedly achieved 100% accuracy on the same database. Using MAP adaptation, Liu and Chen [7] report a 1.2% error rate on images from the Mobo database, and a 4% error rate on images cropped from the Task database. Most of these algorithms focus on wavelet-based and DCT-based features.

Another approach is to use embedded or pseudo 2-dimensional HMMs (P2DHMMs). P2DHMMs are

Table 1.1: Summary of Face Recognition Results on ORL Database

Name	Year	Method	Accuracy
Samaria [3]	1994	Eigenface	90%
Samaria [3]	1994	P2DHMM	94.5%
Othman et al. [9]	2003	P2DHMM	91%
Nefian et al. [10]	1999	P2DHMM	98%
Nefian et al. [10]	1998	1DHMM	84%
Bevilacqua et al. [11]	2007	P2DHMM	100%
Ballot [12]	2005	P2DHMM	100%
Bicego et al. [6]	2003	1DHMM	100%
Geng et al. [13]	2009	SIFT	90%
Geng et al. [13]	2009	PDSIFT	95.5%

very common in the existing literature for image recognition tasks. For example, Kuo and Agazzi [8] used them for spotting keywords in text with some success. For face recognition tasks, many researchers used P2DHMMs in the late 1990s and early 2000s. Othman et al. [9] used P2DHMMs with various architectures to achieve error rates in the range of about 9-12% with the ORL database. Samaria [3] used a P2DHMM architecture on the ORL database and reported a 94.5% accuracy rate. Nefian and Hayes [10] amended their previous 1DHMM model to a P2DHMM and report a 98% correct detection rate (again, on the ORL database with a feature spaced composed of DCT coefficients). Bevilacqua et al. [11] used coefficients generated from a neural network to train a P2DHMM, then applied a maximum likelihood recognizer. They report a recognition rate of 100% on the ORL database. Ballot [12] applied a similar P2DHMM approach to get a recognition rate of 100% on the ORL database, and was also able to get a recognition rate of 93.31% on the XM2VTS database. The above results are summarized in Table 1.1.

With the relevant literature results now established, it is useful to summarize common techniques and approaches used in these and other papers. Most of the literature read during the development of this algorithm showed that most researchers used DCT based coefficients, Gabor wavelets, or PCA vectors for generating their feature spaces. In addition, the recognition results vary greatly depending on what sort of pre-processing is done to the image, and on what sort of environmental impairments (pose, illumination, etc.) are present in the database. In addition, while some literature exists on face recognition using 1DHMMs, most of the literature uses some variation on 2DHMMs. Also, the states generally correspond to a geometric interpretation of the face. Finally, while a few articles in the literature cite MAP adaptation, most do not seem to utilize it.

The above results helped establish the outline of the face recognition algorithm used in this thesis. As a departure from the apparent tradition of using simple DCT features, SIFT (Scale Invariant Feature Transform) vectors were used to create the feature space used to train the HMMs. The idea behind this was that since SIFT vectors are invariant to changes in scale and changes in rotation, they create a more robust description of the face in question, and the features themselves can be used in detection tasks [14]. Also, Geng et al. [13] showed that an error rate of 10% was achievable using just SIFT with no further processing as a recognition scheme, while an error rate of 4.5% was achievable using their own variation of SIFT (PDSIFT). In addition, a universal background model (UBM) was trained over a general subset of the faces in the database to describe faces in general. This model was in turn adapted (using MAP adaptation) to each subject in the database to create a distinct model for each subject. While the idea of having a model for each subject (or in some cases every image in the database) is common to every approach, MAP

adaptation appears to be used in very few publications (relative to the number of publications that don't use it). While MAP adaptation is more commonly used in speech recognition and speaker recognition, the same techniques can be applied to face recognition since the concepts inherent in the two tasks can be compared analogously. Finally, while a maximum likelihood classifier is explored in this thesis, the idea of creating supervectors to create a direct classifier between HMMs is also explored. Hao Tang [15] did a great deal of similar work in his own dissertation. Indeed, his MAP-adapted HMM approach with SIFT vectors informed much of this work. Tang used the technique established above on 1DHMMs for emotion classification using LDA classifiers. This work differs in that the supervector derivations Tang established for 1DHMMs are established to P2DHMMs and then applied to the task of face recognition.

The remainder of this thesis is organized as follows. In chapter 2, the general algorithm design described briefly above is described in much more detail. The database parameters, system block diagram, and general algorithm flow are all shown.

In Chapter 3, the general techniques for pre-processing the image and generating the feature space are examined. It is proposed that the feature space be composed of dense SIFT vectors generated over a decimated grid in the image. In addition, for the purposes of initializing certain parameters for the HMM training algorithms, a cascaded classifier based face detection algorithm is used to segment the image into regions corresponding to regions of interest in the image (forehead, eyes, nose, mouth, chin).

In Chapter 4, the concept of the universal background model (UBM) and how it is trained for both a 1DHMM and a P2DHMM is established. This chapter also establishes much of the underlying theoretical background needed to understand the algorithms used in this thesis, such as what GMMs and HMMs are, and how they can be trained and evaluated.

In Chapter 5, the theory behind MAP adaptation is introduced. In addition, the algorithms used for adapting HMMs are established in this chapter.

In Chapter 6, the concept of supervectors is introduced. In addition, the derivations for the KLD between GMMs and 1DHMMs is shown, as are Tang's [15] derivations for the 1DHMM and GMM supervectors.

In Chapter 7, the more original contributions of this thesis are introduced. Specifically, a more relaxed version of MAP adaptation for P2DHMMs is introduced, and the derivations for the KLD bounds and supervector equations for P2DHMMs are shown.

Finally, Chapter 8 shows the recognition results for the algorithm described in this thesis, and discusses the meaning behind the results.

CHAPTER 2

GENERAL SYSTEM DESIGN

In general, a recognition system can be described as a system with three distinct phases (see Figure 2.1). The first phase is the feature generation phase, where the features that will be used for learning the model are generated from the raw signals. In the case of face recognition, the raw signals are images. The second phase is the system generation phase. In this phase, some sort of model is learned over the feature space. Sometimes this phase is skipped (as is the case for the eigenface system). The third and final phase is the classification phase.

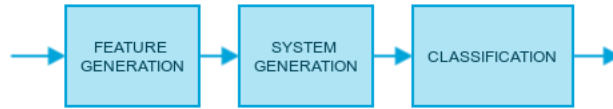


Figure 2.1: Generic Recognition Diagram

The recognition system devised for face recognition in this thesis is composed of several components.

1. Feature Generation
2. Universal Background Model Learning
3. MAP Adaptation
4. Supervector Generation
5. Classification

At the feature generation phase, every input image in the database is segmented into blocks, and dense SIFT [16] is performed in order to create a single SIFT (scale invariant feature transform) vector per block [14]. These vectors are then raster-scanned into a single matrix. An example that illustrates this general idea is shown in Figure 2.2.

In the context of the Figure 2.1, The system generation phase of the algorithm refers to the training of the universal background model (UBM) using a maximum-likelihood training algorithm, and then adapting this model to each subject using maximum a posteriori estimation (MAP adaptation). This approach is widely used in speaker recognition, and here it is adapted to face recognition. This is the same approach used by Hao Tang in his dissertation [15], although he applied it towards facial expression recognition. Again following Hao Tang's work, the HMMs for each subject are used to create a supervector space that can then be used in classification. This is shown in Figure 2.3.



Figure 2.2: Raster - Scanned Uniform Block Segmentation

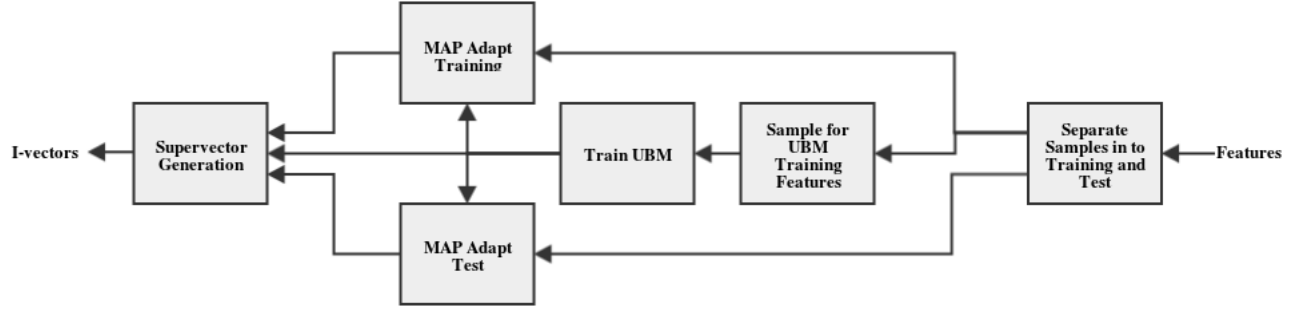


Figure 2.3: System Generation Block Diagram

For the purposes of this thesis, the UBM is trained as either a 1-dimensional HMM (1DHMM) or a pseudo 2-dimensional HMM (2DHMM). In either case, the HMMs have continuous emission models that are generated from Gaussian mixture models (GMMs).

The final stage, classification, is the stage where the subject is classified to a subject in the dataset. In general, there are P classes such that each class L_p exists in the space $L = \{L_1, \dots, L_P\}$. Thus, the classifier assigns the dataset to one of these P classes based on the evaluation of the model in the system generation phase.

CHAPTER 3

FEATURE GENERATION AND IMAGE SEGMENTATION

This chapter focuses on the techniques required to preprocess the image data and generate the features space.

3.1 Image Segmentation using Cascaded Classifiers

3.1.1 Haar-like, Integral Features

The Haar-like Integral image features form the feature space that will be used in the facial detection task. Essentially, this process involves applying a series of rectangular kernels to an image that weight the pixels by some value (in this case, either +1 or -1) and then adding the resulting weighted pixels in the area of the kernel. These kernels are shown in Figure 3.1, in which one can see that the kernels are split evenly into un-rotated kernels and kernels rotated by 45 degrees. This means that two new images are created from the original image (denoted $O(x, y)$) - the integral image (denoted $I_u(x, y)$) and the rotated integral image (denoted $I_r(x, y)$).

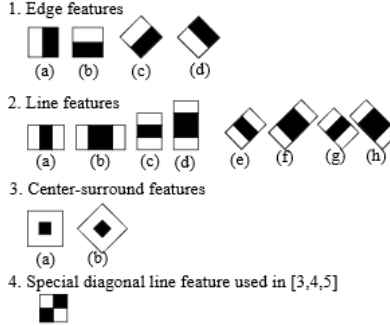


Figure 3.1: Haar-like Image Kernels

$$I_k(x, y) = \sum_{(i,j) \in \text{kernel}} w_{ij} O(i, j), \quad k \in \{u, r\} \quad (3.1)$$

Thus, every pixel within this region is set to the value of the “integral,” thus subdividing the image into a concatenation of blocks. If the image is compressed into an array of these block values, the dimensionality is reduced, allowing for fast feature generation and subsequent learning.

3.1.2 Adaboost

Adaboost is a boosting technique designed to train a series of simple, weak classifiers. Given a set of observation vectors $\mathbf{x}_t \in \mathbf{X}$ and a set of binary classification outputs $y_t \in \{-1, 1\}$, each classifier is assigned a weight denoted α_i . Furthermore, each observation vector is assigned a T-dimensional cost distribution that will assign a cost to each mis-classified observation in the final error computation. Thus, the algorithm is designed such that the classifiers that mis-classify tokens will have their weights updated so that they become weaker and the strong positive classifiers are strengthened. This process is iterated over time until a final strong set of classifiers is output. The overall algorithm is given below.

Adaboost

Given M classifiers $f_m(\mathbf{x}_t)$ with weight vector $\mathbf{w}$ and activation function $g()$ (can be linear).

$$f_m(\mathbf{x}_t) = g(\mathbf{w}_m^T \mathbf{x}_t) \quad (3.2)$$

Compute the expected error according to some loss function, $D_m(t)$.

$$\epsilon_m = \sum_t D_m(t) u(|y_t - f_m(\mathbf{x}_t)|), \quad u(t) = \text{unit step} \quad (3.3)$$

Assign a voting weight α_m .

$$\alpha_m = \ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right) \quad (3.4)$$

Update the weights.

$$\mathbf{w}_m = \mathbf{w}_m \odot e^{\alpha_m u(|y_t - f_m(\mathbf{x}_t)|)} \quad (3.5)$$

Output the classifier.

$$y_t = \text{sign} \left(\sum_m \alpha_m f_m(\mathbf{x}_t) \right) \quad (3.6)$$

3.1.3 Cascaded Classifier Boosted Training

The classifier is trained off-line using a series of positive images (images with a face in it) and negative images (images without a face, i.e., all such background images). The training algorithm is based on the face detection classifier used in Viola and Jones [17]. For each image in the set of observations (i.e., the positive and negative images), a feature space is created using the Haar-like integral images. Next, a sequence of N binary nodes (classify image as face, or do not classify image as face) are used to create cascaded classifier. This is illustrated in Figure 3.2.

Thus, the integral image feature must produce a positive at each classifier to claim that the region contains a face. The weights of each node in the cascade classifier are trained using the Adaboost algorithm.

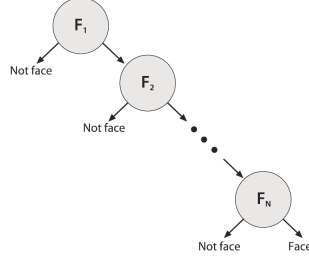


Figure 3.2: Binary Cascade Classifier

3.2 Scale Invariant Feature Transform

The scale invariant feature transform (SIFT) algorithm, developed by David Lowe [14], is an algorithm used to generate distinctive keypoints in an image which are invariant to changes in scale and rotation (as the name suggests). The algorithm is proprietary, but it is widely used in computer vision and image processing to generate distinctive image features or to generate keypoints which are mappable between transformed images (among other applications). The algorithm itself is highly complex and involves several steps, which will be briefly summarized below.

3.2.1 Construction of Scale Space

The first thing to do is create a series of keypoints in the image. This is done by creating a scale space of subsequent Laplacian filtered images. The Laplacian is a useful kernel because it can be used to detect characteristic regions (“blobs”) in the image. The Laplacian definition is awkward to implement (from a computational perspective) so it is approximated using a difference of Gaussians instead.

Given an image with pixels $i(x, y) \in \mathbf{I}$, we first convolve it with Gaussian blurring kernels.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.7)$$

$$\begin{aligned} G_d(\omega_x, \omega_y, \sigma) &= \sum_x \sum_y \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} e^{-j\omega_x x} e^{-j\omega_y y} \\ &= e^{-\frac{(\omega_x^2 + \omega_y^2)\sigma^2}{2}} \end{aligned} \quad (3.8)$$

$$I_d(\omega_x, \omega_y) = \sum_x \sum_y I(x, y) e^{-j\omega_x x} e^{-j\omega_y y} \quad (3.9)$$

$$L(x, y, \sigma) = \frac{\int_{\omega_x} \int_{\omega_y} I_d(\omega_x, \omega_y) G_d(\omega_x, \omega_y, \sigma) e^{j(\omega_y y + \omega_x x)} d\omega_x d\omega_y}{4\pi^2} \quad (3.10)$$

These images are blurred with increasing σ to form an octave. Then, the image is downsampled and blurred again, thus creating another octave. This process is repeated for N octaves and M scales. Next, the difference between subsequent blurred images in each octave is evaluated to create the scale-space.

$$D(x, y, \sigma) = L_j(x, y, k\sigma) - L_j(x, y, \sigma) \quad (3.11)$$

This is shown graphically in Figure 3.3.

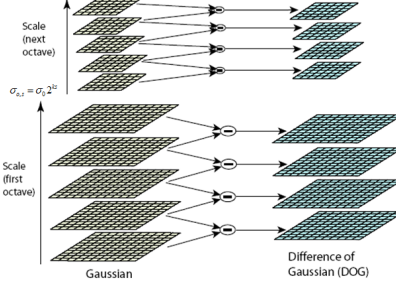


Figure 3.3: Scale Space Visualization

3.2.2 Extrema Search and Filtering

The next logical step is to find the maxima that occur across the scale-space. These maxima indicate the indices in the original image where the keypoints should be. In practice, this is done using a broad search where a $9 + 8 + 9 = 26$ cube grid is used at each pixel. If the pixel in this grid is the maximum, then it is marked as a keypoint. This is shown in Figure 3.4.

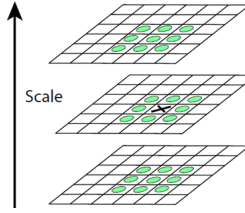


Figure 3.4: Local Extrema Detection

Next, the resulting grid must be refined to a subset of more distinctive features (otherwise the resulting set of keypoints would be far too dense). The first thing that is done is to interpolate the returned candidate keypoints from the search. This is essentially done through interpolation using the gradient. The derivative at a keypoint (x, y, σ) is approximated through a Taylor expansion as:

$$D(\mathbf{x}) = D(x, y, \sigma) + \left(\frac{\partial D}{\partial \mathbf{x}} \right)^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \left(\frac{\partial^2 D}{\partial \mathbf{x}^2} \right) \mathbf{x} \quad (3.12)$$

$$\mathbf{x} = [x \ y \ \sigma]^T \quad (3.13)$$

The vector $\mathbf{x}$ in this case represents an offset from the initial candidate keypoint. The derivative is evaluated such that the result is a function of the offset $\mathbf{x}$. Then, this equation is set equal to 0 and solved for $\mathbf{x}$ in order to find the maximum. If the result in any dimension of $\mathbf{x}$ is more than 0.5, then this implies that the true local maximum is closer to another keypoint. Thus, the current keypoint is discarded and the new keypoint is given by:

$$\hat{\mathbf{x}} = \left(\frac{\partial^2 D}{\partial \mathbf{x}^2} \right)^{-1} \left(\frac{\partial D}{\partial \mathbf{x}} \right) \quad (3.14)$$

The other part of this step is to discard keypoints that have a low contrast. If derivative $D(\mathbf{x})$ evaluated at the solution offset $\hat{\mathbf{x}}$ is less than 0.03 in any direction, then it is discarded.

$$|D(\hat{\mathbf{x}})| < 0.03 \quad (3.15)$$

Finally, edges are filtered out, since edges will always appear as keypoints in the algorithm as it stands. This is done through evaluation of the Hessian matrix at each remaining keypoint.

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad (3.16)$$

The following equation is used as a thresholding function for the keypoints. In the equation, r is essentially a representation of the ratio of the eigenvalues of the Hessian. The eigenvalues of the Hessian are proportional to the principal curvatures of the difference of Gaussians. We seek to reject the keypoint if this ratio of these eigenvalues is very large. Formally, we reject the keypoint if the following condition is met:

$$\frac{\text{Tr}(\mathbf{H})^2}{\det(\mathbf{H})} < \frac{(r+1)^2}{r} \quad (3.17)$$

The parameter r for the thresholding is typically set to 10.

3.2.3 Orientation and Descriptor

Finally, it is time to assign an orientation to each remaining keypoint, as well as return a vector of descriptors for each keypoint.

The Gaussian-smoothed pixel value at scale σ is evaluated at each keypoint. The gradient is approximated at each keypoint according to the following equations.

$$|\nabla L| = \sqrt{(L(x+1, y, \sigma) - L(x-1, y, \sigma))^2 + (L(x, y+1, \sigma) - L(x, y-1, \sigma))^2} \quad (3.18)$$

$$\angle \nabla L = \tan^{-1} \frac{L(x, y+1, \sigma) - L(x, y-1, \sigma)}{L(x+1, y, \sigma) - L(x-1, y, \sigma)} \quad (3.19)$$

Finally, a set of orientation histograms are created on 4×4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16×16 region around the keypoint such that each histogram contains samples from a 4×4 subregion of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. This will yield a descriptor vector with 128 elements, which is then normalized. The set of these descriptor vectors for each image will form the feature space for the image recognition task.

3.3 Implementation

3.3.1 Facial Feature Segmentation using Cascaded Classifiers

The face images can be pre-segmented using the Viola and Jones cascaded classifier. A set of training images for each facial feature (chin, mouth, nose, eyes, forehead) can be pre-segmented manually and then used to train the cascaded classifier. While the training algorithm may take a great deal of time to train, it only has to be trained once and the evaluation of the classifier is far less time-intensive. As an example, the segmentation borders found using the cascaded classifier for a face image from the ORL database are shown below in Figure 3.5.

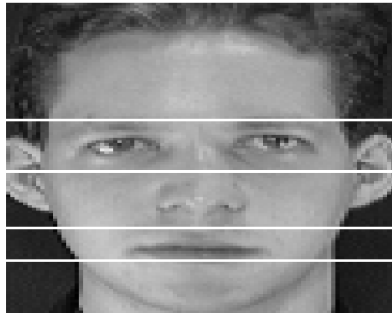


Figure 3.5: Segmentation using Cascaded Classifier

At this point the dense SIFT algorithm can be run on each segment of the image. The resulting SIFT vectors can be used to train a Gaussian Mixture Model for each segment.

3.3.2 Dense SIFT Features

To generate the SIFT vectors, the VLFeat [16] library was used. The dense sift features are generated in a representative grid, as shown in Figure 3.6, in which every line intersection represents a keypoint generating a particular SIFT vector.

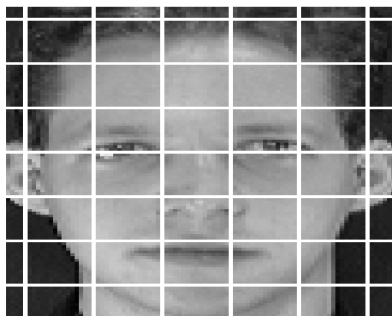


Figure 3.6: Dense SIFT Segmentation

The SIFT vectors from each block are extracted, and the blocks are raster scanned in order to get a sequence of feature vectors that have a sequential dependence. These features are used to train the UBM, and then later to MAP adapt the UBM to the training sequences for each subject.

CHAPTER 4

LEARNING THE UNIVERSAL BACKGROUND MODEL (UBM)

This chapter focuses on the techniques needed to learn the universal background model. In this case, the universal background model is a hidden Markov model (HMM) whose states have continuous emission models. These emission models are given by Gaussian mixture models (GMMs). In order to understand how such a model is trained, a basic understanding of these two models is required. In the context of the problem at hand, a GMM is trained over the extracted SIFT features from each segment of the face image (forehead, eyes, nose, mouth, chin). These GMMs are then used as the initial estimates for the emission distributions of the HMM, which is trained thereafter. What follows is an examination of the basic concepts of GMMs and HMMs, as well as the methods used to train them.

4.1 Introduction to the Universal Background Model

A universal background model (UBM) is used to represent subject-independent features that can be compared with subject-dependent features when testing a hypothesis [18]. In this thesis, the UBM is used for training a subject-specific model by acting as the prior model in MAP adaptation. In general, the UBM is trained over a select subgroup of the data representative of the data you want to ultimately classify, while making sure not to bias the training pool against sub-populations in the database (i.e., only having males in the database when you want to classify for either gender). For the purposes of face recognition in this thesis, a single subject-independent UBM is trained across samples (generally, only one sample from each subject is taken, but this could be changed) from every subject in the database. The idea here is to generate a model that represents the transitions between facial features in general among the faces in the database. This base model is then adapted to every speaker in the next phase of the algorithm.

4.2 Gaussian Mixture Models

Gaussian mixture models (GMMs) are used for clustering data or for creating a representative parametric probability distribution for a set of data. For the purposes of this thesis, GMMs are used as underlying probability models for specific states in an HMM (to be discussed in the next section). An understanding of what GMMs are and how they are trained is required to understand their use in this thesis.

A mixture model can be generally defined as an overall probabilistic model that takes into account the presence of sub-models. The GMM is a weighted combination of K conditional Gaussian probability density functions. Each of these Gaussian pdf's has a weight, a mean vector, and a covariance matrix that describes it. These quantities form the parameter set for the GMM as shown in Equation (4.1).

$$\theta = \{w_k, \Sigma_k, \mu_k\} \quad 1 \leq k \leq K \quad (4.1)$$

In Equation (4.1), the scalar w_m is a weight that corresponds to how much the m -th Gaussian contributes to the overall mixture, the vector μ_m is the mean of the m -th Gaussian, and the matrix Σ_m is the covariance matrix of the m -th Gaussian. The vector μ_m is D-dimensional, and the matrix Σ_m is DxD-dimensional.

The GMM is trained over a set of T observation vectors in a set $\mathcal{X}$, each of which is denoted $\mathbf{x}_t$. This is shown in Equation (4.2).

$$\mathbf{x}_t \in \mathcal{X}, \quad 1 \leq t \leq T \quad (4.2)$$

Each vector $\mathbf{x}_t$ is assumed to be D-dimensional. It is further assumed that these data vectors are independent and identically distributed (IID) with distribution p .

The probability density function that models the likelihood that a vector $\mathbf{x}_t \in \mathcal{X}$ belongs to a particular Gaussian pdf is given by Equation (4.3).¹

$$p(\mathbf{x}_t | \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} \exp \left[(\mathbf{x}_t - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_t - \mu_k) \right] \quad (4.3)$$

In practice, the evaluation of this pdf can yield values that are very close to 0. So close, in fact, that they reach the underflow level of the floating point arithmetic system being used to calculate it. To combat this, the logarithm of the above equation is generally used instead. This makes numbers that are very close to 0 become large negative numbers (although not so large that they cannot be represented). Using the log-probabilities in place of the regular probabilities can make the subsequent training calculations more difficult to implement (especially in languages which rely on vectorization), but the accuracy gained by avoiding underflow is often necessary. The log-form of the above equation is given below.

$$\ln(p(\mathbf{x}_t | \mu_k, \Sigma_k)) = -\frac{D}{2} \ln(2\pi) - \frac{1}{2} \ln(|\Sigma_k|) - \frac{1}{2} (\mathbf{x}_t - \mu_k)^T \Sigma_k^{-1} (\mathbf{x}_t - \mu_k) \quad (4.4)$$

The evaluation of $a = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$ is the computation of most concern, and also the most potentially expensive. Matrix multiplication of a matrix which is $n \times m$ and another which is $m \times p$ (the result of which is an $n \times p$ matrix) is $\mathcal{O}(mnp)$. For multiplying two square matrices of dimension n , it is clear that this is $\mathcal{O}(n^3)$. Matrix inversion is generally $\mathcal{O}(n^3)$. For LU decomposition, the net cost would be on the order of $\frac{2}{3}\mathcal{O}(n^3)$. However, the covariance matrix is symmetric positive definite (SPD), so Cholesky decomposition can be applied instead for half the cost of LU decomposition. Using Cholesky decomposition, the covariance matrix can be written as Equation (4.5).

$$\Sigma = \mathbf{L}\mathbf{L}^T \quad (4.5)$$

The matrix $\mathbf{L}$ is lower triangular. Thus, the argument can be re-written as Equation (4.6).

¹In this equation, $|\mathbf{X}| = \det(\mathbf{X})$, ie, the determinant of matrix $\mathbf{X}$.

$$\begin{aligned}
a &= (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= (\mathbf{x} - \boldsymbol{\mu})^T (\mathbf{L}\mathbf{L}^T)^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= (\mathbf{x} - \boldsymbol{\mu})^T (\mathbf{L}^T)^{-1} \mathbf{L}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= (\mathbf{L}^{-1} (\mathbf{x} - \boldsymbol{\mu}))^T \mathbf{L}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \\
&= \mathbf{y}^T \mathbf{y}, \quad \mathbf{y} = \mathbf{L}^{-1} (\mathbf{x} - \boldsymbol{\mu})
\end{aligned} \tag{4.6}$$

Again, finding the decomposition is order $\frac{1}{3}\mathcal{O}(n^3)$. The equation for y can be re-written as $\mathbf{L}\mathbf{y} = \mathbf{x} - \boldsymbol{\mu}$. Solving this equation for $\mathbf{y}$ is an order $\mathcal{O}(n^2)$ operation because back-substitution can be used. Thus, the net cost is of the same order as Cholesky decomposition itself. In addition, this makes the evaluation of the determinant incredibly easy.

$$\begin{aligned}
\det(\boldsymbol{\Sigma}) &= |\boldsymbol{\Sigma}| \\
&= |\mathbf{L}\mathbf{L}^T| \\
&= |\mathbf{L}||\mathbf{L}^T| \\
&= |\mathbf{L}|^2, \quad (\text{because of symmetry}) \\
&= \left(\prod_{j=1}^D l_{jj} \right)^2, \quad (\text{because of matrix triangularity})
\end{aligned} \tag{4.7}$$

This can be evaluated in the log-domain as Equation (4.8)

$$\ln |\boldsymbol{\Sigma}| = 2 \sum_{j=1}^D \ln(l_{jj}) \tag{4.8}$$

Thus, the log-likelihood of the vector given the distribution can be computed as Equation (4.9).

$$\ln(p_{\mathbf{X}}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})) = -\frac{D}{2} \ln(2\pi) - \sum_{j=1}^D \ln(l_{jj}) - \frac{1}{2} \mathbf{y}^T \mathbf{y} \tag{4.9}$$

The exact same result can be obtained using an upper triangular matrix $\mathbf{U}$ in the Cholesky decomposition. In this case, the solution operator is given as Equation (4.10).

$$\mathbf{y} = (\mathbf{U}^T)^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{4.10}$$

The probability of a particular observation given the total GMM is given by the equation below.

$$p(\mathbf{x}_t|\theta) = \sum_{k=1}^K w_k p(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4.11}$$

With this, the probability of the set of observations can be written as Equation (4.12).

$$p(\mathcal{X}|\theta) = \prod_{t=1}^T p(\mathbf{x}_t|\theta) \quad (4.12)$$

The log-likelihood is applied by taking the logarithm of both sides of this equation. The final result is shown in (4.13).

$$\begin{aligned} L(\theta; \mathcal{X}) &= \ln(p(\mathcal{X}|\theta)) \\ &= \ln\left(\prod_{t=1}^T p(\mathbf{x}_t|\theta)\right) \\ &= \sum_{t=1}^T \ln(p(\mathbf{x}_t|\theta)) \\ &= \sum_{t=1}^T \ln\left(\sum_{k=1}^K w_k p(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) \end{aligned} \quad (4.13)$$

At this point, it is useful to define the “logsum” operation. Consider that the above equation involves taking the logarithm of a summation. The logsum operator allows a shorthand for the former and further increases the accuracy of the above computation. Consider a generalized problem, that of computing $\ln \sum_t \exp(x_t)$.

$$\begin{aligned} \log \text{sum}_t x_t &= \ln \sum_t \exp(x_t) \\ &= \ln \sum_t \exp(x_t - \max_t x_t + \max_t x_t) \\ &= \ln \sum_t \exp(x_t - \max_t x_t) \exp(\max_t x_t) \\ &= \ln \left(\exp(\max_t x_t) \sum_t \exp(x_t - \max_t x_t) \right) \\ &= \ln \left(\exp(\max_t x_t) \right) + \ln \sum_t \exp(x_t - \max_t x_t) \\ &= \max_t x_t + \ln \sum_t \exp(x_t - \max_t x_t) \end{aligned} \quad (4.14)$$

In the above equation, the maximum of the vector is subtracted from each element in the vector. This makes the exponent of the result larger than it otherwise would have been, which makes the result less prone to underflow. Using this new operator, the log-likelihood can be redefined as Equation (4.15).

$$\ln(p(\mathcal{X}|\theta)) = \sum_{t=1}^T \log \text{sum}_k [w_k p(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)] \quad (4.15)$$

At this point, the question becomes how to train the parameters that comprise θ in such a way that the log-likelihood is maximized. This can be accomplished using Baum-Welch training

(expectation-maximization).

In expectation-maximization (EM), the goal is to maximize the log-likelihood the model with some known data given that there are hidden (or latent) variables that exist. In other words, for a single observation $\mathbf{x}_t \in \mathcal{X}$, there is also an associated hidden variable $\mathbf{z}_t \in \mathcal{Z}$. Each of these variables corresponds to a random variable, denoted $\mathbf{O}_t$ and $\mathbf{Q}_t$, such that the observed random variables belong to space $\mathcal{O}$, while the hidden random variables belong to space $\mathcal{Q}$.

The goal here is to attempt to find the optimal model parameters θ^* that maximize the log-likelihood of the observation space given the model.

$$\begin{aligned}\theta^* &= \arg \max_{\theta} [\ln p(\mathcal{O}|\theta)] \\ &= \arg \max_{\theta} \left[\ln \left[\int p(\mathcal{O}, \mathcal{Q}|\theta) d\mathcal{Q} \right] \right]\end{aligned}\tag{4.16}$$

Optimizing such a function is difficult. However, a useful fact can be used to find an auxiliary function which is easier to work with. In general, a θ needs to be found which makes the gradient of $\ln \left[\int p(\mathcal{O}, \mathcal{Q}|\theta) d\mathcal{Q} \right]$ zero in order to find the optimal θ^* . It can be shown that the auxiliary function which makes such a thing true is given by the equation below.

$$\nabla_{\theta} \ln \left[\int p(\mathcal{O}, \mathcal{Q}|\theta) d\mathcal{Q} \right] = 0 \quad \text{if} \quad \nabla_{\theta} Q(\theta, \theta) = \nabla_{\theta} \int \ln [p(\mathcal{O}, \mathcal{Q}|\theta)] p(\mathcal{Q}|\mathcal{O}, \theta) d\mathcal{Q} = 0\tag{4.17}$$

The above equations represent the underlying idea behind the expectation-maximization algorithm. This idea is to find an auxiliary function $Q(\theta_{i+1}, \theta_i)$ that maximizes the expectation of the log-likelihood of the entire space given the model, and iteratively maximize this expectation until convergence is reached at the true solution (i.e., until the optimal parameters are found or, more likely, until the algorithm has converged).

$$Q(\theta_{i+1}, \theta_i) = \int \ln [p(\mathcal{O}, \mathcal{Q}|\theta_{i+1})] p(\mathcal{Q}|\mathcal{O}, \theta_i) d\mathcal{Q}\tag{4.18}$$

This can also be written as Equation (4.19):

$$Q(\theta_{i+1}, \theta_i) = E_{\mathcal{Q}} [\ln p(\mathcal{O}, \mathcal{Q}|\theta_{i+1}) | \mathcal{O}, \theta_i]\tag{4.19}$$

It can be shown that the algorithm converges by proving the following equation:

$$Q(\theta_{i+1}, \theta_i) - Q(\theta_i, \theta_i) \leq \frac{p(\mathcal{O}|\theta_{i+1})}{p(\mathcal{O}|\theta_i)} - 1\tag{4.20}$$

In the case of a Gaussian mixture model, the latent variables correspond to a signal which determines which component Gaussian each observed datum belongs to, while the observed variables correspond to individual observation vectors. Thus, they both form finite sets of T discrete variables with $q_t \in [1, K]$. Therefore, the auxiliary function can be written as Equation (4.21).

$$\begin{aligned}
Q(\theta_{i+1}, \theta_i) &= \sum_{t=1}^T E_{q_t} [\ln p(\mathbf{x}_t, q_t | \theta_{i+1}) | \mathbf{x}_t, \theta_i] \\
&= \sum_{t=1}^T \sum_{k=1}^K p(q_t | \mathbf{x}_t, \theta_i) \ln (p(\mathbf{x}_t, q_t | \theta_{i+1}) p(q_t | \theta_{i+1})) \\
&= \sum_{t=1}^T \sum_{k=1}^K \delta_{tk} \ln (p(\mathbf{x}_t, q_t | \theta_{i+1}) p(q_t | \theta_{i+1})), \quad \delta_{tk} = p(q_t | \mathbf{x}_t, \theta_i) \\
&= \sum_{t=1}^T \sum_{k=1}^K \delta_{tk} \ln (w_k \varphi_k(\mathbf{x}_t)), \quad \varphi_k(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \quad \sum_{k=1}^K w_k = 1
\end{aligned} \tag{4.21}$$

Using Bayes' rule, the quantity δ_{tk} can be expressed using Equation (4.22).

$$\delta_{tk} = \frac{w_k \varphi_k(\mathbf{x}_t)}{\sum_{j=1}^K w_j \varphi_j(\mathbf{x}_t)} \tag{4.22}$$

Thus, the expectation-maximization problem for GMMs becomes a constrained optimization problem that can be solved using the method of Lagrange multipliers. It can be shown that the solution to this problem is given by the following equations [19]:

$$w_k = \frac{1}{T} \sum_{t=1}^T \delta_{tk} \tag{4.23}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{t=1}^T \mathbf{x}_t \delta_{tk}}{\sum_{t=1}^T \delta_{tk}} \tag{4.24}$$

$$\boldsymbol{\Sigma}_k = \frac{\sum_{t=1}^T \delta_{tk} (\mathbf{x}_t - \boldsymbol{\mu}_k)(\mathbf{x}_t - \boldsymbol{\mu}_k)^T}{\sum_{t=1}^T \delta_{tk}} \tag{4.25}$$

Thus, the algorithm for learning a Gaussian mixture model is given by Algorithm 1.

EM inherently converges to (at worst) a local optimum. In other words, convergence to a global optimum is not guaranteed. Indeed, EM is highly sensitive to the choices made for the initial model parameters $\theta_0 = \{\boldsymbol{\mu}_k^0, w_k^0, \boldsymbol{\Sigma}_k^0\}$. A good way to initialize the model with no prior knowledge about it is to initialize the means using the K-means centroid learning algorithm, assume the weights are uniform, and calculate the initial covariance matrices by clustering all the data into one mean and estimating the covariance matrix using the found means [20]. The K-means algorithm is summarized in Algorithm 2. The equations for the initial weights and covariance matrices are given as follows:²

$$w_k^0 = \frac{1}{K} \quad 1 \leq k \leq K \tag{4.26}$$

$$\zeta_t^0 = \arg \min_{1 \leq j \leq K} \|\mathbf{x}_t - \boldsymbol{\mu}_j^0\|_2 \tag{4.27}$$

²Here, $\delta(x)$ represents the Dirac delta function.

Algorithm 1 Expectation Maximization for Gaussian Mixture Model Training

```
function EM_GMM( $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$ ,  $\mathbf{W}_0 = [w_1, \dots, w_K]$ ,  $\mathbf{M}_0 = [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K]$ ,  $\mathbf{S}_0 = [\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_K]$ , MaxIter, Threshold )  
     $i = 0$   
     $L_i = -\infty$   
     $\mathbf{W} = \mathbf{W}_0$   
     $\mathbf{M} = \mathbf{M}_0$   
     $\mathbf{S} = \mathbf{S}_0$   
     $i = i + 1$   
    while  $i < \text{MaxIter}$  and  $\Delta L < \text{Threshold}$  do  
        for  $t = 1$  to  $T$  do  
            for  $k = 1$  to  $K$  do  
                Compute  $\delta_{tk}$  by Equation (4.22)  
            end for  
        end for  
        for  $k = 1$  to  $K$  do  
            Compute  $w_k$  by Equation (4.23) and assign to  $\mathbf{W}[k]$   
            Compute  $\boldsymbol{\mu}_k$  by Equation (4.24) and assign to  $\mathbf{M}[:, k]$   
            Compute  $\boldsymbol{\Sigma}_k$  by Equation (4.25) and assign to  $\mathbf{S}[:, :, k]$   
        end for  
        Compute  $L_i = \ln p(\mathbf{X}|\theta_i)$  using Equation (4.15)  
         $\Delta L = L_i - L_{i-1}$   
        if  $\Delta L < \text{Threshold}$  then  
            break  
        end if  
         $i = i + 1$   
    end while  
    return  $\mathbf{W}$ ,  $\mathbf{M}$ ,  $\mathbf{S}$   
end function
```

$$\Sigma_k^0 = \frac{\sum_{t=1}^T \delta(\zeta_t^0 - k)(\mathbf{x}_t - \boldsymbol{\mu}_k^0)(\mathbf{x}_t - \boldsymbol{\mu}_k^0)^T}{\sum_{t=1}^T \delta(\zeta_t^0 - k)} \quad (4.28)$$

Algorithm 2 K-means Algorithm

```

function KMEANS( $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T], K, \mathbf{M}_0 = [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K], \text{MaxIter}, \text{Threshold}$ )
     $i = 0$ 
     $L_i = -\infty$ 
     $i = i + 1$ 
    while  $i < \text{MaxIter}$  and  $\Delta L < \text{Threshold}$  do
        for  $t = 1$  to  $T$  do
            for  $k = 1$  to  $K$  do
                 $d[t, k] = \|\mathbf{x}_t - \boldsymbol{\mu}_k\|_2$ 
            end for
             $\zeta[t] = \arg \min_{1 \leq k \leq K} d[t, k]$ 
        end for
        for  $k = 1$  to  $K$  do
             $\boldsymbol{\mu}_k = \mathbf{0}$ 
            for  $t = 1$  to  $T$  do
                if  $\zeta[t] == k$  then
                     $\boldsymbol{\mu}_k = \boldsymbol{\mu}_k + \mathbf{x}_t$ 
                end if
            end for
        end for
         $L_i = 0$ 
        for  $k = 1$  to  $K$  do
             $L_i = L_i + \frac{1}{K} \max_t d[t, k]$ 
        end for
         $\Delta L = L_i - L_{i-1}$ 
        if  $\Delta L < \text{Threshold}$  then
            break
        end if
         $i = i + 1$ 
    end while
    return  $W, M, S$ 
end function

```

4.3 Hidden Markov Models

A hidden Markov model (HMM) is a stochastic model in which a system is modelled as a Markov process with hidden (unobserved) states [21]. Thus, it is used for sequential or spatial data where every observation in the series is assumed to be generated from some hidden state in the model. An HMM can take on many architectures. For example, a left-to-right HMM with 5 states is shown in Figure 4.1.

The above architecture is called left-to-right because every transition involves either staying in the same state or moving to the adjacent state in the chain. As another example, a fully ergodic HMM would have an architecture where every state is connected to every other state.

For a sequence $\mathbf{x}_t \in \mathbf{X}$ for $1 \leq t \leq T$ with a hidden Markov model with N states Q_i for $1 \leq i \leq N$, the hidden Markov model is generally characterized according to Equation (4.29).

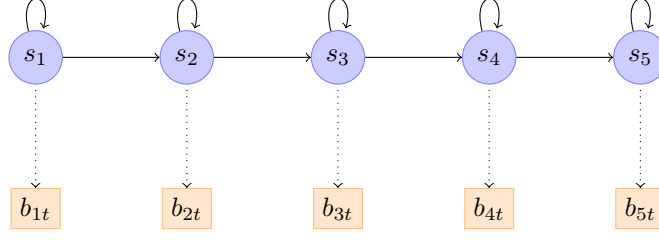


Figure 4.1: A Left-to-Right 5-State Hidden Markov Model

$$\lambda = \{\mathbf{A}, \boldsymbol{\pi}, \mathbf{B}\} \quad (4.29)$$

In the above equation, $\mathbf{A}$ represents the state-transition matrix (STM), which is a matrix that contains the probability of moving from a state i at time t to a state j at time $t + 1$. The vector $\boldsymbol{\pi}$ is a vector of initial probabilities for each state of the HMM. Finally, the $\mathbf{B}$ represents the probability of emitting a certain symbol (or observation) at a certain state at time t (these are often referred to as the emission probabilities).

$$a_{ij} = P(Q_{t+1} = j | Q_t = i), \quad 1 \leq (i, j) \leq N, \quad a_{ij} \in \mathbf{A} \quad (4.30)$$

$$\pi_i = P(Q_1 = i), \quad 1 \leq i \leq N, \quad \pi_i \in \boldsymbol{\pi} \quad (4.31)$$

$$b_j(t) = P(\mathbf{O}_t = \mathbf{x}_t | Q_t = j), \quad 1 \leq j \leq N, \quad b_j(t) \in \mathbf{B} \quad (4.32)$$

Often, the emission probabilities are represented with probability density function (for example, with a GMM with K mixtures).

$$b_j(t) = \sum_{k=1}^K w_{jk} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \quad (4.33)$$

$$b_{jk}(t) = w_{jk} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \quad (4.34)$$

There are three basic problems that are dealt with using HMMs. The first is called the “Evaluation Problem” and deals with finding the probability of a set of observations given a particular HMM λ . The second is called the “Decoding Problem,” and deals with finding which particular state sequence generated a particular set of observations for some model λ . The third problem is called the “Learning Problem,” and deals with learning the parameters of an HMM for some set of observations which maximizes the likelihood of the observations given the model.

The solution to the first problem can be generally computed using Equation (4.35).

$$P(\mathcal{O} | \lambda) = \sum_{\mathbf{q} \in \text{all possible sequences of length } T} \pi_{q_1} b_{q_1}(1) \prod_{t=2}^T b_{q_t}(t) a_{q_t, q_{t+1}}, \quad \mathbf{q} = [q_1, \dots, q_T]^T \quad (4.35)$$

Of course, evaluating the above equation is not really feasible from a purely computational standpoint

since it is far too expensive. The computational complexity of this operation is on the order of $\mathcal{O}(2TN^T)$. To solve this problem, the Forward - Backward algorithm is used. This algorithm uses recursion to determine a set of parameters $\alpha_t(j)$ (the so-called forward probability) and $\beta_t(j)$ (the so-called backward probability) to determine likelihood of the observations given the model. These probabilities are defined in Equations (4.36) and (4.37), respectively.

$$\alpha_t(j) = P(\mathbf{x}_1, \dots, \mathbf{x}_t, q_t = j | \lambda) = b_j(t) \sum_{i=1}^N a_{ij} \alpha_{t-1}(i) \quad (4.36)$$

$$\beta_t(j) = P(\mathbf{x}_{t+1}, \dots, \mathbf{x}_T | q_t = j, \lambda) = \sum_{i=1}^N a_{ji} b_i(t+1) \beta_{t+1}(i) \quad (4.37)$$

The algorithms themselves can be defined according to Algorithms 3 and 4.

Algorithm 3 Forward Algorithm

```

function FORWARD( $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$ ,  $\mathbf{A}$ ,  $\boldsymbol{\pi}$ ,  $\mathbf{B}$ ,  $N$ ,  $T$  )
  for  $j = 1$  to  $N$  do
     $\alpha_1(j) = \pi_j b_j(1)$ 
  end for
  for  $t = 2$  to  $T$  do
    for  $j = 1$  to  $N$  do
       $\alpha_t(j) = 0$ 
      for  $i = 1$  to  $N$  do
         $\alpha_t(j) = \alpha_t(j) + a_{ij} \alpha_{t-1}(i)$ 
      end for
       $\alpha_t(j) = \alpha_t(j) b_j(t)$ 
    end for
  end for
  return  $\alpha$ 
end function

```

Algorithm 4 Backward Algorithm

```

function BACKWARD( $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$ ,  $\mathbf{A}$ ,  $\boldsymbol{\pi}$ ,  $\mathbf{B}$ ,  $N$ ,  $T$  )
  for  $j = 1$  to  $N$  do
     $\beta_T(j) = 1$ 
  end for
  for  $t = T - 1$  to  $1$  do
    for  $j = 1$  to  $N$  do
       $\beta_t(j) = 0$ 
      for  $i = 1$  to  $N$  do
         $\beta_t(j) = \beta_t(j) + a_{ji} \beta_{t+1}(i) b_i(t+1)$ 
      end for
    end for
  end for
  return  $\beta$ 
end function

```

The computational cost of both algorithms is on the order of $\mathcal{O}(N^2T)$, while the memory cost of both algorithms is on the order of $\mathcal{O}(NT)$.

The likelihood of the observation space given the model can be computed using both the forward and backward probabilities. However, in light of the assumption for $\beta_T(i)$ in the backward algorithm, it can be simplified so that only the forward probabilities are used.

$$P(\mathcal{O}|\lambda) = \sum_{i=1}^N \alpha_t(i) \beta_t(i) = \sum_{i=1}^N \alpha_T(i) \beta_T(i) = \sum_{i=1}^N \alpha_T(i) \quad (4.38)$$

Intuitively, the solution to the second problem involves finding the state q_t at time t that maximizes the probability $P(q_t = i|\mathcal{O}, \lambda)$. This probability can be written using Equation (4.39).

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(\mathcal{O}|\lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \quad (4.39)$$

However, the real solution should take into account the sequence dependence among the states. Indeed, the true solution should attempt to maximize $P(q_1, q_2, \dots, q_t = i, \mathbf{x}_1, \dots, \mathbf{x}_t|\lambda)$. In other words, it should attempt to maximize the probability that a particular *sequence* occurred in accordance with a given set of observations given the current model. This is where the Viterbi algorithm comes into play. The Viterbi algorithm attempts to maximize the above probability using recursion.

$$\delta_t(i) = \max_{q_1, \dots, q_{t-1}} P(q_1, \dots, q_t = i, \mathbf{x}_1, \dots, \mathbf{x}_t|\lambda) \quad (4.40)$$

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] b_j(\mathbf{x}_{t+1}) \quad (4.41)$$

$$\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}] \quad (4.42)$$

The Viterbi algorithm can also be used to solve the Evaluation Problem (Problem 1).

$$P(\mathcal{O}|\lambda) = \max_i \delta_T(i) \quad (4.43)$$

The Viterbi algorithm is summarized in Algorithm 5.

The solution to the Learning Problem involves finding the optimal model parameters of λ in order to maximize $P(\mathcal{O}|\lambda)$. In other words, the solution to the learning problem would involve the following:

$$\lambda^* = \arg \max_{\lambda} P(\mathcal{O}|\lambda) \quad (4.44)$$

$$\nabla_{\lambda} P(\mathcal{O}|\lambda)|_{\lambda=\lambda^*} = 0 \quad (4.45)$$

The standard solution involves using Baum-Welch expectation-maximization to iteratively find a local optimum such that the above equations are true. In order to do this, the joint probability of being in state i at time t and state j at time $t+1$ must be defined. The sequence dependence for this calculation is shown in Figure 4.2, and the computation itself is shown in Equation (4.46).

$$\xi_t(i, j) = \frac{a_{ij} \alpha_t(i) \beta_{t+1}(j) b_j(\mathbf{x}_{t+1})}{\sum_{i=1}^N \sum_{j=1}^N a_{ij} \alpha_t(i) \beta_{t+1}(j) b_j(\mathbf{x}_{t+1})} \quad (4.46)$$

Similarly, the parameter $\gamma_t(i)$ can be redefined for this problem.

Algorithm 5 Viterbi Algorithm

```
function VITERBI( $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$ ,  $\mathbf{A}$ ,  $\boldsymbol{\pi}$ ,  $\mathbf{B}$ ,  $N$ ,  $T$ )  
  for  $i = 1$  to  $N$  do  
     $\delta_1(i) = \pi_i b_i(\mathbf{x}_1)$   
     $\psi_1(i) = 1$   
  end for  
  for  $t = 2$  to  $T$  do  
    for  $j = 1$  to  $N$  do  
       $d = 0$   
      for  $i = 1$  to  $N$  do  
         $d(i) = \delta_{t-1}(i) a_{ij}$   
      end for  
       $\delta_t(j) = \max_i (d_i) b_j(\mathbf{x}_t)$   
       $\psi_t(j) = \arg \max_i d_i$   
    end for  
  end for  
   $q_T = \arg \max_i \delta_T(i)$   
  for  $t = T - 1$  to  $1$  do  
     $q_t = \psi_{t+1}(q_{t+1})$   
  end for  
  return  $q$   
end function
```

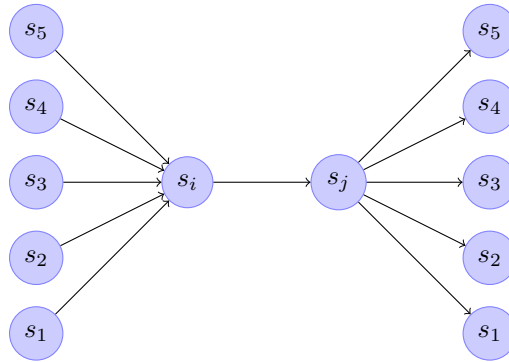


Figure 4.2: Sequence of Operations for Computing $\xi_t(i, j)$

$$\gamma_t(i) = P(q_t = i | \mathcal{O}, \lambda) = \sum_{j=1}^N \xi_t(i, j) \quad (4.47)$$

Since the underlying emission probabilities are assumed to be given by a mixture model, the $\gamma_t(i)$ parameters are augmented to take into account the mixture as well.

$$\gamma_t(j, k) = \gamma_t(j) \times \frac{w_{jk} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})}{\sum_{l=1}^K w_{jl} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jl}, \boldsymbol{\Sigma}_{jl})} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)} \left[\frac{w_{jk} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})}{\sum_{l=1}^K w_{jl} \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_{jl}, \boldsymbol{\Sigma}_{jl})} \right] \quad (4.48)$$

At this point, the update equations can be derived using the same methodology used before for GMMs. That is, calculate the auxiliary function for EM, and then find the optimal solution under some set of constraints using the method of Lagrange multipliers. The hidden variables in this case are the state occupations at each time t in the sequence of observations. The auxiliary function in this case is given by Equation (4.49).

$$Q(\lambda_i, \lambda_{i+1}) = \sum_{\mathbf{q} \in \text{all possible sequences of length } T} P(\mathcal{Q} | \mathcal{O}, \lambda_{i+1}) \ln P(\mathcal{O}, \mathcal{Q} | \lambda_i), \quad \mathbf{q} = [q_1, \dots, q_T]^T \quad (4.49)$$

The solution update equations can be defined according to the equations below (assuming that the emission probabilities are given by GMMs whose parameters also have to be updated at each iteration). In this particular implementation, there is an additional assumption that is made. Before, the equations were all derived under the assumption that there was only one observation sequence that could be used for training. However, in practice, it is both more useful and more realistic to assume that there are multiple training sequences that can be generated from the same model. In this way, the parameters defined above are calculated for each training sequence. Then, the set of all parameters from each training sequence are used to update the parameters. In this case, assume that there are M sequences, each of which have a unique sequence length T_m .

$$a_{ij} = \frac{\sum_{m=1}^M \sum_{t=1}^{T_m} \xi_t(i, j, m)}{\sum_{m=1}^M \sum_{t=1}^{T_m} \sum_{j=1}^N \xi_t(i, j, m)} \quad (4.50)$$

$$\pi_i = \frac{1}{M} \sum_{m=1}^M \sum_{j=1}^N \xi_1(i, j, m) \quad (4.51)$$

$$w_{jk} = \frac{\sum_{m=1}^M \sum_{t=1}^{T_m} \gamma_t(j, k, m)}{\sum_{m=1}^M \sum_{t=1}^{T_m} \sum_{k=1}^K \gamma_t(j, k, m)} \quad (4.52)$$

$$\boldsymbol{\mu}_{jk} = \frac{\sum_{m=1}^M \sum_{t=1}^{T_m} \gamma_t(j, k, m) \mathbf{x}_t^m}{\sum_{m=1}^M \sum_{t=1}^{T_m} \sum_{k=1}^K \gamma_t(j, k, m)} \quad (4.53)$$

$$\boldsymbol{\Sigma}_{jk} = \frac{\sum_{m=1}^M \sum_{t=1}^{T_m} \gamma_t(j, k, m) (\mathbf{x}_t^m - \boldsymbol{\mu}_{jk})(\mathbf{x}_t^m - \boldsymbol{\mu}_{jk})^T}{\sum_{m=1}^M \sum_{t=1}^{T_m} \sum_{k=1}^K \gamma_t(j, k, m)} \quad (4.54)$$

At this point, it is useful to take a step back and consider what has been defined. The quantity

Algorithm 6 EM Learning Algorithm for Hidden Markov Models

```
function EM HMM(X, A,  $\pi$ ,  $W_0$ ,  $\mu_0$ ,  $\Sigma_0$ , N, K, T, MaxIter, Threshold )
     $i = 0$ 
     $L_i = -\infty$ 
     $W = W_0$ 
     $\mu = \mu_0$ 
     $\Sigma = \Sigma_0$ 
     $i = i + 1$ 
    while  $i < \text{MaxIter}$  and  $\Delta L < \text{Threshold}$  do
        for  $t = 1$  to  $T$  do
            for  $j = 1$  to  $N$  do
                for  $k = 1$  to  $K$  do
                    Compute  $b_{jk}(t)$ 
                end for
                Compute  $b_j(t)$ 
            end for
            Compute  $b_j(t)$ 
             $\alpha = \text{forward}(A, \pi, b_j(t))$ 
             $\beta = \text{backward}(A, \pi, b_j(t))$ 
            for  $t = 1$  to  $T$  do
                for  $i = 1$  to  $N$  do
                    for  $j = 1$  to  $N$  do
                        Compute  $\xi_{ij}(t)$ 
                    end for
                    Compute  $\gamma_i(t)$ 
                    for  $k = 1$  to  $K$  do
                        Compute  $\gamma_{ik}(t)$ 
                    end for
                end for
            end for
            Compute  $A$ 
            Compute  $\pi$ 
            for  $j = 1$  to  $N$  do
                for  $k = 1$  to  $K$  do
                    Compute  $w_{jk}$ 
                    Compute  $\mu_{jk}$ 
                    Compute  $\Sigma_{jk}$ 
                end for
            end for
            Compute  $L_i = \ln p(\mathbf{X}|\theta_i)$ 
             $\Delta L = L_i - L_{i-1}$ 
            if  $\Delta L < \text{Threshold}$  then
                break
            end if
             $i = i + 1$ 
        end while
    return  $\lambda$ 
end function
```

$\sum_{m=1}^M \sum_{t=1}^{T_m} \xi_t(i, j, m)$ basically represents the expected number of transitions from state i to state j . Similarly, the quantity $\sum_{m=1}^M \sum_{t=1}^{T_m} \sum_{j=1}^N \xi_t(i, j, m)$ basically represents the expected number of transitions from state i . This sort of reasoning allows the definition of a new algorithm for learning an HMM.

Consider that this sort of reasoning basically implies that all of the above equations represent using the expectation of a particular transition or particular emission occurring. If this is indeed the case, then one could forgo calculating the expectations of these random variables, and instead estimate them using a counting method. This is known as Viterbi learning or segmental K-means. Like EM, such an algorithm is also an iterative algorithm.

This algorithm is not as strong as Baum-Welch. However, it is extremely useful for initializing an HMM before using Baum-Welch, since Baum-Welch's convergence to the correct optimum depends heavily upon the choice of the initial parameters.

4.4 Pseudo Two-Dimensional Hidden Markov Models

One-dimensional hidden Markov models are especially useful for speech recognition. The audio signals are modelled as one-dimensional signals, and then they are broken into frames and an observation vector is generated from each frame using some method (generally Mel-cepstrum coefficients or linear-predictive-coding coefficients). A one-dimensional model can also be used for multi-dimensional data. For example, with images the inherently two-dimensional index map can be flattened to a single dimension and then observation vectors can be generated using a frame-based metric just like with a normal one-dimensional signal. This is one of the approaches that will be used in this thesis. However, one question worth asking is: What happens if the underlying data *isn't* flattened? In other words, what if there is underlying information that is lost when the problem is constrained in a single dimension? To answer this question, the data is assumed to remain two-dimensional and a Pseudo 2-dimensional hidden Markov model (P2DHMM) is used [3].

For the image problem, consider each pixel value as an output in $\mathbb{R}^1$ from an input set in $\mathbb{R}^2$, and assume that each observation vector is generated from the image through a function g such that $\mathbf{o}_{x,y} = g(f(x', y'))$ where (x', y') represent the image indices and (x, y) represent the indices of the lattice of observation vectors. Again, since this is an image, the variables x and y are not assumed to be continuous, but rather are considered to represent pixel indices in a two-dimensional map such that $1 \leq x \leq C$ and $1 \leq y \leq R$.

$$\mathcal{O} = \begin{bmatrix} \mathbf{o}_{1,1} & \mathbf{o}_{2,1} & \dots & \mathbf{o}_{C-1,1} & \mathbf{o}_{C,1} \\ \mathbf{o}_{1,2} & \mathbf{o}_{2,2} & \dots & \mathbf{o}_{C-1,2} & \mathbf{o}_{C,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{o}_{1,R-1} & \mathbf{o}_{2,R-1} & \dots & \mathbf{o}_{C-1,R-1} & \mathbf{o}_{C,R-1} \\ \mathbf{o}_{1,R} & \mathbf{o}_{2,R} & \dots & \mathbf{o}_{C-1,R} & \mathbf{o}_{C,R} \end{bmatrix} \quad (4.55)$$

A fully decoupled 2-dimensional hidden Markov model (2DHMM) would allow a transition between any adjacent observations. In other words, one can consider a stencil in which there are 4 possible movement directions, as shown in Figure 4.3.

However, this will create an NP -hard problem later when it comes to both the conventional learning and decoding problems for a hidden Markov model. A P2DHMM assumes a coupled observation structure that ensures two-dimensional causality at the line-level. It assumes that there is an observation set $\mathcal{O}_y$ that is

$$|S| = \begin{bmatrix} X & X & X \\ X & O & - \\ - & - & - \end{bmatrix}$$

O = current occupation, X = possible transition, $-$ = impossible transition

Figure 4.3: Possible Transition Stencil

composed of R observations $\mathcal{Y}$ that are, in fact, *sequences* of observations such that each $\mathcal{Y}$ is a sequence composed of C observations vectors $\mathbf{o}_{x,y}$.

$$\mathcal{O}_y = \begin{bmatrix} \mathcal{Y}_1 \\ \text{---} \\ \mathcal{Y}_2 \\ \text{---} \\ \vdots \\ \text{---} \\ \mathcal{Y}_{R-1} \\ \text{---} \\ \mathcal{Y}_R \end{bmatrix} = \begin{bmatrix} \mathbf{o}_{1,1} & \mathbf{o}_{2,1} & \dots & \mathbf{o}_{C-1,1} & \mathbf{o}_{C,1} \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ \mathbf{o}_{1,2} & \mathbf{o}_{2,2} & \dots & \mathbf{o}_{C-1,2} & \mathbf{o}_{C,2} \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ \mathbf{o}_{1,R-1} & \mathbf{o}_{2,R-1} & \dots & \mathbf{o}_{C-1,R-1} & \mathbf{o}_{C,R-1} \\ \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ \mathbf{o}_{1,R} & \mathbf{o}_{2,R} & \dots & \mathbf{o}_{C-1,R} & \mathbf{o}_{C,R} \end{bmatrix} \quad (4.56)$$

At this point, the model structure itself can be defined. A P2DHMM has a set of N_S “superstates.” When traversing the observation space, the model can transition from superstate m at “time” y to superstate n at “time” $y + 1$ such that $1 \leq y \leq R$. Thus, the super-state system can be modelled as a one-dimensional HMM. However, within each superstate, the sequence of observations within $\mathcal{Y}_y$ move through *another* one-dimensional HMM, such that each superstate contains a unique one-dimensional HMM that allows transitions from state i at time x to state j at time $x + 1$ such that $1 \leq x \leq C$. In this way, the emission model of each superstate can essentially be considered a separate “sub-HMM”. A graphical model of a P2DHMM is shown in Figure 4.4.

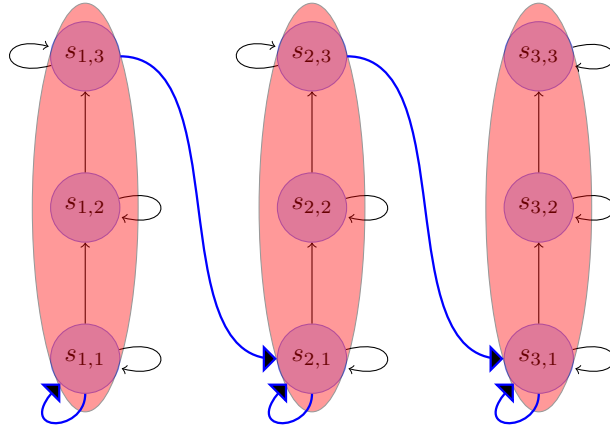


Figure 4.4: Example Left-to-Right Pseudo 2D HMM with $N_s = 3$ Superstates and 3 Substates per Superstate

The parameter set η of the P2DHMM is $\eta = \{\mathbf{\Omega}, \mathbf{\Pi}, \mathbf{\Lambda}\}$ such that $\mathbf{\Omega}$ represents the superstate transition matrix, $\mathbf{\Pi}$ represents the initial probability vector for each superstate, and $\mathbf{\Lambda} = \{\lambda_1, \dots, \lambda_{N_s}\}$ is the set of sub-HMMs for each superstate. Since the $\mathcal{Y}$ sequences are coupled to the $\mathbf{o}_{yx}$ vectors the way defined above, the superstate emissions can be essentially be thought of the probabilities of the individual sub-HMMs defined for each superstate.

$$\omega_{m,n} = P(\hat{s}_{y+1} = n | \hat{s}_y = m), \quad \omega_{m,n} \in \mathbf{\Omega} \quad (4.57)$$

$$\Pi_m = P(\hat{s}_1 = m), \quad \Pi_m \in \mathbf{\Pi} \quad (4.58)$$

$$P(\mathcal{Y}_y | \hat{s}_y = i) = P(\mathbf{o}_{1,y}, \dots, \mathbf{o}_{C,y} | \lambda_i) \quad (4.59)$$

In this case, each λ_i is defined as a continuous HMM with GMM emissions (as in the above section).

At this point, the logical next step is to find the solution to the three standard HMM problems defined in the previous section. First, consider the problem of evaluating the P2DHMM. To solve this problem, the NP hard solution can be determined again.

$$P(\mathcal{O}_y | \eta) = \sum_{\hat{\mathbf{q}} \in \text{all possible sequences of length } R} \Pi_{\hat{q}_1} b_{\hat{q}_1}(1) \prod_{r=2}^R b_{\hat{q}_r}(r) \omega_{\hat{q}_r, \hat{q}_{r+1}}, \quad \hat{\mathbf{q}} = [\hat{q}_1, \dots, \hat{q}_R]^T \quad (4.60)$$

$$b_{\hat{q}_r}(r) = P(\mathcal{Y}_r | \lambda_{\hat{q}_r}) = \sum_{\mathbf{q} \in \text{all possible sequences of length } C} \pi_{q_1} b_{q_1}(1) \prod_{c=2}^C b_{q_c}(c) a_{q_c, q_{c+1}}, \quad \mathbf{q} = [q_1, \dots, q_C]^T \quad (4.61)$$

However, if it is cost-prohibitive to evaluate such an expression for the 1DHMM case, then it will be even more cost-prohibitive for the P2DHMM case. Instead, the analogous forward and backward probabilities will be determined.

$$\alpha_{xy}^m(i) = P(\mathbf{o}_{x,1}, \dots, \mathbf{o}_{x,y}, \hat{s}_y = m, s_x = i | \mathbf{\Lambda}) \quad (4.62)$$

$$\beta_{xy}^m(i) = P(\mathbf{o}_{x,y+1}, \dots, \mathbf{o}_{x,C} | \hat{s}_y = m, s_x = i, \mathbf{\Lambda}) \quad (4.63)$$

Once these have been defined, the superstate forward and backward probabilities can be defined.

$$\alpha_y(m) = P(\mathcal{Y}_1, \dots, \mathcal{Y}_y, \hat{s}_y = m | \mathbf{\Lambda}) \quad (4.64)$$

$$\beta_y(m) = P(\mathcal{Y}_{y+1}, \dots, \mathcal{Y}_C | \hat{s}_y = m, \mathbf{\Lambda}) \quad (4.65)$$

To continue with the analogy to 1DHMMs, the emission probabilities for each superstate can be defined as the probability of each sub-state HMM.

$$b_m(\mathcal{Y}_y) = \sum_{i=1}^{N_m} \alpha_{xC}^m(i) \quad (4.66)$$

Finally, the probability of the P2DHMM can be defined.

$$P(\mathbf{o}_{1,1}, \dots, \mathbf{o}_{R,C} | \eta) = \sum_{m=1}^M \alpha_R(m) \quad (4.67)$$

At this point, it is clear that doing a “double-pass” of the 1D forward algorithm defined previously (first over every sub-state to find the emission probabilities and then over every super-state to define the probability itself) is sufficient to find the probability of the sequence given the model.

Next, the analogous Viterbi decoding algorithm for a P2DHMM can be determined in order to find the solution to the second classical HMM problem.

$$\delta_{x,y}^m(i) = \max_{\hat{q}_1, \dots, \hat{q}_x, q_1, \dots, q_{y-1}} P(\hat{q}_1, \dots, \hat{q}_y = m, q_1, \dots, q_x = i, \mathbf{o}_{1,1}, \dots, \mathbf{o}_{x,y} | \eta) \quad (4.68)$$

Again, it is expedient to break up the evaluation into two passes of the 1D algorithm. For each row of observation vectors, the 1D Viterbi algorithm can be evaluated for each sub-HMM. Then, the 1D Viterbi algorithm can be evaluated over the super-states. This algorithm is broadly outlined below [8].

P2DHMM Viterbi Algorithm

1. Sub-State 1D Viterbi Pass

- (a) Initialize with $\delta_{1y}^m(i) = P(q_1 = i, \mathbf{o}_{1,y} | \lambda_m) = \pi_i^m b_{i1}^m$
- (b) Evaluate the recursion with $\delta_{x,y}^m(j) = b_{jx}^m \max_{1 \leq l \leq N_m} [\delta_{x,y}^m(l) a_{lj}^m]$ and $\psi_{x,y}^m(j) = \arg \max_{1 \leq l \leq N_m} [\delta_{x,y}^m(l) a_{lj}^m]$
- (c) Evaluate $b_{my} = P(\mathcal{Y}_y | \lambda_m) = \max_{1 \leq j \leq N_m} \delta_{C,y}^m(j)$

2. Super-State Viterbi pass

- (a) Initialize with $D_1(m) = P(\hat{q}_1 = j, \mathcal{Y}_1 | \eta) = \Pi_m b_{m1}$
- (b) Evaluate the recursion with $D_y(n) = b_{ny} \max_{1 \leq v \leq N_s} [D_{y-1}(v) \omega_{v,n}]$ and $S_y(n) = \arg \max_{1 \leq v \leq N_s} [D_{y-1}(v) \omega_{v,n}]$
- (c) Evaluate $P(\mathcal{O}_y | \eta) = \max_{1 \leq j \leq N_s} D_R(j)$

3. Embedded Back-Tracking

- (a) Initialize with $\hat{q}_R = \arg \max_{1 \leq i \leq N_s} \delta_R(i)$
- (b) Back-trace along super-states with $\hat{q}_y = S_y(\hat{q}_{y+1})$
- (c) initialize sub-state $q_C = \arg \max_{1 \leq j \leq N_{\hat{q}_y}} \delta_{Cy}(j)$
- (d) Back-trace along sub-states with $q_x = \phi_{x,y}^{\hat{q}_y}(q_{x+1})$

For solving the third problem of training the HMM, a non-traditional approach is utilized. Traditionally,

evaluating the HMM would result in two embedded passes of the Baum-Welch algorithm for training a 1D HMM (but just the expectation step). Then, a modified version of the maximization step using the parameters from both passes would be used to update the P2DHMM parameters. The algorithm proposed in this thesis is an augmented version of segmental K-means (Viterbi learning) extended to the case of a P2DHMM. Essentially, Viterbi for a P2DHMM will be run to determine which rows of observations belong to which superstate. Then, the Baum-Welch algorithm can be run for each sub-state using just the rows that have been allocated to each sub-state HMM. Really, this boils down to CN_s evaluations of 1D Viterbi and RN evaluations of 1D Baum-Welch. This approach was used in Samaria [3], and the general algorithm is outlined below.

Training a P2DHMM

1. Use P2DHMM Viterbi to determine superstate/sub-state sequence
2. Assign each observation to a sub-state i in a superstate m based on the results of Viterbi
3. Train each sub-state HMM λ_m using Baum-Welch over the segmented rows that have been assigned to superstate m
4. Evaluate the new model
5. Exit at convergence

Alternatively, a full Baum-Welch algorithm could be used for this case. Although this was not done in this thesis, the required equations for such an algorithm are given in [22]. The occupation probabilities are given by:

$$\gamma_y(m) = P(\hat{s}_y = \lambda_m | \mathcal{O}_y, \Lambda) = \frac{\alpha_y(m)\beta_y(m)}{\sum_{n=1}^{N_s} \alpha_y(n)\beta_y(n)} \quad (4.69)$$

$$\gamma_{xy}^m(j) = P(\hat{s}_y = m, s_x = j | \mathcal{Y}_y, \lambda_m) = \frac{\alpha_{xy}^m(j)\beta_{xy}^m(j)}{\sum_{l=1}^{N_s} \alpha_{xy}^m(l)\beta_{xy}^m(l)} \quad (4.70)$$

The required update equations would be given by Equation (4.71) through (4.75) for the substate HMMs.

$$\bar{\pi}_j^m = \frac{\sum_{e=1}^E \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \alpha_{1y}^{m(e)}(j) \beta_{1y}^{m(e)}(j)}{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \sum_{l=1}^{N(i)} \alpha_{1y}^{m(e)}(l) \beta_{1y}^{m(e)}(l)} \quad (4.71)$$

$$\bar{a}_{ij}^m = a_{ij}^m \frac{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=1}^{X-1} b_j^{m(e)}(\mathbf{o}_{x,y+1}^e) \alpha_{xy}^{m(e)}(j) \beta_{xy}^{m(e)}(j)}{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=1}^{X-1} \alpha_{xy}^{m(e)}(j) \beta_{xy}^{m(e)}(j)} \quad (4.72)$$

$$\bar{c}_{jk}^m = \frac{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)}{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^{(e)}(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X \sum_{k=1}^K b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)} \quad (4.73)$$

$$\bar{\mu}_{jk}^m = \frac{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^e(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X \mathbf{o}_{xy}^e b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)}{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^e(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X \sum_{k=1}^K b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)} \quad (4.74)$$

$$\bar{\Sigma}_{jk}^m = \frac{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^e(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X (\mathbf{o}_{xy}^e - \boldsymbol{\mu}_{jk}^m)(\mathbf{o}_{xy}^e - \boldsymbol{\mu}_{jk}^m)^T b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)}{\sum_{e=1}^E \sum_{y=1}^Y \frac{\gamma_y^e(m)}{b_m(\mathcal{Y}_y^e)} \sum_{x=2}^X \sum_{k=1}^K b_{ik}^m(\mathbf{o}_{x,y+1}^e) \beta_{xy}^{m(e)}(j) \sum_{l=1}^{N_m} a_{lj}^m \alpha_{x-1,y}^{m(e)}(l)} \quad (4.75)$$

Likewise, the update equations for the superstate HMM are shown in Equations (4.76) through (4.77).

$$\bar{\omega}_{mn} = \omega_{mn} \frac{\sum_{e=1}^E \sum_{y=1}^Y \alpha_y^e(m) b_n(\mathcal{Y}_{y+1}^e) \beta_{y+1}^e(n)}{\sum_{e=1}^E \sum_{y=1}^Y \alpha_y^e(m) \beta_y^e(m)} \quad (4.76)$$

$$\bar{\pi}_m = \frac{\sum_{e=1}^E \alpha_1^e(m) \beta_1^e(m)}{\sum_{e=1}^E \sum_{n=1}^{N_s} \alpha_T^e(n)} \quad (4.77)$$

The last point worth noting is that a P2DHMM can generally be decomposed into a 1D HMM by inserting non-looping, non-emitting states that consume nothing to represent superstate transitions. This allows one to forgo the various complication of running the embedded training algorithm to learn an P2DHMM. An example of the decomposition of the P2DHMM shown in the above figure into a 1DHMM is shown in Figure 4.5.

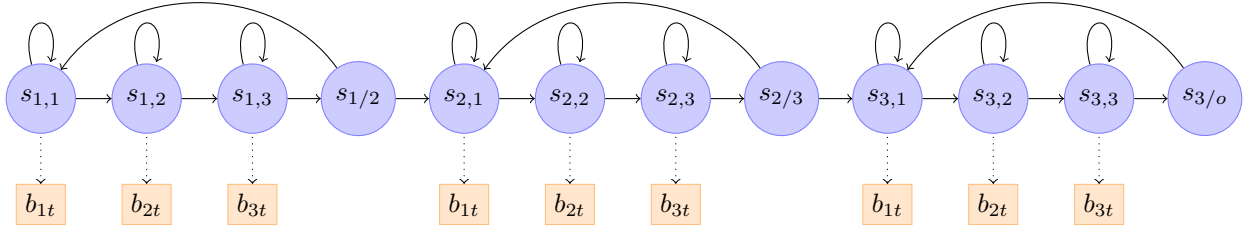


Figure 4.5: Reduction of P2DHMM to 1DHMM

Clearly, although the P2HMM is composed of left-to-right 1DHMMs and can itself be considered a left-to-right 1DHMM with HMM emissions, the conversion process does not result in a net left-to-right 1DHMM. The state transition matrix of the converted HMM is shown in Equation (4.78).

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{4,1} & 0 & 0 & 0 & a_{4,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{5,5} & a_{5,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{6,6} & a_{6,7} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{7,7} & a_{7,8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{8,5} & 0 & 0 & 0 & a_{8,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{9,9} & a_{9,10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,10} & a_{10,11} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11,11} & a_{11,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{12,9} & 0 & 0 & a_{12,12} \end{bmatrix} \quad (4.78)$$

The matrix is still a sparse banded matrix (as long as the number of sub-states in each super-state is constant). However, it is no longer strictly bidiagonal as it was in the 1D left-to-right case. Thus, the 1DHMM formulas already defined above can be used to train the converted P2DHMM.

In this case, it is assumed that there are N_s superstates in the P2DHMM. It is further assumed that the topology of the superstate HMM is left-to-right, and that the sub-state HMMs are also left-to-right HMMs. Finally, it is assumed that all the sub-state HMMs have the same topology (ie, the same number N_m of sub-states per superstate for all $1 \leq m \leq N_s$). For each sub-state HMM, the assumptions on the state transitions and initial state vectors from the previous 1DHMM case are also enforced in this case. In addition, the GMM initial training via image segmentation can be applied in this case as well, since each superstate corresponds to a specific facial feature. In addition, the superstate HMM has the same assumptions enforced above in the 1DHMM case for the initial state vector and the state transition matrix.

That being said, there are some additional assumptions and pre-processing steps for this method. Since the sequence is now 2-dimensional, an end-of-state entry is appended to each row, where this end-of-state entry is a SIFT vector composed of zeroes. This allows the sequence to enter the non-looping states that signify the transition between superstates in the superstate HMM.

At this point, if the P2DHMM to 1DHMM conversion method is being used, then the P2DHMM structure can be decomposed according to the above discussion and the 1DHMM methods can be run. If the more traditional embedded approach is used, then the P2DHMM structure can remain as is and the training algorithms described above for P2DHMMs can be run.

4.5 Training the Universal Background Model (Implementation)

In this case, the universal background model is either an HMM or a P2DHMM. In all cases, HTK [23] played an integral role in training the HMMs. For the 1DHMM case and the P2DHMM conversion to 1DHMM case, HTKs built in functions (HVite, HRest, and HERest) can be used to train the HMMs using segmental K-Means or Baum-Welch, and then MAP adapt the resulting HMM. In the final case of training an embedded P2DHMM, HTK does not have any functions for explicitly training a P2DHMM. However, the algorithm outlined above for training a P2DHMM uses embedded passes of 1D Baum Welch and Segmental K-means. Thus, HTK's functions can still be used in this case - but they need to be called as

subroutines in a larger algorithm instead of being called as standalone programs (as in the 1DHMM case).

4.5.1 1D Hidden Markov Model

For training the 1-dimensional HMM UBM, it is assumed that there are five possible states: the forehead, eyes, nose, mouth, and chin. The image is segmented using a cascaded classifier so that all of the observation vectors can be grouped by which feature they belong to. Once that is completed, a GMM with K mixture components can be trained for each facial feature. These GMMs will be initial estimates for the emission models for each state in the HMM.

The HMM is initialized by assuming that most of the time in each sequence is spent in self-loop transitions. In addition, it is assumed that the HMM must begin at the first state.

$$\mathbf{A}_{init} = \begin{bmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.79)$$

$$\boldsymbol{\pi}_{init} = [1 \ 0 \ 0 \ 0 \ 0]^T \quad (4.80)$$

4.5.2 P2DHMM

In this case, it is assumed that there are N_s superstates in the P2DHMM. It is further assumed that the topology of the superstate HMM is left-to-right, and that the sub-state HMMs are also left-to-right HMMs. Finally, it is assumed that all the sub-state HMMs have the same topology (i.e., the same number N_m of sub-states per superstate for all $1 \leq m \leq N_s$). For each sub-state HMM, the assumptions on the state transitions and initial state vectors from the previous 1DHMM case are also enforced in this case. In addition, the GMM initial training via image segmentation can be applied in this case as well, since each superstate corresponds to a specific facial feature. In addition, the superstate HMM has the same assumptions enforced above in the 1DHMM case for the initial state vector and the state transition matrix.

That being said, there are some additional assumptions and pre-processing steps for this method. Since the sequence is now 2-dimensional, an end-of-state entry is appended to each row, where this end-of-state entry is a SIFT vector composed of zeroes. This allows the sequence to enter the non-looping states that signify the transition between superstates in the superstate HMM.

At this point, if the P2DHMM to 1DHMM conversion method is being used, then the P2DHMM structure can be decomposed according to the above discussion and the 1DHMM methods can be run. If the more traditional embedded approach is used, then the P2DHMM structure can remain as is and the training algorithms described above for P2DHMMs can be run.

CHAPTER 5

MAXIMUM A POSTERIORI (MAP) ADAPTATION

The previous chapter focused on the techniques used to train Gaussian mixture models, hidden Markov models, and Pseudo-2D hidden Markov models. This chapter focuses on how such models can be adapted to a particular subject using new training methods. The underlying technique used for all cases in this thesis is maximum a posteriori (MAP) adaptation.

5.1 Maximum A Posteriori (MAP) Adaptation

The general idea behind MAP adaptation is to use a prior, generalized model trained over a set of data and then to adapt its parameters to fit a new set of data.

To visualize the problem that this technique tries to solve, an illustrative example would be fitting a Gaussian to some set of labelled training data, and then to adapt that model to some new set of data. Consider three sets of 2-dimensional training data drawn from three distinct Gaussian distributions (thus, three distinct classes). A universal background model is trained over a sample of this data from each class, as shown in Figure 5.1.

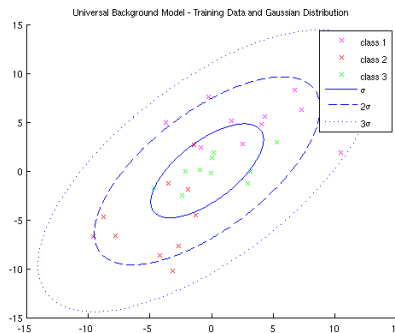


Figure 5.1: Training Data and Overall Gaussian Distribution

Now assume that three new sets of data are drawn from the same underlying Gaussian distributions, and a model needs to be found for each class. MAP adaptation can be used to adapt the initial training model to better fit the new data from each class. This is illustrated in Figure 5.2.

For a dataset using only a single Gaussian, a trivial approach that could accomplish the desired goal would be to fit a Gaussian to each new dataset using the sample mean and sample covariance of each new set of data. However, for a Gaussian mixture model or a hidden Markov model, it is easier to simply adapt from an underlying model. In addition, by using a prior model, it often becomes easier to find a more

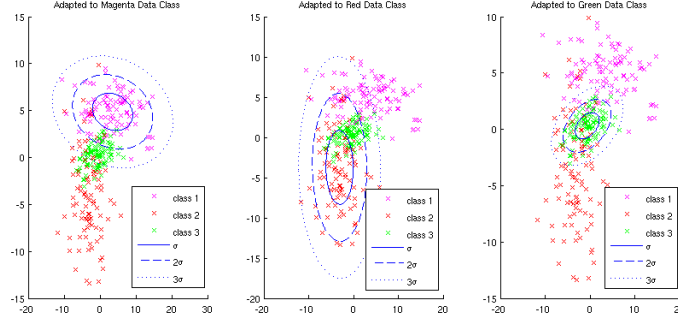


Figure 5.2: New Training Data and Adapted Gaussian Distributions, (left) Adapted to Magenta Data, (center) Adapted to Red Data, (right) Adapted to Green Data

accurate new model for a small set of new data than it is to re-train a unique model from scratch.

MAP adaptation is also linked to the idea of MAP estimation. Maximum likelihood (ML) estimation attempts to maximize the probability of a set of data given a set of underlying parameters.

$$\theta_{ML}^* = \arg \max_{\theta} P(\mathbf{X}|\theta) \quad (5.1)$$

MAP estimation, however, introduces a prior distribution, $P(\theta)$, and attempts to maximize a new expression that arises from Bayes' rule. The prior distribution $P(\theta)$ essentially describes some prior knowledge about the underlying distribution of the parameters. Bayes' rule is shown in Equation (5.2).

$$P(\theta|\mathbf{X}) = \frac{P(\mathbf{X}|\theta)P(\theta)}{P(\mathbf{X})} = \frac{P(\mathbf{X}|\theta)P(\theta)}{\int_{\theta} P(\mathbf{X}|\theta)P(\theta)d\theta} \quad (5.2)$$

Since the point of MAP estimation is to find some optimal θ_{MAP}^* for some given set of data, it is assumed that the probability of the data does not change the maximization problem. Thus, the MAP estimation problem can be described by Equation (5.3).

$$\theta_{MAP}^* = \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta) \quad (5.3)$$

While MAP estimation simply attempts to find some optimal solution given an assumed prior distribution, MAP adaptation differs in that it simply adapts the training data by some given prior model for some adaptation weight τ .

$$\theta_{MAP}^* = \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)^{\tau} \quad (5.4)$$

5.2 MAP Adaptation Algorithm for a 1D Hidden Markov Model

At this point it is useful to use MAP to determine the update equations for a hidden Markov model.

5.2.1 MAP Estimation of State Transition Probabilities for a Hidden Markov Model

As an example of this, consider adapting the state transition probabilities of an HMM. It can be assumed, through the work of Gauvain and Lee [24], that the underlying distribution of transition probabilities follows a Dirichlet distribution with prior parameters α_i . This is shown in Equation (5.5).

$$P(a_i|\alpha_i) = \frac{\Gamma(\sum_{j=1}^N \alpha_{ij})}{\prod_{j=1}^N \Gamma(\alpha_{ij})} \prod_{j=1}^N a_{ij}^{\alpha_{ij}-1} \quad (5.5)$$

It can be assumed that the hyperparameters α_{ij} can be set to $\alpha_{ij} = \bar{a}_{ij}\tau$, where τ is some universal adaptation hyperparameter to be chosen. If this is the case, then the expectation of the prior distribution can be written as Equation (5.6) and the variance of the prior distribution can be written as Equation (5.7).

$$E[a_{ij}] = \bar{a}_{ij} \quad (5.6)$$

$$\text{Var}(a_{ij}) = \frac{\bar{a}_{ij}(1 - \bar{a}_{ij})}{\tau + 1} \quad (5.7)$$

Expectation-maximization can be used to iteratively attempt to find the optimal solution. Following the analysis from the previous chapter, the auxiliary function can be written as $Q(\bar{A}, \theta)$ where $\bar{A}$ represents the state transition probability matrix from the UBM. For some set of hyperparameters α_{ij} , it can be shown [24] that the update equation for the new transition probabilities, $\hat{a}_{ij}$, is given by Equation (5.8).

$$\hat{a}_{ij} = \frac{\tau \bar{a}_{ij} - 1 + \sum_{t=1}^T \xi_t(i, j)}{\sum_{j=1}^N (\tau \bar{a}_{ij} - 1 + \sum_{t=1}^T \xi_t(i, j))}, \quad \xi_t(i, j) = P(q_t = i, q_{t+1} = j | \mathbf{X}, \lambda) \quad (5.8)$$

In this sense, MAP estimation becomes MAP adaptation because the prior parameters in this case are based on the actual state transition matrix from the universal background model. MAP adaptation of the STM would essentially be a simple augmentation of the EM algorithm for HMMs from the previous chapter. Instead of computing a_{ij} directly from the parameter $\xi_t(i, j)$ computed in the E-step, both the $\xi_t(i, j)$ parameter and the prior UBM STM are used to update STM based on some pre-chosen adaptation learning weight τ .

Similar reasoning can be used to find the update equations for the initial probabilities π_i and the emissions using assumptions on the prior distributions of all of these parameters. However, it is assumed that the initial probabilities are not adapted (in other words, the same no matter what subject is being adapted in the context of the face recognition problem). The emission probabilities, on the other hand, will be computed using the slightly looser MAP adaptation approach.

5.2.2 MAP Adaptation for the Emission Parameters

Now, the emission model adaptation can be considered. Once again, the underlying emission probabilities are assumed to follow a continuous density distribution. Specifically, it is assumed that the emission probabilities are generated from an underlying Gaussian mixture model unique to each state. It is assumed that the standard EM algorithm is being used to update the parameters, in which case the posterior probabilities are given by Equation (5.9).

$$\gamma_t(j, k) = \left(\frac{\alpha_t(j)\beta_t(j)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)} \right) \left(\frac{w_{jk}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k=1}^K w_{jk}\mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \right) \quad (5.9)$$

The data sufficient statistics are given by Equations (5.10) through (5.12).

$$\eta_{jk} = \frac{1}{T} \sum_{t=1}^T \gamma_t(j, k) \quad (5.10)$$

$$\boldsymbol{\mu}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k) \mathbf{x}_t}{\eta_{jk}} \quad (5.11)$$

$$\boldsymbol{\Sigma}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k) \mathbf{x}_t \mathbf{x}_t^T}{\eta_{jk}} \quad (5.12)$$

After defining an auxiliary function for the EM algorithm and optimizing, it can be shown [24] that the adapted parameters are given by Equations (5.13) through (5.15). In the initial step of the EM algorithm, $\bar{w}_{jk}$, $\bar{\boldsymbol{\mu}}_{jk}$, and $\bar{\boldsymbol{\Sigma}}_{jk}$ represent the UBM parameters. However, after the first iteration they represent the parameters from the previous iteration.

$$\hat{w}_{jk} = \left(\frac{T\eta_{jk}}{\tau_w + T\eta_{jk}} \right) \eta_{jk} + \left(1 - \frac{T\eta_{jk}}{\tau_w + T\eta_{jk}} \right) \bar{w}_{jk} \quad (5.13)$$

$$\hat{\boldsymbol{\mu}}_{jk} = \left(\frac{T\eta_{jk}}{\tau_m + T\eta_{jk}} \right) \boldsymbol{\mu}_{jk} + \left(1 - \frac{T\eta_{jk}}{\tau_m + T\eta_{jk}} \right) \bar{\boldsymbol{\mu}}_{jk} \quad (5.14)$$

$$\hat{\boldsymbol{\Sigma}}_{jk} = \left(\frac{T\eta_{jk}}{\tau_c + T\eta_{jk}} \right) \boldsymbol{\Sigma}_{jk} + \left(1 - \frac{T\eta_{jk}}{\tau_c + T\eta_{jk}} \right) (\bar{\boldsymbol{\Sigma}}_{jk} + \bar{\boldsymbol{\mu}}_{jk} \bar{\boldsymbol{\mu}}_{jk}^T) - \hat{\boldsymbol{\mu}}_{jk} \hat{\boldsymbol{\mu}}_{jk}^T \quad (5.15)$$

In the above equations, τ_w , τ_m , and τ_c are adaptation weights chosen prior to running the EM MAP adaptation algorithm. More accurately, they are a representation of the ratio between the hyper-parameters and the sufficient statistics. For the weights, as for the transition parameters, it is assumed that the priors follow a Dirichlet distribution. The priors for the mean and covariance, however, are assumed to follow a Wishart distribution, as shown in Equation (5.16). In this way, τ_m should represent the ratio between the determinants of the variance and the hyper-variance. Also, τ_c should represent the confidence parameter of the Wishart prior.

$$g(\hat{\boldsymbol{\mu}}_{jk}, \hat{\boldsymbol{\Sigma}}_{jk} | \alpha_m, \alpha_s, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \propto |\hat{\boldsymbol{\Sigma}}_{jk}|^{\alpha_s/2} \exp \left(-\frac{\alpha_m}{2} (\hat{\boldsymbol{\mu}}_{jk} - \boldsymbol{\mu}_{jk})^T \hat{\boldsymbol{\Sigma}}_{jk}^{-1} (\hat{\boldsymbol{\mu}}_{jk} - \boldsymbol{\mu}_{jk}) \right) \exp \left(-\frac{1}{2} \text{Tr}(\hat{\boldsymbol{\Sigma}}_{jk}^{-1} \boldsymbol{\Sigma}_{jk}) \right) \quad (5.16)$$

All of the hyperparameters, by definition, should lie in the range $[0, 1]$. However, there is no such restriction on the adaptation weights. In all cases, the adaptation weights represent how much confidence the algorithm should have in prior model. If the adaptation weights are small, there is a great deal of faith in the prior model, and the EM algorithm converges quickly. If the weights are large, however, the opposite is true.

CHAPTER 6

SUPERVECTORS

This chapter focuses on the supervectors, and how they are derived and implemented. In this chapter, previous work by Tang [15] is mostly gone over, with additional material about simulating the Kullback-Leibler divergence and finding various bounds.

6.1 Introduction: The Idea of Supervectors

The idea behind the generation of supervectors is to create a new representation of a probabilistic model such that the distance between any two such supervectors is equivalent to some difference measure between their two corresponding probabilistic models. A common difference metric for hidden Markov models or Gaussian mixture models is the Kullback-Leibler divergence, which is what will be used in this derivation. In this case, the goal is to create a linearly-separable space with the supervectors. Thus, the L_2 norm will be the distance metric used for the supervectors.

6.2 Kullback-Leibler Divergence

The Kullback-Leibler divergence (KLD) [25] between a continuous distribution $f(\mathbf{x})$ and another continuous distribution $g(\mathbf{x})$ is given by Equation (6.1).

$$D(f(\mathbf{x})||g(\mathbf{x})) = \int_{\mathbf{x} \in \mathbb{R}^D} f(\mathbf{x}) \ln \left(\frac{f(\mathbf{x})}{g(\mathbf{x})} \right) d\mathbf{x} \quad (6.1)$$

If conditional distributions are considered, the KLD between them is given by Equation (6.2).

$$D(f(\mathbf{y}|\mathbf{x})||g(\mathbf{y}|\mathbf{x})) = \int_{\mathbf{x} \in \mathbb{R}^D} f(\mathbf{x}) \int_{\mathbf{y} \in \mathbb{R}^D} f(\mathbf{y}|\mathbf{x}) \ln \left(\frac{f(\mathbf{y}|\mathbf{x})}{g(\mathbf{y}|\mathbf{x})} \right) d\mathbf{x} d\mathbf{y} \quad (6.2)$$

If joint distributions are considered, the KLD between them is given by Equation (6.3).

$$D(f(\mathbf{x}, \mathbf{y})||g(\mathbf{x}, \mathbf{y})) = D(f(\mathbf{x})||g(\mathbf{x})) + D(f(\mathbf{y}|\mathbf{x})||g(\mathbf{y}|\mathbf{x})) \quad (6.3)$$

The Kullback-Leibler divergence has the general properties that if $f(\mathbf{x}) = g(\mathbf{x}) \forall \mathbf{x}$, then the KLD between the two distributions is 0, ie, $D(f||g)|_{f=g} = 0$. The KLD also has the property that $D(f||g) \geq 0$ regardless of f or g . The Kullback-Leibler Divergence Rate (KLDR) is also sometimes considered.

$$R(f(\mathbf{x})||g(\mathbf{x})) = \lim_{T \rightarrow \infty} \frac{1}{T} D(f(\mathbf{x})||g(\mathbf{x})) \quad (6.4)$$

6.2.1 KLD for a Multivariate Gaussian Distribution

The closed-form Kullback-Leibler divergence can be analytically calculated in some cases. For example, the Kullback-Leibler divergence between two multivariate Gaussian distributions, $f = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $g = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$, is given by Equation (6.5).¹

$$\begin{aligned}
D(f(\mathbf{x})||g(\mathbf{x})) &= \int f(\mathbf{x}) \ln \left(\frac{\frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_1|}} \exp \left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) \right)}{\frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_2|}} \exp \left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right)} \right) d\mathbf{x} \\
&= \int f(\mathbf{x}) \left(\frac{1}{2} \ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) \right) d\mathbf{x} \\
&= \frac{1}{2} \int f(\mathbf{x}) \ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) d\mathbf{x} - \frac{1}{2} \int f(\mathbf{x}) (\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) d\mathbf{x} \\
&\quad + \frac{1}{2} \int f(\mathbf{x}) (\mathbf{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) d\mathbf{x} \\
&= \frac{1}{2} E_{f(\mathbf{x})} \left[\ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) \right] - \frac{1}{2} E_{f(\mathbf{x})} [(\mathbf{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\mathbf{x} - \boldsymbol{\mu}_1)] \\
&\quad + \frac{1}{2} E_{f(\mathbf{x})} [(\mathbf{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\mathbf{x} - \boldsymbol{\mu}_2)] \\
&= \frac{1}{2} \ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) - \frac{1}{2} \text{Tr} (\boldsymbol{\Sigma}_1^{-1} \boldsymbol{\Sigma}_1) - \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_1^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_1) \\
&\quad + \frac{1}{2} \text{Tr} (\boldsymbol{\Sigma}_2^{-1} \boldsymbol{\Sigma}_1) + \frac{1}{2} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&= \frac{1}{2} \left(\ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) - D + \text{Tr} (\boldsymbol{\Sigma}_2^{-1} \boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right) \tag{6.5}
\end{aligned}$$

6.2.2 KLD for a GMM

In other cases either the closed-form solution is too difficult to compute or simply does not exist. This is generally the case for both hidden Markov models and Gaussian mixture models. Thus, it is desirable to be able to numerically approximate the Kullback-Leibler divergence in these cases. Often, this is done using Monte Carlo simulation, which rewrites the integral as an expectation and evaluates an approximation of this expectation over a sufficiently large number of trials such that the sample mean approaches this expectation.

$$D(f(\mathbf{x})||g(\mathbf{x})) = E_{f(\mathbf{x})} [\ln f(\mathbf{x})] - E_{f(\mathbf{x})} [\ln g(\mathbf{x})] \tag{6.6}$$

$$D(f(\mathbf{x})||g(\mathbf{x})) \approx \frac{1}{T} \sum_{t=1}^T [\ln f(\mathbf{x}_t) - \ln g(\mathbf{x}_t)] \tag{6.7}$$

The Kullback-Leibler divergence between two Gaussian mixture models can be approximated using Monte Carlo simulation. That is to say, given a sequence of observation vectors $\mathbf{x}_t \in \mathbf{X}$ and a GMM $\theta = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ for $1 \leq k \leq K$, $p(\mathbf{x}_t; \theta)$ can be calculated for each $\mathbf{x}_t$ and the KLD can be approximated for that particular sequence using Equation (6.9).

¹Using the fact that $E[(\mathbf{x} - \mathbf{m})^T \mathbf{A} (\mathbf{x} - \mathbf{m})] = (\mathbf{m} - \mathbf{m}')^T \mathbf{A} (\mathbf{m} - \mathbf{m}') + \text{Tr}(\mathbf{A} \boldsymbol{\Sigma})$ if $\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \boldsymbol{\Sigma})$ from The Matrix Cookbook [26].

$$p(\mathbf{x}_t; \theta) = \sum_{k=1}^K \frac{w_k}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_k|}} \exp \left(-\frac{1}{2} (\mathbf{x}_t - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{x}_t - \boldsymbol{\mu}_k) \right) \quad (6.8)$$

$$D(\theta_1 || \theta_2) = \lim_{T \rightarrow \infty} \frac{1}{T} \sum_{t=1}^T \ln \left(\frac{p(\mathbf{x}_t; \theta_1)}{p(\mathbf{x}_t; \theta_2)} \right) \quad (6.9)$$

This can be average over multiple trials to get a more representative measure. However, Julier and Uhlmann [27] showed that an even better approximation to the KLD was possible using the so-called “unscented transform”. The approach to this transform is to $E_f[h(\mathbf{x})]$ for all quadratic functions $h(\mathbf{x})$ given a mixture model $f(\mathbf{x}) = p(\mathbf{x}_t; \theta_1)$. They show that it is possible to pick $2D$ (where $\mathbf{x}_t \in \mathbb{R}^D$) points from mixture k of the GMM such that Equation (6.10) is true.

$$\int f(\mathbf{x}) h_k(\mathbf{x}) d\mathbf{x} = \frac{1}{2D} \sum_{t=1}^{2D} h_k(\mathbf{x}_t) \quad (6.10)$$

They further show that one possible choice for the points can be determined by using the eigenvectors $\mathbf{v}_{d,k}$ and eigenvalues $\omega_{d,k}$ of the covariance matrix of f corresponding to mixture k , $\boldsymbol{\Sigma}_k$.

$$\mathbf{x}_{t,k} = \begin{cases} \boldsymbol{\mu}_k + \sqrt{D\omega_{t,k}} \mathbf{v}_{t,k} & \text{if } 1 \leq t \leq D \\ \boldsymbol{\mu}_k - \sqrt{D\omega_{t-D,k}} \mathbf{v}_{t-D,k} & \text{if } t > D \end{cases} \quad (6.11)$$

In this way, the KLD between two GMMs θ_1 and θ_2 can be approximated according to Equation (6.12).

$$D(\theta_1 || \theta_2) = \frac{1}{2D} \sum_{k=1}^K w_1(k) \sum_{t=1}^{2D} \ln \frac{f(\mathbf{x}_{t,k})}{g(\mathbf{x}_{t,k})} \quad (6.12)$$

Algorithm 7 KLD GMM Monte Carlo

```

function KLD GMM MONTE CARLO( $\theta_1, \theta_2, R, T$ )
  for  $r = 1$  to  $R$  do
    for  $k = 1$  to  $K$  do
      Draw samples from mixture  $k$  of  $\theta_1$  using Equation (6.11) and send to  $x_i$ 
      Calculate  $f_i = p(x_i | \theta_1)$  and  $g_t = p(x_i | \theta_2)$  using Equation (6.8) for  $1 \leq t \leq T$ 
      Calculate  $h_k = \sum_{i=1}^{2D} \ln \frac{f(\mathbf{x}_i)}{g(\mathbf{x}_i)}$ 
    end for
     $y_r = \frac{1}{2D} \sum_{k=1}^K w_k h_k$ 
  end for
   $\bar{D} = \frac{1}{R} \sum_{r=1}^R y_r$ 
  return  $\bar{D}$ 
end function

```

Although it is true that a closed form solution for the KLD between two GMM’s does not exist, analytic upper bounds on it do exist. For example, based on the work by Do [28], the chain rule for KLD can be applied to generate an upper bound, assuming that the two GMMs have the same number of mixture components (this is sometimes called the matched bound approximation). This is shown in Equation (6.13).

$$D(f(\mathbf{x})||g(\mathbf{x})) \leq D(\mathbf{w}_1||\mathbf{w}_2) + \sum_{k=1}^K w_2(k) D(\mathcal{N}(\boldsymbol{\mu}_{1,k}, \boldsymbol{\Sigma}_{1,k})||\mathcal{N}(\mathbf{m}\mathbf{u}_{2,k}, \boldsymbol{\Sigma}_{2,k})) \quad (6.13)$$

In addition, a few other papers have established their own upper or lower bounds the KLD between two GMMs. Durrieu et al. [29] established a bound using the mean between the upper and lower bounds of the KLD in order to create a more general bound for the KLD which more closely matched the results of a Monte Carlo simulation. This is shown in Equation (6.20).

$$z_{a\alpha} = \int_{\mathbf{x}} f_a(\mathbf{x}) f_{\alpha}(\mathbf{x}) d\mathbf{x} \quad (6.14)$$

$$\ln z_{a\alpha} = -\frac{D}{2} - \frac{1}{2} \ln |\boldsymbol{\Sigma}_a^f + \boldsymbol{\Sigma}_{\alpha}^f| - \frac{1}{2} (\boldsymbol{\mu}_{\alpha}^f - \boldsymbol{\mu}_a^f)^T (\boldsymbol{\Sigma}_a^f + \boldsymbol{\Sigma}_{\alpha}^f)^{-1} (\boldsymbol{\mu}_{\alpha}^f - \boldsymbol{\mu}_a^f) \quad (6.15)$$

$$t_{ab} = \int_{\mathbf{x}} f_a(\mathbf{x}) g_b(\mathbf{x}) d\mathbf{x} \quad (6.16)$$

$$\ln t_{ab} = -\frac{D}{2} - \frac{1}{2} \ln |\boldsymbol{\Sigma}_a^f + \boldsymbol{\Sigma}_b^g| - \frac{1}{2} (\boldsymbol{\mu}_b^g - \boldsymbol{\mu}_a^f)^T (\boldsymbol{\Sigma}_a^f + \boldsymbol{\Sigma}_b^g)^{-1} (\boldsymbol{\mu}_b^g - \boldsymbol{\mu}_a^f) \quad (6.17)$$

$$D_{prod}(\theta_1||\theta_2) = \sum_a w_a^f \ln \left(\frac{\sum_{\alpha} w_{\alpha}^f z_{a,\alpha}}{\sum_b w_b^g t_{a,b}} \right) \quad (6.18)$$

$$D_{var}(\theta_1||\theta_2) = \sum_a w_a^f \ln \left(\frac{\sum_{\alpha} w_{\alpha}^f \exp(-D_{KL}(f_a(\mathbf{x})||f_{\alpha}(\mathbf{x})))}{\sum_b w_b^g \exp(-D_{KL}(f_a(\mathbf{x})||g_b(\mathbf{x})))} \right) \quad (6.19)$$

$$D_{mean}(\theta_1||\theta_2) = \frac{1}{2} (D_{var}(\theta_1||\theta_2) + D_{prod}(\theta_1||\theta_2)) \quad (6.20)$$

6.2.3 KLD for a 1DHMM

As in the case of the Gaussian mixture model, an approximation of the Kullback-Leibler divergence for the hidden Markov model can be determined using Monte Carlo simulations with Equation (6.21). That is to say, the HMM can be sampled (generally multiple times to create multiple sequences), and the resulting KLD can be averaged across many trials to find an approximation to the true KLD. Based on the work by Juang and Rabiner [30] and Kohler [31], a Monte Carlo approximation to the Kullback-Leibler divergence between two hidden Markov models, $\lambda_1 = \{A_1, \pi_1, B_1\}$ and $\lambda_2 = \{A_2, \pi_2, B_2\}$, could generally be approximated as Equation (6.21) using Monte Carlo simulation with T_m observations per sequence $1 \leq m \leq M$.

$$D(\lambda_1||\lambda_2) \approx \frac{1}{M} \sum_{m=1}^M \frac{1}{T_m} \ln \left(\frac{P(\mathbf{X}_m|\lambda_1)}{P(\mathbf{X}_m|\lambda_2)} \right) \quad (6.21)$$

Juang also gave a symmetric distance metric given by Equation (6.22).

$$D_{sym}(\lambda_1||\lambda_2) = D(\lambda_1||\lambda_2) + D(\lambda_2||\lambda_1) \quad (6.22)$$

Algorithm 8 KLD HMM Monte Carlo

```
function KLD HMM MONTE CARLO( $\lambda_1, \lambda_2, R, M, T$ )  
  for  $r = 1$  to  $R$  do  
    for  $m = 1$  to  $M$  do  
      Draw samples from  $\lambda_1$  to get  $T$  sample vectors  $x_t \in X_m$   
      Calculate  $P(X_m|\lambda_1)$  and  $P(X_m|\lambda_2)$  using the Forward Algorithm  
    end for  
     $y_r = \frac{1}{M} \sum_{m=1}^M \frac{1}{T_m} \ln \left( \frac{P(\mathbf{X}_m|\lambda_1)}{P(\mathbf{X}_m|\lambda_2)} \right)$   
  end for  
   $\bar{D} = \frac{1}{R} \sum_{r=1}^R y_r$   
  return  $\bar{D}$   
end function
```

Again, the above equations are just approximations of the KLD using Monte Carlo simulation. Thus, they do not represent analytic expressions for the KLDs for either HMMs or GMMs. However, analytic expressions for the upper bounds on the KLDs of HMMs with certain architectures *do* exist. The work by Do [28] introduces a general upper bound for the KLD, Equation (6.27), and KLDR, Equation (6.29), between two HMMs (since the HMM parameters are stationary).

$$\mathbf{a}_i(j) = a_{ij} \quad (6.23)$$

$$\mathbf{d}_{KLD}^A = \left[D(\mathbf{a}_1^1 || \mathbf{a}_1^2) \quad \dots \quad D(\mathbf{a}_N^1 || \mathbf{a}_N^2) \right]^T \quad (6.24)$$

$$\mathbf{d}_{KLD}^B = \left[D(b_1^1 || b_1^2) \quad \dots \quad D(b_N^1 || b_N^2) \right]^T \quad (6.25)$$

$$\mathbf{d} = \mathbf{d}_{KLD}^A + \mathbf{d}_{KLD}^B \quad (6.26)$$

$$D(\lambda_1 || \lambda_2) \leq D(\boldsymbol{\pi}_1 || \boldsymbol{\pi}_2) + \boldsymbol{\pi}_1^T \left(\sum_{t=1}^{T-1} \mathbf{A}^{t-1} \mathbf{d} + \mathbf{A}^{T-1} \mathbf{d}_{KLD}^B \right) \quad (6.27)$$

$$\boldsymbol{\nu}^T = \lim_{n \rightarrow \infty} \boldsymbol{\pi}^T \mathbf{A}^n \quad (6.28)$$

$$R(\lambda_1 || \lambda_2) \leq \boldsymbol{\nu}^T \mathbf{d} \quad (6.29)$$

Silva and Narayanan [32] show that there is an analytic upper bound for the KLD of an HMM if said HMM has a left-to-right structure. The equation for the upper bound of the KLD of a left-to-right HMM as given by Silva and Narayanan is given by Equation (6.30).

$$D(\lambda_1 || \lambda_2) \leq D(\boldsymbol{\pi}_1 || \boldsymbol{\pi}_2) + \boldsymbol{\pi}_1^T (\mathbf{I} - \mathbf{A}_1)^{-1} [\mathbf{d}_{KLD}^A + \mathbf{d}_{KLD}^B] \quad (6.30)$$

6.3 Supervectors

Now that the general formulas that bound the KLD for 1DHMMs and P2DHMMs have been established, they can be manipulated to form the supervector equations needed for the classification phase of the algorithm.

6.3.1 KLD Between Two PMFs and Resulting Supervector

In the KLD bounds described above, terms that look like $D_s(\mathbf{p}||\mathbf{q})$ or $D(\mathbf{p}||\mathbf{q})$ (where $\mathbf{p}$ and $\mathbf{q}$ are essentially probability mass functions) often appear. Thus, it is useful to consider the case of breaking down the KLD between two finite sequences $\mathbf{p}_1$ and $\mathbf{p}_2$ of length K adapted from a UBM.

$$\begin{aligned}
D_s(\mathbf{p}_1||\mathbf{p}_2) &= \frac{1}{2} \sum_{k=1}^K \left(p_1(k) \ln \frac{p_1(k)}{p_2(k)} + p_2(k) \ln \frac{p_2(k)}{p_1(k)} \right) \\
&= \frac{1}{2} \sum_{k=1}^K (p_1(k) \ln p_1(k) - p_1(k) \ln p_2(k) + p_2(k) \ln p_2(k) - p_2(k) \ln p_1(k)) \\
&= \frac{1}{2} \sum_{k=1}^K (p_1(k) - p_2(k)) \ln \frac{p_1(k)}{p_2(k)}
\end{aligned} \tag{6.31}$$

Assuming that each PMF is adapted from some underlying UBM, the above expression can be rewritten.

$$\begin{aligned}
D_s(\mathbf{p}_1||\mathbf{p}_2) &= \frac{1}{2} \sum_{k=1}^K (p_1(k) - p_2(k)) \ln \frac{p_1(k)}{p_2(k)} \\
&= \frac{1}{2} \sum_{k=1}^K (p_1(k) - p_2(k)) \ln \frac{p_1(k)/p_{ubm}(k)}{p_2(k)/p_{ubm}(k)} \\
&= \frac{1}{2} \sum_{k=1}^K (p_1(k) - p_2(k)) \left(\ln \frac{p_1(k)}{p_{ubm}(k)} - \ln \frac{p_2(k)}{p_{ubm}(k)} \right)
\end{aligned} \tag{6.32}$$

As $x \rightarrow 1$, $\ln(x)$ converges to $x - 1$ (in other words, $\lim_{x \rightarrow 1} x - 1 = \lim_{x \rightarrow 1} \ln x$, see Figure 6.1). It is reasonable to assume that $p_1(k)/p_{ubm}(k)$ is near one since the PMFs are adapted from the UBM and the components are not likely to change very much unless the UBM is a very poor model of the data used in the adaptation.

With this in mind, the approximation can be substituted into the symmetric KLD expression to get a final simplified equation.

$$D_s(\mathbf{p}_1||\mathbf{p}_2) \approx \frac{1}{2} \sum_{k=1}^K \left(\frac{p_1(k) - p_2(k)}{\sqrt{p_{ubm}(k)}} \right)^2 \tag{6.33}$$

At this point, the result is basically an L2 norm.²

²The division sign in the subsequent equation denotes element by element division in this case, since there are vectors in the fractions.

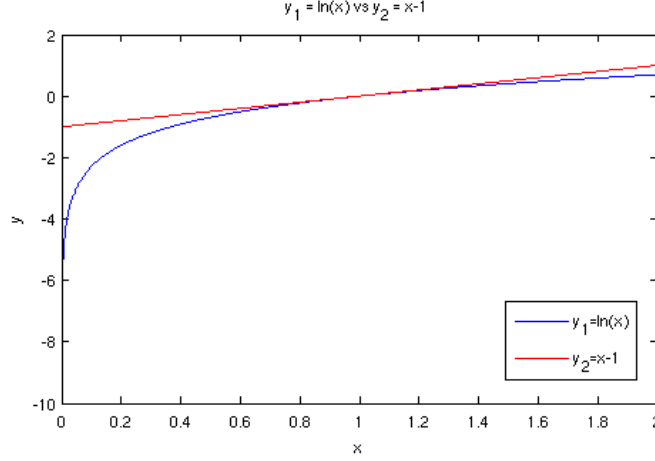


Figure 6.1: $y_1 = \ln x$ vs. $y_2 = x - 1$ Near $x = 1$

$$D_s(\mathbf{p}_1 || \mathbf{p}_2) \approx \frac{1}{2} \left\| \frac{\mathbf{p}_1}{\sqrt{\mathbf{p}_{ubm}}} - \frac{\mathbf{p}_2}{\sqrt{\mathbf{p}_{ubm}}} \right\|_2^2 \quad (6.34)$$

Thus, the supervector that describes this relationship for the p -th PMF is:

$$\mathbf{s}_p = \frac{\mathbf{p}_p}{\sqrt{\mathbf{p}_{ubm}}} \quad (6.35)$$

This relation is used by Tang in his subsequent GMM and 1DHMM supervector derivations, and is used in this work in a later chapter to derive the P2DHMM supervector equations.

6.3.2 GMM Supervectors

At this point, the derivation of the GMM supervector equation can be shown. Although this derivation is broadly shown in Tang's work [15], it is worth going over in this work for the sake of clarity. To approximate the supervector set whose norm describes the KLD between two GMMs, one of the bounds described above is reshaped such that it looks like the norm between two vectors. For example, consider the matched bound from Equation (6.13).

$$D(f(\mathbf{x}) || g(\mathbf{x})) \leq D(\mathbf{w}_1 || \mathbf{w}_2) + \sum_{k=1}^K w_2(k) D(\mathcal{N}(\boldsymbol{\mu}_{1,k}, \boldsymbol{\Sigma}_{1,k}) || \mathcal{N}(\boldsymbol{\mu}_{2,k}, \boldsymbol{\Sigma}_{2,k})) \quad (6.36)$$

The supervector expression will be dominated by the KLD between two Gaussian distributions. This KLD can be further simplified.

$$D(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) = \frac{1}{2} \left(\ln \left(\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} \right) - D + \text{Tr}(\boldsymbol{\Sigma}_2^{-1} \boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \right) \quad (6.37)$$

Consider just the matrix-vector product. This can be simplified into using the symmetry property of the covariance matrix, by defining $\mathbf{R} = \sqrt{\boldsymbol{\Sigma}^{-1}}$ and invoking the symmetry property ($\mathbf{R}^T = \mathbf{R}$).

$$\begin{aligned}
(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \mathbf{R} \mathbf{R} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\
&= (\mathbf{R}^T \boldsymbol{\mu}_1^T - \mathbf{R}^T \boldsymbol{\mu}_2^T) (\mathbf{R} \boldsymbol{\mu}_1 - \mathbf{R} \boldsymbol{\mu}_2) \\
&= (\mathbf{R} \boldsymbol{\mu}_1 - \mathbf{R} \boldsymbol{\mu}_2)^T (\mathbf{R} \boldsymbol{\mu}_1 - \mathbf{R} \boldsymbol{\mu}_2) \\
&= \|\sqrt{\boldsymbol{\Sigma}_2^{-1}} \boldsymbol{\mu}_1 - \sqrt{\boldsymbol{\Sigma}_2^{-1}} \boldsymbol{\mu}_2\|_2^2
\end{aligned} \tag{6.38}$$

If the two Gaussians are adapted from a single UBM, then according to Campbell [33], the symmetric KLD between two Gaussian distribution can be expressed according to Equation (6.39).

$$D_s(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) \approx \frac{1}{4} \text{Tr}((\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2) \boldsymbol{\Sigma}_{ubm}^{-2} (\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2)) + \frac{1}{4} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}) (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{6.39}$$

Assuming that $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are the diagonal elements of the two covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ in vector form, the above expression can be simplified as Equation (6.40).

$$D_s(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) \approx \frac{1}{4} (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)^T \boldsymbol{\Sigma}_{ubm}^{-2} (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2) + \frac{1}{4} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}) (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{6.40}$$

At this point, the symmetric KLD between the two distributions can be simplified into a form composed of clear L2 norms.

$$D_s(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) \approx \frac{1}{4} \|\boldsymbol{\Sigma}_{ubm}^{-1} (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)\|_2^2 + \frac{1}{4} \|(\sqrt{\boldsymbol{\Sigma}_1^{-1}} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))\|_2^2 + \frac{1}{4} \|\sqrt{\boldsymbol{\Sigma}_2^{-1}} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\|_2^2 \tag{6.41}$$

The above algebra is taken from Tang [15]. In Tang's [15] derivation, it is further assumed that average of adapted covariance matrices can be replaced by the covariance matrix of the UBM. This is because if such an assumption is not made, then the supervector equation must be constrained to describe two Gaussians, instead of simply describing one Gaussian. In this case, the symmetric KLD would be simplified to Equation (6.42)

$$D_s(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) || \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) \approx \frac{1}{4} \|\boldsymbol{\Sigma}_{ubm}^{-1} (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)\|_2^2 + \frac{1}{2} \|(\sqrt{\boldsymbol{\Sigma}_{ubm}^{-1}} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))\|_2^2 \tag{6.42}$$

At this point the supervector equation for a multivariate Gaussian can be written by inspection.

$$\mathbf{s}_p = \begin{bmatrix} \frac{1}{\sqrt{2}} \boldsymbol{\Sigma}_{ubm}^{-1} \boldsymbol{\sigma}_p \\ \sqrt{\boldsymbol{\Sigma}_{ubm}^{-1}} \boldsymbol{\mu}_p \end{bmatrix} \tag{6.43}$$

At this point, the above simplification for the symmetric KLD between two GMMs can be used to generate the supervector equation for a GMM.

$$D_s(f(\mathbf{x}) || g(\mathbf{x})) \leq D_s(\mathbf{w}_1 || \mathbf{w}_2) + \frac{1}{8} \sum_{k=1}^K (w_1(k) + w_2(k)) (\|\boldsymbol{\Sigma}_{ubm}^{-1} (\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)\|_2^2 + 2 \|(\sqrt{\boldsymbol{\Sigma}_{ubm}^{-1}} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))\|_2^2) \tag{6.44}$$

If the average of the two weights is replaced with the UBM weight, and the weight outside the logarithm in the symmetric KLD between two weight vectors is replaced with the UBM weight, then the expression simplifies considerably.

$$D_s(f(\mathbf{x})||g(\mathbf{x})) \leq D_s(\mathbf{w}_1||\mathbf{w}_2) + \frac{1}{4} \sum_{k=1}^K w_{ubm}(k) (\|\Sigma_{ubm}^{-1}(\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2)\|_2^2 + 2\|(\sqrt{\Sigma_{ubm}^{-1}}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2))\|_2^2) \quad (6.45)$$

At this point, the symmetric KLD between the weight vectors needs to be dealt with. Using the result from the previous section, this can be computed as Equation (6.46).

$$D_s(\mathbf{w}_1||\mathbf{w}_2) \approx \frac{1}{2} \left\| \frac{\mathbf{w}_1}{\sqrt{\mathbf{w}_{ubm}}} - \frac{\mathbf{w}_2}{\sqrt{\mathbf{w}_{ubm}}} \right\|_2^2 \quad (6.46)$$

The supervector equation for a GMM becomes:

$$\mathbf{s}_p = \left[\begin{array}{c} \frac{w_{p,k}}{\sqrt{2w_{ubm,k}}} \\ \frac{1}{2} w_{ubm,k} \Sigma_{ubm,k}^{-1} \boldsymbol{\sigma}_{p,k} \\ \frac{1}{\sqrt{2}} w_{ubm,k} \sqrt{\Sigma_{ubm,k}^{-1}} \boldsymbol{\mu}_{p,k} \end{array} \right]_{k=1}^K \quad (6.47)$$

6.3.3 1DHMM Supervectors

Based on the left-to-right HMM KLD bound described above, Tang [15] derived an equation for the HMM supervector given that the HMM was adapted from a UBM. Consider that same reasoning used above to derive an equation for the symmetric KLD between two weight vectors can be used for the symmetric KLD between two state transition vectors $\mathbf{a}_i$ or two initial state probability vectors $\boldsymbol{\pi}_i$ since they are all thought to be probability mass functions for the purposes of this analysis.

$$D_s(\mathbf{a}_i^1||\mathbf{a}_i^2) \approx \frac{1}{2} \left\| \frac{\mathbf{a}_i^1}{\sqrt{\mathbf{a}_i^{ubm}}} - \frac{\mathbf{a}_i^2}{\sqrt{\mathbf{a}_i^{ubm}}} \right\|_2^2 \quad (6.48)$$

$$D_s(\boldsymbol{\pi}_1||\boldsymbol{\pi}_2) \approx \frac{1}{2} \left\| \frac{\boldsymbol{\pi}_1}{\sqrt{\boldsymbol{\pi}_{ubm}}} - \frac{\boldsymbol{\pi}_2}{\sqrt{\boldsymbol{\pi}_{ubm}}} \right\|_2^2 \quad (6.49)$$

Consider the ergodic case defined by Equation (7.4). According to Tang [15], this expression can be transformed into a KLD rate such that

$$R(\lambda_1||\lambda_2) \leq \boldsymbol{\pi}^T \mathbf{d}_{KLD}^A + \boldsymbol{\pi}^T \mathbf{d}_{KLD}^B \quad (6.50)$$

The second term in the above equation can be dealt with directly by wrapping the π_j terms into the summation for the KLD. In such a case, Tang showed that the supervector could be written as

$$\mathbf{s}_p = \left[\left[\begin{array}{c} \sqrt{\frac{\pi_i}{2w_{ubm,ki}}} w_{ki}^p \\ \frac{1}{2} \sqrt{\pi_i w_{ubm,ki}} \Sigma_{ubm,ki}^{-1} \boldsymbol{\sigma}_{ki}^p \\ \frac{1}{\sqrt{2}} \sqrt{w_{ubm,ki} \pi_i} \Sigma_{ubm,ki}^{-1} \boldsymbol{\mu}_{ki}^p \end{array} \right]_{k=1}^K \right]_{i=1}^N \quad (6.51)$$

The first term has a L2 norm that is easy to write again.

$$\boldsymbol{\pi}^T d_{KLD}^A \approx \left\| \sqrt{\frac{\boldsymbol{\pi}_{ubm}}{\mathbf{a}_{ubm,i}}} (\mathbf{a}_i^1 - \mathbf{a}_i^2) \right\| \quad (6.52)$$

This makes the net supervector equation

$$\mathbf{s}_p = \left[\begin{array}{c} \left[\sqrt{\frac{\pi_{ubm,ki}}{a_{ubm,il}}} a_{il}^p \right]_{l=1}^N \\ \left[\begin{array}{c} \sqrt{\frac{\pi_i}{2w_{ubm,ki}}} w_{ki}^p \\ \frac{1}{2} \sqrt{\pi_i w_{ubm,ki}} \boldsymbol{\Sigma}_{ubm,ki}^{-1} \boldsymbol{\sigma}_{ki}^p \\ \frac{1}{\sqrt{2}} \sqrt{w_{ubm,ki} \pi_i} \boldsymbol{\Sigma}_{ubm,ki}^{-1} \boldsymbol{\mu}_{ki}^p \end{array} \right]_{k=1}^K \end{array} \right]_{i=1}^N \quad (6.53)$$

For a left-to-right 1DHMM, Tang derived the supervector using the bound established by Silva and Narayanan. Using the symmetric KLD, Tang shows that the supervector equation can be written as

$$\mathbf{s}_p = \left[\begin{array}{c} \left[\sqrt{\frac{1-a_{11}}{(1-a_{ii})a_{il}}} a_{il}^p \right]_{l=1}^N \\ \left[\begin{array}{c} \sqrt{\frac{1-a_{11}}{(1-a_{ii})c_{ik}}} c_{ik}^p \\ \sqrt{\frac{1-a_{11}}{1-a_{ii}}} c_{ik} \boldsymbol{\Sigma}_{ik}^{-1} \boldsymbol{\mu}_{ik}^p \\ \sqrt{\left(\frac{1}{2}\right) \frac{1-a_{11}}{1-a_{ii}}} c_{ik} \boldsymbol{\Sigma}_{ik}^{-1} \boldsymbol{\sigma}_{ik}^p \end{array} \right]_{k=1}^K \end{array} \right]_{i=1}^N \quad (6.54)$$

CHAPTER 7

SEGMENTAL MAP ADAPTATION, KULLBACK-LEIBLER DIVERGENCE, AND SUPERVECTOR DERIVATIONS FOR PSEUDO-2D HIDDEN MARKOV MODELS

This chapter goes over the algorithms for MAP adapting a P2DHMM and for generating the supervector equations for P2DHMMs. The reason these sections are separated from the previous section is that they represent work that is unique to this thesis in terms of implementation, if not both theory and implementation. This work follows directly from the theoretical background work shown in Chapters 4, 5, and 6.

7.1 Segmental MAP Adaptation of P2DHMMs

Instead of deriving a full EM based MAP adaptation algorithm for the P2DHMM, the MAP adaptation parameters for a P2DHMM are trained using a novel combination of the segmental K-means algorithm from the previous section and the MAP adaptation algorithm described above. Viterbi decoding is used to determine which superstate and which substate are being occupied at (x, y) . Alternatively, the superstate can be chosen by evaluating the log-likelihood of each sub-HMM for each row of observations and picking the superstate whose sub-HMM yielded the maximum log-likelihood. At this point, each row-observation sequence can be allocated as part of a set of observation sequences for each super-state, and the parameters of each sub-HMM can be trained using the 1D HMM MAP adaptation equations shown in Chapter 5. This can be written algorithmically as follows.

P2DHMM Segmental MAP Adaptation

1. For every training image row sequence $\mathcal{Y}_r$ for $1 \leq r \leq R$ where R is the total number of sequence rows, compute $\ln P(\mathcal{Y}_r|\lambda_i)$ for $1 \leq i \leq N_s$ where λ_i is a sub-state HMM corresponding to superstate i . Then, find the superstate to which it belongs using $i^* = \arg \max_j [\ln P(\mathcal{Y}_r|\lambda_j)]$.
2. Now that each training row sequence has been allocated to a state, run 1 pass of 1DHMM MAP adaptation for each sub-hmm to generate a new λ_i for $1 \leq i \leq N_s$
3. Repeat until convergence is reached

7.2 Kullback Leibler Divergence Between P2DHMMs

7.2.1 KLD for Converted P2DHMM

It was established in the section on UBMs that a P2DHMM can be decomposed into a 1DHMM under certain assumptions. Consider a left-to-right P2DHMM with sub-state HMMs that are also left-to-right. An example converted structure with 3 superstates and 3 sub-states per superstate is shown in Figure 7.1.

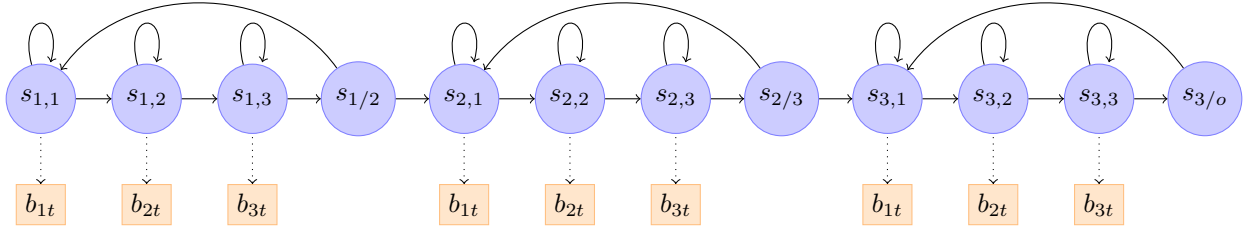


Figure 7.1: Reduction of P2DHMM to 1DHMM

The corresponding state transition matrix is shown in Equation (7.1).

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_{4,1} & 0 & 0 & 0 & a_{4,5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{5,5} & a_{5,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{6,6} & a_{6,7} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_{7,7} & a_{7,8} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{8,5} & 0 & 0 & 0 & a_{8,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{9,9} & a_{9,10} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,10} & a_{10,11} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11,11} & a_{11,12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{12,9} & 0 & 0 & 0 \end{bmatrix} \quad (7.1)$$

Clearly, the system is no longer purely left-to-right, even though it was created from a system that was composed of only left-to-right HMMs. Thus, the KLD bound developed by Silva et al. is not valid here. However, the KLD bound developed by Do could still apply, with some practical modification. From Do [28], the KLDR bound is given by Equation (7.2).

$$R(\lambda_1 || \lambda_2) = \lim_{n \rightarrow \infty} \boldsymbol{\pi}^T \mathbf{A}^n \mathbf{d} = \boldsymbol{\nu}^T \mathbf{d} \quad (7.2)$$

This bound is finite, thus the matrix in the limit must converge. This can be generally shown to be true by plotting the norm of the difference between subsequent computations of $\boldsymbol{\nu}$ for increasing n , as is shown in Figure 7.2.

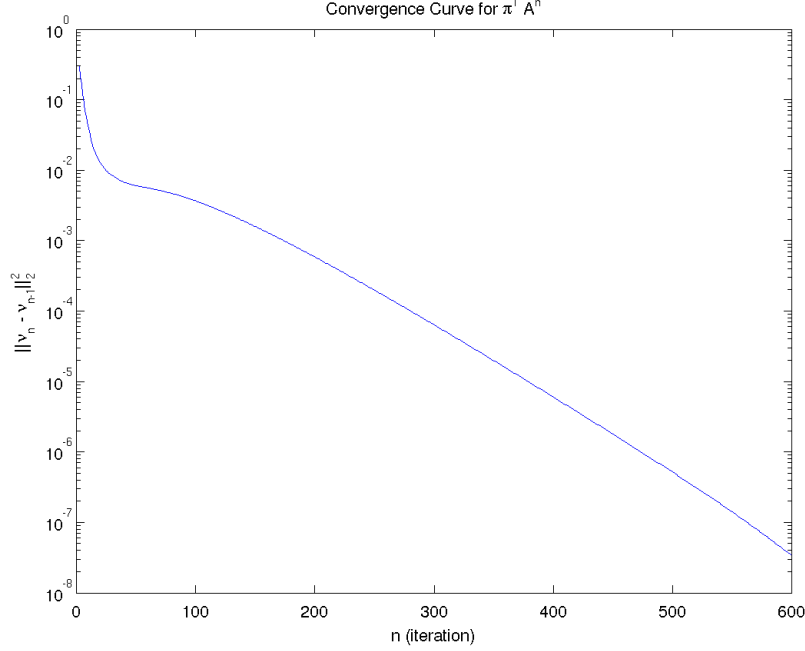


Figure 7.2: Normed Difference between $\pi^T A^n$ and $\pi^T A^{n-1}$ across Iterations

The result of the limit in Do's [28] bound can be roughly thought of as the long-term state residence probability. By convention, the vector $\boldsymbol{\pi}$ of initial state probabilities is $\pi_j = \delta(j)$. However, in the coming analysis of the supervector, it will make more sense to define $\boldsymbol{r}$ as something that is less sparse. Indeed, since conceptually $\boldsymbol{\nu}$ is an expression of the long term residence probability, it makes more sense to define $\boldsymbol{r}$ based on the actual state occupations of the data. In addition, all that is really required of the vector $\boldsymbol{\nu}$ is that it be a stationary distribution vector such that $\boldsymbol{\nu}^T = \boldsymbol{\nu}^T \mathbf{A}$. In other words, let $\boldsymbol{q}$ be the state sequence found using Viterbi decoding on some set of data with T observations, where $1 \leq q_t \leq N$ and $1 \leq t \leq T$. In this case, $\boldsymbol{r}$ can be defined as Equation (7.3).

$$r_i = \frac{1}{T} \sum_{t=1}^T \delta(q_t - i), \quad 1 \leq i \leq N, \quad r_i \in \boldsymbol{r} \quad (7.3)$$

With this new definition, the new KLD bound between two converted P2DHMMs can be defined as Equation (7.4).

$$R(\lambda_1 || \lambda_2) \leq \left(\lim_{n \rightarrow \infty} \boldsymbol{r}^T \mathbf{A}^n \right) (\mathbf{d}_{KLD}^A + \mathbf{d}_{KLD}^B) \quad (7.4)$$

The validity of these assumptions can be tested by looking at the convergence curve for a vector made up of the state occupation probabilities (Figure 7.3), as well as the difference between the $\boldsymbol{\nu}$ vector as the limit converges (Figure 7.4). The first of these plots shows that with $\boldsymbol{\pi}$ replaced with $\boldsymbol{r}$ in the equation for $\boldsymbol{\nu}$, the limit still converges. The second plot shows that even though the initial vectors are considerably different, they converge to values which are very close (within 10^{-6}).

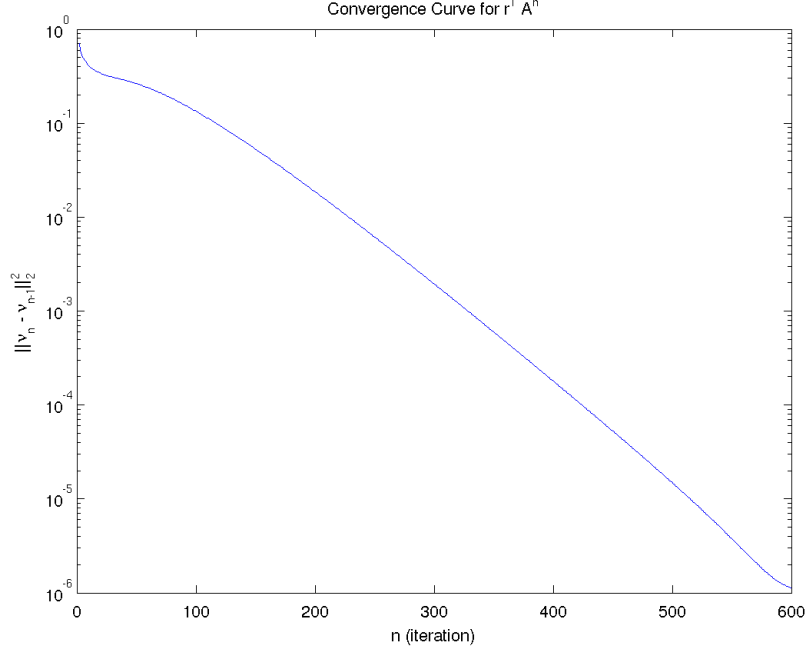


Figure 7.3: Convergence Curve for ν showing $\|r^T A^n - r^T A^{n-1}\|_2^2$

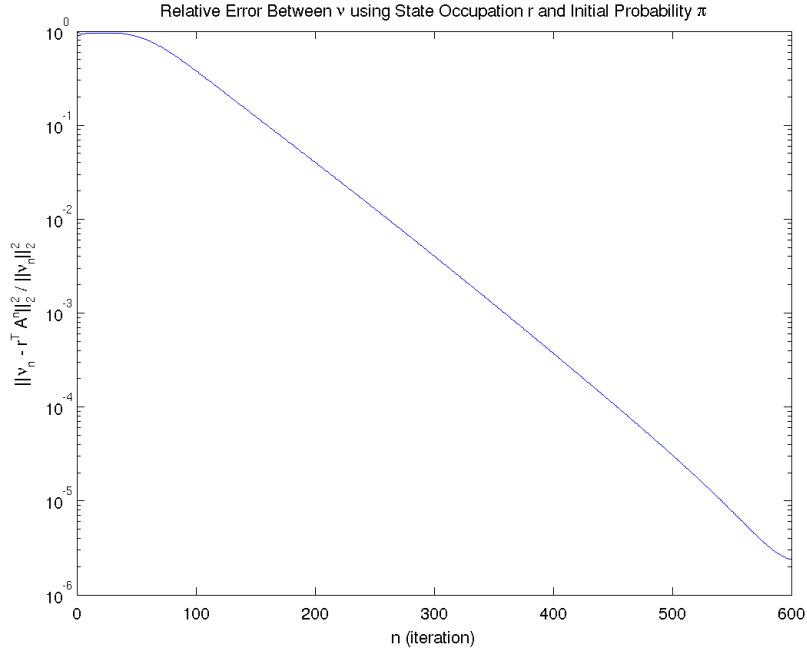


Figure 7.4: Relative Error between $\pi^T A^n$ and $r^T A^n$ across Iterations

The point behind this analysis is that if the KLD between two HMMs can be bounded by $R(\lambda_1 || \lambda_2) \leq \lim_{n \rightarrow \infty} r^T A^n d$, then as $n \rightarrow \infty$ the choice of r has less impact on the overall KLD. This is because the vector ν is assumed to be stationary to uphold the limit in the KLDR.

7.2.2 KLD for Embedded P2DHMM

The second approach to developing a supervector uses the embedded visualization from the previous chapter's discussion on P2DHMMs. An example P2DHMM is reproduced in Figure 7.5. By considering the emissions of the super-state HMM to be driven by each sub-HMM, the bound discussed in the 1DHMM section for left-to-right HMMs can be used to define the bound for the sub-HMM emissions. Then, the same bound equation can be used to define the super HMM bound. This final bound is shown in Equation (7.5).

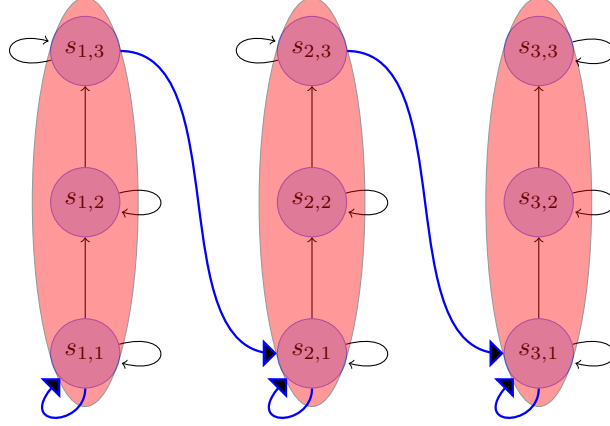


Figure 7.5: Example Left-to-Right Pseudo 2D HMM with $N_s = 3$ Superstates and 3 Substates per Superstate

$$D(\eta_1 || \eta_2) \leq D(\Pi_1 || \Pi_2) + \Pi_1^T (\mathbf{I} - \Omega_1)^{-1} [\mathbf{d}_{KLD}^\Omega + \mathbf{d}_{KLD}^\Lambda] \quad (7.5)$$

$$\omega_n(m) = \omega_{mn}, \quad \omega_{mn} \in \Omega \quad (7.6)$$

$$\mathbf{d}_{KLD}^\Omega = \begin{bmatrix} D(\omega_1^1 || \omega_1^2) & \dots & D(\omega_{N_s}^1 || \omega_{N_s}^2) \end{bmatrix}^T \quad (7.7)$$

$$\mathbf{d}_{KLD}^\Lambda = \begin{bmatrix} D(\lambda_1^1 || \lambda_1^2) & \dots & D(\lambda_{N_s}^1 || \lambda_{N_s}^2) \end{bmatrix}^T \quad (7.8)$$

$$D(\Pi_1 || \Pi_2) = \frac{1}{N_s} \sum_{n=1}^{N_s} \ln \Pi_1(n) - \ln \Pi_2(n) \quad (7.9)$$

$$D(\lambda_n^1 || \lambda_n^2) \leq D(\pi_n^1 || \pi_n^2) + (\pi_n^1)^T (\mathbf{I} - \mathbf{A}_n^1)^{-1} [\mathbf{d}_{KLD}^A + \mathbf{d}_{KLD}^B] \quad (7.10)$$

$$\mathbf{a}_j(i) = a_{ij}, \quad a_{ij} \in \mathbf{A} \quad (7.11)$$

$$\mathbf{d}_{KLD}^A = \begin{bmatrix} D(\mathbf{a}_1^1 || \mathbf{a}_1^2) & \dots & D(\mathbf{a}_N^1 || \mathbf{a}_N^2) \end{bmatrix}^T \quad (7.12)$$

$$\mathbf{d}_{KLD}^B = \begin{bmatrix} D(b_1^1 || b_1^2) & \dots & D(b_N^1 || b_N^2) \end{bmatrix}^T \quad (7.13)$$

7.3 P2DHMM Supervectors

Now that the KLDs for the two P2DHMM cases have been defined, their supervectors can be derived in the same way that Tang [15] derived the 1DHMM supervectors.

7.3.1 P2HMM Supervectors Approach 1 - Conversion to 1D HMM

The work up to this point (i.e., the supervector equations for the 1DHMM case and the GMM case) is basically taken directly from the work done by Tang. The content from this point on represents the departure from Tang's original work by extending Tang's 1DHMM formulas to the two P2DHMM cases described in UBM chapter.

The first approach to developing a supervector uses the fact that a P2DHMM can be converted to a 1DHMM by modelling the superstate transitions as non-emitting states that only consume one observation and by modelling the superstate self-loops as returns from the non-emitting states.

In this case, Do's approach is useful for determining the HMM supervector equations. However, since $\pi_j = \delta(j)$, the vector becomes too sparse to be useful. Instead, the long term state occupation probabilities are used.

$$r_i = \frac{1}{T} \sum_{t=1}^T \delta(q_t - i), \quad 1 \leq i \leq N, \quad r_i \in \mathbf{r} \quad (7.14)$$

With this in mind, the supervector equation for the 1DHMM case can be directly adapted to this case.

$$\mathbf{s}_p = \begin{bmatrix} \left[\sqrt{\frac{r_{ubm,k,i}}{a_{ubm,i,l}}} a_{p,i,l} \right]_{l=1}^N \\ \sqrt{\frac{r_i}{2w_{ubm,k,i}}} w_{p,k,i} \\ \frac{1}{2} \sqrt{r_i w_{ubm,k,i} \Sigma_{ubm,k,i}^{-1}} \boldsymbol{\sigma}_{p,k,i} \\ \left[\frac{1}{\sqrt{2}} \sqrt{w_{ubm,k,i} r_i \Sigma_{ubm,k,i}^{-1}} \boldsymbol{\mu}_{p,k,i} \right]_{k=1}^K \end{bmatrix}_{i=1}^K \quad (7.15)$$

7.3.2 P2HMM Supervectors Approach 2 - Embedded (Superstate) Approach

The supervector equation for the 2D embedded HMM case can be found by expanding the KLDs for each sub-state HMM and then for the superstate HMM. In Tang's [15] thesis, he shows that the inverse matrix $(\mathbf{I} - \mathbf{A})^{-1}$ is upper triangular such that:

$$(\mathbf{I} - \mathbf{A})_{ij}^{-1} = \begin{cases} \frac{1-a_{ii}}{1-a_{jj}} & i \leq j \\ 0 & i > j \end{cases} \quad (7.16)$$

When this vector is multiplied with $\boldsymbol{\pi}^T$, given that $\pi_i = \delta(i)$, the resulting vector is:

$$\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{A})^{-1} = \left[\frac{1-a_{11}}{1-a_{ii}} \right]_{i=1}^N \quad (7.17)$$

Thus, the full product $\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{A})^{-1}\mathbf{d}$ can be written as:

$$\left[\frac{1-a_{11}}{1-a_{11}} \quad \dots \quad \frac{1-a_{11}}{1-a_{NN}} \right]^T \begin{bmatrix} d_1 \\ \vdots \\ d_N \end{bmatrix} = \sum_{i=1}^N \frac{1-a_{11}}{1-a_{ii}} d_i \quad (7.18)$$

As long as the initial state distributions are considered to be equal for all HMMs (thus making the KLD between them 0 by definition), the KLD between two left-to-right HMMs is thus:

$$D_s(\lambda_1 || \lambda_2) \leq \sum_{i=1}^N \frac{1-a_{11}}{1-a_{ii}} (D_s(\mathbf{a}_i^1 || \mathbf{a}_i^2) + D_s(b_i^1 || b_i^2)) \quad (7.19)$$

This same result can be extended to the superstate HMM KLD since the superstate HMM is still left-to-right.

$$\boldsymbol{\Pi}^T(\mathbf{I} - \boldsymbol{\Omega})^{-1}\mathbf{d} = \sum_{m=1}^{N_s} \frac{1-\omega_{11}}{1-\omega_{mm}} d_m \quad (7.20)$$

$$\boldsymbol{\Pi}_1^T(\mathbf{I} - \boldsymbol{\Omega}_1)^{-1}(\mathbf{d}_{KLD}^\omega + \mathbf{d}_{KLD}^\Lambda) = \sum_{m=1}^{N_s} \frac{1-\omega_{11}}{1-\omega_{mm}} (D_s(\boldsymbol{\omega}_m^1 || \boldsymbol{\omega}_m^2) + D_s(\lambda_m^1 || \lambda_m^2)) \quad (7.21)$$

$$D_s(\eta_1 || \eta_2) \leq \sum_{m=1}^{N_s} \frac{1-\omega_{11}}{1-\omega_{mm}} (D_s(\boldsymbol{\omega}_m^1 || \boldsymbol{\omega}_m^2) + D_s(\lambda_m^1 || \lambda_m^2)) \quad (7.22)$$

The symmetric KLD between the state-transition vectors can now be defined using the definition for the KLD between two PMFs defined in the previous section.

$$D_s(\boldsymbol{\omega}_m^1 || \boldsymbol{\omega}_m^2) \approx \frac{1}{2} \sum_{n=1}^{N_s} \left(\frac{\omega_m^1(n) - \omega_m^2(n)}{\sqrt{\omega_m^{ubm}(n)}} \right)^2 \quad (7.23)$$

Using the above equation, the KLD between the two P2DHMMs can be further expanded.

$$D_s(\eta_1 || \eta_2) \lesssim \sum_{m=1}^{N_s} \frac{1-\omega_{11}}{1-\omega_{mm}} \left(\frac{1}{2} \sum_{n=1}^{N_s} \left(\frac{\omega_{mn}^1 - \omega_{mn}^2}{\sqrt{\omega_{mn}^{ubm}}} \right)^2 + \sum_{i=1}^{N_m} \frac{1-a_{11}}{1-a_{ii}} \left(\frac{1}{2} \sum_{j=1}^{N_m} \left(\frac{a_{ij}^1 - a_{ij}^2}{\sqrt{a_{ij}^{ubm}}} \right)^2 + D_s(b_i^1 || b_i^2) \right) \right) \quad (7.24)$$

$$\begin{aligned} D_s(\eta_1 || \eta_2) &\lesssim \frac{1}{2} \sum_{m=1}^{N_s} \sum_{n=1}^{N_s} \frac{1-\omega_{11}}{1-\omega_{mm}} \left(\frac{\omega_{mn}^1 - \omega_{mn}^2}{\sqrt{\omega_{mn}^{ubm}}} \right)^2 + \frac{1}{2} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \sum_{j=1}^{N_m} \frac{1-\omega_{11}}{1-\omega_{mm}} \frac{1-a_{11}}{1-a_{ii}} \left(\frac{a_{ij}^1 - a_{ij}^2}{\sqrt{a_{ij}^{ubm}}} \right)^2 \\ &\quad + \frac{1}{2} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \frac{1-\omega_{11}}{1-\omega_{mm}} \frac{1-a_{11}}{1-a_{ii}} D_s(b_i^1 || b_i^2) \end{aligned} \quad (7.25)$$

From the previous chapter, it has already been established that the symmetric KLD between two GMMs is:

$$\begin{aligned}
D_s(b_i^1 || b_i^2) &\lesssim \frac{1}{2} \sum_{k=1}^K \left(\frac{w_1(k) - w_2(k)}{\sqrt{w_{ubm}(k)}} \right)^2 + \frac{1}{4} \sum_{k=1}^K w_{ubm}(k) \|\Sigma_{ubm,k}^{-1}(\sigma_{1,k} - \sigma_{2,k})\|_2^2 \\
&\quad + \frac{1}{2} \sum_{k=1}^K w_{ubm}(k) \|\sqrt{\Sigma_{ubm,k}^{-1}}(\mu_{1,k} - \mu_{2,k})\|_2^2
\end{aligned} \tag{7.26}$$

Substituting this result into the expression for the KLD between two P2DHMMs, the result is:

$$\begin{aligned}
D_s(\eta_1 || \eta_2) &\lesssim \frac{1}{2} \sum_{m=1}^{N_s} \sum_{n=1}^{N_s} \left(\frac{1 - \omega_{11}}{1 - \omega_{mm}} \right) \left(\frac{\omega_{mn}^1 - \omega_{mn}^2}{\sqrt{\omega_{mn}^{ubm}}} \right)^2 \\
&\quad + \frac{1}{2} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \sum_{j=1}^{N_m} \left(\frac{1 - \omega_{11}}{1 - \omega_{mm}} \right) \left(\frac{1 - a_{11}}{1 - a_{ii}} \right) \left(\frac{a_{ij}^1 - a_{ij}^2}{\sqrt{a_{ij}^{ubm}}} \right)^2 \\
&\quad + \frac{1}{4} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \sum_{k=1}^K \left(\frac{1 - \omega_{11}}{1 - \omega_{mm}} \right) \left(\frac{1 - a_{11}}{1 - a_{ii}} \right) \left(\frac{w_1(k) - w_2(k)}{\sqrt{w_{ubm}(k)}} \right)^2 \\
&\quad + \frac{1}{8} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \sum_{k=1}^K w_{ubm}(k) \left(\frac{1 - \omega_{11}}{1 - \omega_{mm}} \right) \left(\frac{1 - a_{11}}{1 - a_{ii}} \right) \|\Sigma_{ubm,k}^{-1}(\sigma_{1,k} - \sigma_{2,k})\|_2^2 \\
&\quad + \frac{1}{4} \sum_{m=1}^{N_s} \sum_{i=1}^{N_m} \sum_{k=1}^K w_{ubm}(k) \left(\frac{1 - \omega_{11}}{1 - \omega_{mm}} \right) \left(\frac{1 - a_{11}}{1 - a_{ii}} \right) \|\sqrt{\Sigma_{ubm,k}^{-1}}(\mu_{1,k} - \mu_{2,k})\|_2^2
\end{aligned} \tag{7.27}$$

At this point, the corresponding supervector can be written by inspection.

$$\mathbf{s}_p = \left[\begin{array}{c} \left[\sqrt{\frac{1 - \omega_{11}}{(1 - \omega_{mm})\omega_{mn}}} \omega_{mn}^p \right]_{n=1}^{N_s} \\ \left[\left[\sqrt{\frac{1 - \omega_{11}}{1 - \omega_{mm}}} \frac{1 - a_{11m}}{(1 - a_{iim})} a_{ijm}^p \right]_{j=1}^{N_m} \right]_{i=1}^{N_m} \\ \left[\left[\left[\sqrt{\left(\frac{1}{2}\right) \frac{1 - \omega_{11}}{1 - \omega_{mm}}} \frac{1 - a_{11m}}{(1 - a_{iim})} c_{ikm}^p \right]_{k=1}^K \right]_{i=1}^{N_m} \right]_{m=1}^{N_s} \\ \left[\left[\frac{1}{2} \sqrt{\frac{1 - \omega_{11}}{1 - \omega_{mm}}} \frac{1 - a_{11m}}{1 - a_{iim}} c_{ikm} \Sigma_{ikm}^{-1} \mu_{ikm}^p \right]_{k=1}^K \right]_{i=1}^{N_m} \\ \left[\left[\sqrt{\left(\frac{1}{2}\right) \frac{1 - \omega_{11}}{1 - \omega_{mm}}} \frac{1 - a_{11m}}{1 - a_{iim}} c_{ikm} \Sigma_{ikm}^{-1} \sigma_{ikm}^p \right]_{k=1}^K \right]_{i=1}^{N_m} \end{array} \right]_{m=1}^{N_s} \tag{7.28}$$

A more intuitive approach would be to simply concatenate the supervectors from all the sub-state HMMs into a single supervector for this particular case. While this approach will work, the supervector equation derived above is generally a stronger representation since it matches the expression for the KLD bound between two P2DHMMs.

CHAPTER 8

FINAL RESULTS

8.1 Data

For the purposes of testing the algorithms developed in the previous chapters, the ORL [4] database was used. The relevant information regarding the ORL database is shown in Table 8.1.

Table 8.1: ORL Database Relevant Information

Name	ORL Database
Image Size	112 x 92
Subjects	40
Images per Subject	10
Total Images	400
Lighting Variation	slight

Since there are 400 images in the database, 200 were used for the test set, and the remaining 200 were used for the training set. To accomplish this, the first 5 images for each subject were sent to the training set, and the last 5 images for each subject were sent to the test set.

8.2 MAP Adapted HMM Maximum Likelihood Recognizer

For this test, the system was implemented with some UBM topology (1DHMM, converted P2DHMM, or embedded P2DHMM) and a maximum likelihood slicer was used to determine which subject class a current test subject belonged. To accomplish this, the likelihood of the observation space of the current test subject, $\mathcal{O}_{test}$, given a trained model, λ_{train} , is evaluated as $P(\mathcal{O}_{test}, \lambda_{train})$. Whichever model yields the maximum likelihood is the model class to which the test subject must belong.

$$\lambda_{test} = \arg \max_{i \in S_{train}} \ln P(\mathcal{O}_{test} | \lambda_i) \quad (8.1)$$

At this point, the recognition rates can be established simply by comparing the class detected by the recognizer for each test subject with the actual test subject class.

8.2.1 1D HMM

For a 1D HMM, it was assumed that the HMM for the UBM was a five state HMM with each state corresponding to a sequential facial feature (forehead, eyes, nose, mouth, chin). Each emission model was

set to be a GMM with K mixtures and diagonal covariance matrices. The model parameters were pre-trained by segmenting the test images using a cascaded classifier, collecting the dense SIFT vectors from each region, and then pre-training the GMM from each of these segmented feature spaces. These models represented the initial parameters for the state emission models. The UBM was trained using 1 image for each subject in the test database. Then, using the UBM, the models were adapted to each subject in the training database and each subject in the test database using the full set of test image and training images.

At this point, the maximum likelihood recognizer described above was used to classify each test subject. The recognition results were recorded when the MAP adaptation weight parameter (τ) was varied and when the number of mixture components for the emission GMMs was varied. The resulting plot is shown in Figure 8.1.

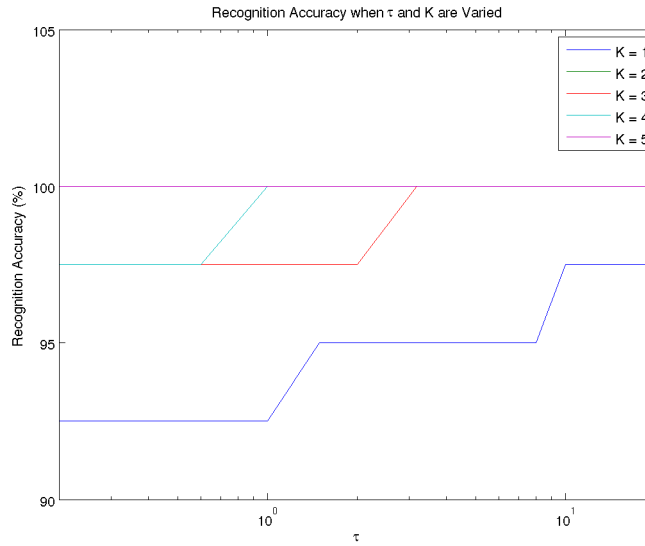


Figure 8.1: Accuracy for Max-LL Recognition when τ and K are Varied

Clearly, the $K = 5$ mixture components case is the optimal for this set of data (the ORL database). In addition, when $K = 5$, the system is invariant to changes in the adaptation parameter τ . However, it is possible to achieve 100% accuracy with a variety of combinations of K and τ .

This plot also shows that the MAP adaptation method with max-log-likelihood slicing is, in general, a valid method of recognition. In addition, it shows that this method can be used to beat the existing results, such as the Eigenface approach or the P2DHMM approach as outlined by Samaria [3].

8.2.2 P2DHMM Decomposition

For a P2DHMM, more configuration parameters were varied than in the 1DHMM case. In addition to τ and K , the number of sub-states per super-state N_i for $1 \leq i \leq N_s$ were varied, and the number of super-states was varied. In this case, the number of super-states N_s was allowed to be 3 or 5. With the $N_s = 3$ case, the three superstates corresponded to the eyes, nose, and mouth facial features. In addition, the eyes and mouth facial features were allowed to encompass the forehead and chin features (respectively)

so that every row in the image could correspond with a specific state. With the $N_s = 5$ case, the segmentation method outlined above still holds.

The tables for the $N_s = 3$ and $N_s = 5$ cases are shown in Tables 8.2 through 8.4 and Tables 8.5 through 8.7 , respectively.

Table 8.2: 3 Superstates, 2 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	95	95	95	95	95	95	95	95	95	97.5
2	97.5	97.5	97.5	97.5	97.5	95	95	97.5	97.5	97.5
3	97.5	97.5	97.5	95	95	95	95	95	97.5	97.5
4	95	95	95	95	95	95	97.5	97.5	95	97.5
5	95	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	100

Table 8.3: 3 Superstates, 3 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	95	97.5
2	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5
3	92.5	92.5	92.5	92.5	92.5	92.5	97.5	97.5	97.5	100
4	90	90	87.5	87.5	90	87.5	92.5	97.5	97.5	97.5
5	85	85	85	85	85	85	87.5	90	97.5	100

Table 8.4: 3 Superstates, 4 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5	97.5
2	95	95	95	95	97.5	97.5	97.5	97.5	97.5	97.5
3	92.5	92.5	92.5	92.5	92.5	95	95	97.5	97.5	97.5
4	95	92.5	92.5	92.5	92.5	92.5	92.5	95	100	100

Table 8.5: 5 Superstates, 2 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	97.5
2	92.5	92.5	92.5	92.5	92.5	90	92.5	92.5	92.5	92.5
3	90	87.5	90	87.5	87.5	90	92.5	92.5	92.5	100
4	87.5	90	87.5	87.5	85	87.5	90	92.5	92.5	97.5
5	77.5	77.5	77.5	77.5	77.5	77.5	80	92.5	92.5	95

Table 8.6: 5 Superstates, 3 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	95	95	95	95	95	95	95	95	95	95
2	85	85	85	87.5	87.5	87.5	92.5	95	95	100
3	85	85	77.5	80	77.5	80	82.5	80	85	95
4	75	72.5	70	62.5	62.5	67.5	75	77.5	85	97.5
5	65	65	65	65	65	75	80	90	92.5	97.5

Table 8.7: 5 Superstates, 4 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	95	95	95	95	95	95	95	95	95	95
2	80	77.5	77.5	77.5	77.5	77.5	80	82.5	85	92.5
3	77.5	77.5	75	75	75	75	82.5	87.5	87.5	95
4	62.5	60	60	60	60	65	75	82.5	85	97.5
5	40	45	40	37.5	37.5	40	50	67.5	77.5	97.5

8.2.3 P2DHMM Embedded

For the embedded P2DHMM, 100% accuracy was achieved with a few combinations of N_s , N , τ , and K . The most consistent match was with 4 mixture components, 4 sub-states per superstate, and 3 superstates. Strangely, the match was less consistent with the embedded case than it was for the decomposition case. This may be due to the fact that full Baum-Welch was not used in this case. However, the larger point is that 100% accuracy was achieved using this method.

8.3 Verification of KLD Bounds

Before getting to the recognition results obtained by applying a supervector-based classifier, it is useful to first establish the validity of the KLD bounds shown in Chapters 6 and 7, since the supervectors arise directly from these bounds. To test the validity of the bound itself, the KLDs obtained between each HMM using the derived bounds and using the Monte Carlo method could be compared. However, this was not done in this analysis due to time constraints. In this case, validity was tested by determining the KLD between every HMM which was MAP adapted for every subject in the database and every HMM which was MAP adapted for every subject in the test database using the bounds established in Chapters 6 and 7. In theory, if using the divergence metric for recognition directly is a valid approach, then classification using the norm of the difference between two supervectors should likewise be valid. In this way, a test subject could be classified by taking the KLD between a test subject j and every training subject i , and classifying it according to minimum KLD.

$$L_j = i^* = \arg \min_i [D_s(\lambda_{train,i}, \lambda_{test,j})] \quad (8.2)$$

In addition, the KLD is between the test HMM and every training HMM is normalized.

$$D_s(\lambda_{train,i}, \lambda_{test,j}) = \frac{D_s(\lambda_{train,i}, \lambda_{test,j})}{\max_l D_s(\lambda_{train,l}, \lambda_{test,j})} \quad (8.3)$$

Likewise, for a P2DHMM, the conversion equations can be written in terms of η .

$$L_j = i^* = \arg \min_i [D_s(\eta_{train,i}, \eta_{test,j})] \quad (8.4)$$

$$D_s(\eta_{train,i}, \eta_{test,j}) = \frac{D_s(\eta_{train,i}, \eta_{test,j})}{\max_l D_s(\eta_{train,l}, \eta_{test,j})} \quad (8.5)$$

In this way, there is no net effect on the classification itself, but the subsequent plots showing recognition trends become considerably clearer. The resulting plots showing the KLD between every test versus train combination are shown in Figure 8.2 for the 1DHMM case, Figure 8.3 for the P2DHMM converted to a 1DHMM case, and Figure 8.4 for the embedded P2dHMM case. In every case, the results very clearly show the KLD between the two HMMs is generally at a minimum when the test subject corresponds to the correct training subject. In addition, the cases below would correspond to 100% accuracy using 1DHMMs, 90% accuracy with a converted P2DHMM, and 95% accuracy with an embedded P2DHMM. The parameters in every case were chosen arbitrarily. It will be shown later that with different parameter choices, 100% accuracy can be obtained using any of the above methods with this dataset. In any case, the results clearly show that this approach at least has merit.

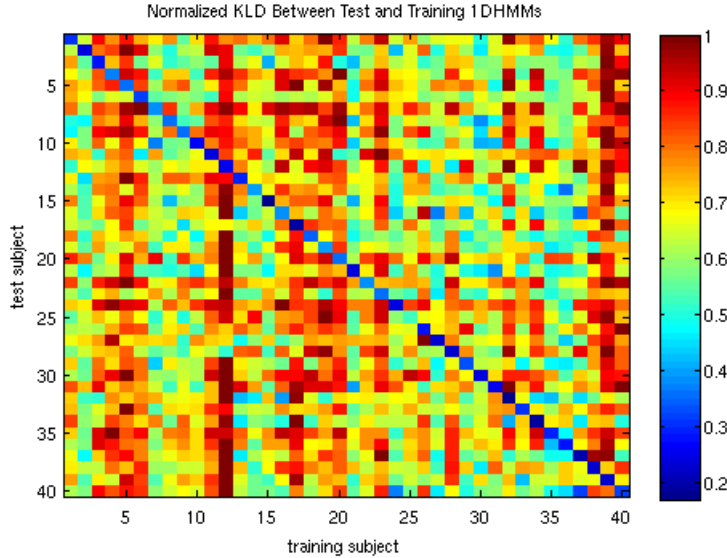


Figure 8.2: KLD Results for 1DHMM

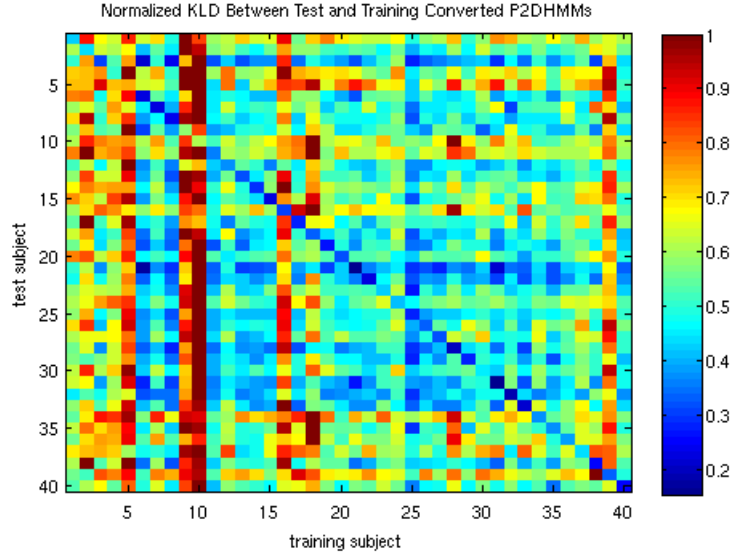


Figure 8.3: KLD Results for Converted P2DHMM

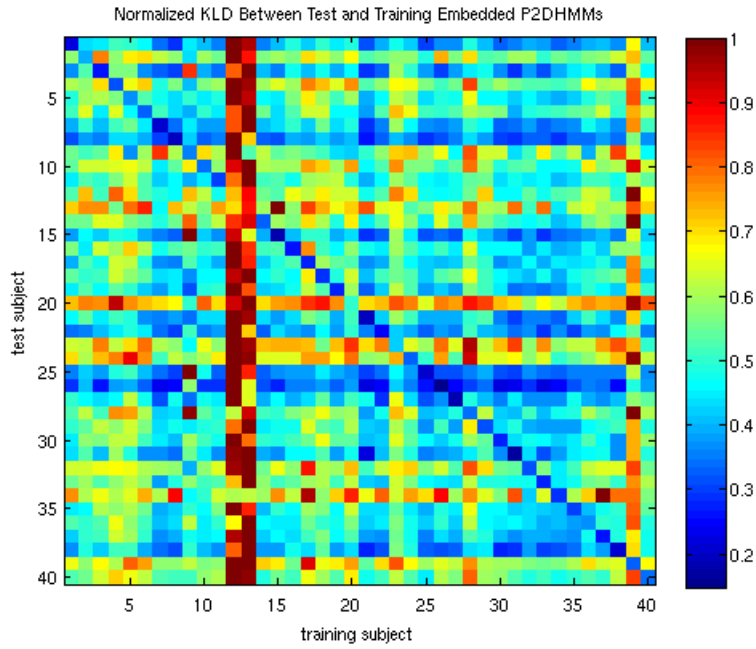


Figure 8.4: KLD Results for Embedded P2DHMM

8.4 Supervector Recognition using Nearest-Neighbor Classifier

For this test, the system was implemented with some UBM topology (1DHMM, converted P2DHMM, or embedded P2DHMM) and a nearest neighbor classifier was used on all of the supervectors generated in the test and training set. The distance metric used for the classifier was the cosine distance metric.

$$d(\mathbf{s}_{train,i}, \mathbf{s}_{test,j}) = 1 - \frac{\mathbf{s}_{train,i}^T \mathbf{s}_{test,j}}{\sqrt{\mathbf{s}_{train,i}^T \mathbf{s}_{train,i}} \sqrt{\mathbf{s}_{test,j}^T \mathbf{s}_{test,j}}} \quad (8.6)$$

The class L_j chosen for test subject j is given by

$$L_j = i^* = \arg \min_i [d(\mathbf{s}_{train,i}, \mathbf{s}_{test,j})] \quad (8.7)$$

8.4.1 1D HMM

The same basic setup for the maximum likelihood recognizer was used for this case as well. The only difference was that the distance metric described above was used for the HMM supervectors between the test and training models. The recognition results were recorded when the MAP adaptation weight parameter (τ) was varied and when the number of mixture components for the emission GMM's was varied. The resulting plot is shown in Figure 8.5.

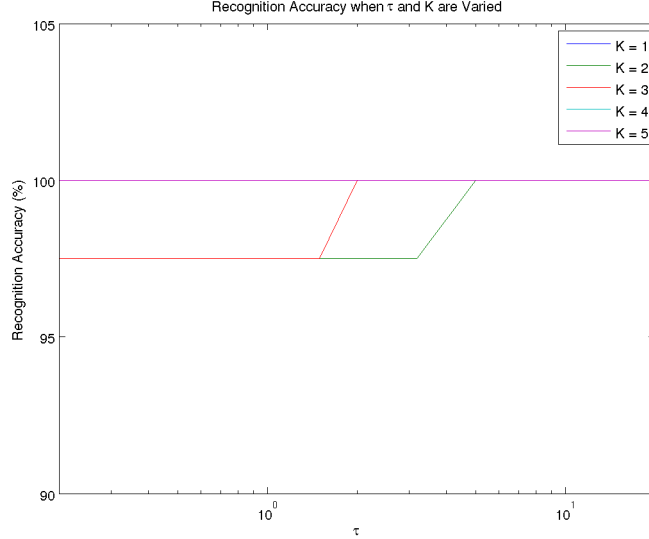


Figure 8.5: Accuracy for Supervector Approach when τ and K are Varied

When comparing the results from the maximum likelihood recognizer and the supervector classifier, the supervector approach performed at least as well as the maximum likelihood recognizer for all cases except when the number of mixture components was 1 (single Gaussian emission case). However, the supervector approach was able to either outperform the maximum likelihood approach or obtain 100% accuracy in 60% of the total test cases. This is shown in Figure 8.6.

8.4.2 P2DHMM Decomposition

The same basic setup for the maximum likelihood recognizer was used for this case as well. The only difference was that the distance metric described above was used for the HMM supervectors between the

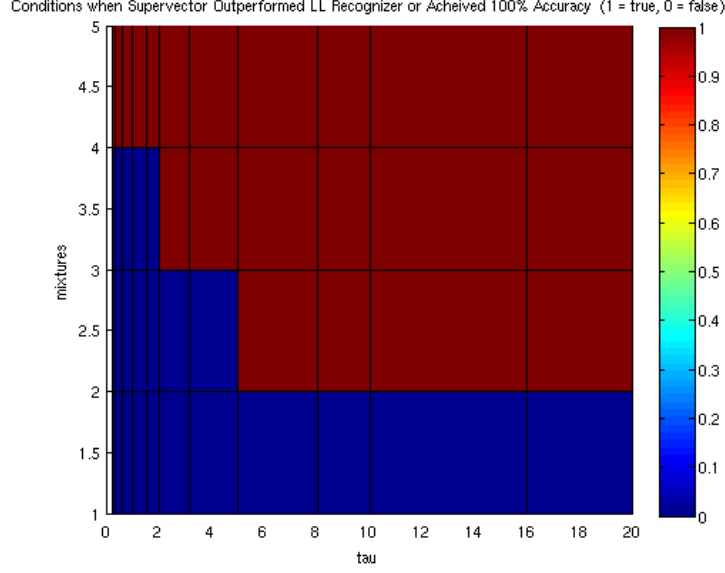


Figure 8.6: Conditions where Accuracy of Supervector Approach was 100% or Supervector Approach Outperformed Likelihood Approach (Red = True, Blue = False) for 1DHMM Case

test and training models. The tables for the $N_s = 3$ and $N_s = 5$ cases are shown in Tables 8.8 through 8.10 and Tables 8.11 through 8.13 , respectively.

Table 8.8: 3 Superstates, 2 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	82.5	87.5	87.5	87.5	85	90	92.5	90	90	90
2	90	90	90	87.5	85	85	85	85	85	85
3	95	95	92.5	92.5	92.5	90	92.5	97.5	97.5	97.5
4	87.5	85	87.5	85	85	85	87.5	90	87.5	87.5
5	92.5	90	85	85	82.5	82.5	82.5	80	80	80

Table 8.9: 3 Superstates, 3 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	95	92.5
2	97.5	97.5	92.5	92.5	92.5	95	95	95	95	92.5
3	90	90	87.5	87.5	87.5	87.5	85	87.5	87.5	90
4	97.5	97.5	95	95	95	95	95	95	95	95
5	97.5	97.5	97.5	95	95	95	92.5	92.5	95	92.5

Table 8.10: 3 Superstates, 4 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	95	95	95	95	97.5	95	95	95	95	95
2	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5
3	87.5	85	82.5	87.5	87.5	87.5	87.5	87.5	87.5	85
4	92.5	92.5	92.5	92.5	92.5	92.5	95	95	92.5	92.5

Table 8.11: 5 Superstates, 2 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	95	97.5	97.5	97.5	97.5	97.5	95	95	95	92.5
2	100	100	100	100	95	95	97.5	97.5	97.5	97.5
3	95	95	95	95	92.5	90	90	90	90	92.5
4	97.5	97.5	95	95	92.5	92.5	90	92.5	92.5	92.5
5	95	95	95	95	95	95	92.5	95	95	97.5

Table 8.12: 5 Superstates, 3 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	87.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5	92.5
2	97.5	97.5	97.5	95	95	95	95	95	95	95
3	95	95	95	92.5	92.5	92.5	92.5	92.5	92.5	92.5
4	97.5	97.5	95	95	95	95	95	95	97.5	97.5
5	95	100	100	100	100	100	100	100	97.5	100

Table 8.13: 5 Superstates, 4 Sub-States per Superstate

K/τ	0.317	0.6	1	1.5	2	3.17	5	8	10	16
1	90	90	90	92.5	92.5	92.5	85	75	90	50
2	92.5	92.5	92.5	92.5	92.5	92.5	80	92.5	92.5	92.5
3	95	92.5	95	95	95	95	55	97.5	97.5	100
4	95	95	90	90	90	85	87.5	87.5	90	92.5
5	87.5	87.5	87.5	57.5	87.5	87.5	90	90	90	87.5

When comparing the results from the maximum likelihood recognizer and the supervector classifier, the supervector approach performed at least as well as the maximum likelihood recognizer in 49.3% of the recorded test cases. 100% accuracy was achieved in 13 distinct test cases. In the majority of these cases, the number of superstates was 5, the number of mixtures was 5, and the number of sub-states was 3.

8.4.3 P2DHMM Embedded

As was the case in the maximum likelihood recognizer, 100% accuracy could be achieved using the supervector approach using the same architectural parameters (K, τ, N_s, N) as in the maximum likelihood recognizer case. One interesting thing to note here is that the supervector approach generally outperforms the maximum likelihood approach in approximately 55% of the test cases run. However, the recognition results are far more stable across subsequent runs of the algorithm, and far more likely to reach 100% accuracy, using the P2DHMM to 1DHMM conversion method.

8.5 Recognition Results when the Number of Subjects used to Train the UBM is Varied

In another experiment, the number of subjects used to train the UBM was varied. The idea behind this experiment was to establish how many subjects could be left out of the UBM database while still getting good accuracy. If a sufficiently large subset of subjects could be left out of the pool for UBM training, then these methods could perhaps be useful for training a representative classifier using the initial data. Furthermore, the MAP adaptation method could still possibly be used on data left out of the initial database to correctly classify the new data (as long as the new images are taken under the same assumptions as the others in the database). The results for the 1DHMM and converted P2DHMM are shown in Figures 8.7 and 8.8, respectively.

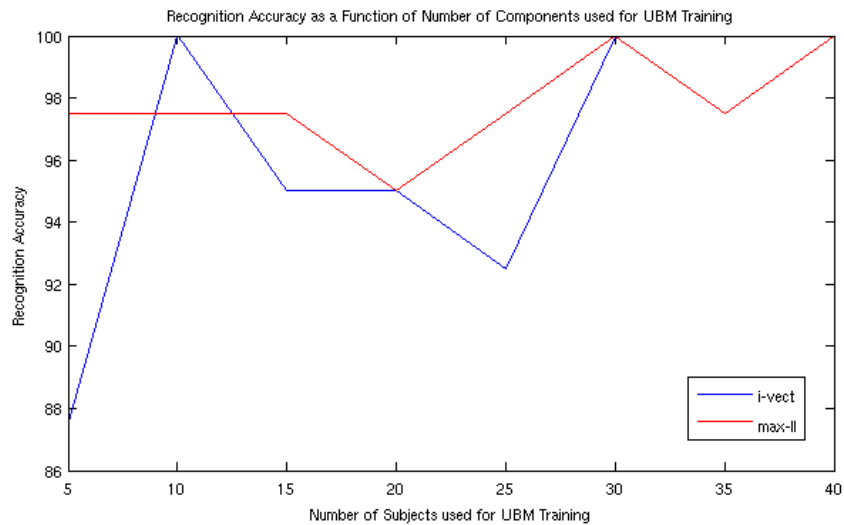


Figure 8.7: Recognition Accuracy as a Function of Number of Components used for UBM Training (1DHMM Case)

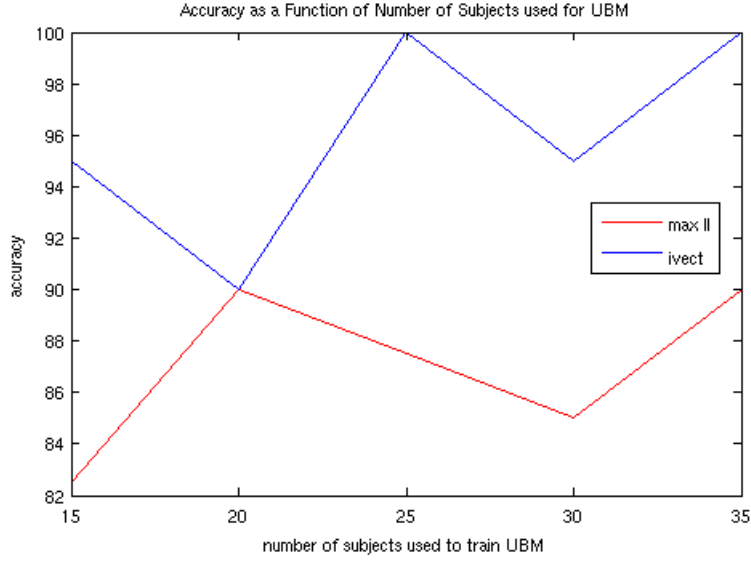


Figure 8.8: Recognition Accuracy as a Function of Number of Components used for UBM Training (Converted PDHMM Case)

The results show that (generally) the accuracy increases as the number of subjects used in the UBM increase. However, in both the 1DHMM and converted P2DHMM cases, not all of the subjects are required to get 100% accuracy. From the combined data, it can be assumed that 30 out of the 40 subjects (75%) are approximately required to get 100% accuracy. This number is reached by assuming that the first 100% in the 1DHMM curve is a fluke. However, the data is very noisy. This is probably partly due to the fact that there were an extremely limited number of trials run. In addition, the curve will vary with the adaptation and architectural training parameters (the above curves are for a single structural configuration). In any case, the results show that while it helps to have all the data in the UBM training pool, it is not necessarily required to get accuracy upwards of 90%.

8.6 Recognition Results when Number of Images per Subject used for MAP Adaptation is Varied

In the final experiment, the number of images per subject used for MAP adaptation was varied. There are 5 images per subject for each of the 40 subjects in both the test and training datasets. However, if all 5 images are not needed for superior recognition results, then that would be desirable from a practical implementation standpoint. The results for the 1DHMM and converted P2DHMM cases are shown in Figures 8.9 and 8.10, respectively. The results show that 100% accuracy cannot be reached with less than 5 samples per subject used for MAP adaptation, and that accuracy increases as the number of MAP adaptation images per subject increases.

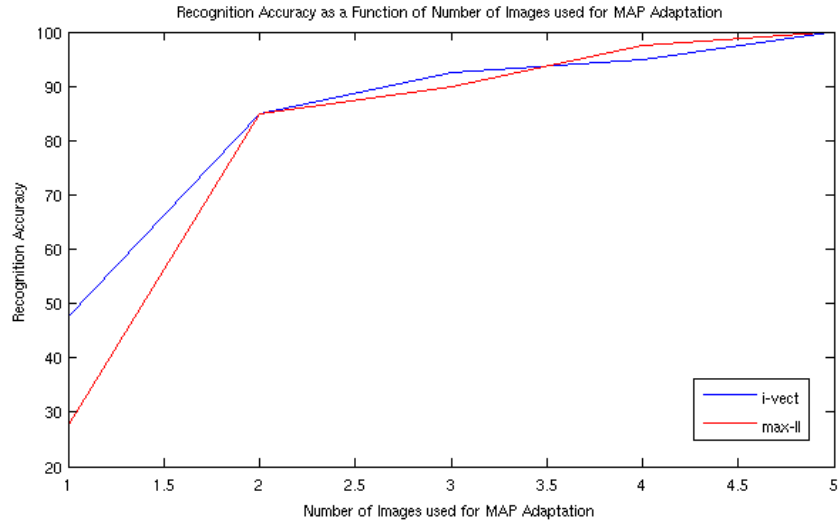


Figure 8.9: Recognition Accuracy as a Function of Number of Components used for UBM Training (1DHMM Case)

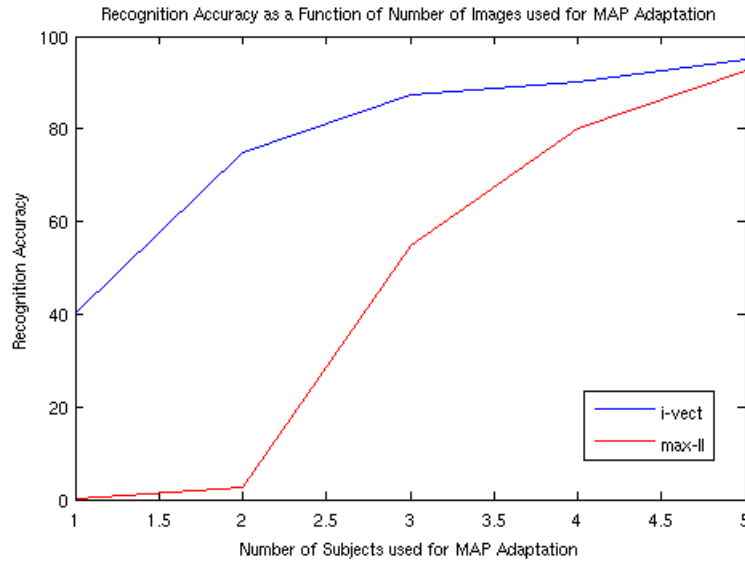


Figure 8.10: Recognition Accuracy as a Function of Number of Components used for UBM Training (Converted P2DHMM Case)

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

The three architectural methods were able to achieve 100% accuracy with both the maximum likelihood recognizer and the supervector method. However, in general it would seem that the biggest boost to recognition accuracy when compared to previous work by Samaria [3] or Nefian and Hayes [10] is that the UBM - MAP adaptation approach taken in this thesis seems to consistently outperform the subject-based full HMM training done in these cases. In addition, while the 1DHMM case consistently achieves 100% accuracy using this approach, the P2DHMM cases reach somewhat less consistently when the architecture parameters are varied. However, this does not necessarily mean that the 1DHMM method is stronger than the P2DHMM methods. At best, it only means that the recognition rates for the P2DHMM cases are far more sensitive to variation in the architectural parameters than they are for the 1DHMM cases. Indeed, since 100% accuracy is achievable in all test cases, all that can be said is that all of the methods put forth in this thesis either meet or exceed previous recognition accuracy results published in the various other papers mentioned in the Introduction. To truly fully test the robustness of the algorithm, it should be tested against other face recognition databases. The ORL database just happens to be a good benchmark for testing whether or not the algorithm under investigation is at least valid.

The other larger point to address is the comparison of not just P2DHMMs to 1DHMMs, but HMM based recognizers in general compared to supervector recognizers. Training and evaluating either a 1DHMM or a P2DHMM is fairly operationally intensive. However, evaluating the distance between two vectors (even if they are arbitrarily large) is much less computationally intensive. In addition, by generating a new (ideally) separable space, the classification tasks can be simplified. In this work, a simple nearest neighbor classifier was used for the recognition task, and 100% accuracy was achieved. Additional improvement (in terms of possible database variation) may possibly be achieved by using a slightly more robust classifier like a neural network.

One additional use for the approach described in this work would be extending it to feature spaces derived from multiple types of media. For example, consider the case of identity verification using speech and video. If the video frames were extracted, the feature generation algorithm described in this thesis could be run on every frame to establish a (potentially) large set of training sequences. This feature space could be coupled with features derived from the audio signal taken in between subsequent frames to generate larger feature space that encompasses both audio and video data for recognition. In addition, the same sort of idea could be extended to multimodal fusion for classification. In that same example, a set of HMMs that describe the subjects to be classified could be generated using both audio data and video data; then a supervector based recognizer could be used for both, but the distance metrics for the two streams could be weighted and combined to create a new recognizer.

REFERENCES

- [1] M. Turk and A. Pentland, "Face recognition using eigenfaces," in *Computer Vision and Pattern Recognition*. IEEE Comput. Soc, 1991, ch. [1] M. Tur, pp. 586–591. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=139758>
- [2] S. Zhang and M. Turk, "Eigenfaces," *Scholarpedia*, vol. 3, no. 9, p. 4244, 2008. [Online]. Available: <http://www.scholarpedia.org/article/Eigenfaces>
- [3] F. S. Samaria, "Face recognition using hidden Markov models," Ph.D. dissertation, University of Cambridge, 1994.
- [4] U. AT&T Laboratories, Cambridge, "The ORL Database of Faces (now AT&T "The Database of Faces")." [Online]. Available: http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.zip.
- [5] A. Nefian and M. Hayes, "Hidden Markov models for face recognition," in *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, vol. 5. IEEE, 1998. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=678085> pp. 2721–2724.
- [6] M. Bicego, U. Castellani, and V. Murino, "Using hidden Markov models and wavelets for face recognition," in *12th International Conference on Image Analysis and Processing, 2003.Proceedings*. IEEE Comput. Soc, 2003. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1234024> pp. 52–56.
- [7] X. Liu and T. Chen, "Video-based face recognition using adaptive hidden Markov models," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc, 2003. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1211373> pp. I-340–I-345.
- [8] S. Kuo and O. Agazzi, "Machine vision for keyword spotting using pseudo 2D hidden Markov models," in *IEEE International Conference on Acoustics Speech and Signal Processing*. IEEE, 1993. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=319752> pp. 81–84 vol.5.
- [9] H. Othman, T. Aboulnasr, and S. Member, "A Separable Low Complexity 2D HMM with Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1229–1238, 2003.
- [10] A. Nefian and M. Hayes, "An embedded HMM-based approach for face detection and recognition," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. IEEE, 1999. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=757610> pp. 3553–3556 vol.6.
- [11] V. Bevilacqua, L. Cariello, G. Carro, D. Daleno, and G. Mastronardi, "A face recognition system based on Pseudo 2D HMM applied to neural network coefficients," *Soft Computing*, vol. 12, no. 7, pp. 615–621, Oct. 2007. [Online]. Available: <http://link.springer.com/10.1007/s00500-007-0253-0>

- [12] J. S. S. Ballot, "Face recognition using Hidden Markov Models," Ph.D. dissertation, University of Stellenbosch, 2005.
- [13] C. Geng and X. Jiang, "Face recognition using sift features," in *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE, Nov. 2009. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5413956> pp. 3313–3316.
- [14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: <http://link.springer.com/10.1023/B:VISI.0000029664.99615.94>
- [15] H. Tang, "One-vector representations of stochastic signals for pattern recognition," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2010.
- [16] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," 2008. [Online]. Available: <http://www.vlfeat.org/>
- [17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–511–I–518, 2001. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990517>
- [18] D. a. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker Verification Using Adapted Gaussian Mixture Models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, Jan. 2000. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S1051200499903615>
- [19] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [20] K. Koutroumbas and S. Theodoridis, *Pattern Recognition*, 4th ed. Elsevier, 2009.
- [21] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989. [Online]. Available: <http://ieeexplore.ieee.org/xpls/abs.all.jsp?arnumber=18626>
- [22] S. Marchand-Maillet, "1D and pseudo-2D hidden Markov models for image analysis: theoretical introduction," Institut Eurecom, Tech. Rep. RR-99-49 Part A, 1999.
- [23] S. Young, G. Evermann, M. Gales, and T. Hain, "The HTK book (for HTK version 3.4)," *Cambridge university*, no. July 2000, 2006. [Online]. Available: <http://www.inf.u-szeged.hu/tothl/speech/htkbook.pdf>
- [24] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, Apr. 1994. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=279278>
- [25] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.
- [26] M. S. Pedersen, K. B. Petersen, J. Larson, K. Strimmer, L. Christiansen, L. He, K. Hansen, L. Thibaut, and M. Barao, "The matrix cookbook," Tech. Rep., 2006.
- [27] S. Julier and J. K. Uhlmann, "A general method for approximating nonlinear transformations of probability distributions," Robotics Research Group, Department of Engineering Science, University of Oxford, Tech. Rep., 1996.
- [28] M. Do, "Fast approximation of Kullback-Leibler distance for dependence trees and hidden Markov models," *IEEE Signal Processing Letters*, vol. 10, no. 4, pp. 115–118, Apr. 2003. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1186768>

- [29] J.-L. Durrieu, J.-P. Thiran, and F. Kelly, “Lower and upper bounds for approximation of the Kullback-Leibler divergence between Gaussian mixture models,” in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2012. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6289001> pp. 4833–4836.
- [30] B.-H. Juang and L. R. Rabiner, “A probabilistic distance measure for hidden Markov models,” *AT&T Technical Journal*, vol. 64, no. 2, pp. 391–408, Feb. 1985. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6770944>
- [31] J. Kohler, “Multilingual phone models for vocabulary-independent speech recognition tasks,” *Speech Communication*, vol. 35, pp. 21–30, 2001.
- [32] J. Silva and S. Narayanan, “Upper bound Kullback-Leibler divergence for transient hidden Markov models,” *IEEE Transactions on Signal Processing*, vol. 56, no. 9, pp. 4176–4188, Sep. 2008. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4599176>
- [33] W. M. Campbell, “A covariance kernel for svm language recognition,” *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4141–4144, Mar. 2008. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4518566>