SPEAKER ADAPTATION: MODELLING VARIABILITIES

THÈSE PRÉSENTÉE AU DÉPARTEMENT DE SYSTÈMES DE COMMUNICATIONS
ECOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Pour l'obtention du grade de Docteur Es Sciences Techniques

PAR
Patrick NGUYEN
Ingénieur en télécommunications, EPFL

Composition du Jury:

Directeur:	Prof. Christian WELLEKENS	Eurécom
Co-supervision:	Dr. Jean-Claude JUNQUA	Panasonic-PSTL
Rapporteurs:	Prof. Hervé BOURLARD	IDIAP/EPFL
	Prof. Pietro LAFACE	Politecnico di Torino
	Dr. George SAON	IBM
	Prof. Dirk SLOCK	Eurécom
Président:	Prof. Sabine SÜSSTRUNK	EPFL

November 24, 2002

Abstract

Most Automatic speech recognition systems make use of very complex HMMs to model speech trajectories. They are estimated with very large databases of annotated speech. Models are usually built to function well with any kind of speaker. Specializing these models to a particular speaker, condition, or gender, when enough data are available, is known to improve performance considerably.

Unfortunately, there are rarely enough data per speaker to build a specific model anew. Therefore, the generic speaker-independent model (SI) is altered with the scarcity of data problem in mind to befit the specific speaker's speech. This is commonly referred to as *speaker adaptation*. Model complexity and scarcity of data are intricately related to each other.

This dissertation addresses issues about speaker adaptation in the context of large-vocabulary speech recognition.

Firstly, the estimation of speaker-adapted models is improved by introducing constraints. The relationship between Euclidean distance and maximum-likelihood in the HMM framework allows us to impose linear constraints in the parametric HMM space effectively. Then, feature-space transformation is extended with a new closed-form solution and a Bayesian estimation formula.

Secondly, the specific applications of speaker adaptation are introduced. Self-adaptation is modeled as a cluster identification problem. Unsupervised adaptation techniques are applied to discriminative adaptation. The interaction with noise adaptation is studied.

Lastly, I have developed large vocabulary continuous speech recognition systems during this thesis. This comprises a miscellany of components, including a scalable acoustic training engine, language model training, self-adaptation, feature parameters normalizations, and a native triphonic trigram Viterbi recognizer. The most complex part is the decoder, which we will describe into more details. It is based on a new fast search algorithm.

Version Abrégée

La majorité des systèmes de reconnaissance de la parole font usage de modèles de Markov très complexes dans le but de modéliser la parole. Ces modèles sont habituellement construits de manière à fonctionner de façon satisfaisante indépendamment du locuteur. Il est bien connu que la spécialisation de ces modèles aux spécificités d'un locuteur, d'une condition, ou d'un sexe contribue à améliorer les performances lorsque sont à disposition des données en suffisance.

Malheureusement, rares sont les cas où il est possible de trouver des données en quantités suffisantes à la création de modèles spécifiques à partir de zéro. A cet effet, le modèle indépendant du locuteur est modifié, considérations dues au problème de paupérisme de données, en vue d'adhérer spécifiquement à la parole du locuteur en cours. Ce processus est nommé *adaptation au locuteur*. Une relation complexe existe entre la complexité du modèle et le manque de données.

Cette dissertation porte sur des problèmes ayant attrait à l'adaptation du locuteur dans le contexte de la reconnaissance de la parole grand vocabulaire.

Premièrement, l'imposition de contraintes améliore l'estimation de modèles adaptés au locuteur. La relation entre la distance Euclidienne et le critère de vraisemblance maximale, dans le formalisme des modèles de Markov cachés (HMM), nous permet d'imposer efficacement des contraintes de nature linéaire dans l'espace paramétrique des HMMs. Ensuite, une nouvelle forme analytique pour la transformation de vecteurs d'observations est développée. Une formulation Bayesienne l'accompagne.

Deuxièmement, des applications particulières à l'adaptation au locuteur sont introduites. L'auto-adaptation est modélisée en tant qu'un problème d'identification de groupement. Des méthodes d'adaptation non-supervisée sont appliquées à l'adaptation discriminative. Dans ce cadre, l'interaction au bruit fait l'objet d'une étude approfondie.

Finalement, j'ai développé un système de reconnaissance de la parole grand vocabulaire dans le cadre de cette thèse. Celui-ci comprend une variété de composants, notamment un moteur d'entraînement acoustique flexible et d'entraînement de modèles de language, l'auto-adaptation, des normalisations de vecteurs d'observation, et un décodeur Viterbi en triphones et trigrames au premier pas. Le décodeur forme le pôle prépondérant de complexité, que nous nous proposons de décrire en plus amples détails. Il est basé sur un nouvel algorithme de recherche rapide.

Acknowledgements

I am indebted to all of the people who have supported and encouraged me during this thesis. My parents and my brother are the first to whom I would like to express appreciation. I'm afraid that the friends whom I would gladly mention are too numerous to be listed individually.

It is my pleasure to have shared housing with such charming people as Sergio Galeani and Adeel Siddiqui. Later, I had the good fortune of sharing a house with Delphine Gourdon, Maria-Teresa Napoli, and Sören Thust.

I shared another great experience with Allan Coignet, Michalis Giannakopoulos, and Christian Zapf when we started a company specializing in computer security, while I was waiting for my H1B. They are brilliant people.

I also take this opportunity to gratefully acknowledge the stimulating environment in PSTL that propelled the Large Vocabulary effort. Jean-Claude Junqua, PSTL's director, most notably offered friendship and supervision, reviewed my publications, supported my every initiatives, providing support for the computational resources hungry tasks and valuable technical advice. Roland Kuhn, another of PSTL's acclaimed researchers, dear to me for inventing the Eigenvoices idea before my internship in 1998, and lending it to me for a while, provided technical advice, and was an invaluable source of information. Luca Rigazio, amongst other things, implemented the Finite State Network decoder, and was the most fervent LVCSR proponent, and eventually talked me into the Switchboard effort. Another respected colleague and friend proved to be David Kryze, computer cluster monger, who was in charge of the parallel computing hardware and software support.

Additionally, I would like to thank all these great people who constitute the speech community. In particular, Long Nguyen encouraged me from as early as my first paper, ultimately made me realize the importance of LVCSR and where real speech research was taking place. Michiel Bacchiani triggered the transition to SWB by dismissing WSJ as "solved problem" when I was just dreaming of upgrading Eigenvoices to WSJ. Andrej Lolje, the SWB maestro in AT&T, continued encouragements and disclosed crucial details about the implementation of a SWB system. We will come to George Saon later. I am very honoured by their friendship.

I am also very proud and grateful to count the most reverred speech recognition and digital signal processing experts amongst my reviewers. All four reviewers were very flexible and responded quickly to accomodate my tight schedule. Hervé Bourlard (EPFL/IDIAP), Pietro Laface (Politecnico), George Saon (IBM) and Dirk Slock (Eurécom Institute) are individuals of the highest technical skill. My thesis supervisor, Christian Wellekens, himself also one of the most established European speech experts, provided unfailing support throughout the entire thesis. I must also thank Christine Roussel at Eurécom for her wonderful job at interfacing with EPFL's administrative mine field.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this introduction, I hope to provide a quick kickstart to the main topic of this thesis. The structural overview of dissertation provides a general summary of the contents. Basic notations used throughout the documents are exposed.

1.1 Overview of the thesis

This section is devoted to providing an overview of the thesis. We are primarily interested in speaker adaptation.

There are three parts in this thesis: Theory, Applied Speaker Adaptation, and Evaluation. They are presented in an increasing link to practical applications. The final goal is to tackle problems stated in the Evaluation, namely LVCSR. The systems deployed in LVCSR provide some restrictions about how the adaptation is going to operate. It will be mainly unsupervised speaker adaptation in block mode. Given the properties of the domain, we will choose adaptation formulæ from the Theory.

First, we pursue theoretical developments. We follow the approach of Eigenvoices. Eigenvoices is an adaptation method that operates in the *model space*, that is, in the space defined by parameters of HMMs of a given topology. We extend and formalize the framework of HMM parameter regression under the likelihood criterion. Additionally, a new solution to feature transformation is explored. This is presented in Part I, Theory.

Secondly, we seek algorithms that tackle the problem of unsupervised adaptation and adaptation in noisy environments. We make use of the equations of Part I: Theory. This is done in Part II: Applications. It takes advantage of the characteristics of the adaptation of the Theory.

In Part III: Evaluation, the algorithms of the Applications are embodied in Large Vocabulary systems. The data-savvy adaptation of the Theory calls for large databases. Large Vocabulary systems, in turn, make use of unsupervised adaptation in varying conditions as envisioned by the Applications.

1.2 Notation

I have decided not to include a generic HMM introduction. Chapter 2 will introduce all notations necessary for the basic comprehension of the thesis. Readers are expected to have a general understanding of speech recognition mathematics.

In the short preamble, we define the symbols and conventions that will be exploited throughout this thesis. Most of the material is fairly standard. However, I made use of some shorthands in notations to outline important steps. The reader is invited to review the table of symbols in Table 2.1 quickly before proceeding with the dissertation.

1.3 Theory

In the first part (Chapters 3 and 4), we attempt to develop methods that contribute to a better estimation of HMM parameters. In particular, in Chapter 3, we construct a set of tools to model the parametric values of HMMs. Chapter 4 takes a lower-level approach. Observation features are transformed via a full, LU-decomposed matrix.

1.3.1 Model-space constraints

The first chapter extends and formalizes the Eigenvoices adaptation. Speaker adaptation allows to increase performance by specializing the speech recognition system to a particular speaker. There are two ways of adapting to the speaker: modify observation features, or the HMM models. We follow the second route: we place ourselves in the *model space*. Since the number of parameters to modify is large compared with the amount of speaker specific data, a method that reduces the amount of parameters to adapt is welcome. This is the main purpose of Eigenvoices: it places a linear constraint on model parameters. This concept of model space constraint is extended in Chapter 3.

Section 3.2 unites the likelihood criterion with the Euclidean distance in a Gaussian framework.

In Section 3.3, we give a brief review of the classic Eigenvoices approach. The mathematical framework is exposed in Section 3.4. A new method for estimating the eigenspace is presented in Section 3.4.2.

The following sections, Section 3.2 and 3.5, take a closer look at the idea of model distance. The premises are found in Section 3.5. We stipulate that posterior probabilities of models are Gaussian (Section 3.5.1, 3.5.2).

The root modulation (Section 3.6) is introduced as a means of reconciling PCA's squared error with ML. In Section 3.7, we expose estimation related to this new model.

Having thus formalized PCA, we relate discriminative Hilbert-geometric concepts, such as the Fisher cost function, with probabilistic concepts such as MMI, in Section 3.8. The Section 3.9 attempts to model HMM parameters with piece-wise linear regression, as an attempt to escape the linear restriction.

1.3.2 Feature-space transformation

The second chapter of this "theoretical" part pertains to an entirely different topic. Chapter 4 is to be understood in the context of feature transformation. It bridges a small gap in the current literature.

The problem of linear feature transformation has no direct closed-form solution in the general case. A closed-form may be established when the transformation matrix is diagonal. The SVD may be exercised when the matrix is orthogonal. Numerical solutions, either conjugate gradient, or row-by-row maximization, exist for full matrices.

We find out that triangular matrices also have a closed-form solution. This is closely related to LU decomposition for matrix inversion. Furthermore, we extend the framework to Bayesian adaptation.

1.4 Speaker adaptation: applications

In the second part of the thesis (Chapters 5–7), we deploy the adaptation techniques. Three aspects were investigated.

1.4.1 EM-based approaches

First, we concentrated on the problem of supervised and unsupervised adaptation, including the problem of local optima in the EM algorithm. The unsupervised adaptation techniques are applied to supervised adaptation for best discrimination. We argue that counter examples can be counted as additional data and also improve the reliability of the estimates. Finally, we analyze the pertinence of MAP and MLLR in different contexts.

1.4.2 Clustering and self-adaptation

The problem of unsupervised adaptation is viewed from another standpoint. We divide a sentence into smaller units, which might be correctly recognized or not. A unit is typically the time segment spanned by a recognized word. These units will be used for adaptation. If a unit is assigned a wrong label, due to misrecognition, it will corrupt the adaptation.

We argue that correctly recognized units will translate into a consistent estimation for the speaker. Errors in the unit labels are due to accidental confusion: they wll translate into inconsitent speaker estimation. In our algorithm, we decide whether a unit is correctly recognized or not based on the estimation of the speaker. The performance of speaker adaptation is improved by filtering out wrongly labelled units.

1.4.3 Noise and speaker adaptation

In many practical instances of ASR, noise is present in test conditions. Since the noise is not known *a priori*, an online noise compensation is applied. Applying speaker adaptation on the top of noise compensation, and vice-versa, is not trivial (see [RNKJ01, NWJ99]). We attack the problem with Eigenvoices in mind. Adaptation of prior knowledge to noise conditions is approached, as well as the difference between the characteristics of noise and speaker adaptation.

1.5 Evaluation framework

In the third part of the thesis (Chapters 8 and 9), approaches described in the previous chapters are evaluated on large-vocabulary continuous speech recognition (LVCSR). A significant effort has been devoted to the development of this LVCSR.

1.5.1 System descriptions

First, the large-vocabulary decoder was developed on WSJ. Then, the decoder was tailored to the SWB task. For that purpose, multi-class adaptation, variance normalization, and other features were added to the system. When the preliminary evaluation specification was released by NIST in January 2002, I began working the BN system. Speaker clustering and segmentation were implemented to tackle this task effectively.

WSJ, BN, and SWB are the three large vocabulary tasks which became the frontier of state-of-the-art speech recognition systems since 1992. From March 2000 until April 2002, I have replicated results roughly equivalent to state-of-the-art in 1998-1999, or about six years of research, and written a continuous speech large-vocabulary decoder for that purpose.

Chapter 8 summarizes choices in the implementation of the system. Since computational and human resources allocated to the development were limited, I opted for a minimal selection of features most influential on performance. This chapter is useful for anybody who would want to replicate LVCSR results.

1.5.2 Decoder

The recognizer is the main difficulty encountered when building large-vocabulary systems.

Chapter 9 begins with an introduction of the chapter. Section 9.2 gives an overview of the large-vocabulary decoder. Section 9.3 describes the search algorithm's theoretical foundation. Contrarily to structural optimizations such as the lexical tree search topology, the novelty comes from the organization of the search space. We define a total order relation of the search state hypotheses. The algorithm will traverse and build the active list of hypotheses according to this order, reaching maximum achievable speed.

Finally, NBest lists may be generated with material of Section 9.4.

Chapter 2

Conventions and Notations

This chapter introduces the notations used throughout this thesis.

Symbols and operators are usually named according to Table 2.1. The same symbol w identifies a word sequence, and eigenvalue vector, and the MLLR supervector. To avoid confusion with the transposition, the duration of an utterance will be $\mathcal{T}$.

Abbreviations in the text are found on Table 2.2.

2.1 Implicit subscripting, and super-vectorization

In this thesis, emission probability density functions are Gaussian mixtures. For the sake of simplicity, we usually drop the subscript of Gaussian components of the mixture. Also, several speaker dependent models are trained or speaker independent models are adapted. Wherever no confusion is introduced, speaker indices are also dropped.

2.1.1 Capitalization and alphabet

In general, capital letters are reserved for matrices, objective/cost functions, and cardinal numbers. Ordinals, vectors, and scalars are Latin lowercase characters. Greek letters are continuous variables, and constants.

2.1.2 Subscripting

All vectors are column vectors unless otherwise specified. A vector x of size D will have elements x_d,

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix}. \tag{2.1}$$

If we have a collection of vectors x_k, each of size D, we denote each component of each vector x_k in such a manner that:

$$x_k = \begin{bmatrix} x_1^{(k)} \\ \vdots \\ x_D^{(k)} \end{bmatrix}, \quad \forall k. \tag{2.2}$$

When clear from the context, or agreement of matrix dimensions, we may write $x_d^{(k)} = x_d, \forall d = 1..D$.

Symbol	Description
Q	Expected log-likelihood (E-step)
ε	Squared error
$\mathcal{N}$	The Gaussian/normal distribution
D	Dimension of feature vector (e.g. MFCC:39, PLP:27)
B	Number of speakers in the training data
q	Speaker index $q = 1, ..., B$
d	Dimension index $d = 1, ..., D$
t	Time index, in frame units
$\mathcal{T}$	Duration of an utterance
G	Total number of Gaussians
$\mathcal{W}$	Word sequence of $\#\mathcal{W}$ words: $\mathcal{W} = [\omega_0, ..., \omega_{\#\mathcal{W}-1}]$
o_t	Observation vector at time t
o_d or $o_d^{(t)}$	d^{th} element of o_t
m	Gaussian index $m = 1, ..., G$ in a Gaussian mixture
μ_m or μ	Mean vector of Gaussian m, size D
μ_d	d^{th} element of μ
ζ	Extended mean vector, $\zeta = [1 \vdots \mu^T]^T$
C_m or C	Covariance matrix of Gaussian m
R_m or R	Precision matrix of Gaussian m. $R_m = C_m^{-1}$
r_{kd}	Element d of k^{th} row of R
r_d	d^{th} diagonal element of R (when R is diagonal)
E	Dimension of the eigenspace
$\gamma_m(t)$	Expected posterior occupation probability of Gaussian m at time t
ϑ or w	Eigenvalues vector of size E
P	Length of supervector, usually $G \times D$
U	Eigenspace of type $P \times E$
U_m or U	Part of eigenspace corresponding to m, i.e. rows $1 + mD, ..., (m + 1)D$
u_j	j^{th} row of U_m
v_j	j^{th} column of U_m
λ	model parameters of an HMM
Λ	a diagonal matrix
W	Regression matrix W of type $D \times (D + 1)$
K	Number of regression classes, usually $K = 1$
w_d	d^{th} row of linear regression matrix W
P	Length of MLLR supervectors, $D(D + 1)K$
w	the MLLR supervector
G or G_d	Precision of MLLR matrix d, of type $(D + 1) \times (D + 1)$
τ	The weight parameter in MAP adaptation
β	A constant
A	A matrix
b	A bias vector, usually of size D
x	A vector
$\mathbb{R}^N$	Vector space of dimension N
$\frac{\partial}{\partial x}$	Derivative w.r.t. x
E	Expectation
$(\cdot)^T$	Transpose of matrix
$\mathrm{tr}(\cdot)$	Trace of a square matrix
$\|\| \cdot \|\|^2$	2-norm of matrix or vector
$< \cdot, \cdot >$	Inner product in canonical space
$\otimes$	Kronecker product

Table 2.1: Taxonomy of symbols

Abbreviation	Description	Definition
ASR	Automatic Speech Recognition	
BN	Broadcast news task	p. 102
CAT	Cluster Adaptive Training	[Gal00]
DR	Dimensionality Reduction	p. 24
EM	Expectation Maximization	[DLR77]
HMM	Hidden Markov Model	
KLT	Karuhnen-Loeve transformation	
LDA	Linear Discriminant Analysis	[DH73]
LM	Language Model	
LU	Lower-upper matrix factorization	p. 56
LVCSR	Large Vocabulary Continuous / Conversational Speech Recognition	
MFCC	Mel-frequency cepstral coefficients	
MD(E)	Meta Data (Extraction)	
ML	Maximum-likelihood	
MLED	Maximum-likelihood estimation	p. 24
MLES	Maximum-likelihood eigenspace	p. 25
MLLR	Maximum-likelihood linear regression	[LW95b], p. 28
MMI	Maximum mutual information	[Nor91]
PCA	Principal Component Analysis	p. 15, [Jol86]
pdf	probability density function	
PLP	Perceptual linear predictive	[Her90]
PSTL	Panasonic Speech Technology Laboratory	
QED	Quod erat demonstrandum	
RT	Real-time factor	
RT-02	NIST's Rich Transcription evaluation 2002	
SI,SD,SA	Speaker independent, dependent, adapted	
SAT	Speaker Adaptive Training	[AMSM96]
SVD	Singular vector decomposition	
SWB	Switchboard task	p. 97
STT	Speech-To-Text	
TDT	Topic Detection and Tracking	
TIMIT	TIMIT task	p. 107
VTLN	Vocal Tract Length Normalization	
WER	Word Error Rate	
w.r.t.	with respect to	
WSJ	Wall Street Journal task	p. 105

Table 2.2: Frequently-used acronyms and abbreviations

When matrices are decomposed into rows, each row will be treated as a column (vertical) vector. Let a matrix A be of type $M \times N$. The elements of the matrix are a_{kj}. Its rows a_k^T will be of type $1 \times N$:

$$a_k = \begin{bmatrix} a_{k,1} \\ \vdots \\ a_{k,N} \end{bmatrix}, \tag{2.3}$$

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_M^T \end{bmatrix}, \tag{2.4}$$

so that k will be a row index.

2.1.3 Super-vectorization

If we have a collection of vectors x_k, $k = 1, ..., G$, we define the supervector:

$$x = \text{super}(x_k)_{k=1}^G = \begin{bmatrix} x_1 \\ \vdots \\ x_G \end{bmatrix}. \tag{2.5}$$

If the original vector had size D, then the new supervector has size $P = D \cdot G$.

Similarly, we build a super matrix by concatenating the vectors as columns:

$$X = [x_1, ..., x_G]. \tag{2.6}$$

The matrix X has dimension $D \times G$. The supervector x is called the supervector of this matrix. Finally, inequalities are matricized: if A is a square matrix $N \times N$, we write:

$$A > 0, \tag{2.7}$$

to mean that for all vectors $x \in \mathbb{R}^N$,

$$x^T A x > 0, \tag{2.8}$$

with $x \neq 0$. In that event, the matrix is called non-negative definite.

2.2 Einstein's notation and lemmas

Einstein [Ein16] discovers that we may drop the summation sign Σ if the index appears (at least) twice in the summation. For instance, the j^{th} element of the k^{th} row of a matrix C, such that:

$$C = AB, \tag{2.9}$$

is given by:

$$c_{kj} = \sum_l a_{kl} b_{lj}, \tag{2.10}$$

which Einstein writes:

$$c_{kj} = a_{kl} b_{lj}. \tag{2.11}$$

Similarly, the trace of C is:

$$\text{tr } C = a_{sl} b_{ls} = \sum_{s,l} a_{sl} b_{ls}. \tag{2.12}$$

Usually, we will reserve k and j as indices. Other Latin lowercase letters, such as l, m, n, q, s, r, t and z can be used as ephemeral summation indices.

The following lemmas are useful when differentiating w.r.t. a matrix.

Lemma 1 $\frac{\partial \log |A|}{\partial A} = A^{-T}$.

It stems from Laplace's expansion into minors. Then since adj$A = |A|A^{-T}$ the rest follows.

Lemma 2 $\frac{\partial}{\partial A} tr\ B^T A = B$.

Using Einstein's notation,

$$\frac{\partial}{\partial a_{kj}} b_{lm} a_{lm} = b_{kj}. \tag{2.13}$$

QED.

Lemma 3 $\frac{\partial}{\partial A} tr\ BA^T RA = RAB + R^T AB^T$. *In particular, when matrices* R *and* B *are symmetric, it is also* $\frac{\partial}{\partial A} tr\ BA^T RA = 2RAB$.

Using Einstein's notation,

$$\frac{\partial}{\partial a_{kj}} b_{lm} a_{nm} r_{nq} a_{ql} = \Big(b_{lj} r_{kq} a_{ql}\Big) + \Big(b_{jm} a_{qm} r_{qk}\Big) \tag{2.14}$$

QED.

Part I

Theory

Chapter 3

Model space constraints

This chapter is devoted to our findings in imposing constraints, or structure, to HMM model parameters. In contrast with other approaches (e.g. [SZP01]), we do *not* apply constraints or transformations on the feature space. On the contrary, we apply constraints on HMM model parameters. For this reason, we call these constraints *model-space constraints*. Model parameters are themselves treated as random variables (e.g. [GL94]).

Our approach relies heavily on the idea of Eigenvoices [KNJ+98a]: we first impose linear constraints on HMM model parameters. These will significantly reduce the number of degrees of freedom. In turn, models will require fewer adaptation data before they attain an acceptable level of reliability. The backside of imposing constraints is the reduction of potential generative potential: strong evidence of a speaker-specific pattern that is not predicted by the constraints can never be learned. Therefore, we trade reliability of estimates for generative potential.

In this chapter,

1. We introduce the problem of speaker adaptation (Section 3.1).
2. We review distance measures such as squared error (used in PCA) and divergence (used in ML-HMM training) in (Section 3.2).
3. We explain Eigenvoices (Section 3.3) and extend it within the Baum-Welch framework (Section 3.4).
4. As a first step towards unfiying divergence and squared error, we verify that estimates of MLLR and MLES are Gaussian (Section 3.5).
5. We show that a normalization, called *root modulation* (Section 3.6), equates squared error and log-likelihood (Section 3.7).
6. We show that MAP, DI (deleted interpolation), and MMI can be cast as an ML problem (Section 3.8).
7. We extend the framework of linear regression to a simple non-linear model (Section 3.9).
8. We present experimental evidence of the performance of these derivations (Section 3.10).

3.1 Speaker adaptation and uncertainty

In this section, we review the main motivation for imposing constraints on model-space parameters. We observe that speaker-dependent models are more accurate than speaker-

independent models. However, due to lack of training data, they are rarely observable in real-world applications.

The balance between accuracy and lack of training data is fundamental in speaker adaptation.

3.1.1 Speaker-dependent models

It has been observed that speaker-dependent (SD) models provide much better recognition accuracy than speaker-independent models. In many applications, for instance, on Personal Digital Assistants (PDAs), one can safely assume that it is always the same speaker that is using the system.

The reason why SD models provide better recognition accuracy is the fact that they are not encumbered with having to model different speaker style at the same time. The entropy of SD models is always lesser than, or equal to, the entropy of SI models. Therefore, tailoring models to a specific speaker is always beneficial.

The reason why we do not always use SD models is due to trainability.

3.1.2 Uncertainty and quantization

In practice, building models specifically for the speaker using the system can prove difficult. This is due to the lack of training data. Usually, the adaptation data that is available to the system is several order of magnitude lower than the amount of data available for SI modelling. It is a classical estimation problem. We cannot take the adaptation data as being very reliable. It contains a large amount of accidental patterns. They are mixed with actual patterns that we would like to learn. Therefore, the conclusion drawns from observing the adaptation data are *uncertain*.

In estimation theory, the classical way of dealing with that problem is to reduce number of the degrees of freedom. This is called quantization. For instance, instead of using a mixture of two Gaussians, we use a single Gaussian. Quantization is known to reduce the number of parameters to estimate, and in turn, the uncertainty. On the other hand, quantization also reduces the potential generative power of the pdf.

There is a trade-off between quantization and uncertainty: it is called the error decomposition theorem [Zhu98].

3.1.3 Introducing structure

There are several ways of reducing the degrees of freedom of models. Reducing parameters towards compact models is a difficult task. Clustering, for instance, is a popular technique that ties some models parameters. Tying decreases the number of parameters by sharing parameters. These parameters also share observed samples, which improves their reliability. At the same time, the generative power of the model decreases because the model must describe two pdfs at the same time.

There is another class of techniques, called transformation-based coding, which aims at transforming the space of observations to recognize important from unimportant components. It has been explored into depth in the *feature* space. In the model-space, however, it has gone relatively unnoticed.

Eigenvoices is a transformation-based coding in the model space. Model parameters discern between pdfs of the same family. Those parameters will be treated as random variables themselves.

3.2 Least-squares, divergence, and log-likelihood

In this section, we show how the divergence, log-likelihood, and squared error are linked.

3.2.1 Squared error

The squared error is most popular in discrete vector quantization. Let there be a set X of B samples. Each observation vector x_k, $k = 1, ..., B$ has a dimension P such that $X = [x_1 ... x_B]$ is a matrix of dimension $P \times B$. In Vector Quantization, we decide to approximate each x_k with a value drawn from a set of predefined values y_k. The values are easier to process because of their nature (for instance integer values) or their small amount. For that reason the set of possible values is called a *dictionary* or *codebook*.

The designer of the codebook must find a satisfactory approximation of x_k with y_k:

$$x_k \approx y_k, \tag{3.1}$$

which can be summarized as the mean squared error of the approximation (MSE), some times referred to as the *distortion*. It is defined as:

$$\varepsilon = B^{-1} \sum_{k=1}^{B} \varepsilon_k \tag{3.2}$$

with the observation error ε_k defined as:

$$\varepsilon_k = ||x_k - y_k||^2 = < x_k - y_k, x_k - y_k > = (x_k - y_k)^T (x_k - y_k). \tag{3.3}$$

In the last equation, we have specialized the error to the canonical Euclidean distance. It is also possible to use a different inner product:

$$< x_k - y_k, x_k - y_k > = (x_k - y_k)^T W (x_k - y_k), \quad W > 0,\ W^T = W \tag{3.4}$$

where we weight the contributions using a non-negative definite matrix W. We will see that this matrix can be interpreted as a rotation and a scaling. The rotation transforms the space which is canonical for the phenomenon. In other words, we rotate the observation so that each component of the rotated vector has a meaning by itself, for instance. The scaling gives more weight, or importance, to some components.

The quantizer will choose naturally:

$$y_k = \arg\min_{y \in \mathcal{C}} ||y - x_k||^2 \tag{3.5}$$

where $\mathcal{C}$ is the codebook, i.e. the possible values for y_k. When designing $\mathcal{C}$ for $N > 1$, it is unfortunately impossible to define a total order of the x_k and therefore it results in an intractable problem. However, one can use common sense to devise fairly good suboptimal approaches in quadratic time.

We have defined the least squares criterion. The concept was illustrated with vector quantization.

3.2.2 Principal Component Analysis

Introduction

Principal Component Analysis [Jol86] is an algorithm very popular in many domains, such as sociology, biology, etc. There are many interpretations of what it produces and why it would work. In this section, we concentrate on defining the basic notations which will appear later in this thesis.

We can revisit the problem of Section 3.2.1 and devise a continuous approximation of x_k:

$$y_k = f(x_k). \tag{3.6}$$

In this section, we assume that $f(\cdot)$ is a linear (or affine) function. $f(\cdot)$ is sometimes called the kernel mapping, and in this case we have a linear kernel, which is defined by the matrix U,

$$y_k = U U^T x_k. \tag{3.7}$$

For this reason, the transformation is called a *linear regression*.

Definition

Let the x_k defined previously serve as samples. Let U be a orthogonal matrix of type $P \times E$, where P is the length of all x_k. By *orthogonal*, we mean that:

$$U^T U = I_E, \tag{3.8}$$

which we call the *resolution of unity*. The matrix U generates a linear vector space in $\mathbb{R}^P$ of rank E. We abuse the notation and call the subspace itself U.

It is notoriously known that the minimum squared error approximation of a vector y_k in U is:

$$y_k = \arg\min_{y \in U} ||x_k - y||^2 = \arg\min_{y \in U} \left\{ ||x_k - UU^T x_k||^2 + ||y - UU^T x_k||^2 \right\} = UU^T x_k. \tag{3.9}$$

Note how Pythagoras' identity helped us here.

The goal of PCA is to find the U that will minimize the error of quantization ε. The error due to the approximation is:

$$\varepsilon = \sum_k ||x_k - UU^T x_k||^2. \tag{3.10}$$

We have:

$$\arg\min_U \varepsilon = \arg\min_U \sum_k \left\{ ||x_k||^2 - 2x_k^T UU^T x_k + x_k^T UU^T UU^T x_k \right\}, \tag{3.11}$$

and because U is orthogonal:

$$\arg\min_U \varepsilon = \arg\max_U \sum_k x_k^T UU^T x_k = \arg\max_U \sum_k \operatorname{tr} U^T x_k x_k^T U. \tag{3.12}$$

Let the *observation matrix* X be:

$$X = [x_1, \ldots, x_B] \tag{3.13}$$

It is easy to show that the solution for U is the spectral decomposition of the auto-correlation matrix:

$$XX^T = \sum_k x_k x_k^T = V \Lambda V^T \tag{3.14}$$

with V an orthogonal $P \times B$ matrix, and Λ a diagonal eigenvalue matrix with diagonal elements $\lambda_1, ..., \lambda_B$ such that:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_B. \tag{3.15}$$

Substituting XX^T with its eigenvalue decomposition into (eq. 3.12), we have:

$$\arg\min_U \varepsilon = \arg\max_U \operatorname{tr} U^T V \Lambda V^T U. \tag{3.16}$$

Clearly, any $U \neq V$ will yield non-diagonal elements whose energy will be subtracted from the trace. Therefore, the optimal U is a truncation of spectral decomposition of the auto-correlation matrix.

We have shown how to use the spectral decomposition to define a linear constraint that will minimize the squared error. Utilizing Pythagoras identity, the problem was be reduced to maximizing the energy of the projected samples.

Orthogonality of eigenvalues

Another noteworthy property of the eigenvalues is that of orthogonality. The eigenvalues ϑ represent the location of x_k in the transformed space:

$$\vartheta_k = U^T x_k, \tag{3.17}$$
$$Y_k = U\vartheta_k. \tag{3.18}$$

We can see that they are independent:

$$\sum_k \vartheta_k \vartheta_k^T = \sum_k U^T x_k x_k^T U = U^T V \Lambda V^T = \Lambda_E, \tag{3.19}$$

where Λ_E^2 is the truncation of the diagonal matrix Λ^2, which is again diagonal. It is not yet appropriate to talk about covariance, but if we interpret:

$$\mathrm{E}\vartheta\vartheta^T = \frac{1}{B-1}\sum_k \vartheta_k \vartheta_k^T, \tag{3.20}$$

it means that the variables ϑ are not correlated, because off-diagonal terms of the covariance matrix cancel.

3.2.3 Divergence

The divergence is a popular criterion for measuring similarities between probabilities. It enjoys two useful properties:

- it is independent of the parameterization,
- it is linked to the log-likelihood, and
- its Pythagorean geometry is well-documented.

Let $p(\cdot)$ and $q(\cdot)$ be two probability density functions defined over the same feature space $\mathbb{R}^D$. The divergence $D(p,q)$ is a directed similarity measure that relates p with q:

$$D(p,q) = \int \mathrm{d}p \, \log \frac{p}{q}. \tag{3.21}$$

$D(p,q) = 0$ iff $p \equiv q$ in probability. Usually, p is understood to be the sought after probability. The pdf q does not necessarily belong to the same family, but we seek to find one that is the closest in some family.

For instance, let $p(\cdot)$ be a mixture of two equiprobable Gaussians with the same variance C and mean μ and $-\mu$ respectively:

$$p(x) = \frac{1}{2}\mathcal{N}(\mu, C) + \frac{1}{2}\mathcal{N}(-\mu, C). \tag{3.22}$$

We want to approximate it with another distribution $q(x)$. We seek:

$$\hat{q}(x) = \arg\min_q D(p,q). \tag{3.23}$$

The pdf q is the *projection* of p into a single Gaussian probability density family. Incidentally, it is also the maximum-likelihood estimate [CT84]:

$$\hat{q}(x) = \arg\min_q D(p,q) = \arg\min_q \int \mathrm{d}p \, \log \frac{p}{q} = \arg\max_q \int \mathrm{d}p \, \log q. \tag{3.24}$$

Suppose now that q is a single Gaussian:

$$q(x) = \mathcal{N}(\mu_q, C_q). \tag{3.25}$$

Its mean and covariance are sought:

$$(\hat{\mu}_q, \hat{C}_q) = \arg\max_{\mu_q, C_q} \int \mathrm{d}p \, \log q. \tag{3.26}$$

The expected log-likelihood is:

$$\int \mathrm{d}p \, \log q = -\frac{1}{2} \int \mathrm{d}x \, p(x) \Big[D \log 2\pi + \log |C_q| + (x - \mu_q)^T C_q^{-1} (x - \mu_q) \Big]. \tag{3.27}$$

Differentiating w.r.t. μ_q and setting to zero, we get:

$$\frac{\partial}{\partial \mu_q} \int \mathrm{d}p \, \log q = 0 = 2C_q(\mu_q - \int \mathrm{d}p \, x), \tag{3.28}$$

or:

$$\mu_q = \int \mathrm{d}p \, x = \frac{1}{2}\mu + \frac{1}{2}(-\mu) = 0, \tag{3.29}$$

regardless of C_q. Replacing into (eq. 3.27), we differentiate w.r.t C_q^{-1}:

$$\frac{\partial}{\partial C_q^{-1}} \int \mathrm{d}p \, \log q = 2C_q - \mathrm{diag}(C_q) + \int \mathrm{d}p \, \Big[2xx^T - \mathrm{diag} \, xx^T \Big] = 0. \tag{3.30}$$

Finally, we can state:

$$\hat{q}(x) = \mathcal{N}(0, C + \mu\mu^T). \tag{3.31}$$

Furthermore, Pythagoras' theorem is also valid:

$$D(p, q) = D(p, \hat{q}) + D(\hat{q}, q), \tag{3.32}$$

for any q in the same family as $\hat{q}$.

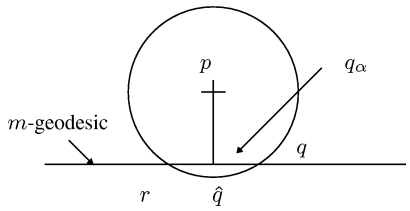


Figure 3.1: Pythagoras identity: $D(p, q) = D(p, \hat{q}) + D(\hat{q}, q)$

Proof: We sketch an algebraic proof of the theorem for completeness. It is heavily based on the concept of sphere and lines. No definition of area is required. We proceed as follows (see Figure 3.1):

1. Draw a line from q to $\hat{q}$. Show that the distance decreases by a certain amount, when we get closer to $\hat{q}$.

2. Extend the line to r. This is done by drawing a line from q to r, and stating that $\hat{q}$ is also on the line.

3. Consider a point z on this line. It was shown by argument 1 that the distance $D(z,p)$ must decrease as we approach $\hat{q}$, and, by symmetry, from that point on increase until r.

4. The derivative of the distance, w.r.t. to the coordinate of z in the line, must then change sign at exactly $\hat{q}$

5. Replace the expression of the derivative into an algebraic expansion of $D(q,p)$ and find the desired result.

We have an arbitrary point in the family of $\hat{q}$. We draw the m-geodesic [Ama99] from q to $\hat{q}$. The m-geodesic is the segment joining the two points. For all points q_α belonging to this m-geodesic, there exists an $\alpha \in [0;1]$ such that:

$$q_\alpha(\cdot) := \alpha q(\cdot) + (1-\alpha)\hat{q}(\cdot). \tag{3.33}$$

The function q_α is a pdf which belongs.Let us assume that the pdf q is further away from $\hat{q}$ to p:

$$D(q,p) \geq D(\hat{q},p). \tag{3.34}$$

Since the divergence is concave in $(1-\alpha)$, then, we know that the distance to p from any point of the m-geodesic is decreasing until we reach $\hat{q}$. That is,

$$\frac{\partial}{\partial\alpha} D(q_\alpha, q) = \int \left[\mathrm{d}p - \mathrm{d}\hat{q}\right] \log\frac{q_\alpha}{p} + \int \left(\mathrm{d}q - \mathrm{d}\hat{q}\right) \tag{3.35}$$

$$= \int \left[\mathrm{d}q - \mathrm{d}\hat{q}\right] \log\frac{q_\alpha}{p}. \tag{3.36}$$

Incidentally, we can compare the distances to p and within the m-geodesic:

$$D(q,p) = \int \mathrm{d}q \, \log\frac{q}{p} \tag{3.37}$$

$$= \int \mathrm{d}q \, \log\frac{q}{\hat{q}} + \int \mathrm{d}q \, \log\frac{\hat{q}}{p} \tag{3.38}$$

$$= \int \mathrm{d}q \, \log\frac{q}{\hat{q}} + \int \mathrm{d}q \, \log\frac{\hat{q}}{p} + \int \mathrm{d}\hat{q} \log\frac{\hat{q}}{p} - \int \mathrm{d}\hat{q} \log\frac{\hat{q}}{p} \tag{3.39}$$

$$= D(q,\hat{q}) + D(\hat{q},p) + \int \left[\mathrm{d}q - \mathrm{d}\hat{q}\right] \log\frac{\hat{q}}{p} \tag{3.40}$$

$$= D(q,\hat{q}) + D(\hat{q},p) + \frac{\partial}{\partial\alpha} D(q_\alpha, p)\bigg|_{\alpha=0}. \tag{3.41}$$

The derivative corresponds to the concept of tangent. Now, let us select another point $r \neq q$ from the sphere around p:

$$D(r,p) = D(q,p). \tag{3.42}$$

Let us draw the m-geodesic from r to $\hat{q}$:

$$r_\gamma = \gamma r + (1-\gamma)\hat{q}, \tag{3.43}$$

the same way we did for q_α. Now, amongst the circle of intersecting the sphere of radius $D(q,p)$ around p and the family of q, we extend the m-geodesic q_α, such that all $\hat{q}$ and q, and r belong to the same m-geodesic. That is, there is a $\hat{\eta}$ such that:

$$z_{\hat{\eta}} = \hat{q}, \tag{3.44}$$

with:

$$z_\eta = \eta q + (1-\eta) r. \tag{3.45}$$

We can see that r exists and is unique. Since both ends of the z_η m-geodesic have equal value,

$$D(r,p) = D(z_\eta,p)|_{\eta=0} = D(z_\eta,p)|_{\eta=1} = D(q,p). \tag{3.46}$$

Since $\hat{q}$ has a divergence different from those,

$$D(\hat{q},p) \leq D(r,p), \tag{3.47}$$

by hypothesis, and by convexity of the divergence on the r_γ and q_α curves, the partial derivative *must* exactly reach its minimum and change sign at $\hat{q}$:

$$\frac{\partial}{\partial\eta} D(z_\eta,p)|_{\eta=\hat{\eta}} = 0. \tag{3.48}$$

Therefore, the tangent part of the decomposition of the divergence also vanishes at exactly $\hat{q}$,

$$\int \left[\mathrm{d}q - \mathrm{d}\hat{q}\right] \log\frac{\hat{q}}{p} = 0. \tag{3.49}$$

Replacing into (eq. 3.41), we obtain the Pythagorean identity:

$$D(q,p) = D(q,\hat{q}) + D(\hat{q},p).$$

QED.

3.2.4 Log-likelihood and Euclidean distance

Now that we have defined distance and projection in the least squares and likelihood sense, we might ask ourselves: when do they ever coincide?

In the Eigenvoices logic (Section 3.3), we draw two distributions from the same family of multivariate Gaussians. Let $p(\cdot)$ and $q(\cdot)$ be:

$$p(x) = \mathcal{N}(\mu_x, C_x), \tag{3.50}$$
$$q(y) = \mathcal{N}(\mu_y, C_y). \tag{3.51}$$

We characterize the distance between p and q as:

$$\varepsilon = (\mu_x - \mu_y)^T W (\mu_x - \mu_y) = ||\mu_x - \mu_y||^2_W, \tag{3.52}$$

where W is a whitening matrix. We will see that rationale underlying.

The divergence $D(p,q)$ between these distributions is:

$$D(p,q) = \int \mathrm{d}x\, p(x) \log \frac{p(x)}{q(x)} \tag{3.53}$$

$$= -\frac{1}{2} \int \mathrm{d}x\, p(x) \Bigg[\log \frac{|C_x|}{|C_y|} + (x-\mu_x)^T C_x^{-1}(x-\mu_x) - ... \tag{3.54}$$

$$- (x-\mu_y)^T C_y^{-1}(x-\mu_y) \Bigg] \tag{3.55}$$

$$= \beta - \frac{1}{2} \Bigg[\mathrm{tr}\, C_x^{-1} C_x - (x-\mu_y)^T C_y^{-1}(x-\mu_y) \Bigg] \tag{3.56}$$

$$= \beta' - \frac{1}{2} \int \mathrm{d}x\, p(x)(x-\mu_y)^T C_y^{-1}(x-\mu_y) \tag{3.57}$$

$$= \beta' - \frac{1}{2} \int \mathrm{d}x\, p(x)(x-\mu_x+\{\mu_x-\mu_y\})^T C_y^{-1}(x-\mu_y) \tag{3.58}$$

$$= \beta' - \frac{1}{2} \Bigg[\mathrm{tr}\, C_y^{-1} C_x + 2 \int \mathrm{d}x\, p(x) \{ (\mu_x - x)^T C_y^{-1}(\mu_x-\mu_y) + ... \tag{3.59}$$

$$+ (\mu_x-\mu_y)^T C_y^{-1}(\mu_x-\mu_y) \} \Bigg] \tag{3.60}$$

$$= \beta' - \frac{1}{2} \Bigg[\mathrm{tr}\, C_y^{-1} C_x + (\mu_x-\mu_y)^T C_y^{-1}(\mu_x-\mu_y) \Bigg] \tag{3.61}$$

$$= \beta'' - \frac{1}{2} \Bigg[(\mu_x-\mu_y)^T C_y^{-1}(\mu_x-\mu_y) \Bigg], \tag{3.62}$$

where β, β' and β'' are constants. Now suppose that $W = C_x^{-1} = C_y^{-1}$. This means that the set of Gaussians p and q is *homoscedastic*. The divergence $D(p,q)$ becomes:

$$D(p,q) = -2\varepsilon. \tag{3.63}$$

We can thus state that there is an equivalence between divergence between two distributions and squared error between the mean vectors.

Lemma 4 *Let two random variables X, Y with pdf $p(x)$ and $p(y)$ respectively. If they are Gaussian with equal covariance C, then the divergence and Euclidean distance between the means are related with:*

$$D(p,q) = -2||\mu_x - \mu_y||^2_{C^{-1}}. \tag{3.64}$$

This condition of Gaussianity is very restrictive, but it restores symmetry of the divergence, and all other geometric properties of a Hilbert space, to the otherwise cumbersome divergence measure.

Furthermore, the Pythagorean identity is the same under both concepts. Let q be a Gaussian such that μ_y is constrained to lie in a linear space $\mathcal{U}$. The projection of p onto $\mathcal{U}$ is:

$$\hat{q} = \arg\min_q D(p,q) = \arg\max_q \varepsilon, \tag{3.65}$$

by virtue of (eq. 3.64). The pdf $\hat{q}$ has mean $\hat{\mu}_y$.

We know that:

$$||\mu_y - \mu_x||^2 = ||\mu_y - \hat{\mu}_y||^2 + ||\hat{\mu}_y - \mu_x||^2, \tag{3.66}$$

$$D(q,p) = D(q,\hat{q}) + D(\hat{q},p) \tag{3.67}$$

and both relationships are equivalent.

We have therefore proved that under certain circumstances maximum likelihood and least-squares projection function equivalently in a homoscedastic Gaussian framework.

3.3 Eigenvoices: introduction

A good introduction of Eigenvoices can be found in [KJNN00]. We will review the main ideas quickly.

3.3.1 Motivation

The goal of Eigenvoices is to introduce a linear constraint model parameters, which will be used during the adaptation process. Eigenvoices represent correlation *within* HMM parameters. They can be thought of as principal speaker characteristics.

By observing speaker-dependent models in training data, Eigenvoices learns what is reasonable for speaker-dependent models. Eigenvoices learns that if a male speaker pronounces phoneme `aa`, he will not pronounce phoneme `ax` the way a female speaker does. In other words, it is able to place a *constraint* on what HMM parameters may model.

These constraints are learnt through Principal Component Analysis (PCA [Jol86]). It is posible to discern a linear vector space structure among a large training set of speaker-dependent models.

When a new speaker is presented to the system, a naive adaptation algorithm would attempt to modify parameters directly in the HMM parameter space. However, there is not much data available for one to estimate all parameters. The uncertainty is very high, and accidental structure in the small adaptation sample might yield unreasonable models. The first tentative is to tie parameters together. For instance, in MLLR [LW95b], the adaptation parameters are summarized into an affine transformation.

Eigenvoices, on the other hand, has gained experience during the training of what speaker-dependent (SD) models look like. It will attempt to find the few speaker characteristics in the transformed Eigenvoices space. These will then be mapped back into the seemingly high-dimensional space.

In the next section, we show in more details how the process is carried out.

3.3.2 Algorithmic Overview

The process is shown on Figure 3.2. There are two phases: in the first one, the goal is to collect prior knowledge. In the second one, this prior knowledge enacts better modeling of unseen speakers. Those two phases correspond to training and decoding phases of a speech recognition system.

Training Eigenvoices

We explain how to acquire prior knowledge necessary for better estimation of models.

Training speaker-dependent models. Observing models is crucial to Eigenvoices. During this phase, we train speaker-dependent models. For all speakers available at training time, we train a speaker-dependent model. They are converted into supervectors as shown on Figure 3.3. These serve as observation samples for PCA.

SD models are trained through pure MAP, or MLLR, depending on the amount of data. It is the collection of adaptation experiments that will teach Eigenvoices what is expected during an adaptation procedure.

Once all SD models are trained, we can build models for speaker variability. We shall compare these models to get an idea of how inter-speaker variability affects HMM parameters.

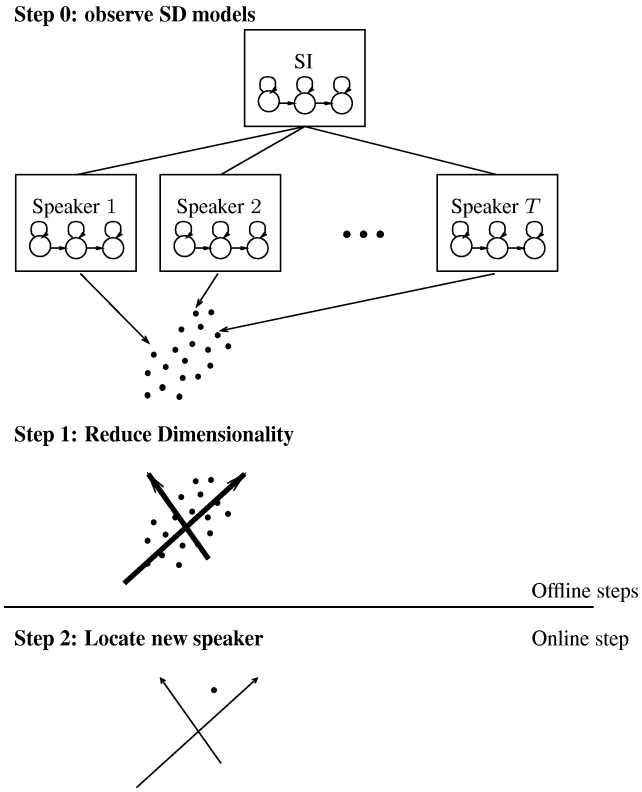


Figure 3.2: Overview of eigenvoices

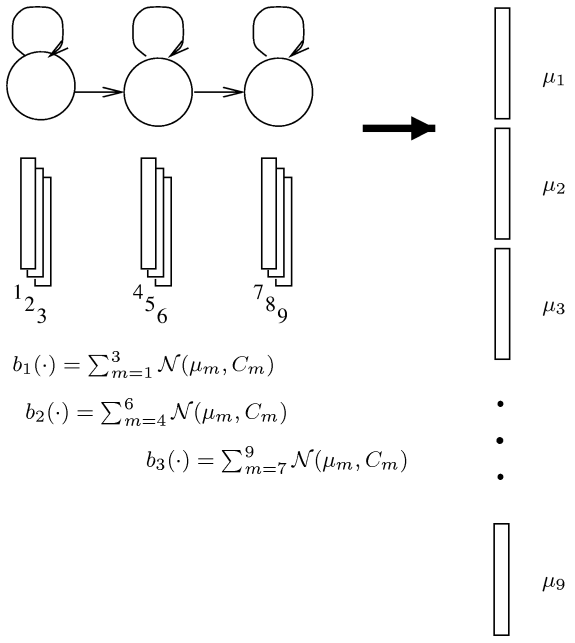


Figure 3.3: Supervectorization of model mean parameters

Reducing dimensionality. Having thus collected SD models, we shall extract the Eigenvoices: they span the space of speaker variability. With the collection of speaker models, we are able to discern a linear pattern hidden in the high-dimensional space.

This is done through dimensionality reduction (DR). In early experiments, we have been using PCA [Jol86], which is known under a variety of names, such as the Karuhnen-Loeve transformation (KLT) or singular vector decomposition (SVD).

PCA is applied to the set of SD models. The KLT maps high-dimensional parameter vectors into a smaller speaker characteristics space. High energy components correspond to the Eigenvoices. Low energy components are discarded, provided that noise is of low energy. This mapping filters out errors in the estimation of SD models. The resulting space is called the *speaker space*, or *Eigenvoices space*. Each vector generating the basis is referred to as an *eigenvoice* or *eigenvoice vector*.

Optimizing Eigenvoices. As we will see later, PCA might not yield an optimal set of principal components. It is possible to re-estimate these vectors in a Baum-Welch algorithm. KLT is not optimal because it seeks to minimize the Euclidean distance between supervectors, which is related to, but not equal to the ML criterion. Other criteria than ML might be considered.

Deploying Eigenvoices

Once these principal directions are obtained, they will form a framework of prior knowledge into which new models are expected to abide.

When a new speaker is presented to the system, we will assume it to lie in the speaker space.

3.4 Estimation problems in Eigenvoices

We will now approach the problems of optimal estimation during all steps of the Eigenvoices approach. In the remainder, we will only adapt the *means of Gaussians*. Adaptation of the variances is an open problem.

We will proceed in an order that differs from what we have used before. We shall begin with the simplest problem, that also serves as a basis for later stages.

We describre two algorithms:

1. MLED: this addresses the estimation of the location of a speaker within the Eigenvoice space.
2. MLES: this optimizes the Eigenvoice space with respect to the likelihood.

These algorithms aim at overcoming a discrepancy between least-squares and HMM likelihood.

3.4.1 MLED

In the following, we attempt to locate a speaker within the Eigenvoice space.

The algorithm called *Maximum Likelihood Eigen-Decomposition* (MLED) was derived prior to this thesis [Ngu98]. The cited reference also analyzed effects due to unseen models.

The problem may be stated as follows. We place ourselves in the decoding step of Eigenvoices. The SI model and speaker space were derived prior to this situation. An incoming speaker is presented to the system, which must then build a speaker-dependent model from some adaptation data $o_t, t = 1, ..., \mathcal{T}$. This is done conveniently via the EM algorithm. The updated mean of each Gaussian component m is:

$$\mu_m = \sum_e w_e u_e^{(m)}. \tag{3.68}$$

The Q function is defined as [Gal97b]:

$$Q = -\frac{1}{2}\sum_{m,t}\gamma_m(t)(o_t - \sum_{e=1}^{E} w_e u_e^{(m)})^T R_m (o_t - \sum_{e=1}^{E} w_e u_e^{(m)}). \tag{3.69}$$

This is only the fraction of the expected log-likelihood associated to the mean parameters. Transition probabilities, mixture weights, and more importantly variances may not be optimized using this incomplete function. Specifically, variances may not be optimized due to the presence of the log determinant (see Section 4.2). Define a $D \times E$ matrix:

$$U_m = [u_1^{(m)} ... u_E^{(m)}]. \tag{3.70}$$

This is the portion of the eigenspace that corresponds to the Gaussian component m. The Q-function may be rewritten as:

$$Q = -\frac{1}{2}\sum_{m,t}\gamma_m(t)(o_t - U_m w)^T R_m (o_t - U_m w). \tag{3.71}$$

We aim to estimate the speaker parameters w_e. This is done by differentiating (eq. 3.71):

$$\frac{\partial Q}{\partial w} = -\sum_{m,t}\gamma_m(t) U_m^T R_m (o_t - U_m w) = 0. \tag{3.72}$$

The solution is:

$$w = \left(\sum_{m,t}\gamma_m(t) U_m^T R_m U_m\right)^{-1} \sum_{m,t}\gamma_m(t) U_m^T R_m o_t. \tag{3.73}$$

3.4.2 MLES

This algorithm aims at optimizing the Eigenvoice space with respect to the likelihood criterion.

The eigenspace U_m extracted from PCA might not befit the likelihood criterion: this is due to a discrepency between likelihood and least squares criteria. The eigenvoices space might be re-estimated in the EM-algorithm. The *Maximum Likelihood EigenSpace* (MLES) algorithm specifies a solution that maximizes the eigenvoices vectors with respect to the likelihood of adapted speakers in the training data. The dimension of the eigenspace E, is supposed to be fixed statically by the system designer.

Procedure: The algorithm must be understood as an extension of the Baum-Welch algorithm. Let U be the eigenspace, which is the variable that we seek. To achieve this, we proceed in four steps:

1. We define the likelihood to be optimized. It is the aggregation of all speaker-dependent likelihoods.

2. We have an insoluble problem, in which we can discern: observation data (speech, transcriptions, speaker ID), and completion data (segmentation, location of speakers in Eigenvoices space). The completed data is the union of both.

3. We apply the EM algorithm, taking the expectation over the completion data.

4. We approximate the problem to achieve a tractable equation, and solve it.

First, let us define the likelihood $l(O;U)$. It is the probability of the training data given the model U:

$$l(O;U) = p(O|U). \tag{3.74}$$

The training data is O. The HMM models are developed using U. We have a partition of the training data $O = (O_1, ..., O_q, ..., O_B)$ which splits the training data amongst speakers. Each subset O_q corresponds to the speech of one given speaker q. The subsets O_q are assumed to independent of each other:

$$l(O;U) = \prod_{q=1}^{B} p(O_q|U). \tag{3.75}$$

This is impossible to solve directly. The observation data (O_q and their labels) is insufficient to solve the problem. We need to complete the problem with more data, which is unobserved. The unobserved data, or completion data, consists of the segmentation and of the location of speakers in Eigenvoices space. In Baum-Welch, the completion data was the segmentation: it will be called γ. It is now augmented with the location of speakers: they are referred to as w_q.

We use the EM algorithm [DLR77]. We begin with an initialization of the space U_0. We complete the data using the expectation conditional on this model:

$$Q = \mathrm{E} \log l(O;U)|U_0, O = \sum_q \mathrm{E}_{\gamma,w_q} \log p(O_q, w_q, \gamma|U)|U_0, O_q. \tag{3.76}$$

We can develop the expectations w.r.t the speaker location:

$$Q = \sum_q \mathrm{E}_\gamma \left[\int \mathrm{d}P(w_q, \gamma|U_0, O_q) \log p(O_q, \gamma, w_q|U)|U_0, O_q \right]. \tag{3.77}$$

This is still not tractable, due to the joint expectation of γ and w_q. We can use the maximium instead of the the expectation:

$$Q = \sum_q \max_{w_q} \left\{ \mathrm{E} \log p(O_q, \gamma, w_q|U)|U_0, O_q \right\}. \tag{3.78}$$

The bracketed part is the same as the log-likelihood found by Baum-Welch, at the particular w_q where it is maximum. It is proportional to the Q function, of equation (eq. 3.69), evaluated at the MLED estimate (eq. 3.73):

$$Q = \sum_q Q_{\text{MLED}}. \tag{3.79}$$

It is possible to arrive at this same result by alternating EM with speaker location and EM with segmentation. Therefore, the rule of the thumb is to use to estimate and use MLED value for all speakers during the training of U.

Derivations: Replacing Q_{MLED} with its value (eq. 3.69):

$$Q \propto -\frac{1}{2} \sum_q \sum_{t,m} \gamma_m^{(q)}(t)(\mu_m - o_t)^T R_m (\mu_m - o_t). \tag{3.80}$$

with as usual:

$$\mu_m^{(q)} = \sum_{e=1}^{E} w_q u_e^{(m)}. \tag{3.81}$$

For the sake of simplicity we will drop the index q from $\mu_m^{(q)}$, $\gamma_m^{(q)}(t)$ and w_q.

If covariance matrices are diagonal, rows $v_j^T, j = 1...D$ of the eigenspace are independent:

$$U_m = \begin{bmatrix} v_1^T \\ \vdots \\ v_D^T \end{bmatrix}. \tag{3.82}$$

Each row is a vector of type $E \times 1$. The speaker-adapted mean is:

$$\mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix} = \begin{bmatrix} v_1^T \\ \vdots \\ v_D^T \end{bmatrix} w = U_m w. \tag{3.83}$$

Differentiating Q with respect to every row v_j, $j = 1...D$,

$$\frac{\partial Q}{\partial v_j} = -\frac{1}{2}\frac{\partial}{\partial v_j} \sum_{m,t,q} \gamma_m(t) \sum_{d=1}^{D} (\mu_d - o_d)^2 r_d \tag{3.84}$$

$$= -\sum_{t,q} \gamma_m(t) r_j (v_j^T w - o_j) w, \tag{3.85}$$

and therefore:

$$v_j = \left(\sum_{t,q} \gamma_m(t) w w^T \right)^{-1} \sum_{t,q} \gamma_m(t) o_j w, \quad j = 1...D. \tag{3.86}$$

A matrix of rank $E \times E$ must be inverted and stored for every Gaussian of the system [Gal00]. For $D = E = 40$, one would need store the equivalent of 41 times SI models. In many applications, this is considered intolerable [Gal00]. We will concentrate on more practical solutions.

MLES with Diagonal Eigen Correlation

One of the most useful features of the KLT is that dimensions are *orthogonal* (Section 3.2.2).

We transpose this assumption to the HMM framework: it is always true in the least squares framework but not necessarily verified in the log-likelihood framework. We will assume it is almost true. This means that the matrices are almost diagonal:

$$\sum_{t,q} \gamma_m(t) w w^T \approx \left(\sum_{t,q} \gamma_m(t) \right) \sum_q w w^T = \left(\sum_{t,q} \gamma_m(t) \right) \sum_q \begin{bmatrix} w_1^2 & & 0 \\ & \ddots & \\ 0 & & w_E^2 \end{bmatrix}. \tag{3.87}$$

Intuitively, it is the same as assuming that speaker characteristics are independent of each other. For instance, if w_1 is the gender, and w_2 is the speaking rate, we assume that gender has no effect on speaker rate, and vice-versa. As we shall see later, the "transposition" of the concept of orthogonality may be done in a mathematically sound framework if some modifications are applied during the PCA reduction.

The equation (eq. 3.86) can now be solved with minimal cost.

MLES with independent updates

It is possible to achieve an exact solution by updating each Eigenvoice column vector u_j one by one. Let:

$$U_m = [u_1, ..., u_E]. \tag{3.88}$$

The vector u_j is of type $D \times 1$. Means will be updated as is:

$$\mu = \sum_k w_k u_k. \tag{3.89}$$

Differentiating the Q function for each u_j yields:

$$\frac{\partial Q}{\partial u_j} = \sum_{t,q} \gamma_m(t) w_j R_m(\mu - o_t) \tag{3.90}$$

$$= \sum_{t,q} \gamma_m(t) w_j \left(\sum_k w_k u_k - o_t \right). \tag{3.91}$$

Define the complement estimated mean, $\tilde{u}_j$ to be contribution of all other $u_k, k \neq j$:

$$\tilde{u}_j = \sum_{k \neq j} w_k u_k. \tag{3.92}$$

If we are interested in updating only the vector u_j, then:

$$u_j = \left(\sum_{t,q} \gamma_m(t) w_j^2 \right)^{-1} \sum_{t,q} \gamma_m(t) w_j (o_t - \tilde{u}_j). \tag{3.93}$$

This formula can be used as it is, by updating each u_j separately at each iteration, or when we are reasonably close to the solution, by updating all u_j in the same iteration.

This method will be preferred in the implementations. It is more exact than the previous approximation.

3.4.3 Maximum-likelihood dimensionality reduction

It is possible, with certain assumptions, to apply the SVD algorithm in optimal conditions under the maximum-likelihood criterion.

The SVD operates on vectors using the least-squares criterion. It coincides with likelihood if the assumed pdf is a Gaussian with unit variance. We shall develop this point in the next section.

3.5 Gaussianisms and least-square equations

In this section, we will establish the relationship between the posterior probability of MLLR and Eigenvoice estimates as the likelihood equations.

We prove that MLLR and MLED estimates are Gaussian We show that the Gaussian log-likelihood is a special case that is related to squared error.

I apologize for my abusive inflections of Gauss.

3.5.1 Gaussianity of MLLR rows

We take a closer look at the distribution of the rows of the MLLR matrices, or *MLLR rows*. A very important component of the next developments is the assumption of Gaussianity of MLLR rows. We show that the posterior probability of MLLR rows is itself Gaussian. This may be understood by the fact that MLLR rows are multiplied by Gaussian means, and since they are a linear combination of Gaussians, then they are Gaussian themselves.

MLLR: MLLR is described into details in [LW95b]. We reproduce the key derivations here. Throughout this thesis, we will assume one global matrix. The extension to multiple matrices is trivial. In MLLR, the means μ of Gaussian components are transformed via an affine transformation W:

$$\mu = W\zeta, \tag{3.94}$$

$$\zeta = \begin{bmatrix} 1 \\ \cdots \\ \mu_0 \end{bmatrix}, \tag{3.95}$$

where ζ is called the extended mean vector, and μ_0 is the original mean, usually is the SI model. Maximizing the likelihood of W corresponds to maximizing the objective function Q:

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)(\mu - o_t)^T R_m(\mu - o_t). \tag{3.96}$$

We usually assume that covariances are diagonal with elements r_d:

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\sum_{d=1}^{D} r_d(\mu_d - o_d)^2, \tag{3.97}$$

with:

$$\mu = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_D \end{bmatrix}, \tag{3.98}$$

$$o_t = \begin{bmatrix} o_1 \\ \vdots \\ o_D \end{bmatrix}, \tag{3.99}$$

$$R_m = \begin{bmatrix} r_1 & & 0 \\ & \ddots & \\ 0 & & r_D \end{bmatrix}. \tag{3.100}$$

We see that each row contributes to the Q function separately. The affine transformation matrix W is constructed as:

$$W = \begin{bmatrix} w_1^T \\ \vdots \\ w_D^T \end{bmatrix}. \tag{3.101}$$

Each row w_d^T contributes to the log-likelihood function Q with a row log-likelihood Q_d:

$$Q_d = -\frac{1}{2}\sum_{t,m}\gamma_m(t)r_d(\mu_d - o_d)^2. \tag{3.102}$$

The ML estimates y_d^T for the rows w_d^T are well known. They satisfy:

$$y_d = \arg\max_{w_d} Q_d. \tag{3.103}$$

$$y_d = G_d^{-1} z_d, \tag{3.104}$$

$$z_d = \sum_{t,m}\gamma_m(t)r_d o_d \zeta, \tag{3.105}$$

$$G_d = \sum_{t,m}\gamma_m(t)r_d \zeta\zeta^T. \tag{3.106}$$

Posterior distribution of rows w_d^T: We will now show how w_d vectors are distributed. We start again from (eq. 3.102):

$$Q_d = -\frac{1}{2}\sum_{t,m}\gamma_m(t)r_d(\mu_d - o_d)^2. \tag{3.107}$$

Replacing (eq. 3.94 and 3.101) into (eq. 3.107):

$$\begin{aligned}
Q_d &= -\frac{1}{2}\sum_{t,m}\gamma_m(t)r_d(w_d^T\zeta - o_d)^2 r_d && (3.108)\\
&= -\frac{1}{2}\sum_{t,m}\gamma_m(t)r_d\Big((w_d^T\zeta)^2 - 2o_d\zeta^T w_d - o_d^2\Big) && (3.109)\\
&= -\frac{1}{2}\sum_{t,m}\gamma_m(t)r_d\Big(w_d^T\zeta\zeta^T w_d - 2o_d\zeta^T w_d - o_d^2\Big) && (3.110)\\
&= -\frac{1}{2}w_d^T\Big(\sum_{t,m}\gamma_m(t)r_d\zeta\zeta^T\Big)w_d + \Big(\sum_{t,m}\gamma_m(t)r_d o_d\zeta^T\Big)w_d - \frac{1}{2}\beta. && (3.111)
\end{aligned}$$

where β is a constant. We replace quantities y_d and G_d (eq. 3.105, 3.106) into (eq. 3.111):

$$\begin{aligned}
Q_d &= -\frac{1}{2}\Big(w_d^T G_d w_d - 2z_d^T w_d + \beta\Big) && (3.112)\\
&= -\frac{1}{2}\Big(w_d^T G_d w_d - 2G_d y_d^T w_d + \beta\Big) && (3.113)\\
&= -\frac{1}{2}\Big([w_d - y_d]^T G_d[w_d - y_d] + \beta'\Big). && (3.114)
\end{aligned}$$

where β' is another constant. We recall that Q_d is the expected log-likelihood given w_d:

$$Q_d = \mathrm{E}\log p(O|w_d). \tag{3.115}$$

When there is no proper prior distribution for w_d, Bayes' rule yields:

$$\begin{aligned}
\mathrm{E}\log p(w_d|O) &= \mathrm{E}\log p(O|w_d) + \log p(w_d) - \log p(O) && (3.116)\\
&= \mathrm{E}\log p(O|w_d) + \beta'' && (3.117)
\end{aligned}$$

where β'' is a constant. The equation may be extended to conjugate priors of w_d (elliptic distributions [Cho99]).

The functional form of (eq. 3.114) is clearly quadratic, meaning that, conditional of the observations, *transformation rows are Gaussian*. This is an important step towards relating least squares with likelihood objective functions.

The mean and covariance of the distribution are y_d and G_d^{-1}:

$$p(w_d|O) \propto \mathcal{N}\Big(y_d, G_d^{-1}\Big) \tag{3.118}$$

The mean y_d is the MLLR solution. The precision increases proportionally to the number of examples, which is the best achievable performance (Cramer-Rao lower bound for the variance).

3.5.2 Gaussianity of Eigenvoices estimates

This section establishes the Gaussianity of eigenspace location w, which will serve as a basis for Section 6.3.

The demonstration is similar to that of the previous section. Once again, we start with equation (eq. 3.96):

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)(\mu - o_t)^T R_m(\mu - o_t). \tag{3.119}$$

Means are updated given the following equation:

$$\mu = U_m w. \tag{3.120}$$

Substituting into (eq. 3.96):

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)(U_m w - o_t)^T R_m(U_m w - o_t) \tag{3.121}$$

$$= -\frac{1}{2}\sum_{t,m}\gamma_m(t)\Big(w^T U_m^T R U_m w - 2o_t^T R_m U_m w - o_t^T R_m o_t\Big). \tag{3.122}$$

We define the quantities:

$$G = \sum_{t,m}\gamma_m(t)U_m^T R_m U_m, \tag{3.123}$$

$$z = \sum_{t,m}\gamma_m(t)U_m^T R_m o_t, \tag{3.124}$$

$$y = G^{-1}z. \tag{3.125}$$

In particular, y coincides with MLED (Section 3.4.1), as defined by (eq. 3.73, p. 25) The quadratic exponent is becomes visible again when we introduce the quantities into (eq. 3.122):

$$Q = -\frac{1}{2}\Big(w^T G w - 2z^T w + \beta\Big) \tag{3.126}$$

$$= -\frac{1}{2}(w - y)^T G(w - y) + \beta'. \tag{3.127}$$

where β, β' are constants.

In the absence of prior, or, with small modifications, with Gaussian priors, then we can state:

$$p(w \mid O) \propto \mathcal{N}\Big(y, G^{-1}\Big). \tag{3.128}$$

The eigenspace location w is Gaussian. The mean of the Gaussian is called the MLED estimate Trivially, its conjugate prior is also Gaussian.

3.6 Root modulation

In this section, we will present the formulation that allows the use of PCA in the HMM's ML framework. It is based on a normalization of the variables called *root modulation*.

3.6.1 Optimal subspace regression

We now show that PCA may be applied to yield maximum-likelihood estimates with the following remarks:

1. The posterior probability of Gaussian mean, or MLLR rows, is Gaussian (Section 3.5.1).

2. Maximum-likelihood estimation is equal to least-squares estimation of the means when variables are Gaussian with equal variances (Section 3.2.4).

3. Pythagoras' theorem is equivalent in divergence and Euclidean frameworks (Section 3.2.4).

4. The source of variability is assumed to be an isotropic variable: we will see it in Section 3.6.1-3.6.1.

MLLR subspaces

We will apply PCA to observations of MLLR rows, as defined in Section 3.5.1. In Section 3.3, we applied the dimensionality reduction to the means of the models themselves. As noted by Gales [Gal00], in LVCSR, the eigenvoices constitute a signficant amount of data that may be considered intolerably large. For instance, an eigenspace of $E = 100$, for the BN task (Section 8.3) will take 500 MB (Mega Bytes of computer memory). Therefore, we replace supervectors constructed from HMM means of Gaussian components, by the supervectors constructed from MLLR rows. Because we have showed that MLLR rows are Gaussians, and because mean parameters are also Gaussian [GL94], it is trivial to go from MLLR to Gaussian mean parameters.

From Q to ε

In this section, we extend the case of homoscedastic Gaussians to MLLR rows (Section 3.2.4) to the HMM framework.

The objective function in the HMM-MLLR transformation is the sum of all likelihood of all speakers q, as defined in Section 3.5.1. We sum (eq. 3.114) over all dimensions d of all speakers q:

$$Q = -\frac{1}{2}\sum_{q,d}(w_d^{(q)} - y_d^{(q)})^T G_d^{(q)}(w_d^{(q)} - y_d^{(q)}). \tag{3.129}$$

The transformations rows w_d^T are the random variables. The transformation mean $y_d^{(q)}$ and covariance $G_d^{(q)}$ are defined in (eq. 3.104-3.106, p. 29). We would like to transform this equation into a least-squares problem, as in (eq. 3.10). Let us define the supervectors w_q and y_q, and super covariance as usual. The supervector for the mean is:

$$w_q = \begin{bmatrix} w_1^{(q)} \\ \cdots \\ w_2^{(q)} \\ \cdots \\ \vdots \\ \cdots \\ w_D^{(q)} \end{bmatrix}, \tag{3.130}$$

and the ML mean:

$$y_q = \begin{bmatrix} y_1^{(q)} \\ \cdots \\ y_2^{(q)} \\ \cdots \\ \vdots \\ \cdots \\ y_D^{(q)} \end{bmatrix}, \tag{3.131}$$

with covariance:

$$G_q = \begin{bmatrix} G_1^{(q)} & & 0 \\ & \ddots & \\ 0 & & G_D^{(q)} \end{bmatrix}. \tag{3.132}$$

The MLLR super mean vectors w_q and y_q are defined for each speaker, and are each of type $D(D+1) \times 1$. Each G_q is a block-diagonal matrix of type $D(D+1) \times D(D+1)$, with each block of size $(D+1) \times (D+1)$. Replacing into the Q function of (eq. 3.129), one sees:

$$Q = -\frac{1}{2} \sum_q (w_q - y_q)^T G_q (w_q - y_q). \tag{3.133}$$

It is still not a *homoscedastic* problem, since G_q matrices are not shared amongst speakers. Even if speakers are supposed to behave the same way, this may not be assumed if they do not speak the same content in the adaptation sentences.

Root modulation

We therefore propose a normalization scheme that will map us back to the least-squares domain. Let us look again at the functional form of each submatrix G_d (eq. 3.106):

$$G_d = \sum_{t,m} \gamma_m(t) r_d \zeta\zeta^T = \sum_{t,m} \gamma_m(t) r_d \begin{bmatrix} 1 & \vdots & \mathbf{1}_D^T \\ \cdots & & \cdots \\ \mathbf{1}_D & \vdots & \mu_m \mu_m^T \end{bmatrix}. \tag{3.134}$$

Except for the weighting $\gamma_m(t)$, there is nothing in G_d that is learned at adaptation time. It is merely a regularization matrix that cancels the SI precision r_d with the autocorrelate of SI means $\mu_m \mu_m^T$. Therefore, G_d is merely a reflection of the linguistic content observed for the speaker, and is speaker-independent.

Therefore, we will assume that G_d may be computed relatively reliably. We proceed to transform supervectors w_q, y_q into:

$$\tilde{w}_q = G_q^{\frac{1}{2}} w_q, \tag{3.135}$$

$$\tilde{y}_q = G_q^{\frac{1}{2}} y_q = G_q^{-\frac{1}{2}} z_q. \tag{3.136}$$

We call this normalization the *root modulation* because of the root of the precision is involved. The transformed space is called the *root space*. Since this is a transformation of models, it is a pure algebraic change of variables that does not affect the log-likelihood of HMMs. Nonetheless, the posterior of the transformation must be multiplied by $|G_q|$ (for more information, see Chapter 4). Since we are not modelling covariances, this is a constant term that is ignored.

Remember that w_q is the approximated mean, and y_q is the exact MLLR estimate. Since $\tilde{w}_q$ and $\tilde{y}_q$ have been normalized by the expected standard deviation due to linguistic content, they are called *linguistically normalized* supervectors.

We substitute these vectors into (eq. 3.133):

$$Q = -\frac{1}{2} \sum_q (\tilde{w}_q - \tilde{y}_q)^T (\tilde{w}_q - \tilde{y}_q) = -2\varepsilon. \tag{3.137}$$

We have reached the functional form of (eq. 3.2, p. 15), which defines a least-squares problem.

In the root space, the least-squares criterion coincides with the expected log likelihood. We can use PCA under optimal conditions to reduce the dimensionality of the space. Euclidean projections are also available under an ML criterion. We shall develop the Eigenvoices estimation framework again. This time, however, estimations will be *exactly* the same as Euclidean projections.

Root disambiguation

The usual definition of $A^{\frac{1}{2}}$, when A is a positive definite symmetric matrix is related with its eigen-decomposition. That is,

$$A = M\Lambda M^T, \tag{3.138}$$

$$A^{\frac{1}{2}} = M\Lambda^{\frac{1}{2}} M^T, \tag{3.139}$$

$$A^{\frac{1}{2}} = A^{\frac{T}{2}}. \tag{3.140}$$

The matrix M is square an orthogonal. The matrix Λ is diagonal, and its exponentiate is assumed component-wise exponentiation.

It is possible to use any other normalization scheme in (eq 3.135,3.136). The root matrices are defined up to any rotation. For instance, one could use a Cholesky decomposition. The rotations have no effect on the objective function. However, if we believe that the source variables, after transformation, are *isotropic normal*, then the definition (eq. 3.139) must be used. We will always use this restriction.

3.7 Re-estimation in the root space

In this section, we solve the three fundamental problems of estimation in Eigenvoices estimation in the root space:

1. How to estimate speaker-adapted models.
2. How to reduce the dimensionality.
3. How to reestimate eigenspaces in the Baum-Welch framework.

The root modulation was defined as:

$$\tilde{w}_q = G_q^{\frac{1}{2}} w_q, \tag{3.141}$$

$$\tilde{y}_q = G_q^{\frac{1}{2}} y_q = G_q^{-\frac{1}{2}} z_q. \tag{3.142}$$

The precision matrix, G_d, is:

$$G_d = \sum_{t,m} \gamma_m(t) r_d \begin{bmatrix} 1 & \vdots & \mathbf{1}_D^T \\ \cdots & & \cdots \\ \mathbf{1}_D & \vdots & \mu_m \mu_m^T \end{bmatrix}. \tag{3.143}$$

It is believed to map variables into a space that is independent of the linguistic content of the utterance. The linguistic content defines what the speaker said during the adaptation phase. Indirectly, it defines what Gaussians will be hit. Although there is a correlation between linguistic content and speaker identification in some tasks, for practical purposes, we can safely ignore it.

The root modulation was shown to turn the likelihood equation into a least-squares:

$$Q = -\frac{1}{2} \sum_{q,d} (w_d^{(q)} - y_d^{(q)})^T G_d^{(q)} (w_d^{(q)} - y_d^{(q)}) = -\frac{1}{2} \sum_q (\tilde{w}_q - \tilde{y}_q)^T (\tilde{w}_q - \tilde{y}_q). \tag{3.144}$$

3.7.1 The estimation of speaker-adapted models

Now, we find best location of a speaker model in the speaker space (MLED). We will see that it is a simple Euclidean projection. Suppose that we have found a linear subspace U in the root modulated space. It was trained off-line.

We are now concerned with finding the best speaker-adapted model that lies in U. A speaker is presented to the system. The E-step of the EM algorithm is performed. We have to maximize the following auxiliary function with respect to $\tilde{w}_q$:

$$Q = -\frac{1}{2}(\tilde{w} - \tilde{y})^T(\tilde{w} - \tilde{y}), \tag{3.145}$$

subject to the constraint:

$$\tilde{w} = U\vartheta. \tag{3.146}$$

ϑ is a vector of size $E \times 1$ that characterizes the location of the speaker in the eigenspace. The solution is found by projecting $\tilde{y}$ onto U as in (eq. 3.9). We differentiate (eq. 3.145) w.r.t. ϑ:

$$\frac{\partial Q}{\partial \vartheta} = -U^T(U\vartheta - \tilde{y}) = 0. \tag{3.147}$$

This is solved by the familiar Moore-Penrose inverse:

$$\vartheta = (U^TU)^{-1}U^T\tilde{y}. \tag{3.148}$$

Replacing with definitions (eq. 3.146, 3.141, and 3.142),

$$\tilde{w} = U\vartheta = U(U^TU)^{-1}U^T\tilde{y}, \tag{3.149}$$

$$w = G^{-\frac{1}{2}}U(U^TU)^{-1}U^TG^{\frac{1}{2}}y = G^{-\frac{1}{2}}U(U^TU)^{-1}U^TG^{-\frac{1}{2}}z. \tag{3.150}$$

Since the matrix G is block-diagonal, we can decompose each $(D+1)\times(D+1)$ subsystems of equations. Let U_j be the eigenspace that corresponds to dimension j. Each row is found separately:

$$w_j = G_j^{-\frac{1}{2}}U_j(U^TU)^{-1}U_j^TG_j^{-\frac{1}{2}}z_j. \tag{3.151}$$

By construction, $U^TU = I_E$ if no re-estimation takes places. The equation may be reduced to:

$$w_j = G_j^{-\frac{1}{2}}U_jU_j^TG^{-\frac{1}{2}}z_j. \tag{3.152}$$

Therefore, The estimation of speaker-adapted models reduces to:

1. Normalization into the root space: $G^{-\frac{1}{2}}z_j$.
2. Projection: $U_jU_j^T$.
3. Re-normalization into the original space: $G_j^{-\frac{1}{2}}$.

3.7.2 Dimensionality reduction

This section parallels Section 3.2.2. We observe a number of speaker-adapted models in the training data. We mean to use PCA for:

- Removing noise due to unreliable estimation of the observed speakers.
- Reduce the dimensionality of the space in order to
    1. Decrease the number of parameters to estimate, and therefore increase the reliability of the estimate (see Section 3.1.2).
    2. Reduce the run-time requirements of speaker-adaptation.

The observed speakers $\tilde{y}_q$ are collected in the root space. The likelihood of speaker-adapted training is given in Section 3.6.1, (eq. 3.137):

$$Q = -\frac{1}{2}\sum_q (\tilde{w}_q - \tilde{y}_q)^T(\tilde{w}_q - \tilde{y}_q). \tag{3.153}$$

We must relate it with Section 3.2.1, (eq. 3.2):

$$\varepsilon = B^{-1}\sum_{k=1}^{B}(x_k - y_k)^T(x_k - y_k), \tag{3.154}$$

which was the foundation of the least-squares PCA in section 3.2.2, (eq. 3.16):

$$\arg\min_U \varepsilon = \arg\max_U \operatorname{tr} U^T V \Lambda V^T U. \tag{3.155}$$

The dimensionality reduction step is therefore summarized as:

1. Collect all supervectors $\tilde{w}_q$.
2. Form the observation matrix $X = [\tilde{w}_1, ..., \tilde{w}_2]$.
3. Compute the SVD of $X = U\Lambda V^T$.
4. Keep only the desired number of $E < T$ eigenvectors in U.

3.7.3 Eigenspace re-estimation

The resolution of unity is central to PCA. It allows us to remove the mathematical burden associated with the matrix determinant (Section 4.2). It might be desirable to alleviate the constraint, and re-estimate the eigenspace in the Baum-Welch framework. This is the equivalent of MLES. It is also useful when the eigenspace is not inferred from correlations, but learned from arbitrary eigenvalues.

Suppose that we have collected eigenspace locations ϑ_q for all speakers in the training database. The goal is to optimize (eq. 3.137):

$$Q = -\frac{1}{2}\sum_q (\tilde{w}_q - \tilde{y}_q)^T(\tilde{w}_q - \tilde{y}_q). \tag{3.156}$$

We define each row of the eigenspace u_d:

$$U = \begin{bmatrix} u_1^T \\ \vdots \\ u_{D+1}^T \end{bmatrix}. \tag{3.157}$$

The transformed vectors are:

$$\tilde{w} = U\vartheta = \begin{bmatrix} u_1^T\vartheta \\ \vdots \\ u_{D+1}^T\vartheta \end{bmatrix}. \tag{3.158}$$

Differentiating the Q function (eq. 3.156):

$$\frac{\partial Q}{\partial u_j} = -\frac{1}{2}\frac{\partial}{\partial u_j}\sum_q\sum_{d=1}^{D+1}(u_d^T\vartheta_q - \tilde{y}_d)^2 \tag{3.159}$$

$$= -\frac{1}{2}\frac{\partial}{\partial u_j}\sum_q\sum_{d=1}^{D+1}\left(u_d\vartheta\vartheta^T u_d - 2\vartheta_q^T\tilde{y}_d\right) \tag{3.160}$$

and thus:

$$u_j = \left(\sum_q \vartheta_q \vartheta_q^T \right)^{-1} \sum_q \tilde{y}_j \vartheta_q. \tag{3.161}$$

The eigenspace is therefore summarized more compactly as:

$$U = \left(\sum_q \vartheta_q \vartheta_q^T \right)^{-1} \sum_q \vartheta_q^T \otimes \tilde{y}. \tag{3.162}$$

3.7.4 PCA vs. MLES

Since we have *two* different algorithms to estimate the eigenspace, it might not be clear whether or not they are redundant. In other words, after PCA, is there any gain of applying MLES? The central distinction is the same as that between *projection* and *transformation*. PCA uses an SVD decomposition of the covariance matrix. The SVD yields *orthogonal* transformations, whereas MLES is not constrained to orthogonality. This point, as seen in Section 4.2, introduces considerable mathematical complications.

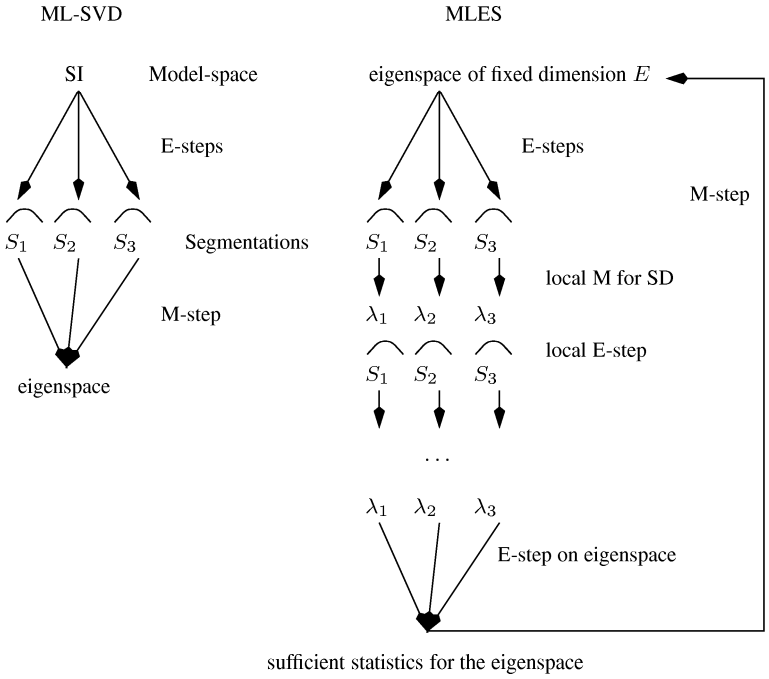


Figure 3.4: Trajectory of the Eigenspace estimation

Table 3.1 summarizes the differences between the algorithms. On Figure 3.4 we can see that the trajectories are different. The main difference is the resolution of unity. The SVD yields an orthogonal decomposition, and the positions of speakers are implicitly determined using (eq. 3.9, p. 16). In the MLES framework, the projection process is separated from the estimation of the eigenspace. Therefore, it is possible to learn arbitrary knowledge by setting speaker locations manually. Also, the eigenspace is not necessarily orthogonal.

3.8 Discriminative approaches

Until now, we have been using PCA as the dimensionality reduction. It will minimize the encoding error, and is suitable for regression tasks.

Characteristic	MLES	SVD
Sufficient statistics	projections of speakers $O(E \times P)$	all speaker models $O(B \times P)$
Dimension of the eigenspace	fixed: E	$B \rightarrow E$
Projective EM steps	Alternates between segmentation and speaker	Only a projection of SI to SD spaces
EM hidden variables	Segmentation and speaker location	segmentation given SI
Resolution of Unity	Non-orthogonal basis	$U^T U = I$
Can introduce External knowledge	Yes	No

Table 3.1: Features of MLES and ML-SVD

However, the optimality of PCA is contested in classification tasks. We will tailor the model-reduction technique to classification tasks.

3.8.1 Speaker-adapted models and discrimination

It is well accepted, but very rarely used, that speaker-dependent speech exhibits more discriminative properties (see Section 5.4).

We are interested in designing a system that learns speaker-dependent phoneme discriminative models. It can be understood as an SAT-MMIE scheme.

3.8.2 Fisher discriminant

Fisher's discriminant is the most popular objective function. It leads to the notorious LDA technique. We will describe LDA as an extension to PCA. Then, we define a new discriminative criterion based on MMI.

PCA: a review

Let us use the notations set forth in Section 3.2.2. We have a collection of samples x_k. We are interested in finding a linear reduction matrix U, such that the squared error (eq. 3.10) is minimized:

$$\varepsilon = \sum_k ||x_k - UU^T x_k||^2. \tag{3.163}$$

We found out that it was also closely related to the eigen-decomposition of the autocorrelation matrix (eq. 3.16):

$$\arg\min_U \varepsilon = \arg\max_U \operatorname{tr} U^T V \Lambda V^T U. \tag{3.164}$$

This was also found to maximize the likelihood in certain conditions (Section 3.6.1). What is important to note in this equation is that *the estimation of the ML projection is equivalent to the maximization of the **trace** of the projected autocorrelation.* In the remainder, we will refer to the optimization of matrices as the maximization of the corresponding projected trace criterion.

We will move on to methods that take into account class discrimination.

LDA: introducing class discrimination

Let us, for the purpose of argument, assume that we segregated speaker models x_k into classes. For simplicity, we will have an equal number of samples associated to classes. Classes will be equiprobable. Suppose furthermore that we are also interested in separating

these classes. It is not appropriate for a speech recognition task, but it is a plausible scenario for gender-classification or speaker identification.

In Fisher analysis [DH73], we identify two matrices that describe regression and discrimination about the training samples. The within-class matrix is defined as:

$$S_w = \sum_{c=1}^{C} S_c, \tag{3.165}$$

$$S_c = \sum_{j \in c} (x_j - m_c)(x_j - m_c)^T. \tag{3.166}$$

Each S_c is the covariance within one class. The vector m_c is the mean of the class c. Maximizing S_w is good for regression.

The between-class matrix is:

$$S_b = \sum_{c} (m_c - m)(m_c - m)^T. \tag{3.167}$$

The vector m is the total mean.

The LDA maximizes the *Fisher discriminant* J [DH73]:

$$J = |S_w^{-1} S_b|. \tag{3.168}$$

The criterion reflects the balance to strike between both conflicting interests: pure regression which attempts to model phenomena as precisely as possible, and pure discrimination which merely attempts to separate training examples. The equation may be reduced to an SVD problem. This will not help speech recognition, however, and we will define the classes differently. The question that also remains is whether this minimization is appropriate according to divergence measures.

Appropriate classes for ASR

In ASR, we are not so much interested in distinguishing speakers, but rather, between phonemes. The discrimination is measured *within* one model. We will define two matrices. One corresponds to the regression problem. The other one does not correspond directly to pure discrimination. The deviation of this matrix from ML simulates model confusion.

The within-class matrix will be defined as usual, as in (eq. 3.155):

$$S_w = V \Lambda V^T. \tag{3.169}$$

The between-class matrix S_b will be defined the same way as S_w. However, instead of using forced alignment, a decoding pass generates a lattice or NBest list. Statistics are collected using the posterior probabilities [WP00].

3.8.3 MAP, deleted interpolation, and ML

We now introduce the equivalence relationships between MAP, deleted interpolation (DI), and ML. By definition, MAP with conjugate priors defines a problem that yields an ML problem with a discounted penalty. In the Gaussian framework, the conjugate prior of a mean is also a Gaussian [GL94, GL92]. The MAP solution is:

$$\mu_{\text{MAP}} = \left(\sum_t \gamma_m(t) + \tau \right)^{-1} \left[\sum_t \gamma_m(t) o_t + \tau \mu_0 \right], \tag{3.170}$$

where τ is a hyper-parameter that specifies the confidence on the prior in number of frames. The prior mean is μ_0. The objective functions are:

$$Q_{\text{ML}} = -\frac{1}{2}\sum_{t,m}\gamma_m(t)(\mu - o_t)^T R_m^{-1}(\mu - o_t), \tag{3.171}$$

$$Q_0 = -\frac{1}{2}(\mu - \mu_0)^T R_m^{-1}(\mu - \mu_0). \tag{3.172}$$

The final MAP objective is:

$$Q_{\text{MAP}} = Q_{\text{ML}} + \tau Q_0, \tag{3.173}$$

and the MAP solution is that of a quadratic function optimization:

$$\mu_{\text{MAP}} = \arg\max_{\mu} Q_{\text{MAP}}. \tag{3.174}$$

Similarly, the deleted interpolation (DI) formula is:

$$\mu_{\text{DI}} = (1-\alpha)\left(\sum_t \gamma_m(t)\right)^{-1}\sum_t \gamma_m(t)o_t + \alpha\mu_0. \tag{3.175}$$

It can be traced backwards into an ML problem:

$$Q_{\text{DI}} = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\Big[(1-\alpha)o_t + \alpha\mu_0 - \mu\Big]^T C^{-1}\Big[(1-\alpha)o_t + \alpha\mu_0 - \mu\Big]. \tag{3.176}$$

With a carefully selected interpolation weight, deleted interpolation performs only slightly worse experimentally than MAP [Ngu98].

In other words, MAP and DI can be recast as a modified ML problem, where we interpolate either the sufficient statistics (MAP) or the observations (DI).

3.8.4 MMI discriminant

We endow the MMI formulation with PCA.

A quick review. For ASR, discriminating *between* speakers is irrelevant. We seek models that maximize, per speaker, the MMI criterion [BBdSM86, NNP88, Val95]. Let us employ the notations of Section 3.4.2 for the partitioning $\{O_q\}$ of the training data. Let also w_q be a MLLR super mean vector defined in Section 3.6.1, with associated eigenspace U. Finally, let $\mathcal{W}$ be a possible transcription for O_q. A sentence O_q uttered by a speaker w_q using word sequence $\mathcal{W}$ has a joint posterior probability:

$$p(O_q, w_q, \mathcal{W}|U) = p(O_q, w_q|U, \mathcal{W}_q)p(\mathcal{W}) \tag{3.177}$$

where $p(\mathcal{W})$ is the language model, for best results set to a unigram [WP00]. The correct transcription for utterance O_q is $\mathcal{W}_q$.

The MMI criterion is:

$$e^H = \prod_q \frac{\int \mathrm{d}w_q\, p(O_q, w_q|\mathcal{W}_q, U)p(\mathcal{W}_q)}{\int \mathrm{d}w_q \sum_{\mathcal{W}} p(O_q, w|\mathcal{W}, U)p(\mathcal{W})}. \tag{3.178}$$

The criterion has a numerator and a denominator. It behaves much like (eq. 3.168): the numerator is an ML regression objective, and the denominator helps discrimination. In the log-domain, we have:

$$H = \sum_q \left[\log\int \mathrm{d}w_q\, p(O_q, \mathcal{W}_q|U) - \log\left\{\int \mathrm{d}w \sum_{\mathcal{W}} p(O_q, \mathcal{W}, w|U)\right\}\right]. \tag{3.179}$$

The maximization of criterion (eq. 3.179) has no closed-form solution. If we just follow the gradient, we have a DI formulation [SMWN99]. Normandin [Nor91] uses Gopalakrishnan's inequality [GKNN89] to devise a fast gradient descent. The results are formalized to continuous densities by Gunawardana [Gun01a]. Define the statistics collected for the numerator as $\gamma_m(t)$ with observations o_t, and the statistics associated to the denominator $\bar{\gamma}_m(t)$ with observations $\bar{o}_t$. The final formula for the SI case is:

$$\mu = \left(\sum_t \gamma_m(t) - \sum_t \bar{\gamma}_m(t) + \tau \right)^{-1} \left[\sum_t \left(\gamma_m(t) o_t - \bar{\gamma}_m(t) \hat{o}_t \right) + \tau \mu_0 \right]. \tag{3.180}$$

The parameter τ is the size of the gradient step, which is large enough to ensure positivity assumptions [Gun01a]. The ML mean is μ_0.

MMI optimization The gradient descent can be cast a *minimization* problem, such as the least-squares PCA problem of (eq. 3.12 p. 16). The equation (eq. 3.180) can be traced backwards as a MAP estimate in the form of (eq. 3.173) where the prior mean is μ_0 with weight τ and the estimates are the difference between observations and unconstrained observations. Let the ML (numerator), unconstrained decoding ML (denominator), and prior functions be respectively:

$$Q_{\text{ML}} = -\frac{1}{2} \sum_{t,m} \gamma_m(t) (\mu - o_t)^T R_m^{-1} (\mu - o_t), \tag{3.181}$$

$$\bar{Q}_{\text{ML}} = -\frac{1}{2} \sum_{t,m} \bar{\gamma}_m(t) (\mu - \bar{o}_t)^T R_m^{-1} (\mu - \bar{o}_t), \tag{3.182}$$

$$Q_0 = -\frac{1}{2} (\mu - \mu_0)^T R_m^{-1} (\mu - \mu_0). \tag{3.183}$$

Up to a constant, these functions are:

$$Q_{\text{ML}} = -\frac{1}{2} \text{tr}\, S_w = \text{E}_\gamma \int \text{d}w_q \, \log p(O_q, w_q, \mathcal{W}_q | U) p(\mathcal{W}_q) \tag{3.184}$$

$$\bar{Q}_{\text{ML}} = -\frac{1}{2} \text{tr}\, S_b = \text{E}_\gamma \int \text{d}\mathcal{W} \int \text{d}w_q \, \log p(O_q, w_q, \mathcal{W} | U) p(\mathcal{W}) \tag{3.185}$$

$$Q_0 = \text{E}_{\gamma, \mathcal{W}} \int \text{d}w_q \, \log p(O_q, w_q, \mathcal{W} | U) \tag{3.186}$$

At each gradient descent step, the MMIE solves the equation:

$$\mu_{\text{MMIE}} = \arg\max_\mu \left[Q_{\text{ML}} - \bar{Q}_{\text{ML}} \right] + \tau Q_0, \tag{3.187}$$

which is equivalent to (eq. 3.174), which in turn is equivalent to an ML problem such as that of (eq. 3.96). This means that the function (eq. 3.179) may be solved iteratively with the minimization of the objective function Q_∇ at each gradient step:

$$Q_\nabla = Q_{\text{ML}} - \bar{Q}_{\text{ML}} + \tau Q_0 = -\frac{1}{2} \text{tr} \left[S_w - S_b \right] + \tau Q_0. \tag{3.188}$$

Algorithm. The criterion (eq. 3.188) is not exactly the same as the Fisher discriminant (eq. 3.168). The discriminative algorithm is almost the same as standard PCA-based eigenvoices:

1. Collect statistics for the numerator lattices S_w.
2. Collect statistics for the denominator lattices S_b.

3. Add a discounted matrix corresponding to Q_0, which is the original Eigenvoices-PCA matrix.

4. Decompose using the SVD.

5. Iterate until convergence ($S_w \approx S_b$ in trace or some threshold is met).

This is a speaker-adaptive MMI algorithm. Note how Q functions may be traded with matrix traces: this is the result of the least-squares criterion being equivalent to ML. We have shown that in MMI *each gradient step can be written as a quadratic function minimization*, which in turn is equivalent to a maximum-likelihood estimation.

3.9 Piecewise linear decomposition

Because of its simplicity and the presence of closed-form solutions, the linear assumption has proven very effective in many pattern regression problems. However, the linearity constraint has no legitimacy. In this section, we investigate a simple non-linear model. Non-linear regression takes on the assumption that dimensions depend on the location of the speaker. In the simplest case, we would assume for instance that loudness of speech depends on the speaker's gender.

There are many alternatives to linear regression. We will simplify the framework as much as possible. We will restrict the model to a two-level hierarchical piecewise linear regression. There is one level of location-independent parameters (gender, accent, ...), and one location-dependent level (loudness, pitch, ...).

3.9.1 Scalar piecewise linear curves

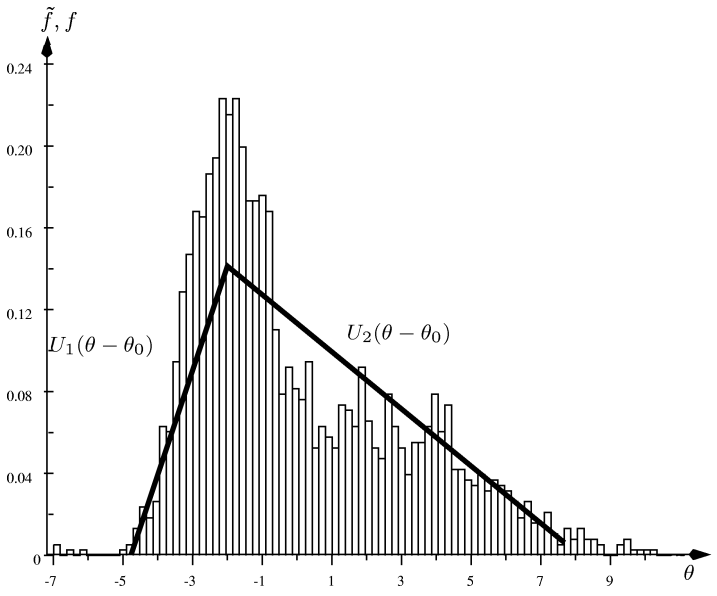


Figure 3.5: Piecewise linear approximation in the scalar case

Before explaining the full model, we first examine the simple scalar piecewise linear approximation. Let us suppose that we would like to model a function $f(\theta)$. The function is drawn on Figure 3.5. Ultimately, the function $f(\cdot)$ represents HMM model parameters. Alternatively, it can be MLLR rows. The parameter θ is the eigenvalues vectors.

In the linear approximation $\tilde{f}_U$, we find a $U \in \mathbb{R}$ such that:

$$\tilde{f}_U(\theta) \approx f(\theta), \tag{3.189}$$

in the least squares sense, or:

$$\hat{U} = \arg\min_U \int \mathrm{d}\theta\, p(\theta)||U\theta - f(\theta)||^2, \tag{3.190}$$

where p is a probability measure on θ. The scalar U is the eigenspace. There are many alternatives to the linear approximation. We use the simplest case, which we believe to be the piecewise linear approximation. We draw two segments: the first one stops at θ_0, and the other one starts at θ_1, as drawn on Figure 3.5. The piecewise linear approximation will be $\tilde{f}_{U_1,U_2}$,

$$\tilde{f}_{U_1,U_2}(\theta) = \begin{cases} U_1(\theta - \theta_0) & \text{if } \theta - \theta_0 \leq 0, \\ U_2(\theta - \theta_0) & \text{if } \theta - \theta_0 > 0. \end{cases} \tag{3.191}$$

The estimation of the function involves three parameters: U_1, U_2, and θ_0. More generally, we have to estimate the *topology* of the function, that is, how many segments comprise the function. The topology defines the complexity of the function, and is dictated by the amount of data and the non-linearity of the function.

There is no closed-form solution that estimates all three parameters U_1, U_2 and θ_0 simultaneously. However, one can use an iterative EM process. The estimation of U_1 and U_2 is easy once θ_0 is known. When U_1 and U_2 are known, the estimation of θ_0 is easy, but only in the scalar case. We will see that generalization to multidimensional linear boundaries is more difficult (Section 3.9.5).

3.9.2 2×2-dimensional piecewise linear hyperplanes

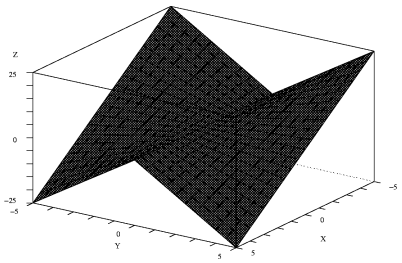


Figure 3.6: Two-dimensional extension: linear in X and piecewise linear in Y

The generalization of the scalar case is not trivial. Amongst all possibilities, we avail ourselves of the simplest. The parameter θ is now a vector of dimension 2, $\theta \in \mathbb{R}^2$. The function also results in a vector of dimension two, $f \in \mathbb{R}^2 \to \mathbb{R}^2$. We show the extension on Figure 3.6. We plot only the first dimension of the function. The criterion to optimize is:

$$\varepsilon = \int \mathrm{d}\theta\, p(\theta)[\tilde{f}(\theta) - f(\theta)]^T[\tilde{f}(\theta) - f(\theta)]. \tag{3.192}$$

We retain the topology: we have only two hyperplanes (previously segments). We decompose the θ-space into a location independent (high), $\theta_h = \theta_1$, and a location dependent (low) $\theta_l = \theta_2$:

$$\theta = \begin{bmatrix} \theta_h \\ \theta_l \end{bmatrix}. \tag{3.193}$$

The location independent part is modeled through linear regression U_0 as in standard Eigenvoices. The location dependent is developed through a piecewise linear approximation. The hyperplane boundaries are defined *using θ_h only*. Therefore, the regression coefficients U_1 and U_2 are of type 2×1. To avoid confusion, we call them U_{l1} and U_{l2}. In general they could be 2×2. The function is approximated thus:

$$\tilde{f} = U_h\theta_1 + \begin{cases} U_{l1}\theta_2 & \text{if } \theta_1 - \theta_0 \leq 0, \\ U_{l2}\theta_2 & \text{if } \theta_1 - \theta_0 > 0. \end{cases} \tag{3.194}$$

The dimension of U_h is 2×1. Compared with a linear model, we double the storage size for the location-dependent part of U (U_{l1} and U_{l2} need to be stored).

3.9.3 Binary $E \times P$ piecewise linear hyperplanes

Now we extend the model to more dimensions. First, the function to be approximated, $f(\theta)$, is now taken to output P-dimensional vectors. In the HMM regression framework, P is GD, the total number of parameters to model mean vectors of the HMMs. In the MLLR framework (Section 3.5.1, discussed in 3.6.1), the total amount of parameters to model is $D(D+1)$.

Then, we extend the θ vector to E dimensions. As in (eq. 3.193), we distinguish two parts, θ_h and θ_l, corresponding to the location-independent and location-dependent eigenvalues. These vectors have dimensions E_h and E_l respectively, such that:

$$E = E_h + E_l. \tag{3.195}$$

Incidentally, as usual, the apparent dimension $P >> E$. The eigenspaces have dimensions that agree: U_h is of type $P \times E_h$. U_{l1} and U_{l2} is of type $P \times E_l$. Again, there were other ways to generalize, but we chose the simplest.

Finally, the hyperplane boundary region must be generalized. We choose a vector $v \in \mathbb{R}^{E_h}$, called *boundary decision vector*. Based on the negativeness of the boundary decision function $b_v(\theta_h)$:

$$b_v(\theta) = \theta_h^T v, \tag{3.196}$$

we will consider ourselves on the hyperplane U_{l1} or U_{l2}. In this case, there are only two possibilities, corresponding to the sign of θ_h. This is a binary decision and there are only two hyperplanes.

Finally, the function approximation can be written as:

$$\tilde{f}(\theta) = U_h\theta_h + \begin{cases} U_{l1}\theta_l & \text{if } b_v(\theta_h) \leq 0, \\ U_{l2}\theta_l & \text{if } b_v(\theta_h) > 0. \end{cases} \tag{3.197}$$

We can match this model to the introduction: we have a set of location-independent features θ_h. In the case when $E_h = 1$, we have the gender. In this case the location is determined by the gender. Then, we have location-dependent (or equivalently gender-dependent) features, θ_l, which determine variations of HMM parameters, based on the gender. For instance, it could be the effect of pitch.

3.9.4 N-ary piecewise linear hyperplanes

Preliminary results on TIMIT showed that gender-dependent eigenspaces were performing better than gender-independent eigenspaces, indicating a non-linearity. We would like to ascertain whether there are more non-linearities to be discovered. Therefore, we generalize again the binary piecewise model to more than two hyperplanes. This is done by introducing a combination of linear boundary decisions. We use a cascade of linear boundary decisions. Limitations of this approach were discussed in the notoriously condemning book by Minsky [MP69]. Hyperplane pieces are found by dichotomy.

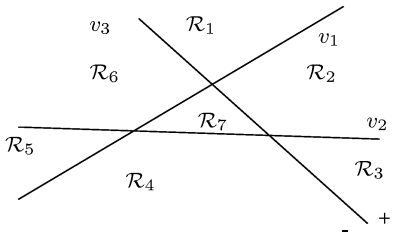


Figure 3.7: Boundary decision vectors and regions: v_1 differentiates regions $\mathcal{R}_1, \mathcal{R}_6, \mathcal{R}_5$ from regions $\mathcal{R}_2, \mathcal{R}_7, \mathcal{R}_4, \mathcal{R}_3$.

Figure 3.7 shows an example of such partitioning. For our experiments, we chose canonical boundary decision vectors:

$$v_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} k-1 \text{ times} \\ \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} E_h - k \text{ times} \end{matrix} \tag{3.198}$$

For the particular case of v_1, it is equivalent to splitting according to the gender. The regions are the quadrants of the eigenspace. We have a set of boundary decision vectors v_k, which we combine. Let the boundary decision functions b_k be:

$$b_k(\theta_h) = \theta_h^T v_k. \tag{3.199}$$

In the example of the figure, we would have:

$$\tilde{f}(\theta) = U_h\theta_h + \begin{cases} U_{l1}\theta_l & \text{if } b_1(\theta_h) > 0 \text{ and } b_3(\theta_h) > 0, \\ U_{l2}\theta_l & \text{if } b_1(\theta_h) \leq 0 \text{ and } b_2(\theta_h) > 0 \text{ and } b_3(\theta_h) > 0, \\ U_{l3}\theta_l & \text{if } b_2(\theta_h) \leq 0 \text{ and } b_3(\theta_h) > 0, \\ \ldots & \ldots \end{cases} \tag{3.200}$$

The model is now complete.

3.9.5 Estimation of parameters

As with the standard Eigenvoices, we are confronted to the estimation of three kinds of parameters:

1. the initial eigenspaces and topology: this is the equivalent of Section 3.6.1,
2. the eigenspaces in the Baum-Welch retraining: this is the equivalent of Section 3.4.2,
3. the location of a speaker in the eigenspace: this is the equivalent of Section 3.4.1.

The first item is the extension of PCA. The second one represents speaker adaptive training. They both have to do with the estimation of hyper-parameters. The third one is the actual adaptation process whereby SI models are altered. For the logic of exposition, we answer these questions in reverse order.

Optimal location

Since we are in a non-linear framework, the solutions are not necessarily unique. In our structure, the search is simpler. It is sufficient to apply the linear programming [Dan63, Dan49, PTVF92] for each region and select the best:

1: **for all** regions k **do**
2: Find the best location θ_k of the speaker in the region, using MLED (Section 3.4.1), and the simplex method [Dan49]. The eigenspace is $U = [U_h; U_{lk}]$.

$$\hat{\theta} = \arg\max_{\theta} p(O|\theta \in \mathcal{R}_k). \tag{3.201}$$

3: For this location θ_k, compute the likelihood Q_k according to (eq. 3.69).
4: **end for**
5: Select the best Q_k and update models with θ_k using (eq. 3.200).

Alternatively, we can use:

$$\theta = \sum_k p(O|\theta_k)\hat{\theta}_k, \tag{3.202}$$

which is equivalent to a mixture of PCA [TB99].

In practice, this complexity proved unnecessary.

The other possibility, which is faster, is to calculate θ_h considering that $\theta_l = 0$. Then, given the region, we can calculate θ_l:

$$\hat{\theta}_h = \arg\max_{\theta_h} p(O|\theta_h, \theta_l = 0) \tag{3.203}$$

$$\hat{\theta}_l = \arg\max p(O|\theta_l, \hat{\theta}_h). \tag{3.204}$$

This is suboptimal but breaks the complexity into two smaller MLED problems of (eq. 3.73). We have used this formula in all experiments. A cursory study proved that it was equivalent to the optimal solution.

Re-estimation of eigenspace parameters

Once eigenvalues and their corresponding associated eigenspaces are determined, we reestimate the eigenspace in the same way we would optimize the linear eigenspace using Section 3.4.2).

Boundary decision vectors: The Perceptron

We can also optimize the boundary decision vectors. The perceptron algorithm [DH73] can be used to update the boundary decision vectors v_k. Suppose we want to find boundary decision vector for an arbitrary dichotomy of the set. For instance, in the Wall Street Journal dictation task, the training set comprises data from two databases WSJ0 (or SI84), and WSJ1 (SI200), recorded in two different occasions. Assume that we would like to separate the database component explicitly. It does not appear to be associated to a particular eigenvector. However, we premise that the impact on recognition will be large. To train a sub-eigenspace per database, consider the following problem. We would like to obtain:

$$v^T\theta \geq 0 \quad \text{if speaker is in WSJ0, and} \tag{3.205}$$

$$v^T\theta < 0 \quad \text{if speaker is in WSJ1.} \tag{3.206}$$

If we switch the sign of all θ corresponding to WSJ1 data, we are left with the problem of solving the inequalities with respect to v:

$$v^T\theta > 0, \quad \forall\theta. \tag{3.207}$$

If the system has a solution, it is called *linearly separable*. Among all 2^B possible dichotomies, there are only:

$$2\sum_{e=0}^{E}\binom{B-1}{e}; \quad B > E, \tag{3.208}$$

which are linearly separable. For $E = 20$ and $B = 284$, this amounts to about 17% of all possible dichotomies. The system of inequalities is solved by defining first the optimization criterion:

$$J(v) := \sum_{\theta\in\Omega} v^T\theta; \quad \Omega := \text{all misclassified}. \tag{3.209}$$

By descending the gradient we obtain the notorious *perceptron algorithm* [DH73], which at each iteration j computes the set of all misclassified θ as Ω_j and update the boundary decision vector v_j using the learning rule:

$$v_{j+1} \leftarrow v_j + \sum_{\theta\in\Omega_j}\theta. \tag{3.210}$$

If v is a solution, then we will converge in at most K steps,

$$K = \frac{\max_e ||\theta_e||^2||v||^2}{(\min_e \theta_e^T v)^2} < \infty. \tag{3.211}$$

There are many extensions to this algorithm, in particular in the case of non-separability.

The perceptron approach is very effective when we would like to specify some prior knowledge manually. It is also useful when we need to update boundary decision vectors when the eigenspaces are re-estimated. Positive signs are enforced when the boundary decision vector maps θ to the eigenspace with highest likelihood.

3.10 Experiments

We carried out two sets of experiments: the first on TIMIT (see Section 8.5), and the other on WSJ (see Section 8.4). PSTL's existing TIMIT system was used during early developments, which led to MLES (Section 3.4.2). The remaining experiments focused on verifying the effectiveness of the algorithms developed in Sections 3.7, 3.8, and 3.9.

3.10.1 MLES results

We verify the effectiveness of MLES. This is done by comparing the PCA-derived eigenspace with MLES an MLES optimized version.

Experimental conditions

The experiments were conducted on the TIMIT database (Section 8.5), using the standard train/test partition. Speech was parameterized using PLP cepstral features without cepstral filtering. There are 9 static coefficients (including energy of the residual) and 9 delta, totalling 18 features. We use 48 context-independent HMM models, with 3 emitting states and 16 Gaussians per mixture, resulting in 2240 distributions. Adaptation is supervised. We report results in phoneme accuracy over the 39 phonemic set. The decoder is a phoneme decoder which applies bigram probabilities.

Method	$E = 5$	$E = 10$	$E = 20$	$E = 50$
LSES	60.67	60.58	61.29	61.56
MLES($E = 10$)	62.53	65.10	-	-
MLES($E = 20$)	63.06	65.01	65.37	-
MLES($E = 50$)	61.74	63.77	64.84	66.96

Table 3.2: Maximum-likelihood eigenspace: unit accuracy for different configurations

MLES vs LSES

Table 3.2 evidences the performance of the maximum-likelihood criterion vs least-squares. The least-squares (LSES) was derived with PCA. MLES was applied for different values of E (first column) and tested the eigenspaces with other values of E (first row). LSES served as the seed eigenspace for MLES. Due to memory limitations, LSES was estimated on a set of only 100 speakers, but balanced with respect to sex. MLES used all 462 speakers. MLES performs best when the number of dimensions is the same during training and decoding. In this database, increasing the number of dimension always proved beneficial. This means that we have to know in advance how many dimensions we want to use in our system when building prior information.

3.10.2 Root modulation and other eigenspaces

For our experiments we chose the Wall Street Journal (WSJ1) Nov92 evaluation test, described in Section 8.4. The baseline system achieves 10.8% Word Error Rate (WER). The systems runs at about 1.1 times real-time each pass, with a search effort of about 9k states (on a Pentium IV at 1.5 GHz).

We used an eigenspace of dimension $E = 40$. This was known to provide reasonably good results with relatively moderate resources. There was one full MLLR regression matrix for each of the following classes: silence, vowels, and consonants.

We operated in self-adaptation mode: a first pass produces the most likely hypothesis. The most likely hypothesis is used for adaptation. The second pass exploits adapted models. Five iterations of within-word Viterbi alignments are performed between passes. To comply with rules dictated by the Nov92 test, adaptation across sentences is forbidden. Each sentence is processed independently.

Table 3.3 summarizes the results for MLLR only (MLLR), eigenspace-constrained MLLR (MLED-MLLR), eigenvoice estimated on MLLR models with MAP smoothing (MLED-MAP/MLLR). Results are significant at an interval of 0.3% WER with 95% confidence. Also, we report the piecewise linear extension applied on MLED-MLLR models in the inverse space, Root space and MMI space results.

	WER
SI	10.8%
MLLR	10.5%
MLED - MLLR	9.8%
MLED - MAP/MLLR	9.6%
Piecewise-linear	9.6%
Root space	9.5%
MMI space	9.1%

Table 3.3: Results for the different models

The largest gain in performance was obtained through MMIE space modeling.

3.11 Summary

In this chapter, we have extended the framework of model-space constraints. The formalism of least-squares criteria applied to model parameters lead to significant formalization of Eigenvoices. This lead to novel approaches. We touched upon the following topics:

- The Gaussianisms interpretation of PCA, and MLLR in the HMM framework: We have established a link between ML and least squares. We have also shown that the posterior probability of MLLR rows is Gaussian. A light-weight ML re-estimation procedure is given for the eigenspace.
- The root space normalization: it is possible to normalize MLLR row estimates so that the squared error criterion corresponds exactly to the likelihood. PCA becomes optimal under the ML criterion. Update formulæ for the eigenspace and the speaker location become simple projections.
- Discriminative method to improve on the estimation of the eigenspace: we show that Gopalakrishnan's gradient update can be written as a MAP minimization of a quadratic function. We state the MMI formulation of Eigenvoices.
- Non-linear extensions via piecewise-linear approximations: we extend the linear regression framework of PCA to a piece-wise linear approximation.

The central element is the link between the trace of the autocorrelation of mean supervectors and the ML criterion. Figure 3.8 shows how components are connected together. Divergence, MAP, ML, projection, Hilbert spaces and least-squares have all been unified in the posterior HMM/MLLR parameter framework.

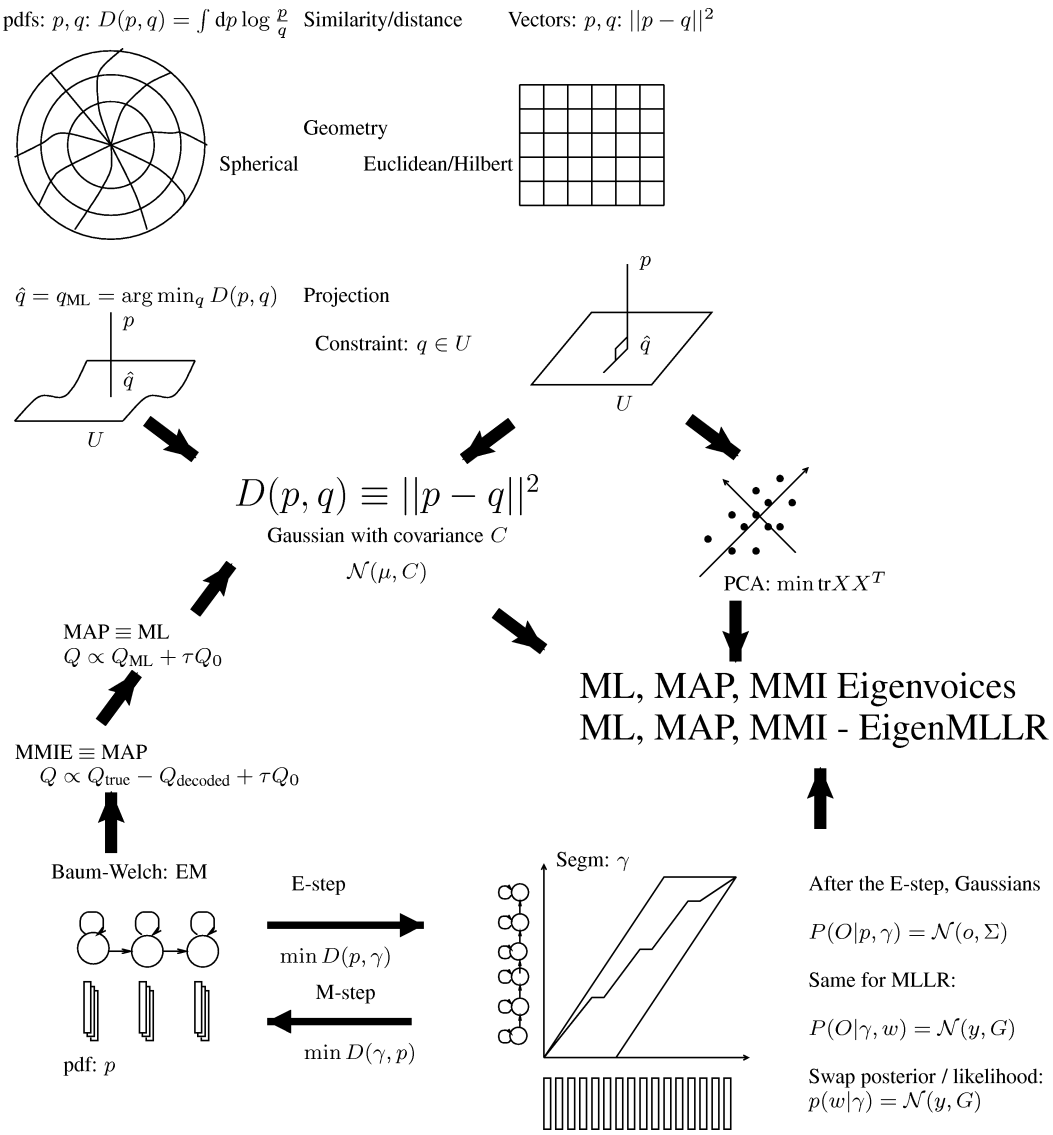


Figure 3.8: Divergence, Gaussianisms and HMMs

Chapter 4

Feature-space transformation

4.1 Introduction

In the case of a mismatch in environmental conditions, it is natural to seek a transformation of the observations to compensate for the difference between training and test conditions.

Linear feature-space transformation is befitted to that task. As we will see, however, the multi-dimensional feature transformation does not admit a closed-form solution in the general case. The literature has so far focused on either diagonal matrices, unitary triangular matrices, and numerical analysis for full matrices. We first review the problem, and the state-of-the-art, and finally our solution.

4.2 Affine transformation of observations

Let X be a random variable with pdf $p_X(\cdot)$. X is a vector of length D. The model $p_X(\cdot)$ will be given by the HMM model. We can only observe another random variable Y, which is modeled as an affine transformation of X:

$$Y = MX + c, \tag{4.1}$$

where M is a $D \times D$ square transformation matrix, and c is called the *bias* vector. The Jacobian J of the transformation from X to Y is known to be:

$$J = \det\left[\frac{\partial X^T}{\partial Y}\right] = |M|^{-1}. \tag{4.2}$$

The bias c does not appear in the expression (eq. 4.2). As we will see later, the Jacobian is the primary cause of analytical difficulties. It is well-known that the pdf is given by:

$$p_Y(y) = |M|^{-1} p_X\left(M^{-1}[y-c]\right). \tag{4.3}$$

For Baum-Welch HMM training, each p_X is assumed Gaussian with mean μ_X and variance C_X, and therefore:

$$\begin{aligned}\log p_Y(y) = -\frac{1}{2}\Big[&D\log 2\pi + \log|C_X| + \log|M|^2 + \\ &+ (\mu_X - M^{-1}y + M^{-1}c)^T C_X^{-1}(\mu_X - M^{-1}y + M^{-1}c)\Big].\end{aligned} \tag{4.4}$$

It is more convenient to define:

$$A = M^{-1}, \tag{4.5}$$

$$b = M^{-1}c. \tag{4.6}$$

The transformed pdf is:

$$\log p_Y(y) = -\frac{1}{2}\Big[D\log 2\pi + \log|C_X| - \log|A|^2 + (\mu_X - (Ay-b))^T C_X(\mu_X - (Ay-b))\Big]. \tag{4.7}$$

Its mean and covariance are:

$$\mu_Y = A^{-1}(\mu_X + b), \tag{4.8}$$

$$C_Y = A^{-1}C_X A^{-T}. \tag{4.9}$$

As aptly noted by Gales [Gal97b], we can transform either means and variances of the models, or observations features.

4.3 Constrained adaptation

These considerations are valid for single Gaussian only. We transpose them into the framework of EM. As we shall see, solutions become rapidly intractable. There are no closed-form solutions available for the general case. We will have to place restrictions on the transformation A. Since the bias does not affect analytical difficulties, we will omit it for simplicity.

The auxiliary function for HMM with Gaussian mixtures is:

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\Big\{\log|R_m| + (\mu - o_t)^T R_m(\mu - o_t)\Big\}. \tag{4.10}$$

Substituting results from (eq.4.7), we obtain:

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\Big\{\log|A|^2 + (\mu - Ao_t)^T R_m(\mu - Ao_t)\Big\}. \tag{4.11}$$

We wish to maximize Q with respect to A. It is not straightforward. First, we will apply the standard gradient search. Then, we examine simplified formulæ found in the literature.

The most common method for solving optimization problems is to follow for the gradient. It is well-known that stationary points correspond to maxima or minima of the objective function.

We differentiate Q w.r.t. A, and solve for$A$:

$$\frac{\partial Q}{\partial A} = 0. \tag{4.12}$$

With the expression of (eq. 4.11), we have:

$$\frac{\partial Q}{\partial A} = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\left\{2A^{-T} + \frac{\partial}{\partial A}\mathrm{tr}\left[o_t o_t^T A^T R_m A\right] - 2\frac{\partial}{\partial A}(R_m\mu)^T A o_t\right\}. \tag{4.13}$$

Using the lemmas (Section 2.2),

$$\frac{\partial Q}{\partial A} = -\sum_{t,m}\gamma_m(t)\left\{-A^{-T} + R_m A o_t o_t^T - R_m\mu o_t^T\right\}. \tag{4.14}$$

We note that A^{-1} and A are present simultaneously. The equation (eq.4.14) is therefore a quadratic equation. For a discussion of multivariate quadratic equations, we refer the reader to [PTVF92]. Since the problem is not linear, there may be more than one solution, and the gradient might indicate a minimum or a maximum.

We will follow three approaches: gradient descent, diagonal matrices, and present our LU-decomposition scheme.

The solution is not unique. In the literature, we find roughly five approaches:

1. Row-by-row optimization [Gal97b]: Gales finds an iterative method for canceling the gradient. He supposes that rows are independent of each other, then solves row by row. The process is iterated until convergence.

2. Conjugate gradient descent [KA98]: Kumar finds out that the equation is quasi-quadratic. He uses the conjugate gradient descent method, but observes instabilities.

3. Diagonal transforms [DRN95]: Digikalis finds a closed-form solution for the diagonal case.

4. Unit triangular transforms [Bil00]: this is a subcase of MLLR. All diagonal elements are set to one. The matrix is constrained to be triangular.

5. MLLT's diagonalizing transform [Gop98]: Gopinath diagonalizes the matrix before diagonal scaling.

4.3.1 Gradient search

Gradient search is the most popular hill-climbing method. It is suited for most problems where objective function is derivable. Models are updated by small ε-corrections in the direction of the gradient:

$$A^{(k+1)} = A^{(k)} + \varepsilon \nabla_A Q = A^{(k)} - \varepsilon \left[\sum_{t,m} \gamma_m(t) \left\{ A^{-1} + R_m A o_t o_t^T - R_m \mu o_t^T \right\} \right], \tag{4.15}$$

using an arbitrarily small step ε.

Since the true solution is:

$$\nabla_A Q = 0 \iff ||\nabla_A Q||^2 = 0 \iff \text{tr} \left([\nabla_A Q]^T \nabla_A Q \right) = 0. \tag{4.16}$$

This is interesting because we have a scalar again, and gradient descent still applies, in other words,

$$\phi = \text{tr} \left([\nabla_A Q]^T \nabla_A Q \right). \tag{4.17}$$

We want to cancel $\phi(A)$, and its gradient is, after a lengthy but straightforward manipulation:

$$\begin{aligned} \frac{1}{2} \frac{\partial \phi}{\partial A} = {} & JC^T + JGA^T R + RA^T JG + ACC^T + \\ & R^{(m)T} A^T A R^{(p)} A G^{(p)} G^{(m)T} + A R^{(p)} A G^{(p)} G^{(m)T} A^T R^{(m)T}, \end{aligned} \tag{4.18}$$

with respect to the following definitions:

$$JC^T = \left[\sum_{t,m} \gamma_m(t)\right] \sum_{t,m} \gamma_m(t) o_t \mu^T R, \tag{4.19}$$

$$JGA^T R = \left[\sum_{t,m} \gamma_m(t)\right] \sum_{t,m} \gamma_m(t) o_t o_t^T A^T R, \tag{4.20}$$

$$ARAG\,GA^T R = A \left[\sum_m R_m A \sum_t \gamma_m(t) o_t o_t^T\right] \left[\sum_m R_m A \sum_t \gamma_m(t) o_t o_t^T\right] A^T, \tag{4.21}$$

$$RA^T ARAGG = \sum_m R_m A^T A \sum_p R_p A \left(\sum_t \gamma_p(t) o_t o_t^T\right) \left(\sum_t \gamma_m(t) o_t o_t^T\right), \tag{4.22}$$

$$RA^T JG = \left[\sum_{t,m} \gamma_m(t)\right] \sum_{t,m} \gamma_m(t) RA^T o_t o_t^T, \tag{4.23}$$

$$ACC^T = A \left(\sum_{t,m} \gamma_m(t) R_m \mu o_t^T\right) \left(\sum_{t,m} \gamma_m(t) o_t \mu^T R_m\right). \tag{4.24}$$

We have used Einstein's notation and lemmas of Section 2.2 extensively.

Note that since the computation and differentiation of A^{-1} is difficult, we left-multiplied with A. The direct computation of all terms of the gradient is too expensive. This avenue will not be pursued further.

4.3.2 Diagonal matrix

When the matrix A is diagonal, then there are two solutions per dimension. Let a_{dd} be the d^{th} diagonal element of A:

$$A = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{DD} \end{bmatrix}. \tag{4.25}$$

Let o_d be the d^{th} element of o_t. The expression for the gradient is quadratic and may be found in [Gal97b, DRN95]. Simplifying (eq. 4.14), we have for each dimension d:

$$\frac{\partial Q}{\partial a_{dd}} = -\frac{1}{2} \sum_{t,m} \gamma_m(t) \left\{-\frac{1}{a_{dd}} + r_d a_{dd} o_d(d)^2 - r_d \mu_d o_d(d)^T\right\}. \tag{4.26}$$

Since the equation is a scalar quadratic expression, it has (at most) two solutions. We choose the solution that is closer to unity:

$$a_{dd} = \frac{1}{2}\left(\sqrt{\beta^2 + 4\eta} + \beta\right) \tag{4.27}$$

with the appropriate definitions of β and η:

$$\alpha = \sum_{t,m} \gamma_m(t) r_d o_d^2, \tag{4.28}$$

$$\beta = \alpha^{-1} \sum_{t,m} \gamma_m(t) r_d o_d \mu_d, \tag{4.29}$$

$$\eta = \alpha^{-1} \sum_{t,m} \gamma_m(t). \tag{4.30}$$

The second derivative in this case indicates which of the two solutions corresponds to a stable point by indicating more negative values. We differentiate (eq. 4.26):

$$\frac{\partial^2 Q}{(\partial a_{dd})^2} = -\sum_{t,m} \gamma_m(t) \left\{ \frac{1}{a^2} + r_d o_d^2 \right\} < 0. \tag{4.31}$$

Both roots of the characteristic equation correspond to maxima in the likelihood. However, our choice guarantees a smaller absolute value of the second derivative, and also a value closer to unity. Since there are two possible solution for every dimension $d = 1..D$, in general the Q-function has 2^D local maxima. We can stress the importance of a closed-form solution. The gradient descent algorithm will converge blindly to any of these solutions. By leveraging insight gained from the analytical study, we are able to select exactly which solution is of interest to us.

4.3.3 Upper-triangular matrix

We will now solve the case of an upper triangular matrix A.

Since all rows of the matrix are independent, thanks to the diagonality of covariances, we may set a dimension d and solve each dimension independently. Let $a_k, k = d, d+1, ..., D$ be the non-zero elements of the d^{th} row of A. Define:

$$a^* = [a_{d+1}, a_{d+2}, ..., a_D, b]^T, \tag{4.32}$$

$$o^* = [o_{d+1}, o_{d+2}, ..., o_D, 1]^T. \tag{4.33}$$

We seek to find $[a_d, a^*]$. Since the determinant only depends on a_d, it is treated differently. First, we solve a $(D-d) \times (D-d)$ linear subsystem for a^* using the $D-d$ last elements of the gradient. Then, we use the special equation for a_d to yield the usual quadratic form.

The objective function for the dimension d is:

$$Q = -\frac{1}{2} \left\{ -\log |a_d|^2 + (a^{*T} o^* + a_d o_d - \mu_d)^2 r_d \right\} \tag{4.34}$$

Differentiating with respect to a_k, $k = d+1, ..., D$ and b, we get a linear system:

$$\frac{\partial Q}{\partial a^*} = -\sum_{t,m} \gamma_m(t) r_d \left[\mu_d - a^{*T} o^* - a_d o_d \right] o^*. \tag{4.35}$$

It is solved by:

$$a^* = M^{-1} \left(a_d y + z \right), \tag{4.36}$$

with the following:

$$M = \sum_{t,m} \gamma_m(t) r_d o^* o^{*T} > 0, \tag{4.37}$$

$$y = \sum_{t,m} \gamma_m(t) r_d o_d \, o^*, \tag{4.38}$$

$$z = \sum_{t,m} \gamma_m(t) r_d \mu_d \, o^*. \tag{4.39}$$

Now we need to find a_d and substitute back.

The solution for a_d is found using the last derivative, which is merely a generalization of the diagonal case:

$$\frac{\partial Q}{\partial a_d} = \sum_{t,m} \gamma_m(t) r_d \left[a_d^{-1} + \quad (\mu_d - a^{*T} o^* - a_d o_d)(o_d + y^T M^{-1} o^*) \right]. \tag{4.40}$$

We can use the linear dependency specified by (eq. 4.36), and finally state that a_d is again the solution of a quadratic expression:

$$a_d = \frac{1}{2}\left(\beta + \sqrt{\beta^2 + 4\eta}\right), \tag{4.41}$$

with:

$$\alpha = \sum_{t,m} \gamma_m(t) r_d o_d^2 - y^T M^{-1} y > 0, \tag{4.42}$$

$$\beta = \alpha^{-1}\left[\sum_{t,m} \gamma_m(t) r_d o_d \mu_d - y^T M^{-1} z\right], \tag{4.43}$$

$$\eta = \alpha^{-1} \sum_{t,m} \gamma_m(t) > 0. \tag{4.44}$$

There is always a solution.

When covariances are not diagonal, we must first solve the quadratic equation for a_{DD}. Then, knowledge of this coefficient will help find $a_{D-1,D-1}$ and $a_{D-1,D}$. We proceed thus up to the top row, in the same fashion as the back substitution step in a Gauss-Jordan matrix inversion.

4.3.4 The LU decomposition

Looking at (eq. 4.14), we see that the crux of the problem resides in the presence of a log determinant, which implies in turn the presence of the inverse matrix. A common way of dealing with inverse matrices involves the LU decomposition of a matrix, that is to say, our matrix A is written as:

$$A = LU \tag{4.45}$$

with U an upper-triangular matrix, and L a unitary, lower-triangular matrix. The diagonal elements of L are all equal to 1.

We embed this decomposition by alternating the maximization step in the EM algorithm:

$$o' = Ao = L(Uo). \tag{4.46}$$

The upper-triangular method was derived above, and the lower-triangular method is found by just setting $a_{dd} = 1$ in (eq. 4.36), as in [Bil00].

4.3.5 Bayesian extension

The Bayesian framework is useful for parameter smoothing. For instance, while using regression trees to define multiple classes, the leaf transforms are derived by smoothing with the parent nodes.

The MAP framework is usually greatly simplified by selecting the prior distribution $p_0(A)$ among the family of conjugate priors for A (see [AS96, GL94]).

Definition 1 (Conjugate prior) *Let X be a random variable. Let the distribution of X be $p(X|\lambda)$. The pdf $p(X|\lambda)$ is the likelihood of X. It has parameters λ. Let the distribution of λ, called* prior *distribution of λ, be noted $p_0(\lambda)$. The posterior distribution is $p(\lambda|X = x)$. If $p_0(\lambda)$ is such that the posterior belongs to the same family as the likelihood, i.d.:*

$$p(\lambda|X) \propto p(X|\lambda), \tag{4.47}$$

then it is called the conjugate prior *for $p(X|\lambda)$.*

The conjugate prior for a Gaussian distribution with fixed covariance is a Gaussian distribution.

MAP estimators and prior distributions were defined for all but the diagonal term. The conjugate prior for the bias is a Normal law. The conjugate prior for non-diagonal elements is elliptic [Cho99]. The probability of diagonal terms has a transcendent shape. We will start from (eq. 4.11) for the case of diagonal components a_d:

$$Q = -\frac{1}{2}\sum_{t,m}\gamma_m(t)\left[-\log|a_d|^2 + r_d(\mu_d - a_d o_t)^2\right]. \tag{4.48}$$

We need to find an $R(a_d) = \log p_0(a_d)$ such that $\exists \alpha, r_0, y \in \mathbb{R}$:

$$S(a_d) := Q(a_d) + R(a_d) = \beta - \frac{1}{2}\left[-\alpha\log(a_d)^2 + r_0(y - a_d)^2\right], \tag{4.49}$$

where β is a constant. The conjugate prior family does not appear frequently enough in nature to justify a name. We proceed to define it.

The Maxwell-Rayleigh-Normal distribution

A subset of the family of conjugate priors $p_0(a_d)$ is a mixture of (extended) Maxwell, Rayleigh, and Gaussian distribution. We christen it hence the *Maxwell-Rayleigh-Normal* (MRN) distribution.

Maxwell's distribution models speeds of molecules in thermal equilibrium. It is defined for $x \geq 0, a > 0$:

$$p_M(x|a) = \sqrt{\frac{2}{\pi}}a^{3/2}x^2e^{-ax^2/2}. \tag{4.50}$$

Furthermore, the Rayleigh distribution models the attenuation in fading channels:

$$p_R(x|s) = \frac{x}{2s^2}e^{-\frac{1}{2}x^2/s^2}. \tag{4.51}$$

Lastly, the Normal distribution is an old acquaintance of ours:

$$p_G(x|s) = \frac{1}{\sqrt{2\pi}s}e^{-\frac{1}{2}x^2/s^2}. \tag{4.52}$$

We define the MRN distribution to be:

$$p_{MRN}(x|\nu) = \Phi^{-1}x^2e^{-(x-\nu)^2r^2}, \tag{4.53}$$

where r is the precision element. For simplicity, we set it to $r = 1$. The regularization constraint Φ is chosen such that:

$$\int \mathrm{d}p_{MRN} = 1. \tag{4.54}$$

The distribution is shown on Figure 4.1.

The value of the hyper-parameter ν with respect to the mean is shown on Figure 4.2.

Figure 4.2 shows how the mean of the distribution varies according to the hyper parameter ν.

We proceed by defining the *raised MRN* law constitutes a family of conjugate priors:

$$p_{RMRN}(x|\nu,\tau) = \Phi_R^{-1}(\nu,\tau)x^{2\tau}e^{-\tau(x-\nu)^2}. \tag{4.55}$$

Unfortunately, unless τ is a multiple of $\frac{1}{2}$, moments have no closed-form expression. Nevertheless, in most cases, we are only interested in values of τ, ν such that:

$$\int \mathrm{d}x\left[xp_{RMRN}(x)\right] = 1. \tag{4.56}$$

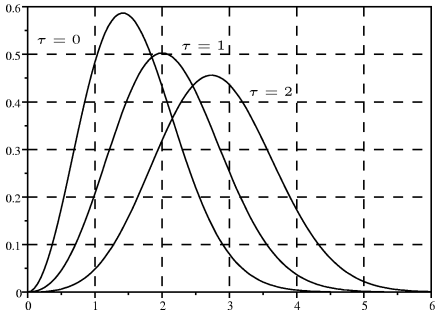


Figure 4.1: The MRN law for different values of ν

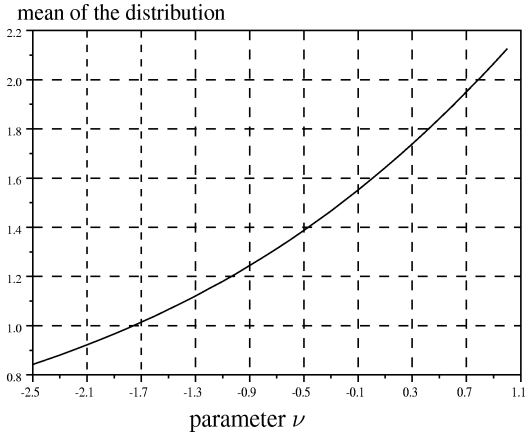


Figure 4.2: The mean of MRN w.r.t. ν. The parameter that corresponds to identity is $\nu = -1.8$.

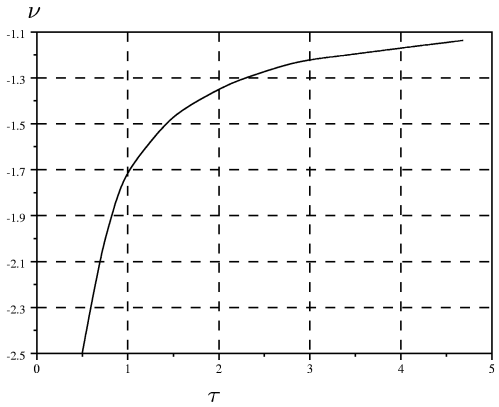


Figure 4.3: We select τ, and choose ν such that the mean is one: this gives more weight to the identity matrix.

That is, the expected value is the identity: since the $p_{RMRN}(\cdot)$ will model the diagonal element of the transformation matrix, in the absence of additional knowledge, a good guess is 1, which corresponds to an identity transformation matrix. It is easier to use numerical integration and tabulate $\nu(\tau)$. On Figure 4.3, we show how to select ν for a given prior weight τ such that the mean of the distribution is again 1.

MAP re-estimation formula

We now verify that the function is a conjugate prior for a_d and find the MAP solution. We can replace (eq. 4.55) into the left-hand side of (eq. 4.49):

$$\begin{aligned} S(a_d) &= Q(a_d) + R(a_d) && (4.57)\\ &= Q(a_d) + \log p_{RMRN}(a_d) && (4.58)\\ &= Q(a_d) - \log\Phi + \tau \log|a_d|^2 - \tau(a_d - \nu)^2, && (4.59) \end{aligned}$$

which from (eq. 4.48) is also:

$$= \beta - \frac{1}{2}\sum_{t,m}\gamma_m(t)\left[-\log|a_d|^2 + r_d(\mu_d - a_d o_t)^2\right] + \tau\log|a_d|^2 - \tau(a_d-\nu)^2, \quad (4.60)$$

with a constant β. The same kind of calculus in Section 3.5.1 yields:

$$S(a_d) = \beta' + \frac{1}{2}\left[2\tau + \sum_{t,m}\gamma_m(t)\right]\log(a_d)^2 - \left[\tau + \frac{1}{2}\sum_{t,m}\gamma_m(t) r_d o_d^2\right](a_d - y)^2, \quad (4.61)$$

with:

$$y = \left[\tau + \frac{1}{2}\sum_{t,m}\gamma_m(t) r_d o_d^2\right]^{-1}\left[\tau\nu + \frac{1}{2}\sum_{t,m}\gamma_m(t)\right]. \quad (4.62)$$

and β' another constant. Matching with the right-hand side of (eq. 4.49), we see that the condition holds for scalars α, r_0 properly defined as:

$$\alpha = 2\tau + \sum_{t,m}\gamma_m(t), \quad (4.63)$$

$$r_0 = \tau + \frac{1}{2}\sum_{t,m}\gamma_m(t) r_d o_d^2. \quad (4.64)$$

Hence, we can assert that the p_{RMRN} is a conjugate prior for a_d:

$$p(O|a_d)p_{RMRN}(a_d) = p(a_d|O) \propto p(O|a_d). \quad (4.65)$$

And therefore from the same token as Section 3.8.3, the MAP estimator can be written as an ML problem. In particular:

$$\frac{\partial S(a_d)}{\partial a_d} = 0 \quad (4.66)$$

implies:

$$a_d^2 - \frac{y}{r_0}a_d + \frac{\alpha}{r_0} = 0, \quad (4.67)$$

which is a quadratic equation similar to the ML case of (eq. 4.26). For the same reasons as in ML, we select the positive root, and uniquely define the MAP estimator for a_d:

$$a_d = \frac{y + \sqrt{y^2 - 4y\alpha}}{2r_0}. \quad (4.68)$$

Asymptotically, when there is an infinity of data, or $\frac{\mathcal{T}}{\tau} \to \infty$, then it is the ML. When there are no data, or $\frac{\mathcal{T}}{\tau} \to 0$, then:

$$a_d = \int \mathrm{d}x \left[x p_{RMRN}(x) \right] = 1, \tag{4.69}$$

or any appropriate value selected on the right-hand side of (eq. 4.56). Because of the presence of the log-determinant, a_d does *not* converge to ν, but to a smaller value.

4.4 Bayesian transformation of the variances

This section is not directly related to feature adaptation. To the best of our knowledge, it is not described in the MAPLR literature. We show that transformation parameters of the variances have very simple conjugate priors.

Fortunately, Bayesian transformation of parameters is almost the same as Bayesian adaptation of parameters. The conjugate prior is a Wishart distribution. We include it here for completeness.

For a transformation of the variances defined as:

$$C \leftarrow A^T C A, \tag{4.70}$$

the objective function becomes:

$$\log p(O|A) + \log p(A) = \beta - \frac{1}{2} \sum_{t,m} \gamma_m(t) \left[\log |A| - (\mu - o_t)^T R^{\frac{T}{2}} A R^{\frac{1}{2}} (\mu - o_t) \right] + \tau \log |A| - \tau \mathrm{tr}\, TA. \tag{4.71}$$

β is a constant term corresponding to regularization. The hyper parameter of the distribution is the prior matrix T. Without the regularization constants, we can distinguish the simpler equation:

$$\log p_0(A) = \text{constant} + \tau \left(\log |A| - \mathrm{tr}\, TA \right). \tag{4.72}$$

We recognize the Wishart distribution:

$$p(W|G) = c(n, \alpha_W) |T|^{\frac{\alpha_W}{2}} |W|^{\frac{\alpha_W - n - 1}{2}} e^{-\frac{1}{2} \mathrm{tr}\, TW} \tag{4.73}$$

with regularization constant:

$$c(n, \alpha_W) = \left[2^{\frac{\alpha_W n}{2}} \pi^{\frac{n(n-1)}{4}} \prod_i \Gamma \left(\frac{\alpha_W + 1 - i}{2} \right) \right]^{-1} \tag{4.74}$$

The update is very similar to the unconstrained covariance adaptation:

$$A^{-1} = \left[\sum_{t,m} \gamma_m(t) + \tau \right]^{-1} \left\{ \tau T + \sum_{t,m} \gamma_m(t) R^{\frac{T}{2}} (\mu - o_t)(\mu - o_t)^T R^{\frac{1}{2}} \right\}. \tag{4.75}$$

We may decompose T into $R_m^{\frac{T}{2}} T_0 R_m^{\frac{T}{2}}$ and T_0 is now a (full) global transformation matrix.

4.5 Experiments

4.5.1 Conditions

To validate our algorithm, we used the Switchboard conversational telephone speech database (Section 8.2). We report results on the first evaluation test set of NIST's 2001 evaluation [MP01], which contains 20 conversations from the Switchboard-I database.

4.5.2 Results with MLLU

In Table 4.1, we report Word Error Rates (WER). The feature space transformation, or MLLU for (Maximum-Likelihood LU transformation), yields an improvement comparable with MLLR when used in isolation. Since there were about 5 minutes of adaptation data in most cases, we have enough data. We disabled the MAP prior described in Section 4.3.5.

There is a .2% WER improvement if we only use block-diagonal matrices. We have observed that MLLR behaves best with 7 regression classes (1 for silence, 4 for vowels, and 2 for consonants). In this case as well, constraining the transformation matrices to be block-diagonal, we get an improvement.

When we use MLLU as a feature normalization, followed by MLLR model adaptation, we obtain a 1.6% WER improvement over the baseline MLLR adapted models.

	WER
SI	34.6%
MLLR 1 global class	32.8%
MLLU 1 global class	32.8%
MLLU block-diag	32.6%
MLLR 7 classes + block	32.2%
MLLU + MLLR(7)	30.6%

Table 4.1: Results

Note that MLLU depends on the order of the coefficients. Best results were obtained with the default order: the energy is first, then standard cepstral coefficients. Since we use block-diagonal matrices, derivatives are not dependent on the static coefficients and vice-versa.

4.6 Summary

In this chapter, we have explored issues related with linear feature transformation. In particular, we have:

- outlined mathematical difficulties due to the determinant,
- illustrated with a gradient descent study,
- then reviewed the diagonal case published by Digilakis [DRN95],
- generalized it to an upper triangular matrix,
- alternated upper and lower triangular transformations in the EM algorithm,
- and presented a Bayesian adaptation framework.

Additionally, we presented formulæ for Bayesian transformation of variances.

Part II

Speaker adaptation: applications

Introduction

Interesting how they may appear, the methods set forth in the previous chapters form only but the premises required for successful speaker adaptation. In contrast with other components in ASR, speaker adaptation is embodied in a running system, which is deployed in certain ways. The specifics of the requirements dictate much of how speaker adaptation will be performed. Such driving proponents are laid out by other considerations which in turn may be rooted in as far as economical constraints.

The study of speaker adaptation is henceforth incomplete without a closer investigation into these matters. There is an infinity of possible applications in which speaker adaptation may be deployed. We hope to encompass a large part of these in the following. There are recurrent characteristics that are encountered in many ASR systems. Some of those may be solved by algorithms developed herein. In other words, this part is dedicated to system developers, whose goal is to integrate existing Speaker Adaptation mathematical algorithms in a way that benefits ASR users' concerns.

We have attacked three specific problems:

1. Unsupervised and supervised adaptation using EM: we study differences between MLLR and MAP, and which one should be used for what reason. We also improve the efficiency of supervised adaptation using a sausage NBest discrimination.

2. Clustering and outliers: for the case of self-adaptation, we employ a clustering technique to remove outliers due to imperfect recognition.

3. Interaction with noise: heavy reliance on prior knowledge learned on training databases, especially in Eigenvoices, underlines the problem of mismatch in training and test conditions. We separate clearly between noise, and speaker adaptation, based on the specifics of the algorithm.

Chapter 5

EM-based approaches to adaptation

Oftentimes, decoders operate in a setting which prevents them from using a lengthy enrolment process. Speaker adaptation is performed on the fly. It is not possible to use labels (transcribed speech) because they are not available. Mathematically, those labels may be guessed via the E-step, which computes the expected segmentation probabilities. The EM algorithm is applied once again. However, there are decision to be made as to how the adaptation process is to be applied into details.

We shall take a closer examination at these matters. Characteristics of adaptation algorithms must be taken into account. Under that light, we study the N-Best algorithm for unsupervised adaptation. Then, we apply concepts to discriminative adaptation. The description of the decoder includes an explanation about how to create sentence-level N-best hypotheses. For adaptation, it is sufficient to use "sausage" NBest lists and examine, in an isolated fashion, alternatives to the first-best given the first-best segmentation.

5.1 Unsupervised adaptation using NBest decoding and Ratio weighting

Integrating the transcriptions in the EM algorithm is mathematically straightforward. The implementation causes much difficulties. We should maximize the expectation of the supervised likelihood over all possible word sequences $\mathcal{W}$ in the grammar:

$$\hat{\lambda} = \arg\max_{\lambda} \mathrm{E}_{\mathcal{W},\vartheta} p(O, \vartheta, \mathcal{W}|\lambda). \tag{5.1}$$

The amount of possible $\mathcal{W}$ is finite but prohibitively large. Therefore we consider the direct computation to be intractable, and it is customary to apply EM hierarchically, once for the word sequence, and thither for the speaker parameters. While this is still a valid EM formulation, it may converge to a lesser optimum than the direct joint estimation.

We decide to apply further approximations. At this point, we should note that recent work by Padmanabhan, Saon and Zweig [PSZ00] integrates directly on a lattice produced by a previous pass. This is theoretically better than the sausage NBest integration scheme. Nonetheless, considerations developed herein are independent of the objects being integrated.

5.2 Three approximations for the calculation of the Expectation with respect to the word sequence

Computing the expectation of (eq. 5.1) directly requires the summation over all possible word sequences. Instead, we may use an approximation. We use three different formulæ . The simplest algorithm consists of using decoded labels as true labels, and we call it the one-best unsupervised adaptation. It is also known as transcription-mode adaptation:

$$\hat{\lambda} = \arg\max_{\lambda} \max_{\mathcal{W},\vartheta} p(O, \vartheta, \mathcal{W}|\lambda). \tag{5.2}$$

Another possibility is to maximize the likelihood of all NBest strings as follows:

$$\hat{\lambda} = \arg\max_{\lambda} \sum_{\mathcal{W}} \max_{\vartheta} p(O, \vartheta, \mathcal{W}|\lambda). \tag{5.3}$$

The expectation, instead of a sum, may be employed for the state sequence. The closest form to the true formula is:

$$\hat{\lambda} = \arg\max_{\lambda} \sum_{\mathcal{W}} \varphi_{\mathcal{W}} \max_{\vartheta} p(O, \vartheta|\mathcal{W}, \lambda). \tag{5.4}$$

Each NBest string is weighted by its (posterior) probability. Let us choose:

$$\varphi_n = \exp\Big[(L_n - L_1)\eta\Big], \tag{5.5}$$

where L_n is the log-likelihood of the n$^{\text{th}}$ best candidate, φ_n is by definition lesser or equal to one, and η is a heuristic parameter that represents prior confidence on the decoded labels. When $\eta \to \infty$, then the best hypothesis is believed to be correct and a one-best adaptation is performed. If $\eta \to 0$, then an equal weighting is applied to NBest hypotheses. Figure 5.1 shows sample weights from different values of the parameter η versus the rank of the words sequence. When the vocabulary is closed and we sum over all possible words, then $\eta = 1$ corresponds to the actual posterior. When η is the inverse of the language model, we have Woodland's weighting scheme [WP00].

In all other cases, $\eta < 1$ leaves a left-over probability mass for unseen words. Contrarily to expectations, experimental work reveals that even though there is an improvement over 1-best adaptation, exponential weighting does not differ much from equal weighting. Therefore, it seems that there is no gain using the theoretically exact formula. Examination of the results showed that the algorithm was very sensitive to the segmentation, which is perhaps due to our sausage NBest generation.

However, we can make a general statement about adaptation techniques themselves before proceeding to the next topic.

5.3 Direct vs indirect adaptation: model complexity

It is not always stated in which case one should use an adaptation method or another. Let us consider two of the most popular ones: MAP and MLLR.

Figure 5.2 depicts a typical learning curve of MAP, MLLR, and ML in supervised adaptation mode. The reason why curves take these shapes is simple:

- when $\mathcal{T} \to \infty$, MAP $\to$ ML, whereas MLLR, with a fixed, finite number of classes, does not converge to ML.
- ML and MLLR being ML techniques, do not behave well when $\mathcal{T} \to 0$. This is because unseen events have no impact on the criterion function.

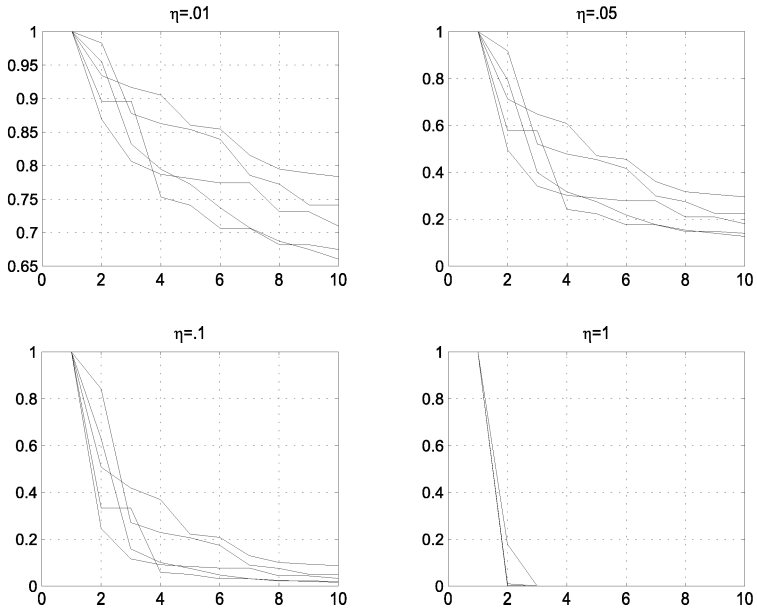


Figure 5.1: Exponential weighting versus the rank of the word sequence n

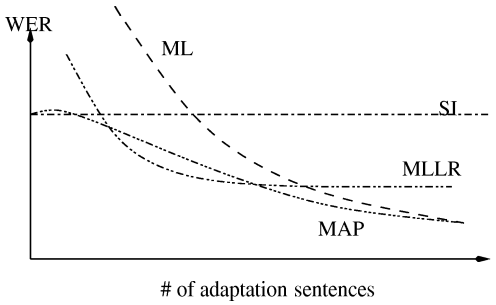
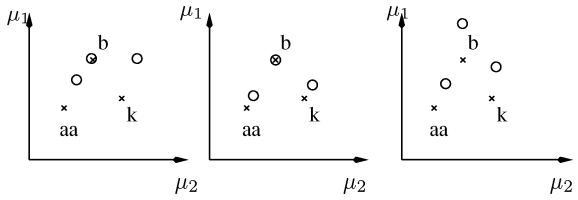


Figure 5.2: Typical learning curve: MAP, MLLR, and ML

- MLLR, however, adapts fewer parameters and converges faster to its plateau.
- MAP $\rightarrow$ SI when $\mathcal{T} \rightarrow 0$.

The main difference is that MLLR should be considered to be an *indirect* parameter optimization technique, whereas MAP changes each component directly.

MAP moves each Gaussian individually, whereas MLLR performs shift, rotation, and scaling (possibly projection). On Figure 5.3, we see examples of phonemes **aa** and **k**. Phoneme **b** remains unseen. MAP will move any seen example cautiously towards their



(a) ML (b) MAP (b) MLLR

Figure 5.3: Transformation of adaptation parameters via ML, MAP and MLLR. Crosses mark SI models. Circles mark updated models.

ML estimates. MLLR, on the other hand, will infer the position of the unseen model **b** because all Gaussians are shifted the same way. If all Gaussians share the same variance, the Euclidean distance may be readily interpreted as the KL distance, which in turn is a measure of log confusability. MAP, in this case, has harmed confusability, whereas MLLR hasn't. This is a key element that provides robustness to MLLR. However, the ability for MAP to move Gaussians individually is of great accountability for discriminative methods. In general, methods using a *coarse* level of granularity should be used for adaptation because they average out errors. Discriminative methods should use MLLR for adaptation (or regression), and on top of this a *finer* granularity adaptation such as MAP or MLLR with more classes to attain discriminative effects.

5.4 Discriminative adaptation

Following this discovery, we may now devise a new algorithm that achieves speaker adaptation and discrimination simultaneously.

There are several reasons why one should be interested in such a pattern. Firstly, from a phonological point of view, there are many reasons to believe that the relative position of phonetic units, divergence-wise, is speaker-dependent. In other words, a person's **aa** may sound similar to the same person's **ax**, but, for another speaker, there may not be a need to develop a fine distinction between the two. The other incentive that compels one to perform discrimination on a speaker-dependent basis follows a parallel argument from an information point of view. According to Jensen's inequality, the conditional entropy of speaker-dependent models must be at most that of the speaker-dependent model. Therefore, one is lead to entertain the hope that speech modelling conditioned by a speaker is but an easy battleground. Figure 5.4 illustrates our point. Let a, b be phonemes and X, Y be speakers. The SI model must model a much more complex landscape, which encompasses the union of the geometries of both speakers at the same time. Additionally, it is impossible for it to discriminate some parts of $b|Y$ and $a|X$. The third reason is perhaps more subtle. Speaker adaptation is intuitively related to dealing with a modest amount of data. Consequently, adaptation is done under a great lot of uncertainty, which causes the estimate to *roam* around the objective. Using the discriminative aspects, we may localise the area by adding soft constraints to prevent the estimate to look too close to another estimate of a

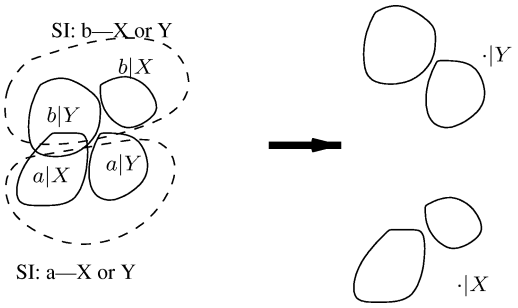


Figure 5.4: Conditional entropy decreases: the SI model must retain a lot of confusability

different class. In other words, we model a point $a|X$ not only as something reminiscent of the $a|X$ examples that we have seen, but also as something as repelled as possible by the $b|X$ that we have also observed. We call the samples of $b|X$ *anti-observations* for the phoneme $a|X$. As genuine observation, but with perhaps less descriptive power, these observations contribute to remove the curtain of uncertainty that shatters the estimate. They are counter examples: they show what the system what $a|X$ does *not* look like.

This third observation will be take literally to design our discriminative algorithm. The previous section constitutes the other contribution that help in the choice of the adaptation geometry. A coarse model is used for global speaker regression with collected data. A fine model is enriched with anti-observations and leads naturally to a finer discrimination. This is a win-win situation.

5.5 An EM-based discriminative algorithm

With these recommendations in mind, we apply a scaled-down version of the MCE criterion. Since both the discrimination term and the likelihood have the same parametric form, they instill a simple alteration of the $\gamma_m(t)$ posterior probability which leads to a different weighting for words:

$$\varphi_n = \begin{cases} \kappa & \text{if correct label,} \\ -\varphi \exp[(L_n - L_1)]\eta & \text{otherwise,} \end{cases} \tag{5.6}$$

where κ represents the weight given to the supervised forced alignment. Figure 5.5 shows a typical plot of the weight φ_n against the rank of the candidate n. The parameter κ is

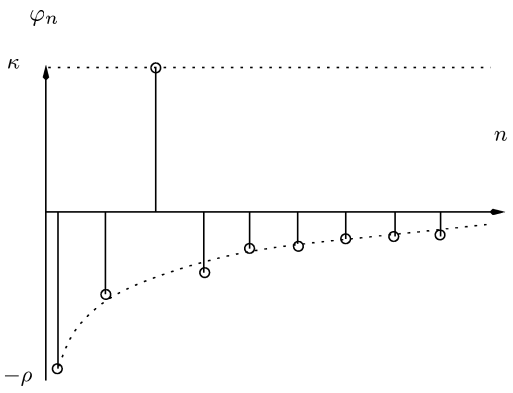


Figure 5.5: Typical assignment of weights

independent of the rank n because we want to recover the label the same way whatever its rank is. Parameters ρ and η contain the amount of back-off that mis-recognized items should receive. A rule of the thumb chooses $\eta > 0$ and $\kappa > (N-1)\rho$ so that the sum of posteriors is positive. In other words, adaptation (regression) is given more emphasis then discrimination (anti-observations). With values such as $\kappa = 2$, $\eta = 10^{-2}$, $\rho = .3$ perform well in the real world.

As any EM variant, perseverance in recurring application of the rule further improves models. The global training protocol may be summarized as

1. Use the NBest decoder to find the best matches.
2. Perform forced alignment according to the true transcription if not found by NBest decoder.
3. Accumulate using φ_n as described in (eq.5.6).

Best performance is achieved with sausage NBest decoding, or possibly with lattice integration. Under the assumptions that discriminative anti-observations "cancel out," we may use a single set of accumulators. Both coarse and fine adaptation may share the same statistics. In our experiments, we exploited this assumption in combination with MAP with MLLR as a prior (MAP|MLLR), that is, if W is the MLLR affine transformation of the mean, then the update is:

$$\mu_m = \left[\tau + \sum_t \gamma_m(t)\right]^{-1} \left[\tau W[1 \vdots \mu^T]^T + \sum_t \gamma_m(t) o_t\right]. \tag{5.7}$$

More generally in MLLR we need to introduce the sufficient statistics G_1, G_2 and z_1, z_2 for the precision and weighted mean of the coarse and fine regression matrices respectively, such that each row w_k is of the form:

$$w_k = \left\{ \left[G_1 + G_2\right]^{-1} \left[z_1 + z_2\right] \right\}. \tag{5.8}$$

We recognize the familiar MAPLR formula. The coarse class serves as a prior to the fine matrix. This is shown on Figure 5.6. We have used the concepts of coarse/fine classes for discriminative adaptation, together with anti-observations to improve reliability of estimates.

5.6 Overcoming local optima

As any iterative optimization algorithm, the EM algorithm is prone to precocious convergence into globally poor local optima. The sparsity of data affords much of a bias in the estimates by itself, which riddles the cost surface and produces shallow, local optima, which head the algorithm into ill-fated deception. The EM algorithm is decomposed into two operations: the expectation step, and the maximization step.

The E-step is the device through which, by means of the initial SI models, the segmentation is estimated. Errors in this initial phase remain imprinted in the following estimates, and performance is thereby impeded. This early precipitation to local optima is another impersonation of the eminent over-training phenomenon: the granularity of the model is greater than the descriptive power of the data. The richness of the model allows accidental structures in the training data to be modeled. There are two common remedies hitherto considered by the literature in this area:

- jack-knifing and the use of distinct models during E and M
- progressive model adaptation

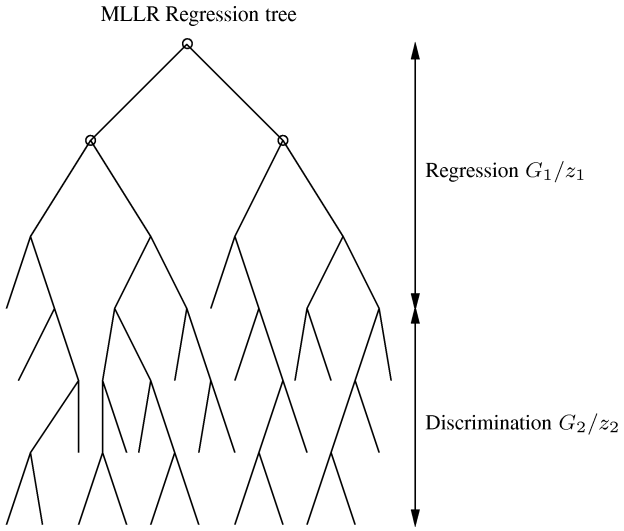


Figure 5.6: Regression trees for G_1/z_1 and G_2/z_2.

5.6.1 Jack-knifing

Jack-knifing was once popular in the early days of the Switchboard task. Several statistically independent observations are available. In a two-wire telephone conversation, they are identified as the different speaker turns originating from the same side of the receiver. The jack-knifing algorithm is a breed of the permutative cross-validation algorithm. We adapt on all cuts but one. The held-out cut is then recognized with this adapted model. The operation is repeated with all cuts. Another variant alternates between SAT and SI models. The design is clear: we try to delineate information processed in the E and M steps. Unfortunately, recent systems abandoned this idea.

However, there is still a form of this idea serviced in state-of-the-art HUB5E systems. Different inputs of the ROVER algorithm (or variant that of) are combined during intermediate steps, after adaptation. We do not leave one system out, but cross breeding still occurs.

5.6.2 Progressive model adaptation

The other typical technique is resilient of the presence of an abundant number of passes in HUB5 systems. While usually it is bantered that low-complexity decoders for the first pass are a necessity supported by computational concerns, it is only true a certain extent. Following that avenue of reasoning, only two passes would have otherwise been selected: a lattice generation, followed by adaptation and any ploy non-causal, and a lattice re-scoring pass producing final recognition, in a backwards pass.

It is well-known that the adaptation commonly reaches a bound in performance after at most four iterations of EM. Afterwards, researchers have employed, with much success, further adaptation passes, by introducing more degrees of freedom. In later stages, variance adaptation, full-matrix instead of block-diagonal, and more classes are all introduced.

The idea is that coarse models are required to flatten the cost surface. Once the region which harbours the sought model is located, a detailed search resumes, which is not confounded by such the pestilence of stumbling blocks. We may insinuate the general concept with the Figure 5.7.

There is not but a single trade-off between model complexity and abundance of data. Acquaintanceship with the EM algorithm reveals that distance (divergence) to the true

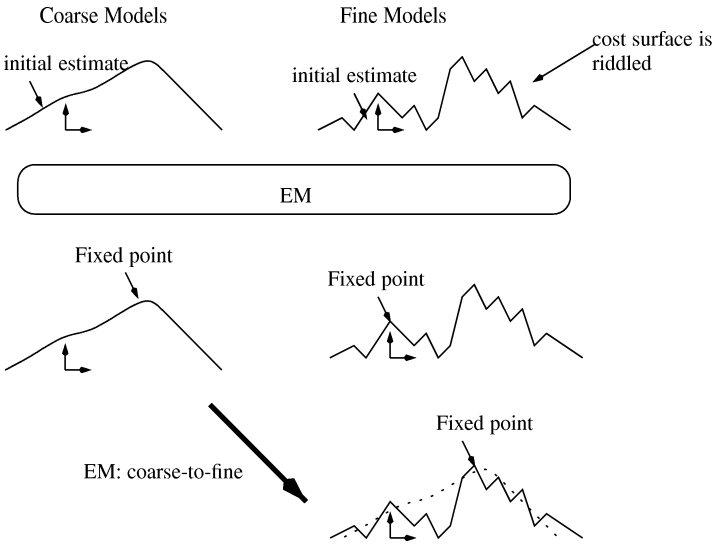


Figure 5.7: Progressive adaptation: in the coarse case, the cost surface is smooth and EM does not saturate in a local optimum

model is a factor that instills moderation in the ostentation of increasing model complexity immediately, and affords patience until we gain more familiarity with the landscape to display more confidence. We need to avail ourselves of the services of rustic low-complexity "myopic" models, that go unencumbered with minute details and trifles, and thereafter a detailed search may proceed. Very akin is the process whereby one looks at a map from afar, to locate a country, or continent, and then, equipped with a magnifying glass, may identify a village.

Such are the ways of accommodating with the locality of EM. We will now turn to a different take of the outlier problem removal, based on clustering techniques.

5.7 Experiments

We have conducted experiments using the NBest algorithms under supervised and unsupervised conditions.

5.7.1 Experimental conditions

Database

Since these results were obtained in early stages of this thesis, the large-vocabulary systems were not available yet. We selected a continuous spelled names database [CRF92]. The training data consisted of 1222 telephone calls. The test data was recorded by PSTL in a car environment with a close-talking microphone. There were 10 speakers:

- 6 native speakers of American English, and 4 non-native speakers.
- Amongst native speakers, there were 4 male and 2 female speakers.
- Amongst non-native speakers, there were:
  1. one Japanese female speaker,
  2. one French male speaker,

System	Unit Accuracy
SI	60%
One corrected (1/5)	70%
Two corrected (2/5)	78%
Three corrected (3/5)	85%
Four corrected (4/5)	94%
All correct (5/5)	100%

Table 5.1: Adding supervision to the adaptation data

3. one Italian male speaker, and
4. one Chinese male speaker.

Each speaker recorded 45 spelled street names in the car at 60 mph. There was a total of 3951 letters. The names were 8.8 letters in average length.

Additionally, speakers recited the alphabet once in continuous mode. The alphabet is divided into five sentences:

1. abcdef
2. ghijkl
3. mnopqr
4. stuvw
5. xyz

During this enrolment phase, the car was parked with the engine turned off. PLP parameters similar to 8.2 with first derivatives served as features.

5.7.2 Experiments

We first compare the unsupervised adaptation with supervised adaptation to calibrate the difficulty of the task. Then, we employ the discriminative scheme for supervised adaptation to improve the estimates.

Unsupervised adaptation

To asses the effect of the corruption of wrong transcriptions for adaptation, we vary the rate of correct labels. The speaker-independent HMMs led to an average of 60% recognition accuracy on the adaptation data. Then, we randomly selected one out of the five adaptation sentences and corrected its transcription. We reiterate the process until we have 100% accuracy, which is supervised adaptation. This is shown on Table 5.1.

We report results on Figure 5.8. We plot three curves: MLLR, MAP, and MLLR followed by MAP (MAP|MLLR). If μ_{MLLR} is the MLLR mean, then the MAP|MLLR is:

$$\mu_{\text{MAP|MLLR}} = \frac{\tau \mu_{\text{MLLR}} + \sum_t \gamma_m(t) o_t}{\tau + \sum_t \gamma_m(t)}. \tag{5.9}$$

It can be seen that in the case of native speakers, MLLR is less sensitive to the recognition accuracy on the adaptation data than MAP. For non-native speakers, this tendency is not present. This may be due to errors which occur sometimes between two letters not belonging to the same confusable set. In this case, the estimation of the MLLR matrix may not be reliable. MLLR combines statistics to estimate a set of parameters shared by several means, whereas MAP updates mean vectors directly. When $\mathcal{T} \to \infty$, MAP converges

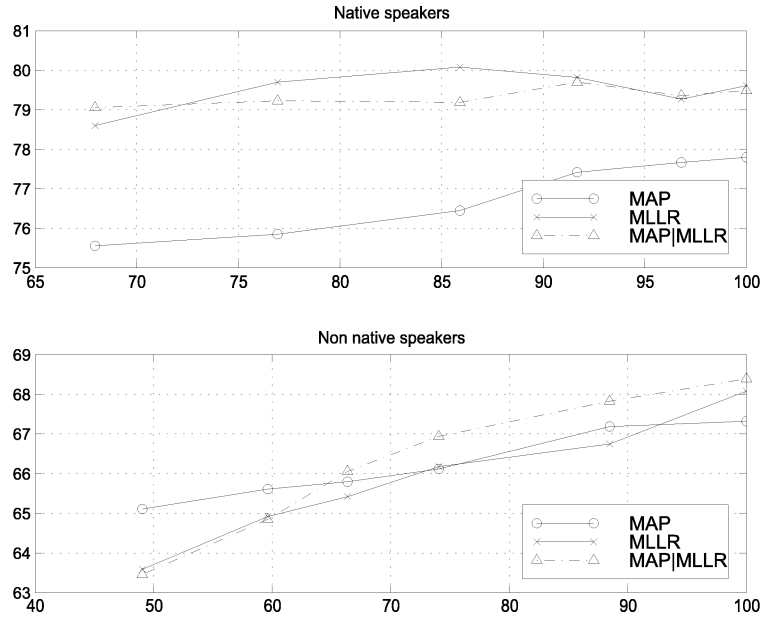


Figure 5.8: MAP and MLLR for native and non-native speakers

to the ML estimate. Therefore, in MLLR, the reliability of statistics is averaged into the estimation of the transformation matrix, whereas in MAP, the granularity is maximal and each errors is propagated directly. It becomes clearer that MLLR is better suited for unsupervised adaptation because parameters are estimated globally combining all statistics, correct and incorrect altogether. MAP, in contrast, updates model parameters for each mean individually, and therefore appears as a good candidate for discriminative training.

Supervised adaptation

We ran experiments with five alternative NBest candidates. The EM was iterated three times. Wrong assignments are weighted negatively, as in Figure 5.5. Varying η, we obtain results on Table 5.2. We set the positive feedback parameter to $\kappa = 2$. The negative magnitude parameter was set to $\rho = 0.3$. The unit accuracy does not vary much when κ and ρ are changed.

System	native	non-native	Average
SI	75.6%	64.3%	71.1%
Unsupervised ($\eta = 10^{-2}$)	79.7%	65.2%	73.9%
Supervised 1-best	79.5%	69.0%	75.3%
Corrective 5-best	81.7%	73.7%	78.5%

Table 5.2: Supervised adaptation using the corrective scheme: unit accuracy

The adaptation update was MAP|MLLR (eq. 5.9): we transform the means using MLLR first, then update each mean individually using MAP.

Chapter 6

Self-adaptation using clustering

6.1 Introduction

We have reviewed classical probabilistic methods and their solutions. Typically, statistical methods have a sensitivity to so-called outliers, which damage the estimation by being away from the true estimate by an inordinate distance. Such accidents are due to mislabeling of the data. Mislabeling occurs when the decoder does not yield the correct transcription. Alternatively, noise may also appear, as well as incorrect phoneme transcription.

An *ad hoc* apparatus is put in place, that steers away and discards such potential mishaps. We design such a device by means of clustering. First, we illustrate our claims by performing a synthetic experiment. We see that the usual model is not appropriate and suggest a solution.

6.2 Modified source model: The Black Sheep Experiment

Bayesian methods are optimal if the hypotheses hold. The need for introducing a non-linear thresholding method is justified. To clarify this, let us once again visit the Bayesian estimation of a Gaussian in several settings.

6.2.1 True observations

Consider an experiment where a set of samples $\{x_k\}$, $k = 1...B$ are produced by instantiating a Gaussian. The ML estimate for the mean is known to be the unbiased MMSE – the minimum of the best criterion:

$$\bar{\mu} = \frac{1}{B}\sum_{k=1}^{B} x_k. \tag{6.1}$$

When $B \to \infty$ it is known that $\bar{\mu} \to \mu$, the true mean in probability. In that case the Bayesian estimate has a closed-form solution and is the best that we can do, thanks to perfect labeling.

6.2.2 Impurities in the sample data

Now suppose that the source guiding the experiment is truthful with his first $B-S$ samples, all drawn from the Gaussian. Then, we introduce S black sheeps, observation with value (on average) $-\mu$. The Bayesian estimate would compute:

$$\mathrm{E}\bar{\mu} = \mathrm{E}\frac{1}{B}\sum_{k} x_k = \frac{B-2S}{B}\mu \tag{6.2}$$

where $\mathrm{E}(\cdot)$ is the expectation over all experiments. The estimate is now spoiled with a bias of:

$$\varepsilon = \frac{2S}{B}\mu \tag{6.3}$$

which is linear with S and inversely proportional to B. This is immediately follows the Cramer-Rao bound.

6.2.3 More realistic errors

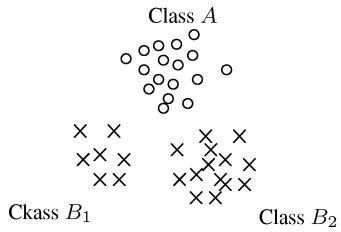


Figure 6.1: Non-zero mean of errors: consider class A. B_k are confusable classes around A. The expected value of B_k is not necessarily the mean of A.

It is quite easy to generalize this experiment to a mixture probability, with multiple classes and a hidden variable choosing among possible sources besides the correct one. Class origin is the hidden variable that we seek to discover ultimately. The EM algorithm is very popular in such environments.

6.3 Self-adaptation using clustering

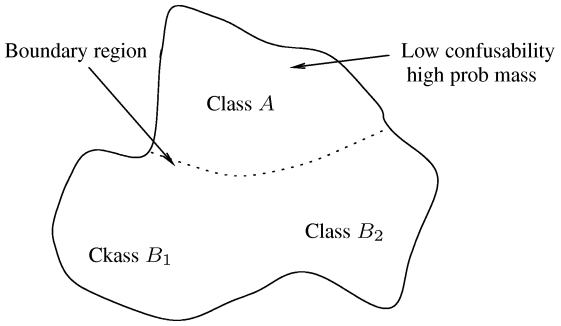


Figure 6.2: Confusability regions: competitors $B_{1,2}$, if introduced in the estimate of A, have a bias. Also, probability regression will detail useless regions.

Let us investigate the Bayesian behaviour when presented with a source such as presented in Section 6.2.3. We have now in command that errors may introduced by our initial estimate. The E-step may not be trusted entirely and misclassified items $B_{1,2}$ may appear with high probability. Second, on the figure, $B_1 + B_2 \neq A$. Errors do not cancel in the least and a Bayes estimate exhibits a bias. We address two issues with the following method:

- first order bias is not zero,
- errors may be assigned a high probability.

We employ the fact that estimates are correlated, whereas error are uncorrelated. That is, a second order assumption that errors are merely coincidental, while the estimate is uncertain but consistent.

The algorithm, however, is not readily explained without another short digression on the sufficient statistics of eigenvoices. These will probe a helpful tool in designing the algorithm.

6.4 Eigenvoices and sufficient statistics

Let us recall briefly how eigenvoices work. Each speaker is assigned a set of eigenvalues $\{w_e\}, e = 1...E$. Means of Gaussian distributions are modelled using a linear interpolation:

$$\mu_m = \sum_{e=1}^{E} \theta_e u_e, \tag{6.4}$$

where m is the index of the Gaussian component. Eigenvoices $\bar{\mu}(e)$ are trained off-line. Let $w = [w_1, ...w_E]^T$. The EM algorithm has a well-known solution (MLED) which goes by solving the weighted quadratic exponent:

$$Q = -\frac{1}{2} \sum_{t.m} \gamma_m(t)(\mu_m - o_t)^T C_m^{-1}(\mu_m - o_t), \tag{6.5}$$

in the usual notations. The MLED equation is equivalent to solving the linear system of equations for w:

$$\sum_{t,m,j} \gamma_m(t)\theta_j u_j^T C_m^{-1} u_e = \sum_{t,m} \gamma_m(t) u_e^T C_m^{-1} o_t, \ \forall e = 1...E \tag{6.6}$$

6.4.1 Compact sufficient statistics for the likelihood

In this section, we find statistics required to compute the likelihood. The idea is that this set of variables S will enable us to compute the likelihood of a segment of speech with respect to some eigenvalues. That is, a segment of speech can be summarized compactly in S as for as the computation of the likelihood of eigenvoices-adapted models is concerned. Define ϑ to be the completion data for the EM algorithm ($\gamma_m(t)$). The likelihood of the entire observation O satisfies:

$$\mathrm{E} \log p(O, \vartheta|\theta) \propto \sum_{t,m} \gamma_m(t) \left[\sum_{e,j} \theta_e \theta_j \bar{\mu}_m(e) C_m^{-1} \bar{\mu}_m(j) - 2 \sum_{e} \theta_e \bar{\mu}_m^T(e) C_m^{-1} o_t^T C_m^{-1} o_t \right]. \tag{6.7}$$

And thus, the following are sufficient statistics for the likelihood:

$$r(e,j) = \sum_{t,m} \gamma_m(t) u_e^T C_m^{-1} u_j, \tag{6.8}$$

$$b(e) = \sum_{t,m} \gamma_m(t) u_e^T C_m^{-1} o_t, \tag{6.9}$$

$$c = \sum_{t,m} \gamma_m(t) o_t^T C_m^{-1} o_t. \tag{6.10}$$

Since the log-likelihood is a quadratic form $(x-y)^2 = x^2 - 2xy + y^2 = r - 2b + c$, we have the autocorrelation matrix r and a crossproduct b. The constant term c is usually discarded. Furthermore, these products may be readily interpreted as correlation, and cross product in the weight inner product defined by $\gamma_m(t) C_m^{-1}$. The posterior is known only during the

adaptation and is a bequest of the E-step projection. The rest is due to the dimensionality reduction step. This is of course due to the Markov chain:

$$(o_t, \lambda_0) \overset{\text{E-proj}}{\rightarrow} \left\{ \sum_t \gamma_m(t), \sum_t \gamma_m(t) o_t \right\} \overset{\text{Dim. reduct.}}{\rightarrow} S \equiv \{r, b, c\}. \tag{6.11}$$

Note that, since eigenvoices usually has fewer degrees of freedom than ML or even MLLR, the sufficient statistics are very compact. This will become of crucial importance in later stages. First, let us review simple operations associated with segment statistics.

6.4.2 Segment algebra and sufficient statistics

We saw that a segment of speech S could be summarized by its statistics r, b, c. There are a few operations that may appear useful to review.

Define two segments of speech O_1 and O_2, with corresponding statistics S_1 and S_2. A rather interesting property of the statistics is that concatenation of the segments, say $O = O_1 + O_2$, have sufficient statistics that may be computed as the arithmetic sum of the segments' statistics: $S = S_1 + S_2$. It is also equivalent to the MAP formula using conjugate priors:

$$\log p(\theta|O_1) = \log p(O_1|\theta) + \log p_0(\theta) - \log p_0(O) \tag{6.12}$$
$$= \log p(O_1|\theta) + \log p(O_2|\theta), \tag{6.13}$$
$$\log p(O|w) = \log p(O_1|\theta) + \log p(O_2|\theta), \tag{6.14}$$

and we used the definition of the empirical Bayes' estimate:

$$p_0(\theta) = \log p(O_2|\theta). \tag{6.15}$$

The segment S_2 was used as a prior density definition with mean θ_2 and covariance A_2, which were calculated on the segment itself. It follows from the previous remarks that the estimation of MLED eigenvalues on arbitrary concatenation of segments, and conditional likelihoods may be computed easily. Moreover, the estimation of the gain or decrease in likelihood given a hypothesized eigenvoice model can be done solely on the basis of the sufficient statistics. Note that MLLR has similar sufficient statistics ([Bac00]). Those familiar with MLLR will recognize G and z matrices of (eq. 3.105, 3.106, p. 29). They are larger and more cumbersome to deal with.

Additionally, since we have linear models, the likelihood is again a Gaussian and therefore attains Cramer-Rao's lower bound for the variance. It is inversely proportional to the amount of data. The squared error due to the introduction of a wrong segment is also inversely proportional to the amount of data, as we have shown previously.

6.4.3 Segment clustering and self-adaptation

The motivation for the approach is based on the transposition of speaker clustering approaches to segment clustering. We have seen that eigenvoices uses fewer parameters and therefore is effective with fewer data. Figure 6.3 shows a typical learning curve for ML training, MLLR adaptation, and eigenvoices. We may reuse (and extend) the same weaponry deployed in speaker clustering. In [Bac00], sufficient statistics for MLLR were gathered with the specific intent to perform speaker clustering. Based on a similarity measure constructed with likelihood ratios, contiguous segments are pooled together.

Instead of speaker clustering, we have to decide which segments are correct. We do not require minutes of data, but instead fractions of seconds. Given an utterance, we divide the speech into segments. These segments may correspond to a uniform time window, phonemes, or words. Figure 6.4 shows an example utterance divided into words.

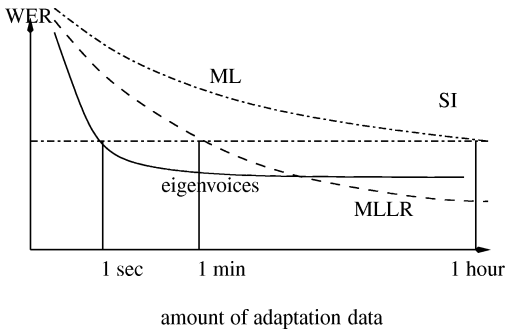


Figure 6.3: Learning curves for ML, MLLR, and eigenvoices

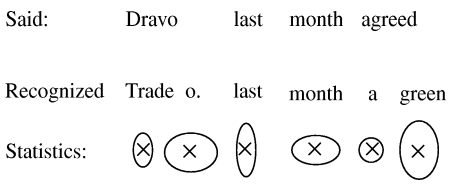


Figure 6.4: Dravo last month: To each recognized word, we assign one Gaussian. The mean and covariance are represented with a cross and an ellipsis.

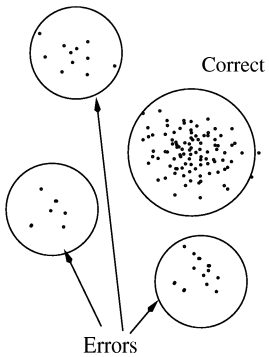


Figure 6.5: Correct estimates are correlated. Errors do not form one consistent cluster.

Each segment defines a pdf with an estimated mean and covariance. We cluster segments to cut off errors: correct estimates (see Figure 6.5) are highly correlated. They form a homogeneous cluster. Errors do not need to have zero mean, but they must be uncorrelated with the correctly labeled segments. They do not form a large cluster.

Unsupervised adaptation reduces to identifying the largest cluster. Given the assumption that errors are not correlated, the largest cluster, which gives a consistent speaker identity, corresponds to the correct segments. Even in high WERs (greater than 50%), if errors are not correlated, correct segments will be identified.

6.4.4 The choice of a distance measure

Now that we have seen how to merge segments, and that we want to single out the correlated segments, we need to define a distance measure between two segments.

Perhaps the most popular criterion in this case is the likelihood. Adaptation gains are used in [PWN00] to reject incorrect transcriptions. Utterance verification techniques are applied to suspicious segments. We introduce the measures in this order:

- likelihood ratio / posterior,
- divergence,
- cross-validation score.

The first distance that comes to mind for measuring similarity between two mean vectors is the canonical Euclidean distance. Let the mean be:

$$w_k = R_k^{-1} b_k \tag{6.16}$$

for a segment $k = X, Y$. The distance between two segments X, Y would be thus defined as the Euclidean distance between w_X and w_Y. Even though, in this chapter, we wish to mark a departure from standard Bayesian EM algorithms, we consider this measure to be inferior. The measure is dependent on the parameterization of w_k. Also, it is independent of the number of frames and observed covariance. In our case, we have, by convention, chosen the initial eigenvectors to be orthogonal with respect to the Euclidean distance in the whitened space. Any non-zero multiple of each vector would have been equally valid. The similarity measure described here would change, giving more or less weight proportionally to the square of the multiple. According to the joint Gaussianity model, one should use a Gaussian with covariance equal to the observed covariance in the training set; the correct weighting scheme, in that event, corresponds to the square root of the covariance, as in the Mahalanobis distance. Unfortunately, the observed Gaussian model corresponds to inter-speaker variability. It is therefore invalid for use with errors in transcriptions. It is merely a way of minimizing the error of the speaker location with respect to another speaker.

Now we shall generalize the use of covariance and attain the likelihood criterion. The likelihood function is not symmetric. By plugging in the eigenvalue in (eq. 6.7), we obtain a Hilbert-like distance between two speakers. This distance is performed in the precision space of the segment which is under examination for scoring. The precision is naturally proportional to the number of frames and its inverse contains both uncertainty due to lack of observations, and the observed variability. The variability is normalized by the expected precision given the SI HMM. The weakness of the likelihood approach comes from the fact that segments with many frames will assume an overwhelming dominating position in the scoring process. This is a common curse in confidence-based approaches. The canonical remedy uses a likelihood normalized by the number of frames.

In our third take, we use a distribution distance measure. Distribution measures are invariant to the parameterization. The Kullback-Leibler distance is almost exactly the likelihood measure, with an additional $\log |R|$ term which accounts for the contribution of the variability. This takes care of the frame normalization.

6.5 Further studies in Self-Adaptation

In the framework of self-adaptation, three extensions immediately spring to mind. The first one concerns the granularity of the segments. The second one pertains to the stationarity of eigenvalues. The third extends the statistical test to disregard highly uncertain eigenvalues to adjust the complexity of the model.

All three performed poorly.

6.5.1 Time segments

In Figure 6.6, it may be questionable whether `trade` should be rejected when most of it is correct. Since our recognizer is based on state lexical tree, recovering phoneme segmentations is possible but awkward. Nevertheless, we may choose to chop the utterance into equally long time segments. This is also takes care of the problem of normalizing to the segment duration. We remove segments containing at least one frame of silence since they may corrupt the estimate.

Surprisingly, this approach performs quite poorly. It may be due to the fact that errors form a slowly varying process with one discontinuity.

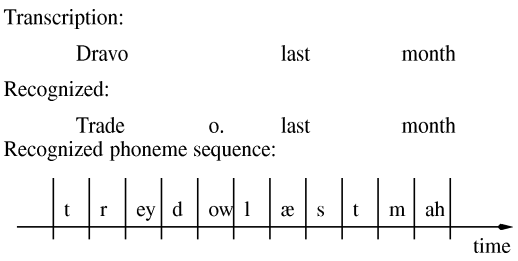


Figure 6.6: Time segments are used instead of word-segmentation

6.5.2 Non-stationarity

Speaker variability stems from many factors. One of them is average speaker rate. Our training and modelling procedure do not afford such subtle details. Still, it may be worth a try. For instance, Molau, Kanthak and Ney [MKN00] report that female seem to speakers exhibit different Vocal Tract Length (VTL) warping factors for each phoneme. On Figure 6.7, we shown a pitch contour, which is determined by the prosody of the sentence. This is a non-stationary effect.

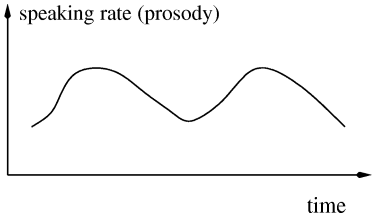


Figure 6.7: Non-stationarity is naturally introduced as speech is affected by prosodic content

This indicates that speaker characteristics may change during the course of the sentence. Experiments did not support these intuitions. We used different flavours, with one

sliding window, and some temporal filtering of eigenvalues. None of them resulted in any significant improvement.

6.5.3 Tuning model complexity

In principle, the same statistical test may determine the complexity of the model E. The variable E controls the dimensionality of the eigenvalues vector ϑ. Setting $E_0 < E$ is the same as forcing some eigenvalues $e = E_0 + 1, E_0 + 2, ..., E$ to zero. It is also similar to using a prior density to be $p_0(\vartheta)$ that sets zero precision to $e > E_0$. In particular the MAPED algorithm, the Bayesian counterpart of MLED, assigns a prior density:

$$p_0(\vartheta) = \mathcal{N}(0, \Sigma_\vartheta). \tag{6.17}$$

Unfortunately, the curve of error rate versus model complexity is rather flat. We blame this as the primary reason for our failure to extract an improvement from these ideas. Also, despite our efforts, eigenvalues are not orthogonal and speaker information is spread over many eigenvalues.

It seems that once again, the simplest method affords the best results, and increments are difficult to obtain.

6.6 Experiments

For our experiments we chose the Wall Street Journal Nov92 evaluation test (Section 8.4).

	SI-84 training (WER)	SI-284 training (WER)
SI	13.7%	10.8%
MLLR	13.1%	10.5%
MLED	12.6%	10.1%
MLED on time segments	12.6%	10.2%
MLED w/ variable dim.	12.6%	10.1%
MLED w/ confidence	12.2%	9.8%
MLED w/ conf + LM weight	12.2%	9.7%

Table 6.1: Self-adaptation: WER with SI-84 and SI-284

Results are shown in table 6.1. Word error rates (WERs) are reported for SI-84 and SI-284. We applied MLLR with one global matrix to get an idea of the difficulty of the task. For calibration standard MLED was also run. The level of significance for this task is 0.3% WER, using the Doddington rule [**?**].

The first pass generated a 1-best transcription. Word segments were clustered using the adaptation gain as a criterion, with a nearest-neighbor clustering scheme. There were on average 17 words per sentence. This resulted in a false acceptance rate (FA) of about 20% and a false rejection rate (FR) of about 40%. Intuitively, we consider the insertion of a wrong segment to be as detrimental as keeping two correct segments.

We tested the assumption of stationarity as follows. We updated the estimate once every 100 ms, based on an window length of 400 ms. Surprisingly the method did not result in a change in WER, even for different values of update period and window span. We believe that the non-stationarity is exactly balanced with uncertainty due to the removal of observation data.

Then, for every utterance, we tuned the complexity of the model E. That is to say, based on the adaptation gain (and amount of training data), we forced all $w_e, e >= \hat{E}$ to be zero for some empirically determined $\hat{E}$. For all values of tuning parameters, permutation of eigenvoices, and maximum E, the system did not outperform the baseline MLED.

However disappointing, it is consistent with our previous unsuccessful experiments with multigaussian prior for w (MAPED).

On the other hand, purging segments based on a ratios of adaptation gains resulted in an improvement. The false acceptance for words was about 20% and false rejection 40%. Errors in the exact transcription may not result in all wrong assignment of gaussian, and conversely a word pronounced poorly, but forced by the language model, may introduce noise in the estimation. However, intuitively, we consider one errors in assignment to be as detrimental as the added uncertainty due to the removal of two correct segments.

In our last set of experiments, we decreased the language model weight proportionaly to our confidence measure. The intuition is that in the case of poor acoustic match, we reduce the gap between first and second best hypotheses, and allow for more changes in the transcription, thereby prevent locking-in errors due to language modeling. We observed no significant improvement.

Chapter 7

Noise and speaker adaptation

In this chapter, we investigate the interaction between speaker and noise adaptation.

7.1 Introduction: why joint adaptation

When ASR systems are deployed, we can usually observe a large drop in performance. Differences in the environment account for much degradation of the system. They are perhaps the first source of errors in real-world systems.

A variety of remedies were proposed by speech researchers to deal with this problem. Contrarily to speaker variability, it is possible to model noise in an explicit way. Therefore, noise, unlike speaker change, may be described mathematically in a miscellany of ways. In this chapter, we will restrict ourselves to stationary noise, for instance, microphone change. When the nature of the mismatch is unknown *a priori*, it is customary to apply speaker adaptation techniques such as MLLR, which have almost no prior knowledge and adapt to the situation regardless of the actual nature of the change. In this event, we mix all variabilities in the same transformation. We are interested in separating the effects for better performance. This is represented in the taxonomy of Figure 7.1.

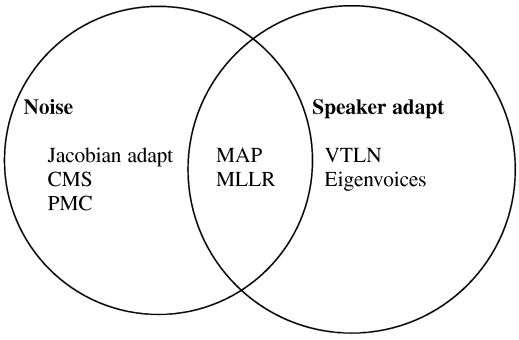


Figure 7.1: Adaptation domains and techniques

The table 7.1 explains the techniques and their properties.
In the same spirit we may distinguish between three kinds of adaptation:

- Noise adaptation: encodes prior knowledge in parametric form (Jac, CMS): the noise is added in the cepstral or log spectral domain.
- Speaker adaptation: encodes prior knowledge in eigenvoices (Eigenvoices): inter-phoneme correlations are present in each eigenvoice.

Speaker adaptation		Noise adaptation	
MLLR	no *a prior*	MLLR	works no matter what
MAP	*a priori* = SI	CMS	channel
Eigenvoices	*a priori*=speaker space	Jacobian	additive noise (Spectrum)
VTLN	prior = parametric form	MAP	works no matter what

Table 7.1: Speaker and noise adaptation: properties

- Speaker adaptation: encodes prior knowledge in parametric form (VTLN): the frequency is warped.
- Blind adaptation: do not assume any prior knowledge specific to the application (MLLR, MAP). The parametric form of the adaptation stems from concerns over mathematical tractability.

Except for VTLN, typical speaker adaptation have no human intervention that specifies a parametric model. Another example is adapting lexicon to speaker dialect. Strik [Str01] gives a very comprehensive review of techniques in pronounciation adaptation.

Both speaker and noise variability share two common properties which have made them targets of choice:

- they have a large impact on performance,
- it is measurable and is relatively controlled, and therefore suitable for data collection.

Other differences, such as stress, or semantic content, command an impact on acoustics. However, they come only second or third after noise and speaker variability. Moreover, they are difficult to label or observe in a database. Additionally, ASR has traditionally been focusing on very specific tasks, tackling problems within a restricted domain.

For these reasons, we have decided to study speaker adaptation in conjunction with noise adaptation.

7.2 MLLR and Eigenvoices

To illustrate the problem, let us show how eigenvoices may be used in conjunction with MLLR to cancel speaker and noise respectively.

Figure 7.2 shows the problem with Eigenvoices. The prior knowledge is built on a training database with homogenous conditions, and usually noise-free. The test conditions, however, have to bear the presence of noise. Both noise and speaker adaptation techniques may be applied. However, the usualy choice is to use one, or the other exclusively. Alternatively, one can use the ubiquitous MLLR to resolve *both* mismatches at the same time.

If we would want to make use of available tools, such as MLLR or eigenvoices, we come across an odd predicament. How do we deploy eigenvoices models amidst noise corruption? Surely the conditions change and all gains given speaker adaptation will be lost. Conversely, if we apply MLLR, gains will be impeded by the dual mission of the adaptation: adapt to the speaker and noise simultaneously. The answer lies in the Figure 7.3.

We shall use two different parametric forms for the different noises. Figure 7.4 goes into further details. We use a simple hierarchical decoding of variabilities: we posit them independent. Then, using alternative projections of EM, we maximize the profit. Since this is joint adaptation, the scheme is very sensitive to local optima. Therefore, care must be exercised to slow down convergence. It is common sense to start with the variability with greatest magnitude: noise.

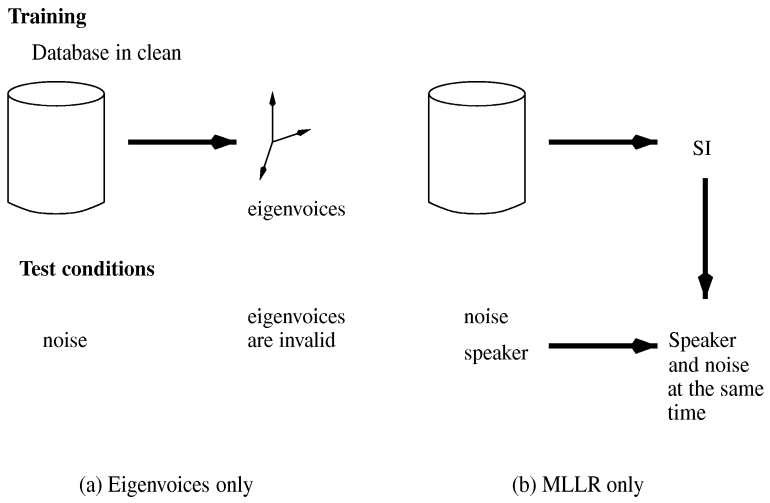


Figure 7.2: Eigenvoices and MLLR: either speaker or noise adaptation

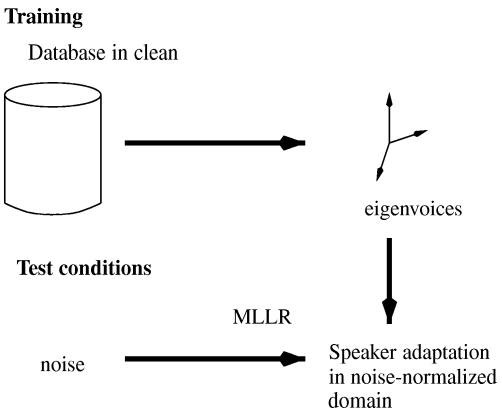


Figure 7.3: Eigenvoices and MLLR: speaker and noise adaptation

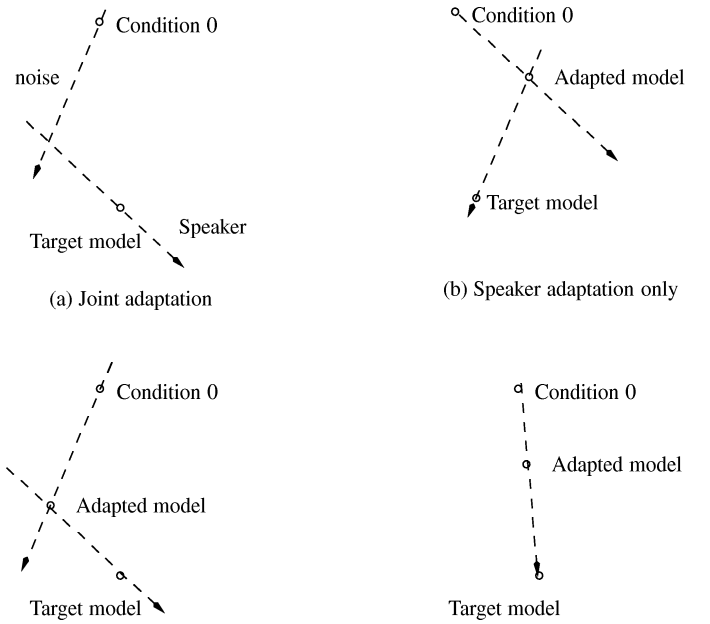


Figure 7.4: Hierarchical decomposition of variabilities

Conjecture 1 (Orthogonality) *Noise variability is orthogonal to the speaker variability. We suppose that ambient noise is independent of the speaker, and that all speakers react to noise in the same way.*

Algorithm 1 (Hierarchical noise decoding) *We initialize the algorithm with:*

- *Noise* $\equiv$ *no noise,*
- *Speaker* $\equiv$ *SI.*

Then we proceed with alternating optimization as shown on Figure 7.5:

- *Noise* k*:* $\max \log p(O|$*speaker* $k-1)$*,*
- *Speaker* k*:* $\max \log p(O|$*noise* $k-1)$*.*

There are risks involved when using the hierarhical decoding in Figure 7.5. In Figure 7.6, we see that a suboptimal estimation of noise results in an incorrect estimation of the speaker. The correct combination of noise and speaker is never recovered.

Specifically for the case when noise is reduced with MLLR and eigenvoices normalizes for the speaker, model means are affected by :

$$\mu_m = W_m \begin{bmatrix} 1 \\ \cdots \\ P_m^T \vartheta \end{bmatrix}, \tag{7.1}$$

where ϑ is the vector of eigenvalues, P_m^T the corresponding eigenvoices, and W_m the regression matrix for the class to which m belongs. In our experiments we used only one global transformation matrix. In that case, it is more advantageous to apply the noise reduction in the feature space:

$$\tilde{o}_t = A^{-1} o_t - b; \qquad W = [b \vdots A]. \tag{7.2}$$

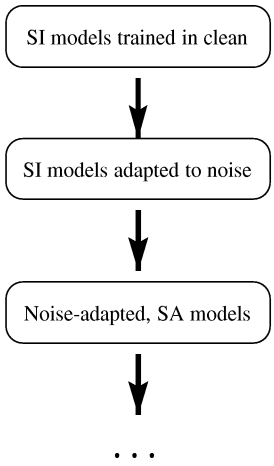


Figure 7.5: Multiple iterations of the hierarchical decoding

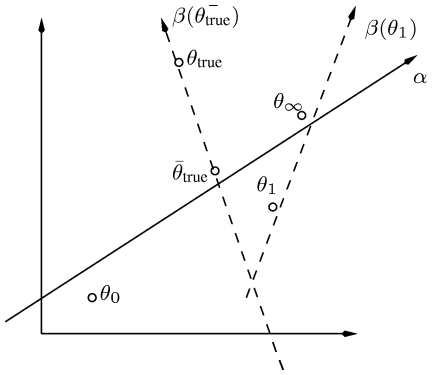


Figure 7.6: EM converges to the incorrect estimate because the conjecture is violated

Thereafter, the eigenvoices work in the noise-cancelled (i.e. cleaned) domain.

The scheme may be extended to a plurality of speakers. In that event, the noise reduction matrix is shared amongst speakers. This consideration becomes of paramount interest when some data may be collected in the test environment, but not sufficiently to train full eigenvoices models. Figure 7.7 shows how this setup occurs. In D_0, where speech is avail-

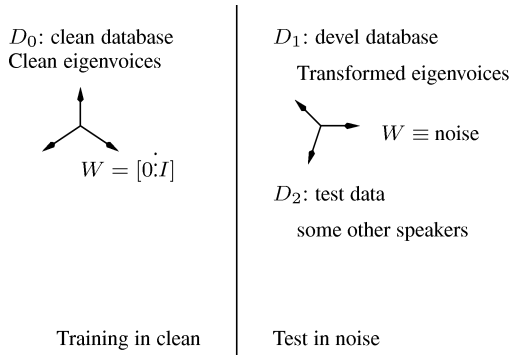


Figure 7.7: Eigenvoices and MLLR: adapting on D_1

able in enormous quantities, we train eigenvoices models. Using a small development set D_1, we estimate the noise transformation. This is the main requirement of the approach. We need to have a development data set. Using unsupervised adaptation, we can avail ourselves of unlabelled data sets. For best performance, we assume that D_1 comes with labels. When come new speakers in the same conditions D_2, we are able to renormalize eigenvoices in that environment and adapt to the speaker in this environment. This scheme should be performed whenever D_2 is available, for it allows for explicit distinction between noise and speaker, which, improves the joint adaptation scheme.

We have solved the final problem with eigenvoices: requirements that we gather infinite amounts of data in test conditions. We are able to simulate unseen speakers with eigenvoices, and with MLLR, unseen speakers in unseen conditions.

7.3 Experiments

7.3.1 Experimental conditions

We used TIMIT in the same experimental framework as in Section 3.10, p. 47. Noise of a car running at 60 mph was added artificially to the test utterances. No noise reduction processing was applied. In Table 7.2, we report results in unit accuracy. The SNR for clean TIMIT is about 70 dB.

7.3.2 Normalization

We employed two databases. The training database D_0 is the standard TIMIT training database. To train to the noise, we have a small development database D_1. The test was done on a third part of the database D_2 with the same noise as D_1. D_1 comprised 30 speakers, each pronouncing 8 sentences. D_2 was made up by 30 speakers, each pronouncing 1 sentence (about 2-7 sec of speech) for adaptation $D_2^{(a)}$ and the rest for decoding $D_2^{(b)}$. All results reported are on $D_2^{(b)}$. SI (D_0) represents the SI model, estimated on the full training set of the TIMIT database. MLLR(D_1) can be interpreted as the SI normalized by the environment learned from D_1. MLLR(D_2) and MLED(D_2) correspond to MLLR and MLED applied normally, without any use of D_1. Finally, normEV(D_1, D_2) symbolizes MLED applied on

Method/SNR	∞	40 dB	30 dB	20 dB
SI (D_0)	60.94	50.13	31.09	10.63
MLLR(D_1)	59.79	56.86	44.82	30.82
LR (D_2)	53.14	52.44	42.78	25.07
EV (D_2)	65.05	57.13	43.14	19.31
normEV (D_1, D_2)	64.25	62.53	52.08	34.54

Table 7.2: Unit accuracy for different SNRs

Size / SNR:	∞	40 dB	30 dB	20 dB
30×8	64.25	62.53	52.08	34.54
10×8	64.46	61.65	51.37	33.78
10×4	63.59	60.83	53.28	33.08
20×2	63.52	60.35	50.74	32.91

Table 7.3: Unit accuracy when reducing data for environment normalization

$D_2^{(a)}$ with priors transformed using an estimation of the environment based on D_1. These sets were sliced randomly (non-overlapping) from the test set of TIMIT. For all tests, E was set to 10.

7.3.3 Further experiments: reducing the amount of data

In a further experiment, we examine how the algorithm reacts when we reduce the size of the re-training database, D_1. Table 7.3 summarizes the results. The first column describes the size of the database by the product of the number of speakers times the number of utterances per speaker. We see that it is better to have fewer speakers, but each pronouncing more utterances, than more speakers with fewer utterances. This conforms to the intuition that we are attempting to comprehend speaker variability with many speakers. Undertraining of each speaker is filtered out after PCA. These results might be dependent on the database.

Part III

Evaluation and decoding

Chapter 8

Evaluation framework

8.1 System descriptions

This section is devoted to the descriptions of the systems that are used to evaluate the algorithms. A large part of my time during the thesis was invested in developing PSTL's large vocabulary decoder and systems, which culminated with the participation to NIST's Rich Transcription 2002 evaluation.

An overwhelming amount of research in speech recognition is carried out under direct or indirect influence from the DARPA programs. In only two years' time, I have implemented a large vocabulary recognizer, with training tools for acoustic and language modeling. We have developed systems corresponding to all three generations of DARPA LVCSR evaluations, namely Wall Street Journal (WSJ/Hub3), Broadcast News (BN/Hub4), and Switchboard (SWB/Hub5).

8.2 Switchboard

In 1989, DARPA contracted Texas Instruments to record a database of telephone conversational speech for use in speaker identification tasks. This database was later transcribed at the LDC, and re-transcribed at Mississippi State University (MSU/ISIP). The re-transcription effort attained completion in August 2002 under the direction of Dan Harking. This database is known as the Switchboard Phase 1 (swbd1) database. In 1992, another round of recordings took place and yielded a database twice the size of the first one, which was christened Switchboard Phase II (swb2). Except for short excerpts, it was never transcribed manually. Recordings have again been revived, this time including cellular phone.

Switchboard (SWB) has enjoyed intense investigation from leading speech recognition sites. Only two tasks are considered more challenging than SWB. The first is CallHome/CallFriend, in the same acoustic quality, but where people call relatives and friends. This task is so difficult that human transcribers themselves consider it impossible to transcribe. The second more difficult task is that of transcribing multi-speaker Meetings. It is currently being collected and will be the focus of the upcoming DARPA MUSE program.

I have, almost single-handedly and with modest resources, made a humble attempt to catch up with multi-million dollars research legacy in that area. I will try render details of the SWB system submitted by PSTL to the evaluation.

8.2.1 Parameterization

The input signal is filtered to retain only 100-4000 Hz. After pre-emphasis, PLP coefficients with an 8-pole model are computed every 10 ms over a window of 25 ms. Residual energy was prepended to the static feature vector. Those 9 coefficients $s(t)$ were augmented

with the first and second order time derivatives. The time derivative was approximated with a five-tap non-causal filter $h(t)$:

$$\frac{\mathrm{d}s(t)}{\mathrm{d}t} \approx \sum_{\tau=-2}^{+2} h(\tau)s(t-\tau). \tag{8.1}$$

These cepstral coefficients were then normalized to have unit variance, and zero mean over each conversation side on all 9 dimensions of the static parameters. Delta and acceleration coefficients were computed after variance and mean normalization. By construction their mean is zero. Normalization excludes long segments of silence. Specifically, during training, we used segments as defined by MSU. During decoding, we used either segments provided by NIST (transcribed with the same specification as MSU), or automatically defined segments.

The convention adopted by MSU results in a padding of at least one second prior to and posterior to speech in every segments. This introduces a large proportion of silence in the speech segments, which is believed to reduce gains due to normalization. Moreover, this silence amounts to about 30h duration, and slows down training. For simplicity we chose not to remove it for this evaluation.

8.2.2 Acoustic training

In our system, acoustic modeling was the primary consumer of computational resources. Training LVCSR models is a discipline that concedes a large influence of parameter tuning. We will describe our choices.

Data

Training data was drawn exclusively from the Phase-1 SWB (SWBD1). LDC provides the sound files under catalog number LDC97S62. There is approximately 265 hours of speech. There were 2436 conversations, or 4872 conversation sides. Due to lack of time, we made no use of additional material such as CallHome, Cellular, etc. The segmentation was bootstrapped from an early automatic word-level segmentation provided by MSU. Training and segmentation were alternated for more than 6 months. Forced-alignments use recognition models adapted to the speaker using MAP-smoothed MLLR. Therefore, we have SAT-grade quality segmentations.

All speakers were identified with a PIN number. They are recurrent in development and evaluation corpora. There were 539 speakers in the training. Some conversations contained unusable transcriptions, or very poor quality due to frame dropping. There were filtered automatically by a likelihood based confidence measure. We kept almost all of the training data. A total of 4400 conversation sides are present in the training. After filtering, speakers had either zero training data, or at least more than one minute of speech.

Lexical units and Language modeling

The language model was kindly provided to us by Andreas Stolcke from SRI [SBB+00]. It contains 34610 words, including 1659 compounds, 4.8 million bigrams, and 11 million trigrams. Training data includes Broadcast News transcripts, Call Home, and Switchboard acoustic transcription data.

We employed 41 phonemes including one for the silence. Cross-word transcriptions were manually inserted for the most frequent compounds. We believe that compounding is an efficient alternative to first-pass cross-word modelling. We leave the decision of selecting cross-word modeling or word-internal modeling to the Viterbi segmentation step. For instance, the compound `you_have_a` may be modeled as:

- Word sequence `you have a`, denoting word-internal units.

- Word sequence `you_have a`, for a cross-word model for the first two words only. The last word `a` is modeled separately because of a pause between the words.
- Word sequence `you have_a` is similar.
- Word sequence `you_have_a` for full cross-word modeling. No silence is allowed between these words, and coarticulation effects may be introduced in the acoustics.

Instead of applying the rule greedily to model across-compounds, we split compounds arbitrarily. For instance, `do_you_have_a` does not appear in the SRI language model. It was split into the two compounds `do_you` and `have_a`, which could be split in smaller units. The reason is that we were concerned that the greedy rule might leave too little data for model synthesis. If cross-words are always selected during training, there is very little data left for word-internal model synthesis of unseen compounds. The problem can be seen easily with the case of acronyms. Acronyms are sequences of letters, such as `C. N. N.`. If all acronyms are trained as compounds, there is no data to left to synthesize unseen acronyms. Similarly, language models will leave almost no backoff probability mass for unseen acronyms.

Triphone clustering

Our best acoustic models were trained using 3-state, left-to-right HMMs, with no skip state. Gaussian mixtures provided the basis for probabilistic modeling on each state. States do not share Gaussians, but may be shared across triphones. This is called State Clustered Tied Modeling (SCTM).

Sharing is settled by a Classification And Regression Tree (CART) algorithm. Trees were grown using half of the acoustic training data. The second half of the training data served as a cross-validation set to prune unreliably trained leaves. After we adjusted the minimum count and likelihood gain thresholds, this resulted in 3892 states. Trees assumed a single Gaussian distribution at each level.

There were 63 phonetic questions, which we obtained from MSU. The questions distinguish between vowel, voiced, front, lenis, etc., to determine the nature and the position of the sound. This set is slightly larger than the one used by Odell [Ode95]. We asked questions about the position of the allophone (immediately after or before a word boundary). This is called position-dependent triphone modeling.

Iterative splitting - merging

The single Gaussian distributions inferred from pooling growing and cross-validation pruning data were then refined using word-internal Baum-Welch training. Distribution were split and perturbed proportionally to their standard deviation. Another round of Baum-Welch occurs. We split by a factor of two exponentially to 128 Gaussians.

In the last step, we had trained 489350 Gaussians. A nearest-neighbor clustering based on least likelihood loss trimmed unreliably trained Gaussians. We retained 256000 Gaussians. Merging of Gaussians was permitted only within states.

8.2.3 Decoder

In order to meet the self-imposed 10x real-time constraints, we truncated the decoding strategy to a single-pass trigram decoding. We were the only ones to participate to the unpartitioned condition. We paved the way for limited resources SWB conditions.

The segmenter uses gender-dependent GMMs to find speech from the four-wire (4W) data. In NIST's terminology, four-wire data was defined as data collected on the switchboard, separately for each channel. Therefore, meta data annotation as defined this year

System	WER
Dev set	34.4%
Eval set	36.7%

Table 8.1: Development versus evaluation performance on SWBD1 data

reduces to a simple speech detector. We adjusted the slackening parameters to an overbearing false acceptance ratio. We preferred to insert pure silence segments that would be detected as such by the LVCSR decoder, rather than missing speech altogether.

Recognizers ran on the partitioned (PEM), and unpartitioned (UEM) conditions were identical up to beam size. During the evaluation, we solicited NIST for a definition of the real-time factor computation. To keep in line with Meeting Room experiments, they decided for a lenient real-time factor computation for the UEM, that leaves twice as much time for the decoder. The real-time factor is defined as:

$$\mathrm{RT} = \frac{\text{Total processing time}}{\text{Reference time}}. \tag{8.2}$$

The definition of the reference time is:

- PEM: total of duration of segments as provided by the NIST (MSU-style annotations),
- UEM: total of duration of channel A **plus** total duration of channel B.

For the sake of comparison, both recognizers used the same single-pass strategy. Encouraged by our preliminary findings, NIST will probably settle for the UEM next year. Moreover, CU-HTk was probably engrossed by our submission when they submitted a late limited resources system.

8.2.4 Development / Evaluation Results

In this subsection, we report the performance of the system in different conditions.

For our first participation to a NIST evaluation, we tried to follow the instructions of the evaluation plan as closely as possible. It states that sites should submit as many conditions as possible, even though the results might be poor. We were the only ones submitting in a number of categories. We submitted the most systems to the evaluation.

Due to a lack of computing resources, our developments were only measured on the earlier SWBD1. The Phase 2 (swb2) and cell phone (swbcell) series were available to us, but ignored for practical reasons. Table 8.1 shows the dev set versus eval set results. Referring to previous evaluations, we estimated that the first pass of the unlimited systems were scoring at about 35-40%, with a real-time factor of about 20 times RT. The systems that we report were 10 times RT, that is two times faster for about the same performance. We can see from the table that there was a significant "overtuning" to the database.

Moreover, the results from the actual evaluation have to be blended with the two other sets, which are more difficult. Table 8.2 shows the results on the evaluation sets. Again, due to lack of time and computing resources, we did not test the submission system on these part of the dev sets. There are two different systems: the manual and the automatic segmentation systems. We were the only ones to submit an automatically segmented Switchboard transcription system. The automatic transcription is believed to be harder. However, because of the larger beam, as explained in Section 8.2.3, the results are slightly better. Despite the preliminary nature of our results, this has encouraged NIST to propose an automatic condition for RT-03.

After the evaluation, and hopefully encouraged by our tentative, CU-HTk decided to develop a "late" system. AT&T submitted a faster-than-RT system timely. Results are shown on Table 8.3.

Test set	WER manual PEM	WER auto UEM
SWBD1	36.7%	35.9%
SWB2	44.3%	46.9%
SWB cell	50.9%	48.0%
Average	44.5%	43.9%

Table 8.2: Evaluation results for PEM and UEM tests

System	RT	WER
CU-HTk	10x	27.2%
PSTL	10x	44.5%
AT&T	1x	28.4%

Table 8.3: Limited resources systems

On Table 8.4, results for the unlimited resources systems are given. These systems can use more than 300 times RT. They typically use two to four passes with two to seven systems for ROVER combination, with full variance adaptation and pentaphone modelling. By design, we have decided to concentrate on a small set of features, which would be directly applicable to a commercial system. The NCE measure is the normalized cross entropy.

System	WER	NCE
AT&T	26.4%	-0.420
BBN	28.4%	0.188
CU-HTk	23.9%	0.289
JHU	33.6%	N/A
LIMSI	30.0%	0.239
SRI	27.4%	0.268

Table 8.4: Unlimited resources

As we can see, the PSTL system is not competitive yet. In absolute WER performance, it is comparable with the open-source MSU-ISIP system, which did not participate this year. However, the PSTL system operates about 80 times faster than ISIP's hierarchical decoder.

In terms of MetaData, the raw results are shown on Table 8.5.

11714.69	Total Speech to be segmented (in seconds)
7834.75	Correctly segmented speech (in seconds)
3879.94	Incorrectly segmented speech (in seconds)
0.33	SEGMENTATION ERROR

Table 8.5: SWB: Meta Data Results (frame error)

During the workshop, NIST presented detailed results where they showed that PSTL performed equally regardless of the set type (SWBD1, SWB2, and SWB-Cell). MIT-LL performed better than PSTL on SWBD1 and SWB2, and approximately equally worse on SWB-Cell.

8.3 Broadcast News

Our efforts on Broadcast news (BN) data were limited to three months (Jan 2002 – Apr 2002). The BN database consists of news broadcasted from various news channels, such as CBS, ABC, NPR/Marketplace. Test data includes shows from VOA and CNN not present in the training data.

News were recorded in 1996 and 1997. Two hundred hours were transcribed. The data were also labeled versus speaker name. There were more than 4000 distinct identification tags. Some of them may be repeated for different speakers across shows (e.g. `spkr_1`). Due to presumably transcription errors, these tags may refer to different speakers within one show. We define *nominative* speakers as speakers whose tag do not contain digits (`john_doe2`), nor three consecutive capital letters (`ABC_Announcer`). Some were manually corrected, for instance:

- `Dave_Bird <=> David Bird`, and
- `Hilary_Rodham_Clinton <=> Hilary_Clinton`.

After normalization, we still had more than 2000 distinct nominative speakers. More than 50% of them appear fewer than or exactly twice in the database.

BN is characterized by its real-environment nature ("found" data). It includes music in the background and interviewees speaker over telephone channels. Some degree of spontaneity is also present. To a lesser extent, a few examples of overlapping speech may be found. Used in the early days for indexing only, BN systems are now being deployed for closed-captioning by the NHK (Japanese Broadcast company) and the BBC (British Broadcasting Company), but is still thwarted by fierce opposition in the US. CNN recently adopted the English connectionist system formerly known as the Abbot system for their indexing needs.

BN is the LVCSR system that is closest to being deployed. Therefore, extra emphasis is put on real-time operation and simplicity versus performance.

8.3.1 Parameterization

The frontend generates MFCC parameters at a frame rate of 100 Hz. After pre-emphasis, the power spectrum of 32 ms is integrated of 20 filter banks. After the Inverse Cosine Transform, we retained 12 cepstral coefficients, excluding c0. Residual energy is prepended to these coefficients. The five-tap filter of equation (eq. 8.1) is applied twice for delta and acceleration processing. To the static coefficients, we applied a purely causal version of online cepstral mean normalization. An average over the 2 previous seconds of speech is subtracted to all 13 static coefficients prior to delta computation.

As opposed to some other sites, we decided not to incur special processing for narrow-band conditions. Narrow-band is understood as telephone speech or speech recorded with low-quality.

8.3.2 Acoustic training

Broadcast news systems were trained on both training corpora available from LDC, train96 (LDC97S44) and train97 (LDC98S71). Overlapping speech and music-only segments were discarded. Segments were cut at each available time reference point, including `SyncTime`. Transcriptions for this task were surprisingly cleaner than MSU transcriptions. They required almost no post-processing for errors.

We applied the same acoustic training strategy as in SWB (Section 8.2.2). Tri-state, position-dependent triphonic HMMs with entropy-merged SCTM mixtures serve as the basis for speech recognition. Specifically, decision tree clustering produces 2739 states, which were split exponentially to 128 Gaussians per state. Out of the resulting 347832

LDC Name	Type	Size (M words)	Weight
1996 CSR-Hub4	Broadcast	140	3
North American News (NAB)	Newswire/paper	500	1
TDT2 + TDT3	Broadcast	31	3
Acoustic training	Broadcast	1.6	12

Table 8.6: LM training data: amount, type, and weight

Gaussians, we keep only 192000. The phonetic questions were the same as those set out in Odell's thesis [Ode95].

Amongst several gender-dependent training strategies, we found the following to work best. SI models are cloned for each gender. Variances, transition probabilities, and mixture weights parameters remain the same as SI. We then update all seen means by Baum-Welch training with the ML criterion. Unseen means are left untouched, as they seem to provide background modelling necessary for recognition. MAP, MLLR, update of variances, and removal of unseen Gaussians all degraded results. Bandwidth-dependent models were not found to bring stable improvements for speech recognition and were not explored further.

8.3.3 Language Modeling

The language models were trained from a variety of sources, all available from LDC, including closed-captioning data from Broadcast News (CNN), newswire (e.g. Reuters), and newspapers (WSJ). Table 8.6 shows the sources. Texts were processed using a modified version of the normalization tools provided with WSJ.

TDT data excluded epoch December 1998, from which RT-02 evaluation data was known to be drawn.

The vocabulary is selected amongst the 60000 most frequent words in all corpora excluding NAB. In that case all corpora had the same weight. From these words, a number were excluded. We chose to include only those words that were present in COMLEX, SWB training data, BN training data, and most common names in the Census database. False-starts were filtered out. Hesitations were normalized into only `uh` and `hmm`. Compositions were permitted with hyphens (across-the-board). After this filtering we had extracted a vocabulary of 53514 words. Rejected words included about 3000 words which were a mixture of rare valid words (according to the Webster 1913 release), and proper names. They were added manually after the evaluation.

Language modeling interpolated corpora at the count level. The Katz backoff topology, with Good-Turing estimation, and CMU-style cut-off pruning brought about 19M bigrams with 68M trigrams.

8.3.4 Segmenter

The meta-data comprises two tasks: speaker segmentation, and speaker clustering.

The speech is first decoded using a GMM with 512 Gaussians per model. Models were trained for silence, and the Cartesian product of bandwidth (narrow/wide) and gender (male/female). Since the train97 data does not include bandwidth tags, a model was trained on train96, and used for further decoding on train97. Bandwidth classification does not enjoy high agreement between transcribers. Therefore, it is never to be evaluated as meta data. Rather, it might be used by sites to improve modeling.

A unigram decoder produced a first-pass output. The decoded output is then heuristically smoothed according to the following rules:

1. Consecutive segments of speech are merged, and assigned to the dominating class.

Total number of classes	Vowels	Consonants	Silence
7 classes	4	2	1
5 classes	3	1	1
3 classes	1	1	1
1 class	1		0

Table 8.7: Allocation of matrices to broad phonetic classes

2. Speech segments surrounded with fewer than 400 ms are merged again with neighbors if speech within the segments lasts for less than 4 seconds.

3. Silence of fewer than 150 ms between two segments of different conditions are collapsed, if either segment is less than 4 seconds. A resulting segment encompassing both is labeled with the longer segment's label.

This was found to minimize the false-rejection (FR) rate. As in the SWB evaluation, we prefer to decode more silence using LVCSR, than to chop words. The BIC criterion further merges segments.

Clustering also makes use of BIC. We begin with all segments alone in their own cluster. Then bottom-up, nearest neighbor hierarchical clustering merges segments until a predefined BIC threshold is met.

8.3.5 Decoder

BN decoding stands in sharp contrast with SWB in terms of performance and required computational resources.

Less than 10 times real-time decoder

The system proceeds in two stages. The first-pass decoding uses gender-dependent models according to the labels provided by the segmentation/clustering step.

The most likely transcription is used for MLLR adaptation. Block-diagonal matrices (3 blocks) constitute the affine transformation. Regression classes were allocated to silence (1 class), vowels (4 classes), and consonants (2 classes). In degenerate cases, we reduced the number of classes to 5, 3 or 1, according to the number of frames (see Table 8.7).

To speed up search, only words hypothesized during the first pass decoding were allowed in the second pass. There were about 400 words per audio cut. The second pass is exactly the same as the first pass, but uses adapted models.

Real-time decoder

The faster system also proceeds in two stages. The segmenter is the same as for the previous system. The first-pass decoder is truncated to 2 to 5 seconds until a minimum amount of true speech is found. Block-diagonal, global MAPLR adaptation on speech only is applied on these words. The prior was the identity matrix.

The second pass runs with adapted models.

8.3.6 Development / Evaluation Results

We present evaluation results in this section. The system has been updated since. Improvements as of mid-June are reported. The nature of the updates is not algorithmic. It is due to a better handling of compounds in the lexicon and in the text normalization. Acoustic training was iterated.

I refer to the submitted system as sys34, and the updated system as sys61. Table 8.8 shows results for the faster than 10 times RT. This time, the evaluation data was not annotated for focus conditions. (Previously, NIST would differentiate between F0/F1, which is high quality speech in planned and spontaneous mode, and other conditions, which include speech over the telephone, music, etc.) hub4-98 was our development set. It corresponds to the test set of the NIST evaluation held in 1998. The results were better for the evaluation

System	WER (RT02)	WER (hub4-98)
LIMSI	12.7%	N/A
PSTL(sys34)	20.1%	22.5%
PSTL(sys61)	19.5%	21.5%

Table 8.8: BN: limited resources (10x RT)

test set. This might be due to the LM, which was trained on TDT. We carefully removed all the December 1998 data from the training epochs, but the test was still drawn from the TDT corpus.

The faster than real-time system is presented on Table 8.9. The speech recognition results (Speech-to-Text / STT) results are significantly worse than those of LIMSI. However, it can be estimated that they are not wildly worse than a typical 1999 evaluation system.

System	WER (RT02)
PSTL(sys34)	23.7%

Table 8.9: BN: limited resources (1x RT)

3268.95	Total speech to be segmented (in seconds)
2652.99	Correctly segmented speech (in seconds)
615.96	Incorrectly segmented speech (in seconds)
0.19	SEGMENTATION ERROR (sys34)
0.036	SEGMENTATION ERROR (sys61)

Table 8.10: BN: Meta Data Results (frame error)

We were the only ones to participate to the Meta Data (MD) evaluation on BN (see Table 8.10). The error rate of 19% was reduced in further experiments to about 4%. Note that NIST changed the scoring algorithm. It is now more pessimistic but still realistic. The previous methods consisted of computing all pairing of reference speaker labels to putative speaker labels. This algorithm is factorial in the number of speakers and diverges rapidly beyond 2-3 speakers. On average there were about 9 speakers in this set. Instead of the optimal method, NIST selected a greedy algorithm that would match a reference speaker label to a hypothesized speaker label if the overlap duration is the greatest. This is penalizes over-detection of speakers even more. We present no results on hub4-98 because the sets are artificial concatenations of 10 min duration excerpts.

For the updated result, we corrected “Dave Bird” to “David Bird”, which afforded about 1% absolute error reduction.

8.4 Wall Street Journal

The Wall Street Journal (WSJ) database was the first amongst LVCSR tasks to be carried out by NIST. It consists of read speech from WSJ articles. As with BN, training was

Training data	SI-284
Frontend	MFCC, CMS
Acoustic models	triphone, word-internal
Language models	trigram
Gaussians	96k
Real-time	2.0
WER	10.5%

Table 8.11: Results on Nov92

released in two parts. They consist of so-called long-term and short-term speakers. Long-term speakers may be used for SD modeling. Most sites prefer using only short-term speakers for training. The first released part of WSJ data is called SI-84 because it contains 84 speakers. SI-84 contains about 12 hours of speech. The second released part of WSJ is called SI-200, and contains 200 additional speakers. It is worth 60 hours of short-term speakers. It is commonplace to carry out experiment on either only SI-84, or both training parts, referred to as SI-284. Except for the early developments results, most site run training on SI-284.

8.4.1 Parameterization

The frontend is very similar to Section 8.3.1. MFCC parameters, with delta and acceleration parameters are computed over the same window. The difference lies in the way Cepstral Mean Subtraction (CMS) is applied. In WSJ, the cepstral mean is computed over each utterance, including both speech and silence.

8.4.2 Acoustic training

Our WSJ system was trained on SI-284. The TIMIT database was used to bootstrap the forced-alignment. A full system was trained on these utterances. About 20% of the database was then removed from the training using a frame-based and utterance log-likelihood threshold. Amongst rejected sentences we discovered some incorrectly recorded data (sentences repeated twice, or additional disgruntled comments appended). After some passes of training and segmentation, we decided to incorporate rejected data again. About 700 sentences were still rejected due to low likelihood. There were about 37500 sentences.

Again, we exercised the strategy of Section 8.2.2. Triphones were trained with 1400 mixtures, split to 128 Gaussians per mixture, and merged down to 98304 Gaussians (96k). From there on, splitting, and merging back to 96k was repeated a few times.

8.4.3 Decoder

The decoder proceeds in one pass only. Alternatively, in self-adaptation experiments, we would apply a second pass identical to the first one, except for adapted acoustic models.

The lightweight decoder runs in 1.3 times real-time with about 64k Gaussians. At that speed we process 10000 active hypotheses per frame. This stands in sharp contrast with other decoders, which process typically 10000 hypotheses in comfortably more than 4 times real-time. This is mostly due to an optimized implementation. Also, the choice of word-internal triphones simplifies the task of the decoder.

Our baseline system was tested on the Nov92 evaluation test set. There were 333 sentences.

System	Unit Accuracy
CI models, PLP (static + delta)	60.94%
CI models, MFCC (static + delta)	63.72%
CD models, same MFCC, 700 leaves, 16 Gaussians per mix	72.34%

Table 8.12: Phoneme accuracy

8.5 TIMIT

The TI-MIT database, or TIMIT can hardly considered a large-vocabulary task. I have used it in early developments, as a springboard to WSJ. There are 462 speakers in the training set (325 males) and 169 in the test set. Each speaker pronounces 8 sentences of a length of about 2-7 sec each. All subjects read speech with a high-quality Sennheiser microphone. The sampling rate was 16 kHz. Sound files are compressed with the SPHERE pack format.

The standard benchmark is the so-called Kai-Fu-Lee phoneme recognition task. The recognizer is evaluated by its phonemic output. Language model is allowed at the phoneme level. Since sentences are selected from a finite, well-known set, language modeling should be kept to the minimum. The aim of TIMIT is to benchmark pure acoustic modeling. There is one hour of manually labeled speech.

We use MFCC parameters, with static parameters and their first derivatives. They are computed as in WSJ. Context dependent triphones share a total of approximately 700 leaves. A Turing-Good, phoneme bigram language model was built atop the acoustic training data. Recognition is based on 48 phonemes, but the scoring down-samples the phoneme set to 39 phonemes. The decoder is a Viterbi word graph search.

Typical results are reported on Table 8.12, in unit accuracy. Best results are given with at a 0.05% interval, with 95% confidence, using the Doddington rule [Por97].

Chapter 9

Large Vocabulary Decoder

In choosing to study adaptation in large vocabulary continuous speech recognition (LVCSR) tasks, we made the implicit decision to develop such a system. The most time consuming part was the decoder.

In this chapter we review the architecture and specificities of our decoder.

9.1 Introduction

As underlined previously, the more technically interesting configurations occur in state-of-the-art LVCSR systems. Due to the difficulty of the task, the systems are in essence rather complex. A major pole of complexity of the system is due to the search. The Table 9.1 shows the approximate proportions in lines of C code in our system devoted to respectively training, decoding, and support components as of October 2001. The miscellaneous components include speaker adaptation, decision tree clustering, various general purpose mathematical algorithms, and data structures.

Conceptually, the decoder, called EWAVES, is quite simple and will be exposed in the remainder. We begin with a general description of the architecture, followed by a more detailed description of each component. The decoder uses two passes. The second pass may be repeated several times with low cost. We can distinguish two components in the first pass: the search algorithm, and the language modelling structure. The second pass re-scores an N-Best list and is an isolated version of the word-internal search algorithm.

9.2 Architecture

The figure 9.1 shows the general architecture of the decoder. Given the paucity of resources allocated to this project, we aimed for a simple, yet flexible architecture. The first pass, shown on Figure 9.2, includes trigrams by default, with word-internal context-dependent phones. The search space is organized to accommodate trigrams, but was extended to n-grams at various levels of sub-optimality. It is possible to use pentaphone models without

Module	Lines of code
Decoder + Frontend + Audio	70k
Training	30k
Support	50k
Total	150k

Table 9.1: Lines of code allocated to each component

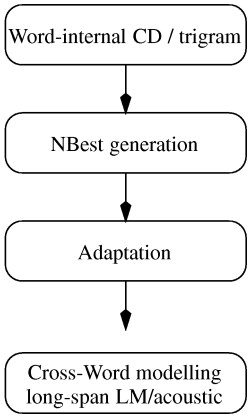


Figure 9.1: Architecture of the decoder

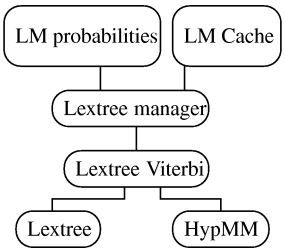


Figure 9.2: Modules in the decoder: LM probabilities and LM Cache are higher-level entities. The lextree manager creates and destroys hypotheses spaces associated to a bigram history. The lextree Viterbi performs Viterbi within a bigram-conditioned space. The Hypotheses memory manager (HypMM) manages state hypotheses.

increase in complexity.

The N-Best generation is a beamed backwards Depth First Search (DFS). Adaptation can be performed between stages. At this point, we may switch to cross-word models, gender-dependent, rate-dependent, VTLN-warped, segmental CMS, and optimal longer span models.

Because of the trend in growing computational power, and the real-time requirements of many practical applications, a lot of complexity is allocated to the first pass. For simplicity, we delayed cross-word modelling to the second pass. Long-span ngrams and pentaphones may be applied in the first pass.

9.3 Search algorithm

In this section, we will describe the within-word search only. Since we have no cross-word contexts at this stage, it is almost the same as an isolated search. We chose a lexical-tree based time-synchronous dynamic programming. The originality of our algorithm resides in the definition of a total order among hypotheses. It is not the same as a structural optimization (for instance usage of a lextree instead of linear lexicon). The algorithm operates without loss of optimality.

For the rest of the explanation, we shall be only concerned in strictly *left-to-right* topologies. It is possible to extend the algorithm to directed acyclic graphs (DAG) with an additional cost proportional the maximum number of outgoing transitions from an emitting state. Intuitively this can be interpreted that since each state can only activate one node at each frame, the space of hypotheses propagates at a controlled speed. In practice, rare is the case when one needs to use loops or even skip transitions.

9.3.1 The lexical tree

For the first pass, there is only one static lexical tree. It is defined *a priori* before the search begins. It can be compacted for unigram and bigram LM lookahead factorization.

While most sites use phoneme-based prefixes, we opted for state-based lexical trees. This allows us to instantiate all needed in-context allophones during the generation of the tree and then discard all data related to decision trees until the next pass. Our lexical trees will be naturally smaller than allophone-based lexical trees.

Definition 2 (Parent and Child Nodes) *If a node p has a non-zero transition to another node $c \neq p$, we call p the parent node of c, and c a child of p's. All nodes except the root node have exactly one parent. A leaf has zero children.*

Definition 3 (N$^{\text{th}}$ order offspring) *The set of all children of a node are called first-order offspring. The set of all children of all N^{th} order offspring are called $(N+1)^{th}$ offspring.*

Definition 4 (Level) *A level N is the set of all N^{th} offspring of the root node.*

Note at this point that there is no prevalence defined between children of the same parent. In fact, from the search point of view, there is no distinction at all between any permutation of children. It is possible to modify the algorithm to ensure that children will be processed in a specific order, for instance to maximize memory caching.

Definition 5 (Lateral lists) *A* lateral *traversal list is a list of nodes to be traversed. All nodes belong to the same level. It is termed thus to differentiate from* hierarchical *traversal which crosses level boundaries.*

9.3.2 Viterbi Search

We will proceed as follows:

- Introduction of Viterbi search
- Array-swapping: a simple Viterbi implementation
- Piggy-backed array: a refinement for DAG HMMs.
- Active envelope: an optimal traversal of hypotheses

The crucial improvement of our algorithm lies not in the structural organization of the search space, but rather in the definition of a total relation of order in the search space. As noted by many authors, the search space explored during Viterbi search is in practice several order of magnitude smaller than the entire potential search space. In other words, the actual search space is very sparse.

In many algorithms implemented in state-of-the-art systems, this implies conceptually that one runs very frequently into representation problems. They can be summarized as asking the question, "what is in my search space? Have I explored this node yet?"

The fact is that algorithms work with a *pool* of hypotheses spread on a network. Searching for a hypothesis in this pool is like search in a bag of stones for a specific stone: the entire bag must be reviewed. Our solution is simple. First, we distinguish smaller sets of equivalence within this pool. These sets are equivalent for the search algorithm. These sets are simply hypotheses in lateral lists. We have a partial order. Then, we complete the order by defining an order within these lists, so that we now have a total relation of order.

The algorithm will review hypotheses religiously in sequence. By construction, we guarantee that it will lookup and create hypotheses following the same relation of order. Therefore, the pool of hypotheses can be encoded as a simple linked list or array. Insertion and lookup of a hypothesis is $O(1)$ per hypothesis, and $O(H)$ in total, where H is the average number of hypotheses per frame. H is also called the *search effort*. Given the premises, it is impossible to design a faster algorithm. All other algorithms with distinct cost are suboptimal.

Viterbi Algorithm

The notorious Viterbi Algorithm [Vit67] is a dynamic programming alignment on a trellis. Its most striking property is that the search can be summarized and built incrementally using the so-called *recursion equations*. Again, we leave out the problem of language modelling for later.

The general layout of the algorithm is shown on Figure 9.3. The words w_1 and w_2 share the same first two states and therefore, given the lexical-tree organization, search is shared. The DP alignment procedure specifies that sufficient statistics for maximum-likelihood ending score can be compacted in the list of active hypotheses at the end of the utterance. Moreover, each list of hypotheses can be generated using recursive equation over time. The list of hypotheses at any time t is $\alpha(t)$. Given our topology, the recursion is simple. Figure 9.4 shows why by zooming the general layout graph. At each point of the column, we only need to look at the immediately following row. In our terminology, only offspring of order 1 are explored.

The direct implementation of the algorithm leads to the so-called array-swapping algorithm.

Array-swapping algorithm

A list of hypotheses $\alpha(t)$ is kept in a linear array. The size of the array is equal to the number of states in the lexical tree. For the recursion, we use the so-called backwards

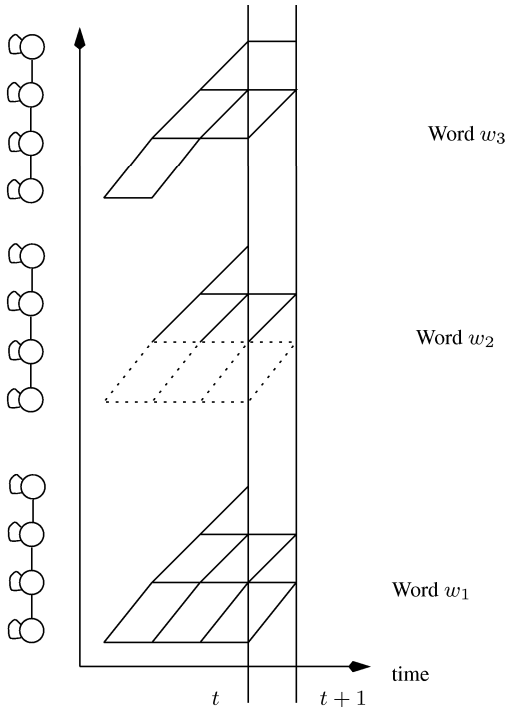


Figure 9.3: General layout of the Viterbi Algorithm

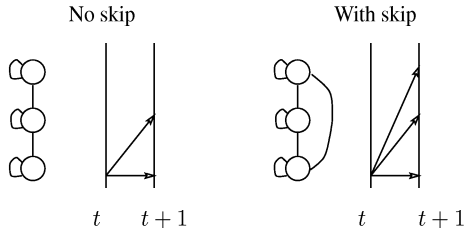


Figure 9.4: Activation without skip transition

recursion equation of the Viterbi algorithm. An array $\alpha(t+1)$ is created empty. The entire list of states is traversed, and for each state k, we find the parent state of the node, called k^*. Inactive states are assigned a score equal to $-\infty$. We assume that scores are log-likelihoods and log-probabilities. The score of k at time $t+1$ is defined by its own score at the current time and the score of its parent:

$$s_k(t+1) = \max\{s_k(t) + l_k, s_{k^*} + i_k\} + d_k(t) \tag{9.1}$$

where:

l_k: the log-probability of the self-transition (loop) from k to k

i_k: the log-probability of the incoming transition k^* to k

$d_k(t)$: the acoustic match $\log p(o_t|k)$.

Since $\alpha(t) \rightarrow \alpha(t+1)$ forms a Markov Chain, at any time we only need the two arrays. For the next time $t+2$, $\alpha(t+1)$ is used in lieu of $\alpha(t)$. The other array $\alpha(t)$ is cleared and can be readily used for $\alpha(t+1)$. For that reason the method is called array-swapping.

Clearly, it is expensive because of the memory requirements. Space must be allocated for the potential search space. On the other hand, lookup and insertion of a hypothesis is $O(1)$. The algorithm is intolerable for LVCSR but optimal. It has two other drawbacks:

- backwards lookup: for each node in consideration we need to find its parent. Of course, this can be hashed.
- backwards recursion: this is a corollary of the previous statement. In practice many states in the trellis are not active, because their parents are not. Supposing that the backwards lookup problem is solved, in theory, we still need to examine all states of the array and look backwards to see if they can be activated;.

We would like to improve the algorithm to only use one array, or hypotheses list, and to look *forward* from existing hypotheses, instead of looking backwards from potential hypotheses.

Forward recursion

The first observation that comes to mind is that $\alpha(t)$ is very sparse. In many an instance of the potential right column $\alpha(t+1)$, parents will have zero likelihood, leading to a waste of time. It is possible to use the forward recursion to construct $\alpha(t+1)$ while traversing only active hypotheses of $\alpha(t)$. This is called the forward recursion. For each node k, we list all children c. The following equation holds:

$$\forall c,\ s_c(t+1) = \max\{s_k(t) + i_k, s_c(t) + l_k\} + d_k(t). \tag{9.2}$$

Unfortunately, this requires the use of a lookup function for $s_c(t)$. It is not known whether the score of c is going to be used further down in the hypotheses list, and thereby impossible to use only one array if no other information is added.

In the introduction, we talked about $\alpha(t)$ as the pool of hypotheses. Now we are going to define small clusters in this pool.

Piggy-backed array

We define a partial order of the hypotheses of $\alpha(t)$ to avoid the problem. From Figure 9.5, we see that it corresponds to traversing each pool of later lists from top to bottom.

By construction, we can override the score of the child c immediately. The algorithm ensures that we have traversed c before and will not use the score again in the same Viterbi step. To avoid another pass through the data, we will modify the beam. Let us define:

$$r_k(t) = s_k(t) - d_k(t); \tag{9.3}$$

$$\tilde{r}_k(t) = r_k(t) + l_k, \tag{9.4}$$

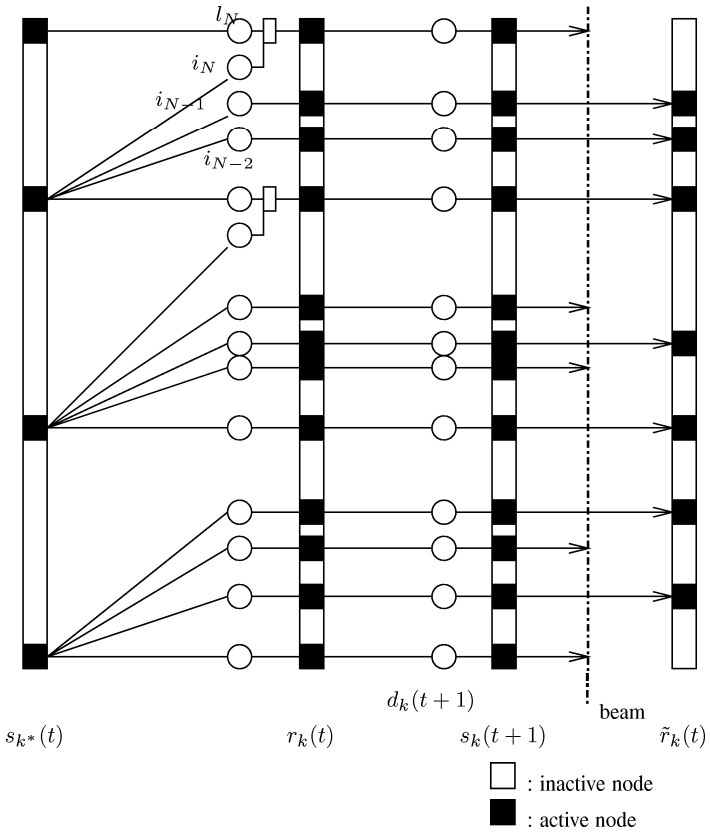


Figure 9.5: Array swapping algorithm: levels

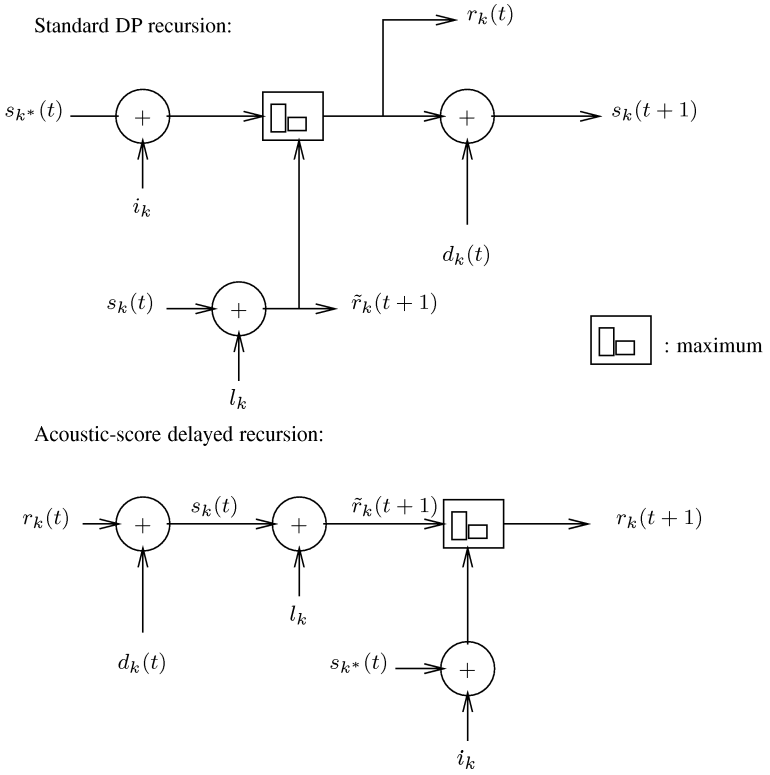


Figure 9.6: Delayed acoustic scoring

which are the topological score and partial topological score respectively. The topological score corresponds to the score at time $(t+1)$ but without the acoustic match. The partial topological score is the same, but with the self-transition. It is more convenient to apply the acoustic score at the beginning of the Viterbi step (V-step), as shown on Figure 9.6.

The follwing forward algorithm is applied to each node k:

1: Acoustic match: $r_k(t+1) \leftarrow s_k(t) = r_k(t) + d_k(t)$.
2: Bequeathal:
3: **for all** children c of k **do**
4: $r_c(t+1) \leftarrow \max\{r_k(t+1) + i_c, \tilde{r}_c(t+1)\}$.
5: By hypothessis, the score $\tilde{r}_c(t+1)$ was either set in line 8, or equals $-\infty$.
6: The bequeathal score $r_k(t+1)$ was computed on line 1.
7: **end for**
8: Self-activation: $\tilde{r}_k(t+1) \leftarrow r_k(t+1) + l_k$.
9: Implicit: $r_k(t+1) \leftarrow \tilde{r}_k(t+1)$ unless seen again in line 4.

Since we have proven that a single array was sufficient, we shall drop the time index. For more performance, we apply the self-activation directly, so that the node k can disappear from the memory cache immediately. We will define a score field, α_k to each node k, which will hold in turn $r_k(t), r_k(t+1)$, and $\tilde{r}_k(t+1)$. Therefore, for each cell k, we summarize the algorithm in Table 9.2.

Application of the acoustic match takes place at the beginning of the iteration. It was moved in order to avoid a full pass on the hypotheses at the end of the iteration. A full pass through the search space costs 5% to 10% of the total search time (including acoustic match) on Switchboard. It cannot be done during the pass because while traversing each cell it is unknown whether the cell score is going to be used again in a later bequeathal.

1: Temporary variable $v \leftarrow \alpha_k + d_k(t)$.
2: Self-activation: $\alpha_k \leftarrow v + l_k$.
3: **for all** children c of k **do**
4: $\quad \alpha_c \leftarrow \max\{v + i_c, \alpha_c\}$.
5: **end for**

Table 9.2: Compact piggy-backed Viterbi

Figure 9.5 shows why the beam is slightly different from standard Viterbi beaming.

So far the children were traversed in any order. They are located in independent search spaces and thus the order is not total. Most decoder architectures stop at this point. If we look carefully at the equations, we will see that the bequeathal process requires the lookup of a hypothesis in $\alpha(t)$ for each child of k. Generally speaking, this means that a linear search through all hypotheses hitherto processed, and hence $O(H)$ for each hypothesis. In total this sums to $O(H^2)$. The active envelope list reduces the cost to a linear function $O(H) < O(H^2)$.

Active Envelope

We define an ordering of the children in the same lateral pool. It is arbitrary and therefore can be assumed to follow the order in the data structure of the lexical tree. Defining this order *within* lateral lists defines an order at each level and thus a total order relation on $\alpha(t)$.

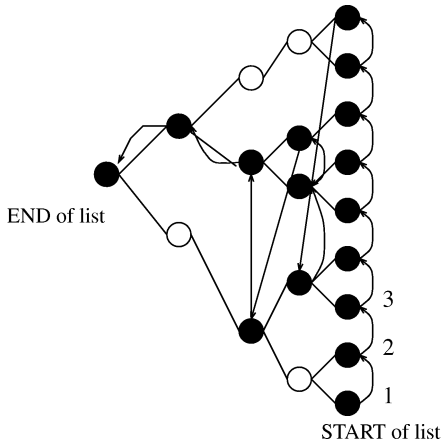
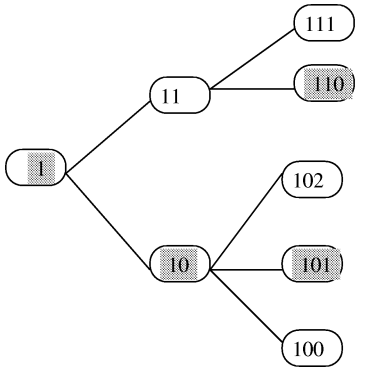


Figure 9.7: Z-traversal of the lexical tree

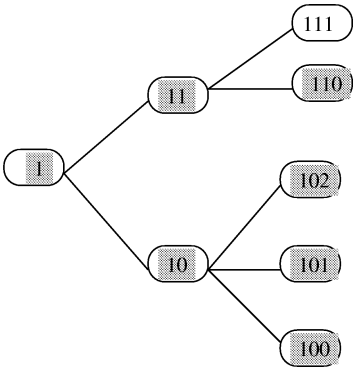
We modify the algorithm to process children in order. By construction, each level of the tree is processed in reverse order. Since offspring are traversed in increasing sequence, the traversal of hypotheses follows a Z-shape, as shown on Figure 9.7.

If we examine the lookup, by induction we can prove that traversing the parents in order insures that children are also going to be inserted in the list following the total order. Lookup is also done in the same order. Therefore, it is sufficient to keep a cursor on $\alpha(t)$ for the activation-bequeathal, and another one for the lookup. Since we proved that inheritance occurs only once, each of the lists is traversed exactly once. The lookup occurs in $O(1)$ per cell, or $O(H)$ in total. This is the minimum achievable cost, which is proportional to the array-swapping method.



L0	L1		L2A		L2B	
1	10			101	111	

(a) Active hypotheses list at t



L0	L1		L2A			L2B	
1	10	11	100	101	102	111	

(b) Active hypotheses list at $(t+1)$

Construction of L2 from L1:

Current Level list is $\alpha = \{\mathbf{101}, 111\}$.
for all hypotheses in L1 $\equiv \{10\}$,
for all children of $10 \equiv \{100, 101, 102\}$:
Activate 100: new list: $\{\mathbf{100}, 101, 111\}$
Lookup 101: since it exists, Viterbi. New list: $\{100, \mathbf{101}, 111\}$
Activate 102: new list: $\{100, 101, \mathbf{102}, 111\}$
Added list: $\alpha_a = \{100, 101, 102\}$
Initial list: $\alpha_i = \{101, 111\}$
Final list (sorted $\alpha_a + \alpha_i$): $\alpha = \{100, 101, 102, 111\}$

(c) Construction of Level 2 (L2A & L2B)

Merging sorted lists:

Merge list $\alpha_a = \{100, 101, 102\}$ into $\alpha(t) = \{101, 111\}$
Add virtual node $-\infty$ to $\alpha(t) = \{-\infty, 101, 111\}$
Define: Next point $n = 101$; Insertion point $i = -\infty$.
for each element a in $\alpha_a = \{100, 101, 102\}$
$a = 100$:
Since $a < n = 111$, insert 100 after i, and let $n \leftarrow 100$.
$\alpha(t) = \{i = -\infty, n = 100, 101, 111\}$.
$a = 101$:
While $a > n$, let $i \leftarrow n$, and n be the next element in $\alpha(t)$.
Since $(a = 101) = (n = 101)$, apply Viterbi to merge hypotheses.
$\alpha(t) = \{-\infty, i = 100, n = 101, 111\}$.
$a = 102$:
While $a > n$, scroll the pointers i and n along $\alpha(t)$.
Since $a < n$, insert a between $i = 101$ and $n = 111$ and let $i \leftarrow a$.
$\alpha(t) = \{-\infty, 100, i = 101, n = 102, 111\}$.
The final list is: $\alpha(t+1) \leftarrow \alpha(t) = \{100, 101, 102, 111\}$.

(d) Illustrated merging algorithm

Figure 9.8: Merging hypotheses list

The construction of $\alpha(t+1)$ from $\alpha(t)$ is equivalent to the merging of two ordered lists, which is done in $O(H)$, thanks to the total order. Figure 9.8 illustrates the process. We show the search space in parts (a) and (b). The Viterbi search is shown in part (c): it is an application of the algorithm of Table 9.2. More detailed about the merging algorithn are laid out in part (d). Part (d) must be written with care, handling cases when there are gaps in the sequences. The new sequence $\alpha(t+1)$ is built in-place, by inserting elements into $\alpha(t)$. As shown in part (c), the merging algorithm of part (d) must be blended with the

Viterbi search.

9.3.3 Language model structure

Now that we have reviewed the search *within* a word, we can describe the structure of the entire search. We follow Ney's Viterbi decoder structure. We keep (conceptually) one tree per word history $p(\cdot|w_2w_1)$. This is depicted on Figure 9.9. The language modelling module can be seen as a higher-level search management in which all nodes are represented by lexical trees given bigram history. Nodes are very strongly connected. They do not form a DAG at all. The trees are very sparsely populated. There is only one topological

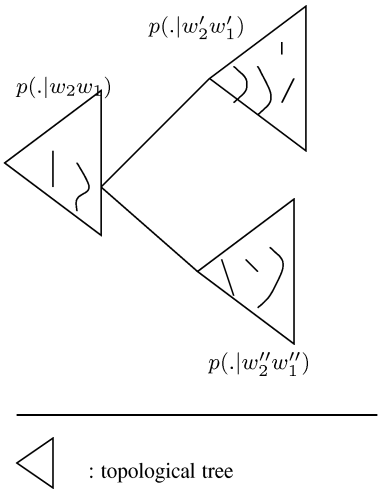


Figure 9.9: Tree copies

tree, defining the lexical tree structure. All tree copies store hypotheses in the compact representation of the active envelope list.

Tree copies

The intra-word search is done using EWAVES. Once a terminal node is reached, the search space has hit a word w_3. If we were in a tree copy of history (w_2, w_1), then the root hypothesis of the tree $p(\cdot|w_3w_2)$ is activated with the cumulative score of our hypothesis plus $p(w_3|w_2w_1)$. There is a loss of optimality at this point. Ney calls this problem the *word-pair approximation*. The correct strategy would activate all $p(\cdot|w_3\bar{w}_1)$, by looking at all previous histories that activated $p(\cdot|w_2w_1)$. This is clearly very expensive. Ney observes that the loss practically negligible. In addition, Demuynck *et al.* note that [DDCW00]:

- Given the Katz backoff topology of the language model, all backoff nodes can share the same search space $p(\cdot|w_3x)$, where x does not matter. The backoff weight $\beta(x, w_3)$ is applied at the entrance the backoff node.
- The use of trigram results in an *increase* in complexity but is beneficial because since the perplexity is reduced, the search space also becomes smaller.

For that reason, we retain the trigram topology, but apply higher-order ngram probabilities at word boundaries whenever available. We have to be aware of the so-called *ripple* effect of a misrecognized word on subsequent words. The error analysis using higher order ngrams becomes more convoluted.

Demuynck *et al.* observe a 10% increase in speed by using the first optimization. This is understandable because only 22% of the time, backoff nodes are actually traversed in the WSJ task. For this reason we chose not to implement this feature.

Other LM issues

Lookahead (smearing) We use a bigram lookahead for the beam pruning. LM lookahead trees are factored starting from the leaf nodes backwards using dynamic programming (DP). The topological tree is compacted to a smaller tree where only nontrivial transition nodes are shown. Nontrivial transition nodes are terminal nodes (no outgoing transition) or have more than one child.

Quantization and storage All probabilities are quantized using linear quantization. We use 8 bits for bigrams and bigram backoffs, and 16 bits for trigrams. Linear quantization allows us to perform arithmetics in the quantized domain directly. This is especially attractive for processors with superscalar integer operations.

All LM probabilities are stored in a sorted array. Other decoders normally employ hierarchical models for faster lookup.

Lookahead cache Bigram lookahead trees are cached using an interpolated cost based on usage (hit counts) and last time of hit. When the cache is overflowing, we diminish the beam size.

Beaming We use a dynamic beam based on the histogram of hypotheses. At the end of each frame, we adjust the beam.

A maximum number of tree copies is maintained. When the count is reached, trees with the least best score are deleted. On WSJ, we keep 64 trees. On SWB, increasing this amount to 256 seems to be profitable.

9.3.4 Parallelization, horizontal caching and defragmentation

In the case when large amounts of computational resources are available for a low-latency or real-time decoding, it is possible, with our architecture, to distribute the load of computation amongst several processors (CPUs). Most decoder architectures are unable untangle the data flow between components of the language model, search, and distribution computation.

First, we use an optimized distribution computation algorithm that is completely independent from the search space. Second, we note that search on each tree copy is independent from one another, and only word ends must be communicated between processors.

The distribution computation algorithm is called *horizontal caching*. It is called thus because instead of computing distributions on demand, frame by frame, we compute stripes of distributions at the same time. If the time axis is horizontal and the Gaussian mixture component is on the vertical axis, then instead of computing vertically, one frame at a time, we compute *all* distributions on a horizontal strip at a time. In this way, we are able to download means and variance parameters into the processor once for many observation vectors. Even in a task such as WSJ, where only 40% of Gaussians are used, computing all of them in horizontal caching mode affords an overall 80% speed improvement. There is no loss of optimality.

The parallelization allows us to dispatch lexical trees to processors, or local processing units. The search space is local to each processor. Only word ends and word beginnings need be transmitted at the end of each frame. For best results, pruning parameters such as histograms or sufficient statistics for the histograms must be transmitted.

Finally, every 5-7 frames, we run a defragmentation of the search space. All active search hypotheses are copied onto a region of contiguous memory. This enhances the memory caching effect.

9.4 Generation of NBest candidates

The generation of the NBest list is done through beamed Depth First Search (DFS). It borrows much from BBN's forward-backward algorithm [NS97]. The first pass is done using the Viterbi algorithm as set forth above. To generate the NBest list, we need to define the partial forward and backward scores. We then define a sentence-level invariant (the posterior probability) that will allow us to prune during the search without loss of optimality.

9.4.1 Partial scores

Suppose we split a sequence of words in the middle. What are the scores up to the end of the word and from the beginning of the word on?

Definition 6 (Observation segment) *We define the observation segment corresponding to an alignment of a word sequence $\mathcal{W} = w_0, w_1, ..., w_{\#\mathcal{W}-1}$ as o_{j-1}^{j}. If t_j is the ending time of word w_j in the utterance, then we let*

$$o_{j-1}^{j} = \left\{ o_{t_{j-1}}, o_{t_{j-1}+1}, ..., o_{t_j} \right\}$$

The acoustic score of the segment is of course the likelihood $p(o_{j-1}^{j}|w_j)$.

Definition 7 (Forward score) *The forward score of a word ω_j ending at time t_j, given history $\omega_0, \omega_1, ..., \omega_{\#\mathcal{W}-1}$ is defined as:*

$$\alpha(\omega_j) = \prod_{k=0}^{j} p(o_{k-1}^{k}|\omega_k) p(\omega_k|\omega_{k-1}\omega_{k-2})$$

[Causality] It is strictly causal. The anti-causal dual is the called backward score, because we begin with the end of the utterance and play it reverse to the beginning.

Definition 8 (Backward score) *The backward score of a word ω_j ending at time o_{t_j}, given history $\omega_0, \omega_1, ..., w_{\#\mathcal{W}-1}$ is defined as:*

$$\beta(\omega_j) = \prod_{k=j}^{\#\mathcal{W}-1} p(o_{k-1}^{k}|\omega_k) p(\omega_{k+2}|\omega_k\omega_{k+1})$$

Unlike other anti-causal definitions in signal processing, we *include* the word in the score.

Trivially, the following, called *recursive update* is true:

$$\alpha(\omega_j) = p(o_{j-1}^{j}|\omega_j) p(\omega_j|\omega_{j-2}\omega_{j-1}) \alpha(\omega_{j-1}) \tag{9.5}$$

$$\beta(\omega_{j-1}) = p(o_{j-2}^{j-1}|\omega_{j-1}) p(\omega_{j+2}|\omega_j\omega_{j+1}) \beta(\omega_j) \tag{9.6}$$

We have shifted the index of $\alpha(\cdot)$ for ease of presentation of the following.

9.4.2 A sentence invariant

Consider the sentence $\mathcal{W}$. We are interested in finding the set of best matching sentences. A depth-first search algorithm (DFS) working backwards in time would work just fine. Unfortunately, it is exponential and extremely costly. Therefore, we retain the DFS idea, but define an A^*-like upper bound for the partial score. The intuition is that it is no use to pursue an utterance at end of word ω_j if it leads to a bad score. If we are able to find a sentence invariant, i.e., the score (or the upper bound that of) of the sentence, then while constructing $\beta(\cdot)$ we can "look ahead" to see whether the path is promising or not.

Definition 9 (Global score) *For a transition word ω_j, we define its forward-backward or global score as*

$$\gamma(\omega_j) = \frac{\alpha(\omega_j)\beta(\omega_j)}{p(o_{j-1}^j|\omega_j)} p(\omega_{j+1}|\omega_{j-1}\omega_j)$$

The normalizing acoustic score is present because it is accounted for in both α and β, and the multiplication LM score is called the *overlap* score, i.e., the score of the concatenation of both sides.

Theorem 1 (Invariant global score) *The global score is constant for all words of the sentence. Furthermore, it is equal to the one-sided final scores, i.e.:*

$$\gamma(\omega_j) = \gamma = \alpha(\omega_{\#\mathcal{W}-1}) = \beta(\omega_0) \tag{9.7}$$

Using the recursive update:

$$\alpha(\omega_{j-1}) = \frac{\alpha(\omega_j)}{p(o_{j-1}^j|\omega_j)p(\omega_j|\omega_{j-2}\omega_{j-1})} \tag{9.8}$$

Plugging into the definition:

$$\gamma(\omega_{j-1}) = \gamma(\omega_j) = \gamma \quad \forall \omega_j \tag{9.9}$$

which is our first claim. The second part of the theorem is also easily seen given that $\alpha(0) = \beta(0) = 1$ by definition, which completes the proof.

Now we have our invariant, that depends on the whole sentence, we will show how to find an upper bound given partial scores only. Let us recall how the best sentence was created.

Definition 10 (Optimal forward solution) *The solution $\hat{\mathcal{W}} = \{\omega_0, \omega_1, ..., \omega_N\}$ is chosen to maximize the forward scores recursively. Denote:*

$$\hat{\omega}_j(\omega_j) = \arg \max_{\mathcal{W}=\omega_0,\ldots,\omega_{j-1}} \alpha(\omega_j) \tag{9.10}$$

$$= \arg\max p(o_{j-1}^j|\omega_j)p(\omega_j|\omega_{j-2}\omega_{j-1})\alpha(\omega_{j-1}) \tag{9.11}$$

$$\approx p(o_{j-1}^j|\omega_j)p(\omega_j|\bar{\omega}_{j-2}\bar{\omega}_{j-1})\bar{\alpha}(\omega_{j-1}) \tag{9.12}$$

where $\bar{(\cdot)}$ means maximized value over $\omega_0, \omega_1, ..., \omega_{j-2}$. The last approximation comes from the "propagate the first best" rule.

Since we have found and stored these values in the forward pass, the following will help us express the upper bound on $\gamma(\mathcal{W})$ based on partial scores $\beta(\omega_{\#\mathcal{W}-1}^j)$ and best partial scores $\hat{\alpha}(\omega_j)$.

Theorem 2 (Global optimum) *The maximum of the score of all optimal forward solutions at the end of the utterance is the maximum achievable score by all combinations permitted by the lattice.*

We proceed to state the following upper-bound on the left-hand side of the sentence.

Theorem 3 (Upper-bound given partial score) *The maximum achievable score over all sentences with (backwards) history $\omega_{\#\mathcal{W}-1}, \omega_{\#\mathcal{W}-2}, ..., \omega_j$ is at most the score of the forward optimum for that word ω_j (ending at that time), id est:*

$$\gamma(\mathcal{W}) \leq \Gamma = \gamma(\{\hat{\omega}_j, \omega_j, \omega_{j+1}, ..., \omega_{\#\mathcal{W}-1}), \quad \forall\, \omega_j, \omega_{j+1}, ..., \omega_{\#\mathcal{W}-1} \tag{9.13}$$

We can now come to the (fairly simple) DFS algorithm.

Algorithm 2 *Generation of the NBest list. We begin at the end of the utterance and:*

- *Set, according to our definition $\beta = 1$.*
- *Start with an infinite threshold ϑ, and our list of NBest hypotheses $\mathcal{H}$ to the empty set.*
- *Take the first node connected to β, use the recursive update to compute the projected β if we follow the link. Compute the upper-bound Γ of the partial score. If it is above the threshold repeat, else try the next node.*
- *When we reach the beginning of the sentence, score it (i.e. apply $p(\omega_0|<s>)$), and push it in $\mathcal{H}$.*
- *If we reach a given card$(\mathcal{H}) = H$, then compute the score g such that :*

$$card\,\{h \,|\, \gamma(h) > g\} = N, \tag{9.14}$$

  where N is the desired number of hypotheses, and set $\vartheta = g$. Proceed.

- *When we have exhausted the list of connected nodes, then we backtrack to the original node and expand another node.*

This is a DFS algorithm that uses the Γ pruning.

Definition 11 (Admissibility) *An algorithm is said to be* admissible *iff for all conditioning the outcome always includes the optimal solution.*

Theorem 4 (Admissibility) *Our generation of the NBest list is admissible.*

Trivial by construction. The first hypothesis inserted in the stack is always the best.

Another optimization is the in the computation of the threshold T. We model the distribution of $\gamma(\cdot)$ as an exponential function before the current maximum, i.e., if we define T as $\max_{h\in\mathcal{H}} \gamma(h)$, the pdf of a score x is approximately:

$$f(T - x) = \alpha^{-1} e^{-\alpha(T-x)}, \tag{9.15}$$

with a fixed dispersion ratio α. The expectation is $T - \alpha^{-1}$. The Q^{th} part of the distribution is $T - \alpha^{-1} \log Q$. In particular, when $Q = 2$, i.e. we set $T = 2N$ we have the median $T - \alpha^{-1} \log 2$. We can compute alpha from the expectation. The algorithm is linear time. There is no theoretical improvement in speed over the exact algorithm. Nonetheless, We avoid the computation of the histogram.

9.5 Recognizer speed

In this section, we measure how much time is spent during a decoding pass. Measuring time spent in each module is not a trivial task. For most measurements, we adopted the following strategy: in the code, replicate N times the computation for a certain component; then, the different between this new code and the baseline is counted as approximately N times the time spent in the baseline system. Different decoders, compiled with different replicated codes, are compared. The final result is averaged: it is the solution of an overdetermined system of equations. Each decoder is repeated 5 times over 5 examplary utterances. The same machine ran all tests. It is a Pentium-IV, 1.5 GHz system, with 1 GB of RDRAM. It is running Linux and all code was compiled with the best combination of optimization flags with the Intel Compiler.

Despite the very expensive measuring procedure, we do not guarrantee more than 0.1 x RT precision at 95% confidence.

	WSJ	BN	SWB
Total decoding time	**1.3 x RT**	**4.8 x RT**	**8.5 x RT**
Total Likelihood computation	**0.9 x RT**	**3.9 x RT**	**5.1 x RT**
Gaussian likelihood computation	0.8 x RT	2.8 x RT	3.4 x RT
Number of Gaussians	64000	192000	256000
Mixture recombination	0.1 x RT	1.1 x RT	1.7 x RT
Number of Mixtures	1404	2542	3889
Estimated total search	**0.4 x RT**	**0.9 x RT**	**3.4 x RT**
Search effort (Hyps per frame)	9.8×10^3	3.6×10^4	1.4×10^5

Table 9.3: LVCSR Decoder statistics

Table 9.3 shows timings for different LVCSR tasks. We measure the total time spent in user mode. The likelihood computation can be broken down into two parts: computation of each Gaussian component, and the mixture recombination. The mixture recombination weights the score of the Gaussian likelihood and adds all weighted scores. Since scores are stored in the log-domain, we use a maximum approximation instead of a sum of exponentiated scores. We measured the search effort in number of state hypotheses per frame.

We usually think of WSJ as a small task with good acoustics and predicatable language. BN has more challenging acoustics, but the language model is still reliable. SWB has relatively easier acoustic conditions, but a low sampling rate. The challenge in SWB is the language model. This is reflected in a significantly larger search space for SWB. The search space in BN is markdely more controlled. However, BN acoustics are considerably more complex than WSJ. Nonetheless, we see that timings are directly proportional to the search space and number of Gaussians. We have therefore verified that EWAVES processing is proportional to the search space.

Estimated total search	**0.9 x RT**
Histogram pruning statistics	0.15 x RT
Defragmentation	0.14 x RT
Minimum traversal cost	0.1 – 0.14 x RT
LM factorization (cached)	0.14 x RT
LM lookahead cache size	300 trees / 34 MB
LM lookahead cache hit	$> 98\%$
Estimated time for Viterbi	0.4 x RT

Table 9.4: Breakdown of search cost for BN

On Table 9.4, we try to break down the cost of the search effort. They are very approximate. The decoder employs histogram pruning: the beam is adjusted so that an approximately constant maximum amount of state hypotheses can be processed per frame. For this, we need to compute the histogram, or the number of frames associated with each score. This computation is proportional to the size of the search space. We estimate that traversing the search space to *count* the hypotheses takes between 0.1 to 0.14 x RT. This is the minimal traversal cost. We also measure the time spent in LM lookahead: it is mostly due to factoring the bigram scores, and interpolating with the backoff unigram tree. Since the factored trees are cached, factoring is relatively inexpensive. The defragmentation of the search space is ran every 7 frames, and copies the search space from the fragmented area, into a defragmented area.

Finally, EWAVES is estimated to account for less than 50% of the total search time, or 9% of the total system time. This includes score combination and comparison, and construction of the search space for the next frame. Those comparisons imply branching (conditional execution), which are known to be the slowest operations for a processor. Given that the minimal traversal time of the search space is at least 0.1 x RT, EWAVES spends approximately four times this amount.

Estimated total search	**0.9 x RT**
Search effort (Hyps per frame)	3.6×10^4
Number of clocks / second	1.5×10^9
Clocks per hypothesis	500

Table 9.5: Average clock cycles per hypothesis

On Table 9.5, we estimate that processing one state hypothesis is equivalent to about 500 processor clock cycles. Any memory operation, such as reading from the memory, loading instructions, etc, will take at least 3 clock cycles. For reference, the square root operation in float uses 70 clock cycles. The least expensive operation is the addition with 4 clock cycles once operands are loaded. The processor works at 1.5GHz while the memory can deliver 64 bits at 400MHz in optimistic conditions. We believe that the cost of EWAVES is minimal.

9.6 Summary

This concludes the description of the decoder. We have a fairly standard architecture except for the following points:

- a fast Viterbi implementation called EWAVES (Section 9.3),
- trigram topology of the search space and application of higher-order ngrams in the first pass (Section 9.2),
- word-internal context-dependent only (implies only one pre-computed lexical topological tree),
- a fast distance computation, called horizontal caching (Section 9.3.4), and
- a parallelizable architecture of the search,

For simplicity, we opted for an NBest re-scoring second pass. Most state-of-the-art decoders avail themselves of word graphs instead. In many systems, we just run the first pass decoder with adapted model.

Part IV

Conclusion

Chapter 10

Final word

10.1 Short summary

We have attempted to address three aspects of speaker adaptation for speech recognition. In the first part, we lay the framework for modeling HMM mean parameters. Regression via ML and least-squares are unified. A discriminative model dimension reduction is presented. A non-linear extension to the regression is also explained. Additionally, we extend the current state-of-the-art feature transformation mathematical apparatus with a closed-form solution for triangular matrices. A Bayesian formulation of the diagonal elements is discovered.

In the second part, we study more application-oriented aspects. The EM formulation is applied to supervised and unsupervised adaptation. A new approach using Eigenvoices locations yields a clustering for confidence-based unsupervised adaptation. The interaction of speaker and noise adaptation is further investigated.

In the third and last part, we describe our large-vocabulary systems. We explain into details the main component: an efficient Viterbi decoding algorithm. It is based on the definition of a total relation of order amongst state hypotheses.

10.2 Achievements

We review again what new elements are explored in the three sections.

Theory: We establish the link between divergence, ML, and least squares in the case of HMM model parameters.First, we remark that the distribution of model parameters is Gaussian. Then, we see that with an appropriate normalization, the likelihood criterion coincides with least squares, which gives a justification for using SVD. Then, we cast the MMIE gradient descent equation as a quadratic form optimization through the MAP framework. Finally, we suggest a non-linear extension to PCA regression via piece-wise linear regression. Estimation formula are given for all cases. Furthermore, we give a closed-form solution for triangular feature transformation. This allows us to select the solution closer to the identity amongst a number of possible transformations. Moreover, we can also find a Bayesian compensation formula.

Applications: Supervised and unsupervised adaptation with MLLR and MAP are studied with sausage NBest decoding. The unsupervised adaptation is improved with a cluster-based rejection. Noise and speaker adaptation are separated. To our knowledge, we were the first to examine joint adaptation to noise and speaker simultaneously.

Evaluation: I have initiated the large vocabulary effort in PSTL. I have designed and implemented a large vocabulary decoder. It achieves the maximum achievable theoretical speed without loss of optimality. In practice we estimate that it can process about 10 times as many hypotheses than a conventional decoder, not only because its algorithmic optimality but also because of implementation.

Most of these contributions, along with joint speaker-noise adaptation and speaker recognition, were also set forth in a number of publications ([KNJ+99, NRJ00, NWJ99, KJNN00, NRMJ02, NRK+01, TKNJ00, NRJW02, RNKJ01, PKNJ01, KPN+01, SRN+02, NGJC99, NKJ+99, NWK+00, NRWJ02, NRWJ01]).

Chronologically, supervised and unsupervised adaptation were investigated first. Then, MLES and noise adaptation were studied. A first attempt at using Eigenvoices for speaker verification emerged. Moving in PSTL implied re-engineering of PSTL's code base. At this time, the decoder was developed and LVCSR was launched in PSTL. Research on adaptation was able to proceed with self-adaptation. Model-space constraints were refined. Developments on SWB led to the LU decomposition scheme. LVCSR was given the ultimate boost for NIST's RT-02 evaluation.

10.3 Afterthoughts

In retrospect, we can always find a myriad of ideas and aspects upon which to improve.

In hindsight, the experiments seem under developed. Perhaps we should have restricted experimental framework to fewer items. The amount of work required to develop the large vocabulary systems was probably overwhelming. It is possible also that I could have started the large vocabulary effort earlier in the thesis, or concentrated the thesis around large vocabulary decoder and system development. I may have been too greedy or diluted my efforts with too many directions.

As for the adaptation, results are very encouraging and sometimes even bring considerable improvements. However, the costs of deploying the techniques, as well as the instability of research features, prevented me from incorporating promising elements into the RT-02 submissions. Piece-wise linear models bring an improvement, but taking into account the increased resource requirements, they seem to be premature now. There were many issues left unsolved: non-uniqueness of estimates, efficient training, and initial partitioning. Piece-wise linear regression was designed in the spirit of generalizing the concept of gender-dependent, condition-dependent modelling. It is comparable to the unlimited modeling power of context-dependent phones: when more data becomes available, it is possible to increase the length of the context. The generative potential of piece-wise linear regression is also asymptotically unlimited. When even more data is available, piece-wise linear might make a difference. We shall wait until then. Unfortunately, variance adaptation is largely ignored in this thesis. The Eigenvoices framework is notoriously mute about variance adaptation. There is an enormous unexploited potential there. Self-adaptation using clustering also seemed interesting but did not bring much improvement. The root modulation, although more correct theoretically, does not outperform the standard eigenvoices as much as expected. The Bayesian extension to the LU feature decomposition was not tested thoroughly.

10.4 Conclusion

In closing, I would like to state the main contributions of this thesis once again. They range from simply theoretical modeling of HMM parameters to decoder development. Perhaps the achievements about which I am most enthusiastic in these areas are exposed in the theoretical section. In particular, the link between ML, least squares, and MMI are fully

exploited in the context of speaker modeling by MLLR matrices. In the applications, we have been pioneers in joint noise and speaker adaptation. A complete LVCSR system was built from scratch, including a new decoder, so that adaptation could be validated on real tasks.

Bibliography

[AB98] T. Anastasakos and S. V. Balakrishnan, *The Use of Confidence Measures in Unsupervised Adaptation of Speech Recognizers*, International Conference on Spoken Language Processing (ICSLP) (Sydney, Australia), vol. 5, Dec. 1998, pp. 2203–2306.

[AH98] M. Afify and J.-P. Haton, *Minimum Cross-Entropy Adaptation of Hidden Markov Models*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (1998).

[Ama99] S. Amari, *Information geometry on hierarchical decomposition of stochastic interactions*, 1999.

[AMSM96] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul, *A compact Model for Speaker-Adaptive Training*, International Conference on Spoken Language Processing (ICSLP) (1996), 1137–1140.

[AS96] S. M. Ahadi-Sarkani, *Bayesian and Predictive Techniques for Speaker Adaptation*, Ph.D. thesis, University of Cambridge, Cambridge, UK, Jan. 1996.

[Bac00] M. Bacchiani, *Using Maximum Likelihood Linear Regression for Segment Clustering and Speaker Identification*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 4, Oct. 2000, pp. 536–539.

[Bau72] L. E. Baum, *An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes*, Inequalities **3** (1972), 1–8.

[BBdSM86] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer, *Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Tokyo), vol. 1, May 1986, pp. 49–52.

[BBW01] H. Bourlard, S. Bengio, and K. Weber, *New Approaches Towards Robust and Adaptive Speech Recognition*, vol. 13, MIT Press, 2001.

[BHK97] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (97), no. 17, 711–730.

[Bil00] J. Bilmes, *Factored Sparse Inverse Covariance Matrices*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), vol. II, 2000, pp. 1009–1012.

[BM97] E. Bocchieri and B. Mak, *Subspace distribution clustering for continuous observation density Hidden Markov Models*, Proc. of Eurospeech, 1997.

[Bot00] H. Botterweck, *Very fast Adaptation for large vocabulary continuous Speech Recognition using Eigenvoices*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 4, Oct. 2000, pp. 354–359.

[Bou95] H. Bourlard, *Towards Increasing Speech Recognition Error Rates*, Proc. of Eurospeech, Sep. 1995, pp. 883–894.

[BT98] C. Bishop and M. Tipping, *A Hierarchical Latent Variable Model for Data Visualization*, IEEE Transactions on Pattern Analysis and Machine Intelligence **20** (1998), no. 3, 281–293.

[BW89] H. Bourlard and C. Wellekens, *Pattern Discrimination and Multi-Layered Perceptrons*, Computer Speech and Language **3** (1989), 1–19.

[Byr92] W. Byrne, *Alternating minimization and Boltmann machine learning*, IEEE Transactions on Neural Networks **3** (1992), 612–620.

[Byr96] ———, *Information Geometry and Maximum-Likelihood Criteria*, Proc. of Conference on Information Sciences and Systems (Princeton, NJ, USA), 1996.

[CFGJ96] R. Cole, M. Fanty, M. Gopalakrishnan, and R. Jansen, *Speaker-independent name retrieval from spellings using a database of 50,000 names*, IEEE **S5.19** (1996), 325–328.

[Cho99] W. Chou, *Maximum A Posteriori Linear Regression with Elliptically Symmetric Matrix Variate Priors*, Proc. of Eurospeech (Budapest, Hungary), vol. 1, Sep. 1999, pp. 1–4.

[CJG99] J.-T. Chien, J.-C. Junqua, and P. Gelin, *Extraction of Reliable Transformation Parameters for Unsupervised Speaker Adaptation*, Proc. of Eurospeech (Budapest, Hungary), vol. 1, Sep. 1999, pp. 207–210.

[CLJ93] W. Chou, C. H. Lee, and B. H. Juang, *Minimum Error Rate Training Based on N-Best String Models*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Minneapolis), 1993, pp. 652–655.

[CLR97] C. Chesta, P. Laface, and F. Ravera, *Bottom-Up and Top-Down State Clustering for Robust Acoustic Modeling*, Proc. of Eurospeech, vol. 1, Sep. 1997, pp. 11–14.

[CLR98] ———, *HMM Topology Selection for Acurate Acoustic and Duration Modeling*, International Conference on Spoken Language Processing (ICSLP) (Sydney), vol. 7, Nov. 1998, pp. 2951–2954.

[CMF94] R. Cole, Y. Muthusamy, and M. Fanty, *The ISOLET Spoken Letter Database*, Tech. report, Oregon Graduate Institute of Science and Technology (OGI), 19600 N.W. von Neumann Drive, Beaverton, OR 97006, Nov. 1994.

[Cox92] S. Cox, *Predictive speaker adaptation in speech recognition*, Computer speech and language **9** (1992), no. 1, 357–365.

[CP96] F. Carreira-Perpinan, *A review of dimension reduction techniques*, Tech. report, Department of Computer Science, University of Sheffield, UK, Sep. 1996.

[CRF92] R. Cole, K. Roginski, and M. Fanty, *A telephone speech database of spelled and spoken names*, International Conference on Spoken Language Processing (ICSLP), 1992, pp. 891–893.

[Csi75] I. Csiszár, *I-divergence geometry of probability distributions and minimization problems*, Annals of Probability **3** (1975), no. 1, 146–158.

[CSML00] W. Chou, O. Siohan, T. André Myrvoll, and C.-H. Lee, *Extended Maximum A Posteriori Linear Regression (EMAPLR) Model Adaptation for Speech Recognition*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 4, Oct. 2000, pp. 616–619.

[CT84] I. Csiszár and G. Tusnady, *Information geometry and alternating minimization procedures*, Statistics & Decisons, Supplement Issue No 1 (1984), 205–237.

[CW97] J.-T. Chien and H.-C. Wang, *Telephone speech recognition based on Bayesian adaptation of hidden Markov models*, Speech Communication **22** (1997), 369–384.

[Dan49] G. B. Dantzig, *Programming of Interdependent Activities: A Mathematical Model*, Econometrica, vol. II, 1949, pp. 200–211.

[Dan63] ———, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ, USA, 1963.

[DDCW00] K. Demuynck, J. Duchateau, D. Van Compernolle, and P. Wambacq, *An efficient search space representation for large vocabulary continuous speech recognition*, Speech Communication **30** (2000), no. 1, 37–54.

[DGP99] N. Deshmukh, A. Ganapathiraju, and J. Picone, *Hierarchical Search for Large Vocabulary Conversational Speech Recognition*, IEEE Signal Processing Magazine **6** (1999), no. 5, 84–107.

[DH73] R. O. Duda and P. B. Hart, *Pattern Classification and Scene Analysis*, Wiley, 1973.

[DKW99] P. Delacourt, D. Kryze, and C. Wellekens, *Speaker-based segmentation for audio data indexing*, 1999.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin, *Maximum-Likelihood from Incomplete Data via the EM algorithm*, Journal of the Royal Statistical Society B (1977), 1–38.

[Doh00] S.-J. Doh, *Enhancements to Transformation-Based Adaptation: Principal Component and Inter-Class Maximum-Likelihood Linear Regression*, Ph.D. thesis, Carnegie Mellon University, Jul. 2000.

[DRN95] V. Digilakis, D. Ritchev, and L. Neumeyer, *Speaker Adaptation Using Constrained Estimation of Gaussian Mixtures*, IEEE Trans. SAP **3** (1995), 129–136.

[DS00] S.-J. Doh and R. Stern, *Inter-class MLLR for Speaker Adaptation*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2000, pp. 1755–1758.

[Ein16] A. Einstein, *Grundlagen der allgemeinen Relativitätstheorie*, Ann. der Physik **49** (1916), 769–822.

[EW00] G. Evermann and P. C. Woodland, *Large Vocabulary Decoding and Confidence Estimation using Word Posterior Probabilities*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Istanbul), 2000.

[Gal96] M. J. F. Gales, *The Generation and Use of Regression Class Trees for MLLR Adaptation – TR.263*, Tech. report, Cambridge University Engineering Department, Aug. 1996.

[Gal97a] ———, *Adapting Semi-Tied Full-Covariance Matrix HMMs – TR.298*, Tech. report, Cambridge University (CUED), Aug. 1997.

[Gal97b] ———, *Maximum Likelihood Linear Transformations for HMM-based Speech Recognition – TR.291*, Tech. report, Cambridge University (CUED), May 1997.

[Gal99a] ———, *Maximum Likelihood Multiple Projection Schemes For Hidden Markov Models – TR.365*, Tech. report, Cambridge University (CUED), Nov. 1999.

[Gal99b] ———, *Semi-tied covariance matrices for hidden Markov models*, IEEE Transactions on Speech and Audio Processing (SAP) (1999), no. 7, 272–281.

[Gal00] ———, *Cluster adaptive training of hidden Markov models*, IEEE Transactions on Speech and Audio Processing (SAP) **8** (2000), 417–418.

[GC89] L. Gilllick and S. J. Cox, *Some statstical issues in the comparison of speech recognition algorithms*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), vol. 1, 1989, pp. 532–535.

[GG01] N. K. Goel and R. Gopinath, *Multiple Linear Transforms*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2001.

[GKNN89] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, *A generalisation of the Baum algorithm to rational objective functions*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Glasgow), 1989, pp. 631–634.

[GL92] J.-L. Gauvain and C.-H. Lee, *Bayesian Learning for Hidden Markov Model with Gaussian Mixture Observation of Markov Chains*, Speech Communication **11** (1992), 205–213.

[GL94] ———, *Maximum* a posteriori *estimation for multivariate Gaussian mixture observations of Markov Chains*, IEEE Transactions on Speech and Audio Processing (SAP) **2** (1994), no. 2, 291–298.

[Gop98] R. A. Gopinath, *Maximum Likelihood Modeling with Gaussian Distributions for Classification*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Seattle), 1998.

[Gun01a] A. Gunawardana, *Maximum Mutual Information Estimation of Acoustic HMM Emission Densities*, Tech. report, JHU-CSLP, 2001, CLSP Research Note No. 40.

[Gun01b] ———, *The Information Geometry of EM Variants for Speech and Image Processing*, Ph.D. thesis, JHU-CSLP, Apr. 2001.

[GY93] M. J. F. Gales and S. J. Young, *Parallel model combination for speech recognition in noise – TR.135*, Tech. report, Cambridge University (CUED), Jun. 1993.

[HCCZ01] C. Huang, T. Chen, E. Chang, and J. Zhou, *Analysis of Speaker Variability*, Proc. of Eurospeech (Aalborg, Denmark), vol. 2, Sep. 2001, pp. 1377–1380.

[Her90] H. Hermansky, *Perceptual linear predictive (PLP) analysis of speech*, Journal of the Acoustic Society of America **87** (1990), no. 4, 1738–1752.

[HHW85] H. Hermansky, B. Hanson, and H. Wakita, *Low-Dimensional Representation of Vowel Based on All-Pole Modelling in the Psychophysical Domain*, Speech Communication **4** (1985), 181–187.

[Hu99] Z. Hu, *Understanding and Adapting to Speaker Variability*, Ph.D. thesis, Oregon Graduate Institute of Science and Technology(OGI), 1999.

[Jel97] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1997.

[Jel99] ———, *Speech And Language Technology At The Johns Hopkins University*, Proc. of International Workshop on Automatic Speech Recognition and Understanding (ASRU) (Keystone, Colorado), vol. 1, Dec. 1999, pp. 9–13.

[JH96] J.-C. Junqua and J.-P. Haton, *Robustness in Automatic Speech Recognition – Fundamentals and Applications*, Kluwer Academic Publishers, Speech Technology Laboratory, Santa Barbara; CNRS, France, 1996.

[Jol86] I. T. Jolliffe, *Principal Component Analysis*, Springer-Verlag, Berlin, 1986.

[KA98] N. Kumar and A. G. Andreou, *Heteroscedastic Linear Discriminant Analysis and Reduced Rank HMMs for Improved Speech Recognition*, Speech Communication (1998), no. 26, 283–297.

[KJNN00] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, *Rapid Speaker Adaptation in Eigenvoice Space*, IEEE Transactions on Speech and Audio Processing (SAP) **8** (2000), no. 6, 695–707.

[KK00] D. K. Kim and N. S. Kim, *Bayesian Speaker Adaptation of HMM Parameters Based on Probabilistic PCA*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), Oct. 2000, pp. 734–737.

[KMH99] S. Kajarekar, N. Malayath, and H. Hermansky, *Analysis Of Speaker And Channel Variability In Speech*, Proc. of International Workshop on Automatic Speech Recognition and Understanding (ASRU) (Keystone, Colorado), vol. 1, Dec. 1999, pp. 9–13.

[KNJ+98a] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, K. Field, and M. Contolini, *Eigenvoices for speaker adaptation*, International Conference on Spoken Language Processing (ICSLP) **5** (1998), 1771–1774.

[KNJ+98b] R. Kuhn, P. Nguyen, J.-C. Junqua, L. Goldwasser, N. Niedzielski, S. Fincke, and K. Field, *Eigenfaces and eigenvoices: dimensionality reduction for specialized pattern recognition*, MMSP (1998), 71–76.

[KNJ+99] R. Kuhn, P. Nguyen, J.-C. Junqua, R. Boman, N. Niedzielski, S. Fincke, K. Field, and M. Contolini, *Fast speaker adaptation using* a priori *knowledge*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Phoenix, AZ, USA), vol. 2, May 1999, pp. 749–752.

[KPN+01] R. Kuhn, F. Perronnin, P. Nguyen, J.-C. Junqua, and L. Rigazio, *Very Fast Adaptation with a Compact Context-Dependent Eigenvoice Model*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2001.

[Kum01] N. Kumar, *Investigation of Silicon Auditory Models and Generalization of Linear Discriminant Analysis for Improved Speech Recognition*, Ph.D. thesis, JHU-CSLP, Mar. 2001.

[LR00] E. Lleida and R. C. Rose, *Utterance Verification in Continuous Speech Recognition: Decoding and Training Procedures*, IEEE Transactions on Speech and Audio Processing (SAP) **8** (2000), no. 2, 126–139.

[LW95a] C. J. Leggetter and P. C. Woodland, *Flexible speaker adaptation for large vocabulary speech recognition*, ESCA – European Conference on Speech Communication and Technology **2** (1995), 1155–1158.

[LW95b] ———, *Maximum likelihood linear regression for speaker adaption of continuous density hidden Markov models*, Computer Speech and Language **9** (1995), 171–185.

[MAZG97] J. McDonough, T. Anastasakos, G. Zavaliagkos, and H. Gish, *Speaker-Adapted Training on the Switchboard Corpus*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Munich, Germany), vol. 2, Apr. 1997, pp. 9–13.

[McN47] I. McNemar, *Note on the sampling error of the difference between correlated proportions and percentages*, Psychometrika **12** (1947), 153–157.

[MHK97] N. Malayath, H. Hermansky, and A. Kain, *Towards decomposing the sources of variability in speech*, Eurospeech (1997).

[MKN00] S. Molau, S. Kanthak, and H. Ney, *Efficient Vocal Tract Normalization in Automatic Speech Recognition*, Proc. of ESSV (Cottbus, Germany), Sep. 2000, pp. 209–216.

[MNP96] B. Moghaddam, C. Nastar, and A. Pentland, *A Bayesian Similarity Method for Direct Image Matching*, International Conference on Pattern Recognition (1996).

[MP69] M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, Massachussetts, 1969.

[MP01] A. Martin and M. Przybocki, *Analysis of results*, 2001 NIST LVCSR Workshop, 2001.

[NGJC99] P. Nguyen, P. Gelin, J.-C. Junqua, and J.-T. Chien, *N-Best Based Supervised and Unsupervised Adaptation for Native and Non-Native Speakers in Cars*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Phoenix, Arizona), vol. 1, May 1999, pp. 173–176.

[Ngu98] P. Nguyen, *Fast Speaker Adaptation*, Tech. report, Speech Technology Laboratory (STL), Institut Eurécom, Jul. 1998.

[NKJ+99] P. Nguyen, R. Kuhn, J.-C. Junqua, N. Niedzielski, and C. Wellekens, *Une représentation compacte de locuteurs dans lespace des modèles*, CORESA (Sophia-Antipolis, France), June 1999.

[NNP88] A. Nádas, D. Nahamoo, and M. A. Picheny, *On a Model-Robust Training Method for Speech Recognition*, IEEE Transactions on Acoustics, Speech, and Signal Processing **36** (1988), no. 11, 1432–1436.

[NO99] H. Ney and S. Ortmanns, *Dynamic Programming Search for Continuous Speech Recognition*, IEEE Signal Processing Magazine **6** (1999), no. 5, 63–83.

[Nor91] Y. Normandin, *Hidden Markov Models, maximum mutual information estimation, and the speech recognition problem*, Ph.D. thesis, Dept. of Elect. Eng., McGill University, Montreal, 1991.

[NRJ00] P. Nguyen, L. Rigazio, and J.-C. Junqua, *EWAVES: an efficient decoding algorithm for lexical tree based speech recognition*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 4, Oct. 2000, pp. 286–289.

[NRJW02] P. Nguyen, L. Rigazio, J.-C. Junqua, and C. Wellekens, *Piecewise Linear Constraints for Model Space Adaptation*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2002.

[NRK+01] P. Nguyen, L. Rigazio, R. Kuhn, J.-C. Junqua, and C. Wellekens, *Self-adaptation using Eigenvoices for Large-Vocabulary Continuous Speech Recognition*, ISCA ITR-Workshop on Adaptation Methods for Speech Recognition (Sophia-Antipolis, France), Aug. 2001, pp. 37–40.

[NRMJ02] P. Nguyen, L. Rigazio, Y. Moh, and J.-C. Junqua, *PSTL System Descriptions*, Proc. of Rich Transcription Workshop (Vienna, USA), 2002.

[NRWJ01] P. Nguyen, L. Rigazio, C. Wellekens, and J.-C. Junqua, *Construction of Model-Space Constraints*, Proc. of International Workshop on Automatic Speech Recognition and Understanding (ASRU) (Trento, Italy), 2001.

[NRWJ02] ———, *LU Factorization for Feature Transformation*, International Conference on Spoken Language Processing (ICSLP) (Boulder, USA), 2002, p. To appear.

[NS97] L. Nguyen and R. Schwartz, *Efficient 2-Pass N-Best Decoder*, DARPA-97 (1997).

[NWJ99] P. Nguyen, C. Wellekens, and J.-C. Junqua, *Maximum likelihood Eigenspace and MLLR for speech recognition in noisy environments*, Proc. of Eurospeech, vol. 6, Sep. 1999, pp. 2519–2522.

[NWK+00] P. Nguyen, C. Wellekens, R. Kuhn, J.-C. Junqua, and N. Niedzielski, *Eigen Voices: a Compact Representation of Speakers in Model Space*, Annales des Télécommunications **55** (2000).

[Ode95] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph.D. thesis, Cambride University, 1995.

[OENC96] S. Ortmans, A. Eiden, H. Ney, and N. Coenen, *Language-model look-ahead for large vocabulary speech recognition*, Proceedings of the Fourth European Conference on Speech Communication and Technology (Philadelphia, PA), Oct. 1996, pp. 2095–2098.

[OENC97] ———, *Look-ahead techniques for fast beam search*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Munich, Germany), vol. 3, Apr. 1997, pp. 1783–1786.

[PFW+95] J. Pitrelli, C. Fong, S.H. Wong, J. R. Spitz, and H. C. Lueng, *Phonebook: A phonetically-rich isolated-word telephone-speech database*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 1995, pp. 1767–1770.

[PHDG00] J. Picone, J. Hamaker, N. Deshmukh, and A. Ganapathiraju, *MSU-ISIP Automatic Speech Recognition Software: Prototype 5.6 (Source code: lxt_cstr_2.cc)*, Aug. 2000, "http://www.isip.msstate.edu/projects/speech/software/index.html".

[PKNJ01] F. Perronnin, R. Kuhn, P. Nguyen, and J.-C. Junqua, *Maximum-Likelihood Training of a Bipartite Acoustic Model for Speech Recognition*, Proc. of Eurospeech (Aalborg, Denmark), vol. B24, Sep. 2001, pp. 618–621.

[Por97] J. E. Porter, *On the 30 error criterion*, National biometrics center: Collected works, San Jose State University, Apr. 1997.

[PSZ00] M. Padmanabhan, G. Saon, and G. Zweig, *Lattice-based Unsupervised MLLR for Speaker Adaptation*, Proc. of International Workshop on Automatic Speech Recognition and Understanding (ASRU) (Paris, France), 2000.

[PTVF92] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1992.

[PWN00] M. Pitz, F. Wessel, and H. Ney, *Improved MLLR speaker adaptation using confidence measures for conversational speech recognition*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 4, Oct. 2000, pp. 548–551.

[Rav96] M. Ravishankar, *Efficient Algorithms for Speech Recognition*, Ph.D. thesis, Carnegie Mellon University, 1996.

[RJ94] F. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ, 1994, ISBN 0-13-015157-2.

[RNKJ01] L. Rigazio, P. Nguyen, D. Kryze, and J.-C. Junqua, *Separating Speaker and Environment Variabilities for Improved Recognition in Non-Stationary Conditions*, Proc. of Eurospeech (Aalborg, Denmark), vol. E14, Sep. 2001, pp. 2347–2351.

[SBB+00] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. Ramana Rao Gadde, M. Plauch, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng, *The SRI March 2000 Hub-5 Conversational Speech Transcription System*, Proc. of 2000 Speech Transcription Workshop, 2000.

[SMWN99] R. Schlüter, B. Müller, F. Wessel, and H. Ney, *Interdependence of Language Models and Discriminative Training*, Proc. IEEE ASRU Workshop (Keystone, Colorado), 1999, pp. 119–122.

[SP00] G. Saon and M. Padmanabhan, *Minimum Bayes Error Feature Selection for Continuous Speech Recognition*, International Conference on Spoken Language Processing (ICSLP) (Beijing), Oct. 2000, pp. 800–806.

[SPGC00] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, *Maximum Likelihood Discriminant Feature Spaces*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Istanbul), 2000, pp. 1747–1750.

[SRN+02] Y. Souilmi, L. Rigazio, P. Nguyen, D. Kryze, and J.-C. Junqua, *Blind channel estimation based on speech correlation structure*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2002.

[Str94] N. Ström, *Experiments with a New Algorithm for Fast Speaker Adaptation*, International Conference on Spoken Language Processing (ICSLP), 1994, pp. 459–462.

[Str96] ———, *Speaker adaptation by modeling the speaker variation in a continuous speech recognition system*, Proc. of Eurospeech, 1996, pp. 989–992.

[Str99] G. W. Strong, *The Future Of Speech Research At NSF And DARPA*, Proc. of International Workshop on Automatic Speech Recognition and Understanding (ASRU) (Keystone, Colorado), vol. 1, Dec. 1999, pp. 7–11.

[Str01] H. Strik, *Pronunciation adaptation at the lexical level*, ISCA ITR-Workshop on Adaptation Methods for Speech Recognition (Sophia-Antipolis, France), Aug. 2001, pp. 123–130.

[SZP01] G. Saon, G. Zweig, and M. Padmanabhan, *Linear Feature Space Projections for Speaker Adaptation*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2001.

[TB97] M. Tipping and C. Bishop, *Probabilistic principal component analysis*, 1997.

[TB99] _____, *Mixtures of Probabilistic Principal Component Analysers*, Neural Computation **11** (1999), no. 2, 443–482.

[TKNJ00] O. Thyes, R. Kuhn, P. Nguyen, and J.-C. Junqua, *Speaker Identification and Verification using Eigenvoices*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), Oct. 2000.

[TP91] M. Turk and A. Pentland, *Eigenfaces for Recognition*, Journal of Cognitive Neuroscience **3** (1991), no. 1, 71–86.

[UkH00] T. Uchibe, S. kuroiwa, and N. Higuchi, *Determination of threshold for speaker verification using speaker adaptation gain in likelihood during training*, International Conference on Spoken Language Processing (ICSLP) (Beijing, China), vol. 2, Oct. 2000, pp. 326–329.

[Val95] V. Valtchev, *Discriminative Methods in HMM-Based Speech Recognition*, Ph.D. thesis, Cambridge University Engineering Dept., 1995.

[VFL00] C. Vair, L. Fissore, and P. Laface, *Dynamic Adaptation of Vocabulary Independent HMMs to an Application Environment*, International Conference on Spoken Language Processing (ICSLP) (Beijing), 2000.

[Vit67] A. J. Viterbi, *Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, IEEE Transactions on Information Theory **IT-13** (1967), 260–266.

[VOWY97] V. Valtchev, J. Odell, P. C. Woodland, and S. Young, *MMIE training of large vocabulary recognition systems*, Speech Communication **Vol. 22** (1997), 303–314.

[WEG+02] P. Woodland, G. Evermann, M. Gales, T. Hain, A. Liu, G. Moore, D. Povey, and L. Wang, *CU-HTK April 2002 Switchboard System*, Proc. of Rich Transcription Workshop (Vienna, USA), 2002.

[Wes00] R. Westwood, *Speaker Adaptation using Eigenvoices*, Master's thesis, Cambridge University, 2000.

[WHM+99] P. C. Woodland, T. Hain, G. Moore, T. Niesler, D. Povey, A. Tuerk, and E. Whittaker, *The 1998 HTK Broadcast News Transcription System : Development and Results*, Proc. DARPA Broadcast News Workshop, Morgan Kaufmann, 1999, pp. 265–270.

[WLSL01a] N. Wang, S. Lee, F. Seide, and L. Lee, *Rapid speaker adaptation using* a priori *knowledge by eigenspace analysis of MLLR parameters*, Proc. of ICASSP, vol. I, 2001, pp. 317–320.

[WLSL01b] N. J.-C. Wang, S. S.-M. Lee, F. Seide, and L.-S. Lee, *Rapid Speaker Adaptation Using A Priori Knowledge by Eigenspace Analysis of MLLR Parameters*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Salt Lake City, USA), May 2001.

[WP00] P. C. Woodland and D. Povey, *Large scale discriminative training for speech recognition*, ISCA ITRW Automatic Speech Recognition: Challenges for the Millenium (Paris), 2000, pp. 7–16.

[WSN00] F. Wessel, R. Schlüter, and H. Ney, *Using Posterior Word Probabilities for Improved Speech Recognition*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP) (Istanbul, Turkey), Jun. 2000, pp. 1587–1590.

[WW00] K.-T. Wen-Wei, *Fast speaker adaptation using eigenspace-based maximum likelihood linear regression*, International Conference Aucoustics, Speech, and Signal Processing (ICASSP), 2000.

[Zhu97] H. Zhu, *Bayesian geometric theory of learning algorithms*, Proc. of International Conference on Neural Networks, vol. 2, 1997, pp. 1041–1044.

[Zhu98] ———, *Error Decomposition and Model Complexity*, Submitted to NC., Santa Fe Institute, Aug. 1998.

Curriculum Vitæ

Full Name: Patrick (An-Phú) Nguyen
Birth date: April 25, 1975.

Address:
1117 East Cota Street, # A
Santa Barbara, CA 93103, USA

Current Research Interests:
Speaker adaptation, Eigenvoices, Large Vocabulary Conversational Speech Recognition, Broadcast News and Non-broadcast speech recognition, Speaker Recognition, Meta Data Annotation, Spoken Language Systems, Audio Indexing, Language Modeling, Acoustic Modeling.

Education:

2000 – 2002	Graduate courses in UCSB (K. Rose, S. Mitra, U. Madhow)
1999 –	Doctoral student, Eurécom Institute / EPFL
1997 – 1998	Eurécom Institute, specialized in Multimedia Communications
1994 – 1998	EPFL, Telecommunications Systems
1993 – 1994	EPFL, Electrical Engineering

Work experience:

Jul 2002 –	Senior Engineer, PSTL
Oct 2001 –	Head of Speech Recognition, PSTL
Mar 2000 –	Team leader of Large Vocabulary project, PSTL
Jan 2000 –	Joined Panasonic Speech Technology Lab (PSTL)
Sep 1999 – Sep 2002	Founder of beTrust concepts, SàRL
Sep 1999 – Dec 1999	Team leader and developer at RealTimeForex.com
Sep 1998 – Sep 1999	Doctoral student in Eurécom Institute
Jan 1998 – Sep 1998	Intern at PSTL
Summer 1997	Spelled name recognition in Swiss Telecom
Summer 1996	Developed Virtual Reality simulation of laparoscopic surgery
1996	Ingénieur du Monde (NGO), Committee member

Awards:

October 2002	Created the Diplôme award (EPFL/Eurécom)
April 2002	Recipient of the PSTL Employee Excellence Award 2002
July 1998	Recipient of Hitachi Award 1998

Selected Publications and Patents:

P. Nguyen, L. Rigazio, Y. Moh, and J.-C. Junqua, *PSTL System Descriptions*, Rich Transcription Workshop, Washington, 2002.

R. Kuhn, J.-C. Junqua, P. Nguyen and N. Niedzielski, *Rapid Speaker Adaptation in Eigenvoice Space*, IEEE Trans. on SAP, 2000, vol 8, no. 6, pp. 695–707.

P. Nguyen, L. Rigazio, and J.-C. Junqua, *EWAVES: an efficient decoding algorithm for lexical tree based speech recognition*, Proc. of ICSLP, 2000, vol. 4, pp. 286–289.

P. Nguyen, L. Rigazio, C. Wellekens, and J.-C. Junqua, *Construction of Model-Space Constraints*, Proc. of ASRU, 2001, Trento, Italy.

Pat. #6,343,267 : Dimensionality reduction for speaker normalization and speaker and environment adaptation using eigenvoice techniques

Pat. #6,272,462 : Supervised adaptation using corrective N-best decoding

Served as a reviewer for the IEEE Transactions on Speech and Audio Processing. Appears on 12 international and 15 US patent applications. 5 US Patents were approved to date. Appears as co-author in 20 publications.