Spontaneous Speech Recognition Using Statistical Dynamic Models for the Vocal-Tract-Resonance Dynamics

by

Zongqiang Ma

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2000

0-612-53993-8

Abstract

The conventional hidden Markov model (HMM) has achieved significant progress in speech recognition area. However. it is far from perfect. To overcome the well-known limitations of HMM. a new statistical dynamic model is developed and investigated in this dissertation. The main novelty of this new model is the introduction of the vocal tract resonance (VTR) as the internal. structured hidden state for representing phonetic reduction and target undershoot in human production of spontaneous speech and the incorporation of pre-knowledge about the VTR dynamics into the model design. training, and likelihood computation processes.

The earliest nonlinear version of the model. originally proposed in [33]. is first evaluated and investigated on the Switchboard speech database. Compared with a baseline HMM system it turns out better performance. Based on investigation on the nonlinear version and in consideration of the systematic variations in speech. two new versions are then developed. One is called a mixture linear dynamic model and the other one a mixture linear dynamic model with switching parameters on measurement equations. Both versions overcome the inefficiency in the parameter learning and likelihood computation process of the nonlinear version. The later version uses piece-wise linear functions rather than linear functions to alleviate the inaccuracy of the former version in approximating the physically nonlinear relationship between the hidden state space and the acoustic space. The later version is a more general case of the former version. Evaluation experiments demonstrate that both versions produce large improvements. Search. a challenging problem for the new dynamic model. is finally addressed. Based on analyses of that problem. three decoding algorithms (a path-stack decoding algorithm. a second-order generalized pseudo-Bayesian decoding algorithm and an interacting multiple model decoding algorithm) are designed. Experiment results show that they all are effective. Consistent improvements are observed when the most efficient one is used on different versions of the dynamic model.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor Li Deng, for his suggestions, encouragements and helps in many aspects, academically and personally, during the entire course of my Ph.D study program. I am also very grateful to him for his original idea about a new statistical dynamic model for speech recognition, which leads to the further work in my dissertation.

I would also like to convey my many thanks to my committee members, Professor Sherman Shen, Professor Amir Khandani, Professor Paul Fieguth, Professor Enhui Yang and Dr. Hsiao-wuen Hon, for their useful suggestions and comments to this dissertation.

I acknowledge John Hopkins University, University of Waterloo, Government of Ontario and NSERC Canada for their financial support during my studies.

I have special thanks to my wife for her love and to my son for the happiness he has brought to me.

The last, but not least, I am indebted to my parents and parents-in-law for their unselfish and unyielding support.

To my wife, Chen Chen, and my son, Cody Machen

Contents

List of Tables

List of Illustrations

Chapter 1

Introduction

1.1 Automatic Speech Recognition

To date. automatic speech recognition(ASR) has achieved significant progress during the past two decades. Although the success. as well as the huge commercial markets. has attracted many companies and researchers into this area. many difficulties associated with ASR. which are mostly unsolved at the present time. prevent them from building a good practical recognition system of use in the real world. Two major ones of them are

1. Coarticulation and phonetic reduction problems

   Studies of the way in which language is organized provide strong evidence that underlying the production and perception of speech is a sequence of discrete segments that are concatenated in time [38]. These segments, called *phonemes*, are assumed to have unique articulatory and acoustic correlates. For a specific language, the inventory of these basic sound units is remarkably limited. However. speech is generated through the closely coordinated and continuous movements of a set of articulators with different degrees of sluggishness. As a result, the acoustic properties of a given phoneme can change as a function of immediate phonetic context. This contextual effect, which is known as *coarticulation*, causes the overlap of the acoustic information for the neighboring segments. Coarticulation is also responsible for the smearing of the segmental boundaries. It is very difficult to determine precisely the phoneme boundaries which segment the time function of spectral envelopes. For example, it is hard to segment a succession of voiced sounds. Furthermore, in continuous ASR it is almost impossible to segment a sentence of speech into words merely based on their acoustic features. Although the coarticulation problem can be avoided in the case of isolated word recognition by using words as the reference templates, the problem still remains in other speech recognition systems where phonemes or smaller parts of the words are used as templates. With continuous speech recognition, the difficulty is compounded by elision, where the speaker runs words together and "swallows" most of the syllables.

2. Individuality and other variation problems

   Acoustic features vary from speaker to speaker, even when the same words are uttered, according to differences in manner of speaking and articulatory organs. Even for the same speaker, acoustic features vary due to the variations in the speaker's mode (relaxed, stressed, shouting, etc.) and due to the changes in the environmental factors (e.g background noise).

How to tackle those problems still remains the biggest challenge in ASR today.

1.2 Hidden Markov Model

The hidden Markov model (HMM) [7, 8, 9] has dominated the ASR field for many years. The most successful speech recognition systems [10, 11, 12] today have been based on HMM. The distinct strength associated with this method is the powerful statistical formalisms based solidly on mathematical optimization principles. The precise mathematical framework gives rise to ease of automatic parameter learning(training) and to optimal decision rules for speech-class discrimination. The systems can do the training automatically with little or no human supervision. It has been shown [92] that for HMMs the decision rule that is used performs as well as the optimal Bayesian decision rule(asymptotically with the number of observations) and there exist efficient search algorithms that implement this decoding rule [29]. So, it is not a surprise that it currently performs the best.

However, the powerful training and recognition algorithms are based on the i.i.d. assumption of the observations, which itself, unfortunately, is not true [42, 62]. To relax this assumption, differentials (first-order and second-order) of the observations are used. But the simultaneous use of a feature vector and its derivative raises one additional problem, which has been well demonstrated in [62]. In Fig. 1.1, the solid curve is a sine wave, and the dotted curve is the differential of the solid one, which is a cosine. As shown in the figure, the optimal HMM state partitioning of the feature sequence and its differential are not consistent. To force them to be synchronous must result in reduced quality of approximation.

To overcome the speech coarticulation problem (or context-sensitivity), all state-of-the-art HMM systems adopt left and right context dependent phones (or triphones) as their unit models. It is a reasonable way for some types of phonological variations. But it results in very large amounts of training data required if the

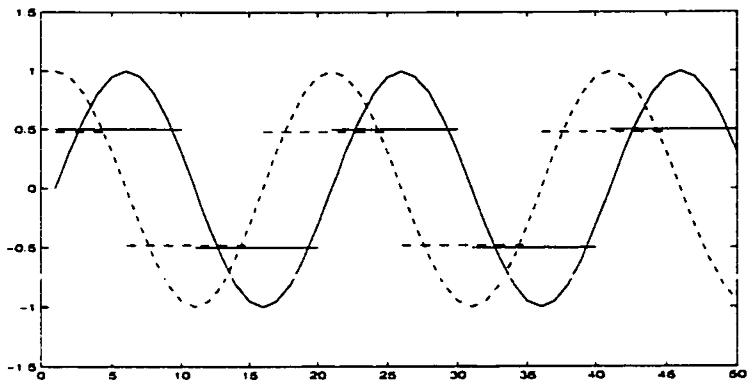


Figure 1.1: Piece-wise constant approximation of sinusoid and its derivative (cosine)

system wants to perform well with a large vocabulary because of the greatly increased number of triphone units.

To deal with the speech variation (speaker, environment, etc.) problem, the only choice for the HMM systems seems to increase the number of Gaussians. The increased number of Gaussians not only again brings about the problem of large amount of training data needed, but also increases the confusability (overlapping) between the Gaussians. In order to reduce the requirement of a huge amount of training data states or Gaussians must be tied. So how to tie states or Gaussians becomes another important issue for HMM systems.

The stationarity in each HMM state is also a serious weakness for HMM systems. It is well known that speech is a non-stationary process. The stationarity in HMM states enforce HMM to approximate the trajectory of speech observations piece-wise constantly. In theory HMM can unlimitedly increase the number of states to approximate the trajectory accurately, but it brings up again the problem of requirement of a large amount of training data. It is not realistic.

Another weakness of HMM is its data-driven mechanism. It totally ignores the phonological and phonetic perspectives of speech. First, nearly all currently popular speech recognition strategies use triphones arranged in strictly linear sequences, like "beads-on-a-string". This, however, is not how human language faculty organizes its phonological primitives. Second, the weak theoretical foundation of the current speech recognition technology from phonetic perspective is reflected in the weak structure of the HMM in use and in the simplistic strategy of surface data fitting to the observable acoustics (equipped with virtually no underlying data generation mechanisms). A consequence of this weakness is that the sample paths of the HMM as a non-stationary stochastic process deviate significantly from true speech data trajectories.

The above weaknesses associated with the current speech recognition technology lead to speech recognizers which inherently lack robustness, and cannot generalize from training data to mismatched test data. The problem is particularly serious when little supervised adaptation data are available to recognizers, as in most real-world speech recognition applications.

Therefore, although HMM has achieved many progresses in the ASR field, the problems are far from being solved. The recognizers built from this method (mostly built for a specific domain and a constrained task) perform poorly for unconstrained tasks and other domains, and even break down easily when porting from one speaking mode or environment to another, or from one language to another. Even for the best recognition systems, performance degrades rapidly in more difficult test conditions, such as spontaneous speech, noisy environment and non-cooperative speakers. This situation results from the fundamental weakness of HMMs.

1.3 Generalized Models

To relax the HMM's limitations of the i.i.d. assumption of observations and the stationarity in each state, many generalized HMM models have been proposed [42, 44, 45, 49, 50, 51, 52, 53, 54, 55, 58, 59, 62, 63]. All those models were called "segment" models in [60] (A detailed survey of those models was given).

To overcome the HMM's ignorance of the phonological and phonetic perspectives of speech, several new speech models have been proposed and investigated recently [64], [65], [66], [68], [69]. All those models try to introduce the speech production mechanism into speech recognition process. By the introduction of a speech production mechanism, they also expect to be able to tackle the speech coarticulation (context-sensitivity) problem. The HMM's triphone solution is an extravagant approach to coarticulation and ignores some well-known properties of real human speech.

The following is a brief review of all the above models. To relax the i.i.d. assumption of observations, a more reasonable and realistic i.i.d assumption, a Gauss-Markov (the first or second order) assumption, has been tried. The models based on the Gauss-Markov assumption was first investigated by Wellekens [42] and Brown [43]. Then, more work along this direction was carried out by Deng [58]. More explicit modeling of the correlation of observations has also been investigated by Kenny [45], Woodland [46] and Takahasi [47] *et al.*

To capture the dynamics in speech observations, non-parametric and parametric approaches have been used. The first non-parametric approach used was the stochastic segment model proposed in [44]. It assigns a Gaussian distribution to the entire segment time-warped to a fixed length. Ghitza and Sondhi [49] developed a similar approach to model the non-stationarity, but this work was done under the

HMM framework. HMM states were defined as diphones and a different way (dynamic time warping) was used to do time-warping. Kimball [56] extended the work in [44] and suggested that each segment be modeled by a discrete mixture of non-parametric mean trajectories. In [57] a mixture model on segment level was also proposed. Most recently Goldberger *et al.* [62] proposed a new model which used a continuous mixture of mean trajectories to model each segment. Deng *et al.* and Gish and Ng separately introduced very similar parametric approaches to model the mean trajectory in each HMM state. In both cases, the mean trajectory was modeled by a polynomial function, $\mu_i = \sum_{l=0}^{L} a_{i,l} t^l$ (L is the order of polynomial function). However, they defined t according to different ways. Deng *et al.* defined t as the absolute time (the number of frames staying at HMM states). Gish and Ng defined it as a normalized one ($t \in [0, 1]$). Russell, Holmes, Gales and Young extended the above parametric models to the continuous mixture case. In [50] and [52] the mean of the trajectories is modeled by a Gaussian distribution. In [59, 63] the coefficients of the linear trajectory function were defined as random variables and they followed Gaussian distributions.

Digalakis *et al* [54] introduced a stochastic linear dynamic model to model the speech dynamics. The model comes directly from control area where it is used to model non-stationary signals.

$$Z(k) = FZ(k-1) + W(k-1) \tag{1.1}$$

$$O(k) = HZ(k) + V(k) \tag{1.2}$$

where $O(k)$ is an observation vector, $Z(k)$ is a hidden state vector, $W(k)$ and $V(k)$ are Gaussian noises. F and H are system parameters. Each phone segment is modeled by one of the above systems, so the system parameters (F, H, and the means and covariances of W and V) are phone-dependent. The hidden states are continuous, so it can be treated as a continuous-state HMM. If $F = I$ and $W = 0$, HMM becomes a special case of this model. Eventually, most of the segment models mentioned above can be considered as special cases of this model. The work in this thesis could be viewed as the extension of this dynamic model, but there are some fundamental changes which will be elaborated later.

To deal with speech coarticulation, which is an important characteristic in the speech production process, one early attempt was made by Bridle *et al.* Bakis [64] suggested a general system with targets, linear dynamics and nonlinear output mapping to model coarticulation. Blackburn *et al* [65] and Richards *et al* [68] both incorporated speech synthesis processes into speech recognition. Both systems use multi-layer perceptrons (MLPs) to describe the relationship between "articulatory" (or dynamic) space and acoustic space. The "articulators" in both systems don't have physical meanings (pseudo-ones). In [66] and [69] the "articulators" were given physical meanings (real ones), which, however, makes the systems harder to be implemented.

1.4 A New Statistical Dynamic Model

The fundamental nature of the acoustic modeling strategy HMM and the so-called "segment" models use is such that they explore only the surface-level observation data and not its internal structure or generative mechanisms.

All the "segment" models reviewed in [60] try to model speech dynamics at acoustic level, or they try to capture the mean trajectory in observations (Mel-frequency cepstral coefficients (MFCCs)). However, the dynamics at the acoustic level is apparently caused by some more fundamental dynamics in a human's vocal tract (the generative mechanism). Hence it appears necessary to explore some underlying dynamics in the speech patterns, which is responsible for the overwhelming variability in the surface-level data. The so-called "articulatory" models try to do this. But the dynamics in [65, 68] are pseudo and totally hidden, no pre-knowledge is allowed to be added to those models. The dynamics in [66, 69] are physical ones and some pre-knowledge (target asymptotic property) are able to be added to the model, but the high dimensionality and the complicated behaviors of the articulators make the systems infeasible.

In this thesis a new statistical coarticulatory dynamic model is proposed and investigated. This model was already derived mathematically in [69], but a small modification is made here. In this work the dynamics of vocal-tract-resonance (VTR) is used to replace the original dynamics of articulators. The VTR domain is internal to the domain of surface acoustic domain (such as MFCCs). The VTRs are pole locations of the vocal tract configured to produce speech sounds. They have acoustic correlates of formants which are directly measurable for vocalic sounds, but often are hidden or perturbed for consonantal sounds due to the concurrent spectral zeros and turbulence noises. Hence, formants and VTRs are related but distinct concepts: the former is defined in the acoustic domain and the latter is associated with the vocal-tract properties. According to the goal-based speech production theory, articulatory structure and the associated VTRs necessarily manifest asymptotic behavior in their temporally smooth dynamics. Therefore, by this replacement, not only is the target-directed property of the dynamics kept, but also the model becomes feasible because of the low dimensionality of VTR space (just the first several VTRs are used).

The coarticulation modeling in this new model is accomplished via two separate but related mechanisms. First, the mechanism of duration-dependent *phonetic reduction* allows the VTR variables and the associated surface acoustic variables to be modified automatically according to the varying speech rate and hence the duration of the speech units (e.g., phones). This modification is physically established according to the structured dynamics assigned to the VTR variables in the model. Second, the "*continuity*" mechanism at the utterance level employed in the model constrains the VTR variables so that they flow smoothly from one segmental unit to another.

This new dynamic model could be classified into the so-called "segment" models because it does have the "segment" concept in its model design. The targets of the dynamics are based on segments. But it has two fundamental differences with those "segment" models.

1. All the "segment" models try to model the speech dynamics at the acoustic level, but the new model tries to do it at the internal level (VTR domain).
2. Due to the first difference, the target-directed property in the internal dynamics is able to be incorporated into the model design.

1.5 Thesis organization

In chapter 2 the original nonlinear version of the new dynamic model is introduced. The mathematical formulation of the model is first derived, then the model parameter training algorithm and model likelihood computation algorithm is derived using the Maximum Likelihood (ML) method. Finally, the new model is evaluated on Switchboard data under a N-best list re-scoring paradigm and some exploration experiments carried out. In chapter 3, a new version of the dynamic model, a mixture linear dynamic model, is developed. Model training and likelihood computation algorithms are then derived for this version. Finally, evaluation experiments are carried out for this new version and an analysis experiment is also performed to demonstrate the success of incorporation of dynamics. In Chapter 4, a more general version, a mixture linear dynamic model with switching parameters on measurement equations, is developed. The model parameter learning and likelihood computation algorithms are derived, then evaluation experiments are done for it. In Chapter 5, three efficient decoding algorithms are developed for the new dynamic models. They are first compared on a small amount of data and the most efficient one is chosen. Then the most efficient one is evaluated on different versions of the new dynamic model. Finally, in Chapter 6. conclusions are made and future work is discussed.

Chapter 2

A Statistical Coarticulatory Dynamic Model

The statistical coarticulatory model presented in this chapter is a drastic departure from the conventional HMM-based approach to speech recognition. In the conventional approach, the variability in observed speech acoustics is accounted for by a large number of Gaussian distributions, each of which may be indexed by a discrete "context" factor. The discrete nature of encoding the contextual (or coarticulatory) effect on speech variability leads to explosive growth of free parameters in the recognizers, and when the true source of the variability originates from causes of a continuous nature (such as in spontaneous speech), this approach necessarily breaks down. In contrast, the new model developed here focuses directly on the continuous nature of speech coarticulation and speech variability in spontaneous speech.

2.1 Mathematical Formulation

2.1.1 State equation

In [32], a deterministic, continuous-time task-dynamic speech production model is established. Starting with that original model but incorporating random noise $\mathbf{w}(t)$, we end up with the following model:

$$\frac{d^2\mathbf{z}(t)}{dt^2} + 2\mathbf{S}(t)\frac{d\mathbf{z}(t)}{dt} + \mathbf{S}^2(t)(\mathbf{z}(t) - \mathbf{Z}^0(t)) = \mathbf{w}(t),$$

where $\mathbf{S}^2$ is normalized, gesture-dependent stiffness parameter (which controls fast or slow movement of tract variable $\mathbf{z}(t)$), and Z^0 is gesture-dependent point-attractor parameter of the dynamical system (which controls the target and hence direction of the movement). Here, for generality, we assume that the model parameters are (slowly) time-varying.

After discretization[1], we get the following noisy, causal, and first-order "state" equation,

$$Z(k) = \Phi Z(k-1) + (I - \Phi)T + W(k-1), \tag{2.1}$$

where $Z(k)$ represents the dynamics of vocal tract constriction variables, Φ is the system matrix or "time-constant", which controls the speed of the movement of the dynamics, and T is the goal (or target) of the dynamics, which attracts the dynamics to go to it. $W(k-1)$ is a white noise with covariance Q.

The inclusion of T gives rise to the target-directed property of the dynamics. This target-directed behavior of the dynamics can be seen by setting

$$k \to \infty,$$

[1]See a derivation of this discrete-time state equation from the continuous-time system in [69].

which forces the system to enter the local, asymptotic region where

$$Z(k+1) \approx Z(k).$$

With the assumption of mild levels of noise $W(k)$, Eqn. (2.1) then directly gives the target-directed behavior in $Z(k)$:

$$Z(k) \rightarrow T.$$

The above model has been investigated before in our laboratory [66, 35, 34]. In the work here, we move from modeling the vocal tract constriction dynamics to modeling the vocal tract resonance dynamics. The late one has the same dynamic property as the former one. So the hidden dynamics $Z(k)$ become the VTR dynamics. This modification offers several significant advantages.

In [66, 35, 34], all models have the dynamic state variables completely hidden (i.e., unobservable). In the case of articulatory-dynamic model, the state variables are articulatory parameters, and in the case of task-dynamic model, the state variables are vocal tract constriction parameters. The current model uses VTRs as the hidden state variables, which are observable for vocalic sounds. In addition to the smaller dimensionality in the dynamic system state (three versus a dozen or so), the use of the partially observable VTRs as the system state has been critically important in the model development (model learning and diagnosis) and in the recognizer implementation.

Some background work which leads to the development of this particular version of the model (i.e., with use of VTRs as the partially hidden dynamic states) has been the extensive studies of spontaneous speech spectrograms and of the associated speech production mechanisms. The spectrogram studies on spontaneous speech have highlighted the critical roles of smooth, goal-directed formant transitions (in vocalic segments, including vowels, glides, and liquids) in carrying underlying phonetic information in the adjacent consonantal and vocalic segments. The smoothness in formant movements (for vocalic sounds) and in VTR movements (for practically all speech sounds [2]) reflects the dynamic behavior of the articulatory structure in speech production. The properties of the dynamic behavior change in a systematic manner as a function of speaking style and speaking rate, and the contextual variations of phonetic units are linked with the speaking style and rate variations in a highly predictable way.

The smoothness of VTR dynamics is not only confined within phonetic units but also across them. This cross-unit smoothness or continuity in the VTR domain becomes apparent after one learns to identify, by extrapolation, the "hidden" VTRs associated with most consonants, where the VTRs in spectrograms are either masked or distorted by spectral zeros, wide formant bandwidths, or by acoustic turbulence. The cross-unit smoothness (or global smoothness) is realized by extending the local smoothness in state vector $Z(k)$ across each pair of adjacent dynamic regimes, making $Z(k)$ continuous or smooth across an entire utterance. This is an important characteristics of this new model. However this constraint makes the search for the new model infeasible, which will be coped with in Chapter 5.

[2]Some limited exceptions to this smoothness across vowel-consonant boundaries include nasal consonants whose production involves sudden opening of nasal tract. In this case, the acoustic resonances are determined by both nasal and oral tracts, and can be discontinuous across vowel-nasal or nasal-vowel boundaries where the new nasal tract is suddenly introduced. To make the model consistent for such an exceptional case, the hidden dynamic variables are defined to be the acoustic resonances resulting from only the oral tract portion. Such variables then satisfy the cross-segment continuity constraint applied to the normal cases since the movements of the articulators responsible for forming the oral tract area function are smooth even while a nasal sound is produced.

2.1.2 Measurement equation

The measurement equation is to describe the relationship between observations and hidden dynamics. In this version we use the following nonlinear, noisy and static equation for it.

$$O(k) = h(Z(k)) + V(k), \qquad (2.2)$$

where the acoustic observation $O(k)$ is MFCC measurements, $V(k)$ is the additive observation noise modeled by an i.i.d., zero-mean, Gaussian process with covariance matrix R, [3] intended to capture residual errors in the nonlinear mapping from $Z(k)$ to $O(k)$. $h(\cdot)$ is a nonlinear function.

The nonlinearity is necessary because the physical mapping from VTR frequencies to MFCCs is highly nonlinear in nature. The noise used in the model Eqn.(2.2) captures the effects of VTR bandwidths (i.e., formant bandwidths for vocalic sounds) and relative VTR amplitudes on the MFCC values. These effects are secondary to the VTR frequencies but they nevertheless contribute to the variability of MFCCs. Such secondary effects are quantified by the determinant of matrix R, which, in combination with the relative size of the state noise covariance matrix Q, plays important roles in determining relative amounts of state prediction and state update in the state estimation procedure.

2.1.3 Summary of the new model

The two components, state equation and measurement equation, of the new coarticulatory speech model have been presented. The two components accommodate separate sources of speech variability. The first component, the state equation, has

[3]Diagonal covariance matrix R has been used in the current model and all the models developed later.

a smooth dynamic property, and is linear but non-stationary. It has a phonetic-goal-directed linear dynamic process rather than an i.i.d. process in HMM to model a phone segment. The second component, the observation equation, is static and nonlinear. This lower-level component in the speech generation chain handles speech variabilities including spectral tilts, formant bandwidths, relative formant magnitudes, frication spectra, and voice source differences.

The two components combined form a non-stationary, nonlinear dynamic system whose structure and properties are well understood in terms of the general process of human speech production.

This new dynamic model can be treated as a generalized HMM. If we set Φ and $h(\cdot)$ to identity matrices and $W(k-1)$ to zero, we have the following model.

$$Z(k) = Z(k-1) \tag{2.3}$$

$$O(k) = Z(k) + V(k) \tag{2.4}$$

It is a HMM with a single state.

Obviously, the model in Eqns.(2.3) and (2.4) is a stationary process, the model in Eqns.(2.1) and (2.2) is a non-stationary process. The new model hence tries to overcome the HMM's limitation of stationarity in each state.

For the new model, the parameters include: $\Theta = \{\Phi, T, h(\cdot), Q, R\}$.

2.2 Comparison with other models

The mathematical model described above can be viewed as a significant extension of the linear dynamic system model as a thus-far most general formulation of stochastic segment models for speech described in [54, 60]. The extension is in the following major aspects.

First, the continuous state variable is endowed with a physically meaningful entity in the realistic speech process (i.e., VTRs), which allows special structures to be built into the state equation and which has been instrumental in the model development (especially in model initialization, learning, and diagnosis). In contrast, in the linear dynamic system model described in [54, 60], the continuous state variable was treated merely as a smoothed version of the noisy acoustic observation. Second, special structures are built into the state equation to ensure the target-directed property of the VTR dynamics. Third, while maintaining linearity in the state equation, the observation equation is extended to a nonlinear one with use of physically motivated nonlinear functions. Finally, due to the introduction of nonlinearity in the observation equation and of the structural constraints in the state equation, the model learning and scoring algorithms described in [54, 60] have been substantially extended.

The current model shares similar motivations and philosophies of other work aiming at developing better, more compact coarticulatory models than the HMM. The models described in [65, 64, 67] have all used fully hidden internal dynamics, similar to the models described in [66, 34, 35]. Some models explicitly use articulatory parameters as the dynamic variable (e.g., [65]), others use more abstract, automatically extracted variables for the purpose of modeling coarticulation (e.g., [64, 67]). One main difference between these and the model described here lies in mathematical formulation of the models. The models described in [65, 64, 67] are largely deterministic, where the output of the models need to be explicitly synthesized and compared with the unknown speech in order to reach recognition decision. In contrast, the statistical nature of the current model permits likelihood-score computation against the unknown speech (similar to the conventional HMM formulation in this aspect) directly from the model parameters where the model synthesis is only carried out implicitly. In addition, the deterministic and statistical natures render the models with different learning criteria and hence different learning algorithms.

2.3 Model parameter learning

For notational clarity, we use O to denote the observation sequence,

$$O = \{O(1), O(2), \dots, O(K)\},$$

and Z the corresponding hidden state sequence,

$$Z = \{Z(1), \dots, Z(K)\}.$$

where K is the total number of frames of the observation.

Because the hidden dynamics $Z(k)$ is missing, the Expectation-Maximization (EM) algorithm is adopted for the model parameter estimation.

2.3.1 The joint probability

The joint log likelihood of Z and O is defined as [39]:

$$\begin{aligned} L(Z, O|\Theta) &= -\frac{1}{2}\sum_{k=1}^{K}\{\log|Q| + e1_k'Q^{-1}e1_k\} \\ &\quad -\frac{1}{2}\sum_{k=1}^{K}\{\log|R| + e2_k'R^{-1}e2_k\} \ + \ const. \end{aligned} \tag{2.5}$$

where $e1_k = Z(k) - \Phi Z(k-1) - (I-\Phi)T$ and $e2_k = O(k) - h(Z(k))$.

2.3.2 E-step

According to the EM algorithm [36], the Q-function is equal to

$$Q(\Theta|\bar{\Theta}) = E[L(Z, O|\Theta)|O, \bar{\Theta}]$$

$$\begin{aligned} &= -\frac{K}{2}\log|Q| - \frac{K}{2}\log|R| - \frac{1}{2}\sum_{k=1}^{K} E_O[e1_k' Q^{-1} e1_k] \\ &\quad -\frac{1}{2}\sum_{k=1}^{K} E_O[e2_k' R^{-1} e2_k] \ + \ const. \end{aligned} \tag{2.6}$$

where E_O denotes the conditional expectation $E[\cdot|O, \bar{\Theta}]$.

2.3.3 M-step

In this step all parameters will be re-estimated. Let derivatives of the Q-function with respect to all those model parameters equal to zeros, we can get estimates for them.

Estimates for covariances Q and R:

Let's first estimate the two noise covariances, Q and R. (note: here Q is the covariance of noise W(k-1), which is different from the Q-function in E-step). The partial derivatives of Q-function with respect to them are:

$$\begin{aligned} \frac{\partial Q(Z,O,\Theta)}{\partial Q^{-1}} &= \frac{K}{2}\frac{\partial}{\partial Q^{-1}}\log|Q^{-1}| - \frac{1}{2}\sum_{k=1}^{K}\frac{\partial}{\partial Q^{-1}}E_O[e1_k' Q^{-1} e1_k] \\ &= \frac{K}{2}Q - \frac{1}{2}\sum_{k=1}^{K} E_O[e1_k e1_k'] \end{aligned} \tag{2.7}$$

$$\begin{aligned} \frac{\partial Q(Z,O,\Theta)}{\partial R^{-1}} &= \frac{K}{2}\frac{\partial}{\partial R^{-1}}\log|R^{-1}| - \frac{1}{2}\sum_{k=1}^{K}\frac{\partial}{\partial R^{-1}}E_O[e2_k' R^{-1} e2_k] \\ &= \frac{K}{2}R - \frac{1}{2}\sum_{k=1}^{K} E_O[e2_k e2_k'] \end{aligned} \tag{2.8}$$

Let them equal to zero, we can obtain the estimates for Q and R,

$$\hat{Q} = \frac{1}{K}\sum_{k=1}^{K} E_O[e1_k e1_k'] \tag{2.9}$$

$$\hat{R} = \frac{1}{K}\sum_{k=1}^{K} E_O[e2_k e2_k'] \tag{2.10}$$

Estimates for Φ and T:

Since Φ and T are just related to $e1_k$ in the Q-function, the partial derivatives of Q-function with respect to Φ and T are [4]:

$$\begin{aligned}\frac{\partial Q(\Theta|\bar{\Theta})}{\partial \Phi} &= \frac{1}{K}\frac{\partial}{\partial \Phi}\sum_{k=1}^{K} E_O[e1_k' e1_k|O,\Phi,T] \\ &= Q^{-1}\frac{1}{K}\sum_{k=1}^{K} E_O[-Z(k)Z(k-1)' + Z(k)T' + TZ(k-1)' - TT' \\ &\quad +\Phi(Z(k-1)Z(k-1)' - Z(k-1)T' - TZ(k-1)' + TT')] \end{aligned} \tag{2.11}$$

and

$$\begin{aligned}\frac{\partial Q(\Theta|\bar{\Theta})}{\partial T} &= \frac{1}{K}\frac{\partial}{\partial T}\sum_{k=1}^{K} E_O[e1_k' e_{k1}|O,\Phi,T] \\ &= Q^{-1}\frac{1}{K}\sum_{k=1}^{K} E_O\{-Z(k) + \Phi' Z(k) + \Phi Z(k-1) - \Phi'\Phi Z(k-1) \\ &\quad + (I - \Phi - \Phi' - \Phi'\Phi)T\} \end{aligned} \tag{2.12}$$

Let them equal to zero, solve the two equations [5], we obtain the estimates for Φ and T:

$$\hat{\Phi} = (D - \hat{T}A' - B\hat{T}' + N\hat{T}\hat{T}')(C - \hat{T}A' - A\hat{T}' + N\hat{T}\hat{T}')^{-1} \tag{2.13}$$

and

$$\hat{T} = \frac{(I-\Phi)^{-1}(B - \Phi A)}{K} \tag{2.14}$$

[4]In the derivation, the following matrix calculus formulas are used: $\frac{\partial(x'Ay)}{\partial A} = xy'$, $\frac{\partial(x'A'y)}{\partial A} = yx'$, and $\frac{\partial(x'A'BAy)}{\partial A} = B'Axy' + BAyx'$, where x and y are vectors, and A and B are matrices.

[5]To make the system stable and realistic, all roots of Φ must be located within the unit circle and not be equal to zero

where

$$A = \sum_{k=1}^{K} E_O[Z(k-1)], \qquad B = \sum_{k=1}^{K} E_O[Z(k)],$$
$$C = \sum_{k=1}^{K} E_O[Z(k-1)Z(k-1)'], \qquad D = \sum_{k=1}^{K} E_O[Z(k)Z(k-1)'].$$

In Eqn.(2.14) the previous Φ value is used, which results in the generalized EM algorithm. The calculation of those sufficient statistics, $E_O[Z(k)], E_O[Z(k-1)Z(k-1)']$ and $E_O[Z(k)Z(k-1)']$, in A, B, C, and D will be given later.

Estimate for the nonlinear function $h(\cdot)$

In this version the nonlinear function $h(\cdot)$ is implemented by two different neural networks, a multi-layer perceptron (MLP) and a radial-basis-function neural network (RBFNN).

MLP case: For a multi-layer perceptron, three layers (input, hidden and output) are used. Suppose w_{jl} is used to denote the MLP weights from input to hidden units and W_{ij} the MLP weights from hidden to output units, where l is the input node index, j the hidden node index and i the output node index. Then the output at node i is equal to

$$h_i(Z) = \sum_{j=1}^{J} W_{ij} \cdot g_j(\sum_{l=1}^{L} w_{jl} \cdot Z_l), \quad 1 \le i \le I, \tag{2.15}$$

where I, J and L are the numbers of nodes at output, hidden and input layers, respectively. g_j is the hidden unit's activation function, which is the standard sigmoid function

$$g(x) = \frac{1}{1+exp(-x)} \tag{2.16}$$

with its derivative

$$g'(x) = g(x)(1 - g(x)). \tag{2.17}$$

So in this case the estimation of $h(\cdot)$ is equivalent to the estimation of w_{jl} and W_{ij}. To estimate w_{jl} and W_{ij}, we have to take partial derivatives of the Q-function with respect to them. Since $h(\cdot)$ is only related to the $e2_k$ term in the Q-function, the derivatives become

$$\frac{\partial Q(\Theta|\bar{\Theta})}{\partial W_{ij}} \propto \frac{\partial(\sum_{k=1}^{K} E_O[\{O(k) - h(Z(k))\}'\{O(k) - h(Z(k))\}])}{\partial W_{ij}} \tag{2.18}$$

$$\frac{\partial Q(\Theta|\bar{\Theta})}{\partial w_{jl}} \propto \frac{\partial(\sum_{k=1}^{K} E_O[\{O(k) - h(Z(k))\}'\{O(k) - h(Z(k))\}])}{\partial w_{jl}} \tag{2.19}$$

The calculation of expectation of the nonlinear function $\{O(k)-h(Z(k))\}'\{O(k)-h(Z(k))\}$ is very expensive. To make the algorithm more feasible, we adopt the following approximation,

$$\begin{aligned} & E_O[\{O(k) - h(Z(k))\}'\{O(k) - h(Z(k))\}] \\ \approx\ & \{O(k) - h(E_O[Z(k)])\}'\{O(k) - h(E_O[Z(k)])\}. \end{aligned} \tag{2.20}$$

The expectation is moved from the outside to the inside of the nonlinear function. By this approximation the normal MLP training algorithm (back-propagation) can be directly applied for the estimates of the weights.

Hence, for the calculation of the estimated $\hat{R}$ in Eqn.(2.10) the expectation is also moved to the inside of the nonlinear function as well.

$$\hat{R} = \frac{1}{K}\sum_{k=1}^{K}\{O(k) - h(E_O[Z(k)])\}\{O(k) - h(E_O[Z(k)])\}' \tag{2.21}$$

RBFNN case: If $h(\cdot)$ is implemented by a radial-basis-function neural network, it can be expressed as:

$$h(Z(k)) = W \cdot Y(k) \tag{2.22}$$

where W is a matrix which is organized by the weights connecting the middle layer and the output layer. As depicted in Fig.(2.1), the (i,j)-th element of W is the connecting weight between node i in the output layer and node j in the middle layer. $Y(k)$ is a vector, which is equal to

$$Y(k) = [y_1(Z(k)), y_2(Z(k)), \cdots, y_j(Z(k)), \cdots, y_J(Z(k))], \tag{2.23}$$

where $y_j(Z(k))$ is the output of the j-th radial-basis-function (or kernel function) in the middle layer.

Here Gaussians are chosen for the kernel functions.

$$y_j(Z(k)) = \exp\{-\frac{1}{2}(Z(k) - \mu_j)'\Sigma_j^{-1}(Z(k) - \mu_j)\} \tag{2.24}$$

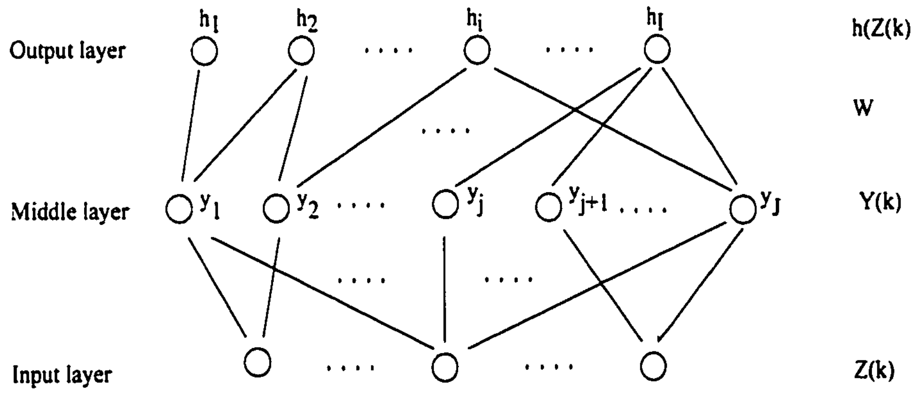


Figure 2.1: A radial-basis-function neural network

Then, for the RBFNN three sets of parameters have to be estimated, they are the weights W, the kernel centers μ_j and the kernel width Σ_j $(j = 1, 2, ..., J)$.

To estimate W, set the partial derivative of the Q-function with respect it to zero (Note: just the term $e2_k$ in the Q-function, Eqn.(2.6), is related to W)

$$\frac{\partial Q(Z, O|\Theta)}{\partial W} = \sum_{k=1}^{K} E_O[R^{-1}(O(k) - WY(k))\, Y(k)'] = 0 \tag{2.25}$$

Solve it, we have

$$\hat{W} = \left\{\sum_{k=1}^{K} O(k)\ E_O[Y(k)]'\right\}\left\{\sum_{k=1}^{K} E_O[Y(k)Y(k)']\right\}^{-1} \tag{2.26}$$

To estimate μ_j and Σ_j $(j = 1, 2, \cdots, J)$, we should follow the same way, letting the derivatives of the Q-function with respect to them equal to zeros, solving the equations and getting the solutions. However, the derivatives are high-order (more than three) nonlinear functions, close-form solutions can not be obtained for μ_j and Σ_j. To make the algorithm feasible, a separate EM algorithm is used for the training of the centers and widths of RBFNNs, which is given in [74]. We make a similar approximation to that in the MLP case, moving the expectation to the inside of the nonlinear function. But in this case the nonlinear function is not the whole $h(\cdot)$ function, it is just a part of the $h(\cdot)$ function, the radial-basis-functions.

Replacing the estimated $\hat{W}$ into Eqn.(2.10), we get

$$\hat{R} = \frac{1}{K}\left(\sum_{k=1}^{K} O(k)O(k)' - \hat{W}\sum_{k=1}^{K} E_O[Y(k)]O(k)'\right) \tag{2.27}$$

The expectations in Eqn.(2.26) and (2.27), $E_O[Y(k)]$ and $E_O[Y(k)Y(k)']$ are implementable. They are calculated according to the following equations (Derivations are given in appendix B).

$$E_O[y_j(Z(k))] = |\Sigma_{k/K}|^{-\frac{1}{2}} \cdot |\Sigma_j^{-1} + \Sigma_{k/K}^{-1}|^{-\frac{1}{2}} \cdot \exp(\delta_{jk}) \quad (1 \le j \le J) \tag{2.28}$$

where

$$\begin{aligned}\delta_{jk} \ = \ & -\frac{1}{2}[\mu_j'\Sigma_j^{-1}\mu_j + \hat{Z}_{k/N}'\Sigma_{k/N}^{-1}\hat{Z}_{k/N} \\ & - (\Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N})'(\Sigma_j^{-1} + \Sigma_{k/N}^{-1})^{-1}(\Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N})]\end{aligned}$$

and

$$E_O[y_r(Z(k))\ y_j(Z(k))] = |\Sigma_{k/K}|^{-\frac{1}{2}} \cdot |C_{rjk}|^{\frac{1}{2}} \exp(\delta_{rjk}) \tag{2.29}$$

where

$$\begin{aligned} C_{rjk} &= (\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})^{-1} \\ \delta_{rjk} &= \frac{1}{2}(\mu_r'\Sigma_r^{-1}\mu_r + \mu_j'\Sigma_j^{-1}\mu_j + \hat{Z}_{k/N}'\Sigma_{k/N}^{-1}\hat{Z}_{k/N} - M_{rjk}'C_{rjk}^{-1}M_{rjk}) \\ M_{rjk} &= (\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})^{-1}(\Sigma_r^{-1}\mu_r + \Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N}) \end{aligned}$$

where $\hat{Z}_{k/N}$ and $\Sigma_{k/K}$ are the smoothed mean and covariance at time k.

2.3.4 Calculation of the sufficient statistics

To estimate state variables from observations, Kalman-type filters [93, 94] can be used. Because of the nonlinearity of function $h(\cdot)$ in Equation (2.2), the extended Kalman filter is applied and the Jacobian matrix of $h(\cdot)$ will also be used.

For our estimation problem, the extended Kalman filter takes the following form:

Forward recursion (or Kalman filtering) :

$$\hat{Z}_{k|k-1} = \Phi\hat{Z}_{k-1|k-1} + (I - \Phi)T \tag{2.30}$$

$$\Sigma_{k|k-1} = \Phi\Sigma_{k-1|k-1}\Phi + Q \tag{2.31}$$

$$\tilde{O}_k = O(k) - h(\hat{Z}_{k|k-1}) \tag{2.32}$$

$$\Sigma_{\tilde{O}_k} = H_{Z(k|k-1)}\Sigma_{k|k-1}H_{Z(k|k-1)}' + R \tag{2.33}$$

$$K_k = \Sigma_{k|k-1}H_{Z(k|k-1)}'(\Sigma_{\tilde{O}_k})^{-1} \tag{2.34}$$

$$\hat{Z}_{k|k} = \hat{Z}_{k|k-1} + K_k\tilde{O}_k \tag{2.35}$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - K_k\Sigma_{\tilde{O}_k}K_k' \tag{2.36}$$

where $\tilde{O}_k$ and $\Sigma_{\tilde{O}_k}$ are the mean and covariance of the innovation sequence at time k. $H_{Z(k|k-1)}$ is the Jacobian matrix of $h(\cdot)$ at point $\hat{Z}_{k|k-1}$.

If $h(\cdot)$ is implemented by a three-layer MLP, its Jacobian matrix is computed as follows. The (i,l)-th element of the Jacobian matrix is

$$H_Z(i,l) = \sum_{j=1}^{J}\{W_{ij} \cdot g_j(y) \cdot (1 - g_j(y)) \cdot W_{jl}\}, \qquad 1 \le i \le I, \;\; 1 \le l \le L, \tag{2.37}$$

where $y = \sum_{l'=1}^{L} w_{jl'} Z_{l'}$.

If $h(\cdot)$ is implemented by a RBFNN, its Jacobian matrix is equal to

$$H_Z = W \cdot \begin{bmatrix} y_1(Z(k)) \cdot (Z(k) - \mu_1)' \cdot \Sigma_1^{-1} \\ y_2(Z(k)) \cdot (Z(k) - \mu_2)' \cdot \Sigma_2^{-1} \\ \vdots \\ y_J(Z(k)) \cdot (Z(k) - \mu_J)' \cdot \Sigma_J^{-1} \end{bmatrix} \tag{2.38}$$

Backward recursion (or Kalman smoothing) [93] :

$$A_k = \Sigma_{k|k}\Phi'(\Sigma_{k|k-1})^{-1} \tag{2.39}$$

$$\hat{Z}_{k|K} = \hat{Z}_{k|k} + A_k[\hat{Z}_{k+1|K} - \hat{Z}_{k+1|k}] \tag{2.40}$$

$$\Sigma_{k|K} = \Sigma_{k|k} + A_k[\Sigma_{k+1|K} - \Sigma_{k+1|k,m}]A_k' \tag{2.41}$$

Based on the Kalman smoothing results above, the three expectations needed for the parameter estimates are equal to

$$E_O[Z(k-1)] = \hat{Z}_{k|k} \tag{2.42}$$

$$E_O[Z(k-1)Z(k-1)'] = \Sigma_{k|K} + \hat{Z}_{k|K}(\hat{Z}_{k|K})' \tag{2.43}$$

$$E_O[Z(k-1)Z(k-1)'] = \Sigma_{k,k-1|K} + \hat{Z}_{k|K}(\hat{Z}_{k-1|K})' \tag{2.44}$$

where $\Sigma_{k,k-1|K}$ is recursively calculated by [40]

$$\Sigma_{k,k-1|K} = \Sigma_{k|k}A_{k-1}' + A_k(\Sigma_{k+1,k|K} - \Phi\Sigma_{k|k})A_{k-1}' \tag{2.45}$$

for $k = K, \cdots, 2$, where

$$\Sigma_{K,K-1|K} = (I - K_K H_z)\Phi\Sigma_{K-1|K-1}. \tag{2.46}$$

2.4 Parameter learning for multiple tokens

For multiple tokens, the joint log-likelihood is written as the summation of the log-likelihoods of each individual token (assume the independence between them).

$$L(\{Z,O\}^N|\Theta) = \sum_{n=1}^{N} L(Z^n, O^n|\Theta) \tag{2.47}$$

where N is the total number of tokens, Z^n and O^n are the state-variables and observations corresponding to the n-th token, $\{Z,O\}^N$ is the joint set of all Z^ns and O^ns

Following the same logic used for the single token case, we obtain the following estimates:

For Q and R:

$$\hat{Q} = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[e1_k^n e1_k^{n'}]}{\sum_{n=1}^{N} K_n} \tag{2.48}$$

$$\hat{R} = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[e2_k^n e2_k^{n'}]}{\sum_{n=1}^{N} K_n} \tag{2.49}$$

For Φ and T:

$$\hat{\Phi} = (\bar{D} - \hat{T}\bar{A}' - \bar{B}\hat{T}' + \sum_{n=1}^{N} K_n\hat{T}\hat{T}')(\bar{C} - \hat{T}\bar{A}' - \bar{A}\hat{T}' + \sum_{n=1}^{N} K_n\hat{T}\hat{T}')^{-1} \tag{2.50}$$

and

$$\hat{T} = \frac{(I-\Phi)^{-1}(\bar{B} - \Phi\bar{A})}{\sum_{n=1}^{N} K_n} \tag{2.51}$$

where

$$\bar{A} = \sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[Z^n(k-1)], \qquad \bar{B} = \sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[Z^n(k)],$$
$$\bar{C} = \sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[Z^n(k-1)Z^n(k-1)'], \quad \bar{D} = \sum_{n=1}^{N}\sum_{k=1}^{K_n} E_O[Z^n(k)Z^n(k-1)'].$$

2.5 Likelihood computation

The log-likelihood of a sequence of observations produced by a model Θ is defined as [94, 93]:

$$L(O|\Theta) = \log p(O(1), O(2), \cdots, O(K)|\Theta) \tag{2.52}$$

For HMM it is equal to the following summation because of the i.i.d. assumption among observations,

$$L(O|\Theta) = \sum_{k=1}^{K} \log p(O(k)|\Theta) \tag{2.53}$$

However, for our dynamic model the i.i.d. assumption doesn't exist any more and the observations are correlated with each other. How to calculate Eqn. (2.52)? It is calculated from the innovation sequence. The innovation sequence is generated by the Kalman filter. The Kalman filter can convert the correlated observation sequence into the uncorrelated innovation sequence, so the likelihood can be calculated from innovation sequence easily. As given in Eqns. (2.30) to (2.36), the Kalman filter is a recursive algorithm, it recursively produces the innovation sequence. Input $O(1)$, it produces $\tilde{O}_1$; $O(2)$, $\tilde{O}_2$; and so on, as illustrated in Fig. 2.2.

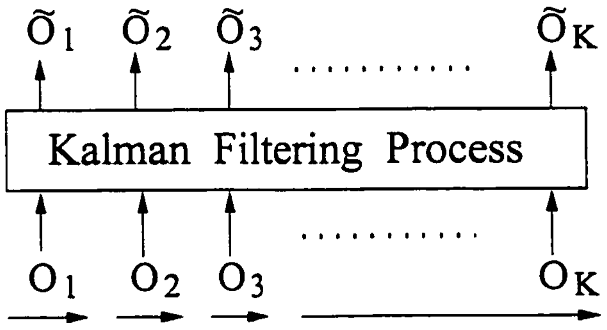


Figure 2.2: A diagram to show how KF generates the innovation process

The innovations are usually Gaussians or approximated by Gaussians, So Eqn.(2.52) can be written as

$$L(O|\Theta) = -\frac{1}{2}\sum_{k=1}^{K}\{\log|\Sigma_{\tilde{O}_k}| + \tilde{O}_k'\Sigma_{\tilde{O}_k}^{-1}\tilde{O}_k\} + const. \tag{2.54}$$

where $\tilde{O}_k$ and $\Sigma_{\tilde{O}_k}$ are the mean and covariance of the innovation sequence at time k.

The new model hence gets rid of the HMM's limitation of i.i.d. assumption of observations.

2.6 Implementation of the continuity constraint on the dynamics

The continuity of VTR dynamics is physically required for this new dynamic model. For either training or recognition (re-scoring), the VTR dynamics in a whole utterance must move continosly from the beginning to the end.

In both the parameter training and likelihood compuation, the KF algorithm is required. The KF is a recursive state estimation algorithm. As listed in Eqns. (2.30) - (2.36), the state estimate at previous time point $k-1$, $\hat{Z}_{k-1|k-1}$ and $\Sigma_{k-1|k-1}$, is used as the initial values for state estimation at current time point k.

To implement the continuity property of VTRs, the following strategy is adopted. When the dynamics switches from the previous phone to the current phone (model parameters will change), the state estimates at the last point of previous phone is used for the initial values for the state estimation at the first point of the current phones. By this way, the dynamics is enfored to go continuously from one phone to the next phone.

This continuity constraint is of importance for the new dynamic model. To some extent it is responsible for the phone context-dependence. It makes the search for the dynamic model a big problem, which will be elaborated in Chapter 5.

2.7 Evaluation Experiments

In this section, a series of experiments for the evaulation of the new statistical dynamic model on Switchboard database will be reported.

2.7.1 Experiment design

In all the experiments we use the spontaneous Switchboard data to evaluate the new dynamic model.

Experimental paradigm

We choose the N-best list re-scoring paradigm to evaluate the new recognizer in all the following experiments. The reason why this re-scoring methodology has been chosen is that efficient decoding algorithms are not available at this earlier stage. So the integration of the new model's scoring module into the lattice search paradigm is impossible.

In order to focus on acoustic modeling issues we ignore language model(LM) scores. LM itself is a research topic. Any improvements due to LM effects, should however be equally applicable to the new system.

The 100-best list of word transcription hypotheses and their phone-level segmentation (i.e., alignment) are obtained from a conventional triphone-based HMM

Physically, a phone VTR dynamic boundary is ahead of its phonetic (or HMM) boundary. In all the following experiments except specially mentioned, the phone dynamic boundaries are sub-optimally derived from their HMM phone boundaries. The derivation is carried out by setting the starting point of the current phone's VTR dynamics at the middle point of the previous phone's HMM segment. By investigating spectrograms of many Switchboard utterances, we found that the above setting is accurate for most cases. In Chapter 5, We will specially deal with the optimization of the dynamic regime boundaries.

Generation of the N-best lists

The 100-best hypotheses for each utterance in the Switchboard test set "test-ws97-dev-l" are generated by a state-of-the-art HMM system. The HMM system was developed for the Workshop'97 [6], we name it **"ws97-baseline"** system. It has been described in some detail in [70, 67]. Briefly, the system has word-internal triphones clustered by a decision tree, with a bigram language model. The system has been trained on "train-ws97-a" Switchboard data set, which consists of about 160-hour spontaneous data.

The total number of parameters in the HMM system is approximately 3.276.000 that can be broken down to the product of: 1) 39, which is the MFCC feature vector dimension; 2) 12, which is the number of Gaussian mixtures for each HMM state; 3) 2, which includes Gaussian means and diagonal covariance matrices in each mixture component; and 4) 3.500, which is the total number of the distinct HMM states clustered by the decision tree.

With bi-gram language model used, the **"ws97-baseline"** system achieves about

[6]see *http://www.clsp.jhu.edu/ws97/ws97_general.html*)

48% word error rate (WER) on the Switchboard test set "test-ws97-dev-l". This system will also serve as the benchmark system to gauge the recognizer performance improvement via use of the new speech model in some of the experiments in later chapters.

Selection of training and test data

For training, in these earlier experiments, we want to make the situation simple, so just a single speaker's data (speaker ID: 1028) is extracted from the Switchboard training set "train-ws97-a" as training data. It consists of several telephone conversations and a total of 30 minutes of the data. Due to the use of only a single speaker, we avoid normalization problems for both the VTR targets and for the MFCC observations. On the other hand, the small amount of data enables the training to be finished within an endurable time. It takes much time to train the nonlinear functions, which occupies most of the training time. We name this training set **"1/2 hour"** training set.

A HMM system is also trained on this small amount of training set. We call it **"HMM baseline"** system, which will serve as the benchmark system in all the experiments.

For test, all the male speakers from the "test-ws97-dev-l" test data are selected. It results in a total of 23 male speakers comprising 24 conversation sides (each side has a distinct speaker), 1243 utterances (sentences), 9970 words, and 50 minutes of speech as the test data. All the 100-best hypotheses for each of those utterances have been generated by the **"ws97-baseline"** HMM system.

Without language model scores, the 100-best re-scoring results of the two HMM systems on the above test set are listed in Table 2.1. The "Oracle" and "By chance" performances are also shown in the table to calibrate the recognizer's performance. The "Oracle" WER is calculated by always choosing the best one hypothesis and the "By chance" WER is computed by randomly picking up one out of all hypotheses. "100 best" column means re-scoring results on 100-best lists and "Ref+100" column re-scoring results on 100-best lists and the reference.

systems	Ref+100	100 best
Oracle	0.0	32.5
By chance	59.6	60.2
HMM-baseline	56.1	58.9
ws97-baseline	56.2	56.9

Table 2.1: Performance (WER) of the two HMM systems

2.7.2 Design parameters of the new recognizer

First, we choose a total of 42 distinct phone-like symbols, including 8 context dependent phones, each of which is intended to be associated with a distinct three-dimensional (F1, F2, and F3) vector-valued target (T^j) in the VTR domain. The phone-like symbol inventory and the VTR target values used to initialize the model training discussed in Section 3.1 are shown in Tables 1 and 2, one for context-independent symbols and the other for context-dependent ones.

A total of eight phones in Table 2 are made context dependent in the recognizer because their target VTRs are affected by the anticipatory tongue position associated with the following phone. The targets of a phone with subscript f (such as b_f) are conditioned on the following phones being front vowels (iy, ih, eh, ae, and y). The

Units	VTR1(F1)	VTR2(F2)	VTR3(F3)	Units	VTR1	VTR2	VTR3
aa	0.730	1.090	2.440	d	0.180	1.800	2.700
ae	0.660	1.720	2.410	t	0.180	1.800	2.700
ah	0.640	1.190	2.390	s	0.250	1.900	2.700
ao	0.570	0.840	2.410	th	0.250	1.300	2.500
ax	0.500	1.500	2.500	z	0.250	1.900	2.700
eh	0.530	1.840	2.480	zh	0.250	1.900	2.500
uh	0.440	1.020	2.240	sh	0.250	1.900	2.500
uw	0.300	0.870	2.240	dh	0.250	1.300	2.500
er	0.490	1.350	1.690	n	0.250	1.800	2.700
ih	0.390	1.990	2.550	en	0.500	1.500	2.500
iy	0.270	2.290	3.010				
l	0.450	1.060	2.640				
el	0.450	1.000	2.700				
r	0.460	1.240	1.720				
w	0.350	7.70	2.340				
y	0.360	2.270	2.920				

Table 2.2: Context-independent units and their VTR target values (in unit of khz) used to initialize model training

targets of a phone without the subscript are conditioned on the following phones being the remaining phones. The initial VTR target values in Tables 1 and 2 are based on the Klatt synthesizer setup [72], and are slightly adjusted by examining some spectrograms of the Switchboard training data.

2.7.3 Experiment I: using class MLPs

This is the earliest version of the dynamic model which was evaluated in the Workshop'97 held in John Hopkins University. In this version we used tied MLPs to implement the VTR-to-MFCC mapping. Unlike the target and system matrix parameters which are phone dependent, we tied the MLPs approximately according to

Units	VTR1	VTR2	F3 VTR3	Units	VTR1	VTR2	F3 VTR3
b	0.180	1.100	2.300	b_f	0.180	1.800	2.300
g	0.180	1.500	2.200	g_f	0.180	2.200	2.800
k	0.180	1.500	2.200	k_f	0.180	2.200	2.800
p	0.180	1.100	2.300	p_f	0.180	1.800	2.300
f	0.250	1.100	2.300	f_f	0.250	1.800	2.300
m	0.250	1.100	2.300	m_f	0.250	1.800	2.300
ng	0.250	1.500	2.200	ng_f	0.250	2.200	2.800
v	0.250	1.100	2.300	v_f	0.250	1.800	2.300

Table 2.3: Context-dependent units and their VTR target values (in unit of khz) used to initialize model training

the distinct classes of manner of articulation (and voicing) [7] By not tying all phones into one single MLP. we also ensure effective discrimination of phones using differential nonlinear mapping (from the smoothed physical VTR state variables to the MFCCs) even if the VTR targets are identical for different phones (a few phones have nearly identical VTR targets). The ten classes resulting from the tying and used in this version are:

1) aw. ay. ey. ow. oy, aa, ae. ah. ao, ax. ih. iy, uh. uw. er. eh. el;

2) l. w. r. y; 3) f. th, sh; 4) s, ch;

5) v. dh. zh: 6) z, jh; 7) p. t. k;

8) b. d. g; 9) m, n, ng, en; 10) sil. sp.

For each of the ten distinct MLPs, we used 100 (nonlinear) hidden units, three (linear) input units, and 12 (linear) output units. This gives a total of $10 \times 100 \times (3 + 12)$ MLP weight parameters.

The results are listed in Table 2.4. For convenience. in Table 2.4, the "by chance" and **"HMM-baseline"** results are also given.

[7]Why tied MLPs were used and How they were tied have been explained in detail in [67, 4]

systems	Ref+100	100-best
By chance	59.6	60.2
HMM baseline	56.1	58.9
tied MLP version (fixCov)	49.6	60.1
tied MLP version (trnCov)	55.5	59.1

Table 2.4: Performance (WER) comparison of the new model with tied MLPs and benchmark HMM system

In Table 2.4. "fixCov" indicates that the system's noise covariances. Q and R. are fixed by experience and "trnCov" indicates that the covariances are trained from data.

For the system with noise covariances fixed by hand. for the "Ref+100" case. the VTR recognizer performs significantly better than the benchmark HMM recognizer. which is slightly better than the chance performance. For the "100-best" case. the VTR recognizer performs nearly the same as the chance. and slightly worse than the benchmark HMM recognizer. This contrasts sharply with the superior performance of the VTR recognizer when it is exposed to references.

A reasonable explanation for this phenomenon is that the long-span context-dependence property of the VTR model naturally endows the model with the capability to "lock-in" to the correct transcription and it at the same time increases the tendency for the model to "break-away" from partially correct transcriptions due to the influence of wrong contexts. Since nearly all the hypotheses in the N-best list contain a large proportion of incorrect words. they affect the matching of the model to the remaining correct words in the hypotheses through the context-dependence mechanism much stronger than the conventional triphone HMM. It is an error propagation problem for the new dynamic model.

For the system with noise covariances trained, for the cases with references included the performances are worsened significantly, but for the cases without references included the performances are improved slightly. It demonstrates that the new model is sensitive to the noise level. That is why the noises are fixed by hand during the earlier work [67, 3]. To make the new model more realistic, the noise covariances will be automatically trained in all the following experiments.

2.7.4 Experiment II: using more MLPs

During the earlier evaluation experiments we found that the prediction errors of the MLPs were much larger than we originally expected. In Fig. 2.3 the solid line shows the average MLP prediction error of the tied MLP version changing with EM training iterations. After training (4 EM iterations) the average prediction error is about 220. It is large. [8] It means that we didn't have an accurate function to describe the relation between the hidden dynamic space and the observation space.

How does the prediction accuracy affect the system performance? To investigate it, we have carried out the following experiments to gradually increasing the number of MLPs used, hoping to decrease the prediction error. First, we get rid of those tied MLPs and build one MLP for each individual phone. We name it "phone MLP version". The dashed line in Fig. 2.3 showes its prediction error changing with EM iterations. Compared with "tied MLP version", the average error, as expected, goes

[8]The prediction error is calculated according to

$$pred\_err = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n}|O_n(k) - h(Z_n(k)|^2}{\sum_{n=1}^{N} K_n}$$

where N is the number of tokens and K_n is the length (frames) of the n-th token. 12 MFCC coefficients are used here, so one can guess what the prediction difference is on each coefficient.

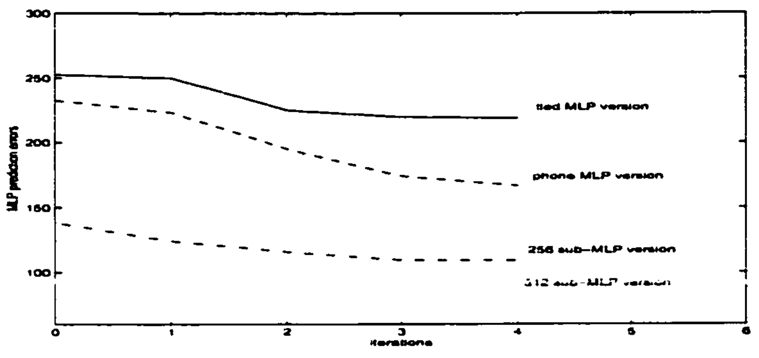


Figure 2.3: Average MLP prediction errors for different versions

down from 220 to about 160 after training. At the same time the system performance, which is listed in Table 2.5, is improved, but very limitedly (about 1%).

systems	Ref+100	100-best
tied MLP version	55.5	59.1
phone MLP version	54.1	58.2
256 sub-MLP version	54.5	57.9
512 sub-MLP version	55.7	58.9

Table 2.5: Re-scoring performance (WER) of the systems with multiple MLPs

Second, we go further step to separate, using VQ algorithm, the whole observation space into 256 and 512 sub-spaces. Each of these sub-spaces is described by one MLP. By this way we are expecting much less prediction errors. However, on the other hand, we are also taking the risk of losing some distinguishable information because we are separating the space by a data-driven approach rather than accoring to the phonetic property (or the articulation manner). We name these two versions "256 sub-MLP version" and "512 sub-MLP version".

The dashed-dotted line in Fig. 2.3 showes the average prediction error for the

"256 sub-MLP version" and the dotted line for the "512 sub-MLP version". The average prediction error is further decreased to about 110 in the "256 sub-MLP version" and to about 85 in the "512 sub-MLP version" version. However, compared with the "phone MLP version", the system performances, which are listed in Table 2.5. are not improved at all, they are even worsened. The reason is probably that more distinguishable information has been lost during the data-driven sub-space separation process, which is not able to be compensated by the reduced prediciton error. Therefore, it is not the direction we are going to explore farther.

By this set of experiments, we conclude:

- The relation between the hidden dynamic space and the acoustic space is highly complicated. Even with 512 sub-space MLPs we still have a large prediction error (about 85).
- A more accurate description of the nonlinear relationship between the hidden space and acoustic space is not the most important issue for the new model at current stage.

2.7.5 Experiment III: using phone RBFNNs

In experiment II, we have observed that the MLP prediction error is not an important issue for the new model at current stage. So, in this experiment we go to investigate another issue, using other kinds of nonlinear functions rather than MLPs to implement the $h(\cdot)$ in the measurement equation.

Radial-basis-function neural networks(RBFNN) are considered to be able to approximate nonlinear functions more smoothly than MLPs [74]. So Jacobian matrices of RBFNNs should vary more smoothly than those of MLPs. In the extended Kalman filter algorithm listed in section 2.3.4 the Jacobian matrix is an important factor. So in this experiment we replace the MLPs with RBFNNs to check how it affects the system behavior. Since the "phone MLP version" turns out the best performance among all the former experiments, we build one RBFNN for each individual phone in this experiment. We call it "phone RBFNN version".

The re-scoring results are listed in Table 2.6. For convenience, the results for "phone MLP version" and "HMM-baseline" are also listed in the table. Compared with the "phone MLP version", limited improvement (less than 1%) is obtained. It implies that the smoothness of nonlinear functions indeed affects the system behavior, but the influence is again limited.

systems	Ref+100	100-best
phone RBFNNs	53.4	57.9
phone MLPs	54.1	58.2
HMM baseline	56.1	58.9

Table 2.6: Re-scoring performance (WER) of the systems with phone RBFNNs and phone MLPs

Compared "phone MLP version" and "phone RBFNN version" with the HMM baseline system, they both turn out better performance no matter if the references are included or not. Although the improvement is limited, it demonstrates the promise of this new model.

From this experiment, we observe:

- The new dynamic model is promising.

- Different nonlinear versions do not turn out much difference on the system performance. It motivates the work in the following chapter.

2.8 Conclusions

The spontaneous speech process is a combination of cognitive (linguistic or phonological) and physical (phonetic) sub-processes. The new statistical coarticulatory dynamic model presented in this chapter focuses on the physical aspect of the spontaneous speech process, where a main novelty is the introduction of the VTR as the internal, structured model state (continuous-valued) for representing phonetic reduction and target undershoot in human production of spontaneous speech.

The continuity constraint imposed on the VTR state across speech units as implemented in the model is physically motivated. Such continuity is not valid in the acoustic domain because of the nonlinear, "quantal" nature of the distortion in the peripheral speech production process [73], and in order for the model to ultimately score on the acoustic domain, we explicitly represent the nonlinear distortion as a model component integrated with the VTR dynamic component. With the complex model structure formulated mathematically as constrained, nonstationary, and nonlinear dynamic system, a version of the generalized EM algorithm has been developed and implemented for automatically learning the compact set of model parameters.

A series of evaluation experiments have been carried out using the recognizer built from the new speech model and using the spontaneous speech data from the Switchboard corpus. The promise of the new recognizer is demonstrated by showing its superior performance, over a benchmark HMM system under similar experimental conditions.

The impact of accuracy and smoothness of the nonlinear function $h(\cdot)$ on the new model's performance has been investigated by running a series of experiments. The results show that the impact is limited and imply that they are not the important aspects we should focus on at current stage. It motivates the work in next chapter.

Chapter 3

A New Version of the Statistical Dynamic Model: A Mixture Linear Dynamic Model

In this chapter, a new version for the dynamic model, a mixture linear dynamic model (MLDM), is developed and evaluated, where several linear dynamic models are combined to represent different VTR dynamic behaviors and the mapping relationship between the VTRs and the observations. Each linear dynamic model is formulated as a state-space model, where the state equation is the same as before, but the measurement equation becomes a linear regression function that approximates the nonlinear relationship between the VTRs and the observations. A version of the generalized EM algorithm [37] has been developed for the learning of the model parameters, where a constraint that the VTR targets change at the segmental level (not at the frame level) is imposed on the parameter learning algorithm. A set of speech recognition experiments are carried out to evaluate the new model using the N-best re-scoring paradigm in a Switchboard task.

3.1 Motivation

Physically the relation between the hidden dynamic (VTR) space and acoustic space (MFCC) is nonlinear. That is why we used the nonlinear functions (MLPs and RBFNNs) to approximate it in the previous chapter. However, by running a series of experiments we found: first, the relation between the two spaces is a highly complicated one, it is difficult to approximate it accurately; second, the accuracy and smoothness of the nonlinear function just have limited influence on the new model's performance. Different nonlinear models don't turn out much difference. This observation makes me to try a linear model to check what we benefit from the use of nonlinear models.

A most straightforward linear method is to use a linear regression function to implement the $h(\cdot)$ function in the measurement equation, while keeping the target-directed, linear state dynamics of the state equation intact. This gives the measurement equation of the state-space model as follows:

$$O(k) = a + HZ(k) + V(k) \tag{3.1}$$

which can be re-written as

$$O(k) = \dot{H}\,\dot{Z}(k) + V(k) \tag{3.2}$$

where $\dot{H} = [a, H]$ and $\dot{Z}(k) = [1, Z(k)']'$, a is a vector and H a matrix. (Note: the dot here doesn't denote differential sign.)

Hence, each phone is modeled by the following linear dynamic model (LDM).

$$\begin{aligned} Z(k) &= \Phi Z(k-1) + (I - \Phi)T + W(k-1) && (3.3) \\ O(k) &= \dot{H}\,\dot{Z}(k) + V(k) && (3.4) \end{aligned}$$

The re-scoring results for this version, named "phone LDM version", are listed in the bottom line of Table 3.1. Compared with those of the versions using MLPs and RBFNNs (also shown in the table), the performance of the LDM version is worsened. It implies that we do benefit from the using of the nonlinear models, MLP version and RBFNN version. But the benefit is very limited (1% or 2%).

systems	Ref+100	100-best
phone MLP version	54.1	58.2
phone RBFNN version	53.4	57.9
phone LDM version	55.7	58.9

Table 3.1: Performance (WER) Comparison of nonlinear and linear systems

This comparison result surprises us somewhat in terms of the improvement on the system performance. The main reasons are probably due to the approximations made in the nonlinear system design. The use of the nonlinear function causes difficulties in state estimation. and specifically, in computation of the conditional expectation of the nonlinear function. One approximation was made in the M-step of the EM algorithm for the estimation of the $h(\cdot)$ (Eqn.(2.20)):

$$E[h(Z(k))] \approx h(E[Z(k)])$$

which moves the expectation from the outside to the inside of the nonlinear function. The other one is the nonlinear filtering algorithm used due to the nonlinearity. While there are many nonlinear filtering approaches already developed to cope with the nonlinearity by employing approximation, the effect of approximation has been poorly understood, especially for the current model of speech. In the previous chapter we adopted the iterated extended Kalman filtering (EKF) algorithm to approximate the sufficient statistics of the hidden states, as was required in the M-step of the EM algorithm. However, once the nonlinear observation equation is replaced by a linear one, we no longer need the above two approximations. Nevertheless, the relationship between the VTR hidden space and the acoustic (MFCC) space, which is physically nonlinear, is approximated by a linear function.

It appears that the effectiveness in learning the linear model parameters has adequately compensated for its weakness in representing the nonlinear relationship between the VTR and the MFCC spaces by the linear approximation with apparent low accuracy.

One way of improving modeling accuracy while maintaining effectiveness in model learning is to extend the linear dynamic system model discussed above to its mixture version. That is, rather than using one single set of model parameters to characterize each phone, we can use multiple sets of model parameters. The multiple mixtures will be responsible for speech systematic variations. This gives rise to the mixture linear dynamic model reported in this chapter.

3.2 A Mixture Linear Dynamic Model (MLDM)

The VTR dynamics and the resulting measurable acoustic dynamics, whose parameters are distinct for each separate phone, are represented mathematically by a combination of a set of linear dynamic models (LDM). This is called the mixture linear dynamic model, which can be written succinctly in the following form:

$$\sum_{m=1}^{M} \pi_m \cdot LDM_m \tag{3.5}$$

where M is the total number of linear dynamic models (or mixtures) for each phone, π_m is the mixture weight and LDM_m is the m-th LDM, which is expressed in the same state-space form as given in Eqn.(3.3) and (3.4) except that the parameters are indexed by m,

$$\begin{aligned} Z(k) &= \Phi_m Z(k-1) + (I - \Phi_m)T_m + W_m(k-1) && (3.6)\\ O(k) &= \dot{H}_m \dot{Z}(k) + V_m(k) && (3.7) \end{aligned}$$

where $W_m(k)$ and $V_m(k)$ have covariances Q_m and R_m respectively.

For this version. the model parameters include:

$$\Theta = \{\pi_m, \Phi_m, T_m, Q_m, R_m, \dot{H}_m, \quad m = 1, 2, \ldots, M\},$$

where each of the parameters is indexed by the mixture component m.

3.3 An Important Constraint

An important constraint. which we call mixture-path constraint. must be imposed on the above mixture linear dynamic system model. That is. for each sequence of acoustic observation associated with a phone, being either a training token or a test token. it is constrained to be produced from a fixed mixture component. m. of the dynamic model. This means that the target of the VTR in a phone is not permitted to switch from one mixture component to another at the frame level [1]. The constraint of such a type is motivated by the physical nature of the speech model — the target which correlates with the phonetic identity is defined at the segment (phone) level, not at the frame level. For example. suppose there are two mixtures for a phone, each has different targets and "time constants". So they turn out different dynamics which are depicted in Fig. 3.1. If a sequence of observations chooses mixture 1 at the beginning, it must follow the dynamics of mixture 1 to the end of the segment, it is not allowed to switch to mixture 2 during that segment.

[1]This same mixture-path constraint has been imposed on the mixture trended HMM; see [55].

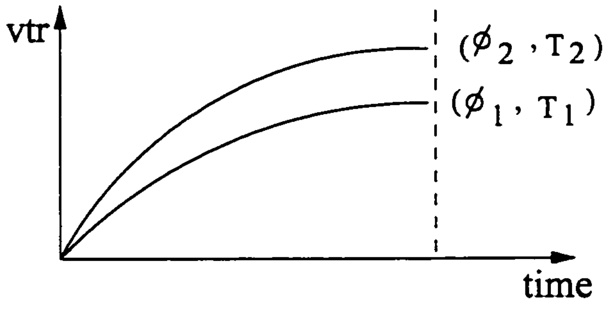


Figure 3.1: An example to show the mixture-path constraint

Use of segment-level mixtures is intended to represent the sources of speech variability including speakers' vocal tract shape differences and speaking-habit differences, etc. This constraint must be imposed on both the model training and likelihood computation processes.

3.4 Parameter Estimation Algorithm

One principal contribution of this chapter is the development of the parameter estimation (or learning) algorithm, which allows automatic determination of all parameters of the mixture linear dynamic system model discussed above from a given set of training data. The algorithm developed is based on the Expectation-Maximization (EM) principle for maximum likelihood.

To proceed the development of the parameter estimation algorithm, we first define a discrete variable X, which indicates the observation-to-mixture assignment for every sequence of observation. For example, for a give sequence of observation of a phone, if $X = m$, $(1 \leq m \leq M)$, it means the m-th mixture model is the true one to generate that observation. For simplicity purposes, we will use m to denote $X = m$.

To impose the mixture-path constraint on the training algorithm, we must define the joint variable at segment level. Suppose there are N training tokens for a phone. We define a joint variable as

$$\{O, Z, X\}^N = \{(O^1, Z^1, X^1), (O^2, Z^2, X^2), ..., (O^n, Z^n, X^n), ..., (O^N, Z^N, X^N)\}$$

where O^n, Z^n and X^n are the n-th training token, its corresponding hidden state sequence, and its corresponding mixture assignment, respectively. All discrete random variables X^n, $(1 \leq n \leq N)$ are assumed to have an identical distribution. We assume further that the N tokens are independent of each other.

The EM algorithm described in this section treats Z and X as missing data, and treat measurements O as observation or training data. The development of the EM algorithm described below consists of several steps. First, we develop an explicit expression for the joint probability density function (PDF) of the observation and missing data. We also develop an expression for the mixture weighting factor. These expressions are then used to compute the conditional expectation as required in the E-step of the EM algorithm. This conditional expectation is expressed as a function of a set of sufficient statistics computed from the linear Kalman filter. Finally, re-estimation formulas are derived using the conditional expectation in the M-step of the EM algorithm.

3.4.1 The joint PDF of observation and missing data

Due to the token-independence assumption, the joint PDF of observation and missing data $\{O, Z, X\}^N$, given the parameter set Θ, can be written as

$$\begin{aligned} p(\{O, Z, X\}^N|\Theta) &= \prod_{n=1}^{N} p(O^n, Z^n, X^n|\Theta) \\ &= \prod_{n=1}^{N} p(O^n, Z^n|X^n, \Theta)\ P(X^n|\Theta) \qquad (3.8) \end{aligned}$$

In Eqn.(3.8), $p(O^n, Z^n|X^n, \Theta)$ is the conditional joint PDF of O^n and Z^n given that mixture component is fixed. It can be further expressed as [39]

$$p(O^n, Z^n|X^n, \Theta) = p(Z_0^n|X^n, \Theta) \prod_{k=1}^{K_n} p(Z_k^n|Z_{k-1}^n, X^n, \Theta)\, p(O_k^n|Z_k^n, X^n, \Theta) \tag{3.9}$$

where K_n is the total number of frames of the nth training token (MFCC sequence), $p(Z_0^n|X^n, \Theta)$ is the distribution of initial value of the hidden dynamics at time $k = 0$ given the mixture component. Let $X^n = m^n$, $(1 \le m^n \le M)$, and for notational clarity, we drop Θ and use $p(Z_{k,m^n}^n|Z_{k-1,m^n}^n)$ and $p(O_k^n|Z_{k,m^n}^n)$ to represent $p(Z_k^n|Z_{k-1}^n, X^n, \Theta)$ and $p(O_k^n|Z_k^n, X^n, \Theta)$, respectively. We then have

$$p(O^n, Z^n|X^n, \Theta) = p(Z_0^n|m^n) \prod_{k=1}^{K_n} p(Z_{k,m^n}^n|Z_{k-1,m^n}^n)\, p(O_k^n|Z_{k,m^n}^n) \tag{3.10}$$

Substituting this into Eqn.(3.8), we obtain the conditional joint PDF $\{O, Z, X\}^N$ of the explicit form:

$$\begin{aligned} & p(\{O, Z, X\}^N|\Theta) \\ = & \prod_{n=1}^{N} \left\{ p(Z_0^n|m^n) \left[\prod_{k=1}^{K_n} p(Z_{k,m^n}^n|Z_{k-1,m^n}^n)\, p(O_k^n|Z_{k,m^n}^n) \right] P(m^n|\Theta) \right\} \end{aligned} \tag{3.11}$$

3.4.2 Computation of the mixture weighting factor

The conditional joint PDF for $\{O, X\}^N$ is

$$\begin{aligned} p(\{O, X\}^N|\Theta) &= \prod_{n=1}^{N} p(O^n|X^n, \Theta) P(X^n|\Theta) \\ &= \prod_{n=1}^{N} p(O^n|m^n, \Theta) P(m^n|\Theta) \end{aligned} \tag{3.12}$$

The PDF for the observation sequence is

$$\begin{aligned} p(\{O\}^N|\Theta) &= \prod_{n=1}^{N} \sum_{X^n=1}^{M} p(O^n|X^n, \Theta) P(X^n|\Theta) \\ &= \prod_{n=1}^{N} \sum_{l=1}^{M} p(O^n|l, \Theta) P(l|\Theta) \end{aligned} \tag{3.13}$$

In the above equation, all $X^n (1 \leq n \leq N)$ follow an identical distribution so we use one common variable l to replace them.

The conditional PDF of $\{X\}^N$ given $\{O\}^N$ is

$$\begin{aligned} & p(\{X\}^N|\{O\}^N, \Theta) \\ = & \frac{p(\{O, X\}^N|\Theta)}{p(\{O\}^N|\Theta)} \\ = & \frac{\prod_{n=1}^N p(O^n|m^n, \Theta) P(m^n|\Theta)}{\prod_{n=1}^N \sum_{l=1}^M p(O^n|l, \Theta) P(l|\Theta)} \\ = & \prod_{n=1}^N \omega_{m^n}^n \end{aligned} \tag{3.14}$$

where we define the (token-dependent) mixture weighting factors to be

$$\omega_{m^n}^n = \frac{p(O^n|m^n, \Theta) P(m^n|\Theta)}{\sum_{l=1}^M p(O^n|l, \Theta) P(l|\Theta)}. \tag{3.15}$$

The mixture weighting factors have two notable properties which we will use later. First, they sum to unity: $\sum_{m^n=1}^M \omega_{m^n}^n = 1$. Second, they satisfy:

$$\sum_{\{X\}^N/X^n} p(\{X\}^N|\{O\}^N, \Theta) = \omega_{m^n}^n, \tag{3.16}$$

where $\{X\}^N/X^n$ means the set of $\{X\}^N$ excluding X^n.

3.4.3 E-step

Given the various PDF's computed above, we are now in a position to derive an iterative EM algorithm for parameter estimation. For the MLDM presented in this chapter, both $\{Z\}^N$ and $\{X\}^N$ are treated as missing data. The Q-function in the E-step of the EM algorithm is computed below as the conditional expectation over the missing data [36, 41]:

$$\begin{aligned}
& Q(\Theta|\bar{\Theta}) \\
= & \sum_{\{X\}^N} \int \log p(\{O,Z,X\}^N|\Theta) \cdot p(\{Z,X\}^N|\{O\}^N,\bar{\Theta})\, d\{Z\}^N \\
= & \sum_{\{X\}^N} \int \log p(\{O,Z,X\}^N|\Theta) \cdot p(\{Z\}^N|\{O,X\}^N,\bar{\Theta})\, d\{Z\}^N \\
& \qquad \cdot p(\{X\}^N|\{O\}^N,\bar{\Theta}) \qquad (3.17)
\end{aligned}$$

where $\bar{\Theta}$ denotes the model parameters associated with the immediately previous iteration of the EM algorithm.

Substituting Eqn.(3.11) into Eqn.(3.17), using the independence between tokens, we have

$$\begin{aligned}
Q(\Theta|\bar{\Theta}) = & \sum_{\{X\}^N} \Big\{ \int \sum_{n=1}^{N} \Big\{ \log p(Z_0^n|m^n) \\
& \qquad + \sum_{k=1}^{K_n} \left[\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n) + \log p(O_k^n|Z_{k,m^n}^n)\right] \\
& \qquad + \log p(m^n|\Theta) \\
& \qquad \Big\} \cdot p(\{Z\}^N|\{O,X\}^N,\bar{\Theta}) \cdot d\{Z\}^N \\
& \Big\} \cdot p(\{X\}^N|\{O\}^N,\bar{\Theta}) \\
= & \sum_{\{X\}^N} \Big\{ \sum_{n=1}^{N} \int \Big\{ \log p(Z_0^n|m^n) \\
& \qquad + \sum_{k=1}^{K_n} \left[\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n) + \log p(O_k^n|Z_{k,m^n}^n)\right] \\
& \qquad \Big\} \cdot p(Z^n|O^n,m^n,\bar{\Theta})\, dZ^n \\
& \quad + \sum_{n=1}^{N} \log p(m^n|\Theta) \\
& \Big\} \cdot p(\{X\}^N|\{O\}^N,\bar{\Theta}) \qquad (3.18)
\end{aligned}$$

Substituting Eqn.(3.14) into the above equation, using the property shown in Eqn.(3.16), changing the order of the summations, and using the common variable

m to replace all m^n's, we can have (derivation is given in appendix C)

$$\begin{aligned} Q(\Theta|\bar{\Theta}) &= \sum_{n=1}^{N}\sum_{m=1}^{M} \{ \int \{ \log p(Z_0^n|m) \\ &\qquad + \sum_{k=1}^{K_n} \left[\log p(Z_{k,m}^n|Z_{k-1,m}^n) + \log p(O_k^n|Z_{k,m}^n)\right] \\ &\qquad \} \cdot p(Z^n|O^n, m, \bar{\Theta})\, dZ^n \} \cdot \bar{\omega}_m^n \\ &\quad + \sum_{n=1}^{N}\sum_{m=1}^{M} \log p(m|\Theta) \cdot \bar{\omega}_m^n \end{aligned} \tag{3.19}$$

where $\bar{\omega}_m^n$ has the same expression as ω_m^n except that the Θ in the expression is replaced by $\bar{\Theta}$ of the previous EM iteration.

We can express $Q(\Theta|\bar{\Theta})$ above as

$$Q(\Theta|\bar{\Theta}) = Q_Z + Q_p. \tag{3.20}$$

where

$$\begin{aligned} Q_Z &= \sum_{n=1}^{N}\sum_{m=1}^{M} \{ \int \{\log p(Z_0^n|m) + \sum_{k=1}^{K_n} [\log p(Z_{k,m}^n|Z_{k-1,m}^n) + \log p(O_k^n|Z_{k,m}^n)]\} \\ &\qquad \cdot p(Z^n|O^n, m, \bar{\Theta})\, dZ^n \} \cdot \bar{\omega}_m^n \end{aligned} \tag{3.21}$$

and

$$Q_p = \sum_{n=1}^{N}\sum_{m=1}^{M} \log p(m|\Theta) \cdot \bar{\omega}_m^n. \tag{3.22}$$

From the model definition by Eqns. (3.6) and (3.7), $p(Z_{k,m}^n|Z_{k-1,m}^n)$ is a Gaussian with mean $\Phi_m Z^n(k-1) + (I - \Phi_m)T_m$ and covariance Q_m, and $p(O_k^n|Z_{k,m}^n)$ is a Gaussian as well with mean $\dot{H}_m \dot{Z}^n(k)$ and covariance R_m. Fixing $p(Z_0^n|m)$ as a Gaussian with zero mean and a given covariance, we simplify Q_Z to

$$\begin{aligned} Q_Z &= -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} \{K_n \log|Q_m| + K_n \log|R_m| \\ &\qquad + \sum_{k=1}^{K_n} E_m[e1_{k,m}^{n}{}'(Q_m)^{-1} e1_{k,m}^n] \end{aligned}$$

$$+\sum_{k=1}^{K_n} E_m[e2_{k,m}^{n\,\prime}(R_m)^{-1}e2_{k,m}^n]\ \} \cdot \bar{\omega}_m^n$$
$$+const. \tag{3.23}$$

where $e1_{k,m}^n = Z^n(k) - \Phi_m Z^n(k-1) - (I - \Phi_m)T_m$ and $e2_{k,m}^n = O^n(k) - \dot{H}_m \dot{Z}^n(k)$. $E_m[\cdot]$ represents $E[\cdot|O^n, m, \bar{\Theta}]$.

3.4.4 M-step

All parameters will be re-estimated in this step.

Re-estimating π_m

π_m, $m = 1, 2, \ldots, M$, is the mixture weighting probability, which is equal to $P(m|\Theta)$. Since in the Q-function only Q_p is related to π_m, we can obtain the re-estimation formula by setting the partial derivative of Q_p with respect to π_m to zero, and then solving it subject to the constraint:

$$\sum_{m=1}^{M} \pi_m = 1.$$

To proceed, we define the Lagrangian equation of

$$L_p = Q_p + \lambda(1 - \sum_{m=1}^{M} \pi_m). \tag{3.24}$$

Taking the derivative of L_p with respect to π_m, we obtain

$$\frac{\partial L_p}{\partial \pi_m} = \sum_{n=1}^{N} \frac{1}{\pi_m} \bar{\omega}_m^n - \lambda. \tag{3.25}$$

Setting the derivative equal to zero, we have the re-estimate for π_m:

$$\hat{\pi}_m = \frac{1}{\lambda} \sum_{n=1}^{N} \bar{\omega}_m^n. \tag{3.26}$$

Taking $\sum_{m=1}^{M}$ on both sides of Eqn. (3.26) and using the property of $\sum_{m=1}^{M} \pi_m = 1$, we obtain

$$\lambda = \sum_{n=1}^{N} \sum_{m=1}^{M} \bar{\omega}_m^n. \tag{3.27}$$

This gives the re-estimation formula for π_m:

$$\hat{\pi}_m = \frac{\sum_{n=1}^{N} \bar{\omega}_m^n}{\sum_{n=1}^{N} \sum_{m=1}^{M} \bar{\omega}_m^n} = \frac{\sum_{n=1}^{N} \bar{\omega}_m^n}{N} \quad \textit{for} \ \ 1 \leq m \leq M. \tag{3.28}$$

Re-estimating Φ_m and T_m

Before deriving the re-estimation formula for these new parameters, we adopt the following notations first for notational simplicity:

$$\begin{array}{ll}
A0_m^n = \sum_{k=1}^{K_n} E_m[Z^n(k-1)Z^n(k-1)'], & A1_m^n = \sum_{k=1}^{K_n} E_m[Z^n(k)Z^n(k)'], \\
A2_m^n = \sum_{k=1}^{K_n} E_m[Z^n(k)Z^n(k-1)'], & C_m = (I - \hat{\Phi}_m)\hat{T}_m, \\
B0_m^n = \sum_{k=1}^{K_n} E_m[Z^n(k-1)], & B1_m^n = \sum_{k=1}^{K_n} E_m[Z^n(k)], \\
D^n = \sum_{k=1}^{K_n} O^n(k)(O^n(k))', & F_m^n = \sum_{k=1}^{K_n} O^n(k)E_m[\dot{Z}^n(k)]' \\
G_m^n = \sum_{k=1}^{K_n} E_m[\dot{Z}^n(k)(\dot{Z}^n(k))']. &
\end{array}$$

where $\hat{\Phi}_m$ and $\hat{T}_m$ stand for the newly re-estimated values,

$$E_m[\dot{Z}^n(k)] = [1, E_m[Z^n(k)]']'$$

and

$$E_m[\dot{Z}^n(k)(\dot{Z}^n(k))'] = \begin{bmatrix} 1 & E_m[Z^n(k)]' \\ E_m[Z^n(k)] & E_m[Z^n(k)(Z^n(k))'] \end{bmatrix}.$$

To re-estimate Φ_m and T_m we note that they are related only to Q_Z. Furthermore, only $e_{k1,m}$ includes Φ_m and T_m. The relevant partial derivatives are [2]

$$\begin{aligned}
\frac{\partial Q_Z}{\partial \Phi_m} &= Q_m^{-1} \sum_{n=1}^{N} (-A2_m^n + B1_m^n T_m' + T_m B0_m^{n\,\prime} - K_n T_m T_m') \cdot \bar{\omega}_m^n \\
&\quad + Q_m^{-1} \Phi_m \sum_{n=1}^{N} (A0_m^n - B0_m^n T_m' - T_m B0_m^{n\,\prime} + K_n T_m T_m') \cdot \bar{\omega}_m^n
\end{aligned} \tag{3.29}$$

[2]In deriving the derivatives, we use the same matrix calculus formulas listed in Chapter 2.

and

$$\frac{\partial Q_Z}{\partial T_m} = -Q_m^{-1}(I-\Phi_m)\sum_{n=1}^{N}\{B1_m^n - \Phi_m B0_m^n - K_n(I-\Phi_m)T_m\}\cdot\bar{\omega}_m^n \tag{3.30}$$

Setting the above derivatives to zero, we obtain the re-estimates for Φ_m and T_m:

$$\begin{aligned}\hat{\Phi}_m &= \left\{\sum_{n=1}^{N}(A2_m^n - B1_m^n\hat{T}_m' - \hat{T}_m B0_m^{n\prime} + K_n\hat{T}_m\hat{T}_m')\cdot\bar{\omega}_m^n\right\} \\ &\quad\cdot\left\{\sum_{n=1}^{N}(A0_m^n - B0_m^n\hat{T}_m' - \hat{T}_m B0_m^{n\prime} + K_n\hat{T}_m\hat{T}_m')\cdot\bar{\omega}_m^n\right\}^{-1}\end{aligned} \tag{3.31}$$

and

$$\hat{T}_m = \frac{(I-\Phi_m)^{-1}\sum_{n=1}^{N}\{B1_m^n - \Phi_m B0_m^n\}\cdot\bar{\omega}_m^n}{\sum_{n=1}^{N}K_n\cdot\bar{\omega}_m^n}. \tag{3.32}$$

Note that in the above, the parameters Φ_m and T_m are updated alternatively at separate EM iterations. This gives rise to the generalized EM algorithm, i.e., local optimization in the M-step, rather than global optimization.

Re-estimating $\dot{H}_m$

To re-estimate $\dot{H}_m$, we note that it is included only in $e_{k2,m}$ term of the Q_Z in Eqn.(3.23). The relevant partial derivative is

$$\frac{\partial Q_Z}{\partial \dot{H}_m} = -\sum_{n=1}^{N}\bar{\omega}_m^n\sum_{k=1}^{K_n}R_m^{-1}E_m[(\dot{H}_m\dot{Z}^n(k) - O_k^n)(\dot{Z}^n(k))']\tag{3.33}$$

Setting the above to zero, we have the re-estimate:

$$\hat{\dot{H}}_m = \left\{\sum_{n=1}^{N}\bar{\omega}_m^n\cdot F_m^n\right\}\left\{\sum_{n=1}^{N}\bar{\omega}_m^n\cdot G_m^n\right\}^{-1} \tag{3.34}$$

Re-estimating Q_m and R_m

Since the noise covariances Q_m and R_m are included only in Q_Z, we compute the following derivatives:

$$\frac{\partial Q_Z}{\partial Q_m^{-1}} = \frac{1}{2}\sum_{n=1}^{N} K_n Q_m \bar{\omega}_m^n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[e1_{k,m}^n e1_{k,m}^{n\,'}]\cdot\bar{\omega}_m^n. \tag{3.35}$$

and

$$\frac{\partial Q_Z}{\partial R_m^{-1}} = \frac{1}{2}\sum_{n=1}^{N} K_n R_m \bar{\omega}_m^n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[e2_{k,m}^n e2_{k,m}^{n\,'}]\cdot\bar{\omega}_m^n \tag{3.36}$$

Let the derivatives equal to zero, we obtain the estimates for Q_m and R_m.

$$\hat{Q}_m = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[e1_{k,m}^n e1_{k,m}^{n\,'}]\cdot\bar{\omega}_m^n}{\sum_{n=1}^{N} K_n\bar{\omega}_m^n} \tag{3.37}$$

and

$$\hat{R}_m = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[e2_{k,m}^n e2_{k,m}^{n\,'}]\cdot\bar{\omega}_m^n}{\sum_{n=1}^{N} K_n\bar{\omega}_m^n} \tag{3.38}$$

In $\hat{Q}_m$ above, $\sum_{k=1}^{K_n} E_m[e1_{k,m}^n e1_{k,m}^{n\,'}]$ is calculated according to

$$\begin{aligned}\sum_{k=1}^{K_n} E_m[e1_{k,m}^n e1_{k,m}^{n\,'}] &= A1_m^n + \hat{\Phi}_m A0_m^n \hat{\Phi}_m' - A2_m^n \hat{\Phi}_m' \\ &\quad -\hat{\Phi}_m (A2_m^n)' - B1_m^n (C_m)' - C_m (B1_m^n)' \\ &\quad +\hat{\Phi}_m B0_m^n (C_m)' + C_m (B0_m^n)'\hat{\Phi}_m' + K_n C_m C_m' \end{aligned} \tag{3.39}$$

In $\hat{R}_m$ above, $\sum_{k=1}^{K_n} E_m[e2_{k,m}^n e2_{k,m}^{n\,'}]$ is calculated according to

$$\sum_{k=1}^{K_n} E_m[e2_{k,m}^n e2_{k,m}^{n\,'}] = D^n - F_m^n(\hat{H}_m)' - \hat{H}_m(F_m^n)' + \hat{H}_m G_m^n (\hat{H}_m)' \tag{3.40}$$

3.4.5 Calculation of the sufficient statistics

In order to obtain the re-estimates for the model parameters according to the formulas derived above as the M-step of the EM algorithm, a set of conditional expectations need to be calculated. Essentially, three conditional expectations (all conditioned on the observation sequences), $E_m[Z^n(k)]$, $E_m[Z^n(k)Z^n(k)']$ and $E_m[Z^n(k)Z^n(k-1)']$, are required by the M-step.

The conditional expectation $E_m[\cdot]$, which denotes $E[\cdot|O^n, m, \bar{\Theta}]$, is precisely the Kalman smoother for the m-th mixture and for the n-th observation (token). All conditional expectations required in the M-step can be calculated by using the results of the Kalman smoothing algorithm. We now list the computational steps of the Kalman smoothing algorithm in the following below. [3]

Forward recursion (or Kalman filtering) :

$$\hat{Z}^n_{k|k-1,m} = \Phi_m \hat{Z}^n_{k-1|k-1,m} + (I - \Phi_m)T_m \tag{3.41}$$

$$\Sigma^n_{k|k-1,m} = \Phi_m \Sigma^n_{k-1|k-1,m} \Phi_m + Q_m \tag{3.42}$$

$$\tilde{O}^n_{k,m} = O^n(k) - \dot{H}_m \dot{\hat{Z}}^n_{k|k-1,m} \tag{3.43}$$

$$\Sigma^n_{\tilde{O}_{k,m}} = H_m \Sigma^n_{k|k-1,m} H'_m + R_m \tag{3.44}$$

$$K_{k,m} = \Sigma^n_{k|k-1,m} H'_m (\Sigma^n_{\tilde{O}_{k,m}})^{-1} \tag{3.45}$$

$$\hat{Z}^n_{k|k,m} = \hat{Z}^n_{k|k-1,m} + K_{k,m}\tilde{O}^n_{k,m} \tag{3.46}$$

$$\Sigma^n_{k|k,m} = \Sigma^n_{k|k-1,m} - K_{k,m}\Sigma^n_{\tilde{O}_{k,m}} K'_{k,m} \tag{3.47}$$

Backward recursion (or Kalman smoothing) :

$$A^n_{k,m} = \Sigma^n_{k|k,m} \Phi'_m (\Sigma^n_{k|k-1,m})^{-1} \tag{3.48}$$

$$\hat{Z}^n_{k|K_n,m} = \hat{Z}^n_{k|k,m} + A^n_{k,m}[\hat{Z}^n_{k+1|K_n} - \hat{Z}^n_{k+1|k,m}] \tag{3.49}$$

$$\Sigma^n_{k|K_n,m} = \Sigma^n_{k|k,m} + A^n_{k,m}[\Sigma^n_{k+1|K_n,m} - \Sigma^n_{k+1|k,m}]A'_k \tag{3.50}$$

Based on the Kalman smoothing results above, the three required conditional

[3]Details of the algorithm and the derivations can be found in ([94], [84], [93], [77], [89]).

expectations are computed as follows:

$$E_m[Z^n(k)] = \hat{Z}^n_{k|K_n,m} \tag{3.51}$$

$$E_m[Z^n(k)Z^n(k)'] = \Sigma^n_{k|K_n,m} + \hat{Z}^n_{k|K_n,m}(\hat{Z}^n_{k|K_n,m})' \tag{3.52}$$

$$E_m[Z^n(k)Z^n(k-1)'] = \Sigma^n_{k,k-1|K_n,m} + \hat{Z}^n_{k|K_n,m}(\hat{Z}^n_{k-1|K_n,m})' \tag{3.53}$$

where $\Sigma^n_{k,k-1|K_n,m}$ is recursively calculated by [40]

$$\Sigma^n_{k,k-1|K_n,m} = \Sigma^n_{k|k,m}{A^n_{k-1,m}}' + A^n_{k,m}(\Sigma^n_{k+1,k|K_n,m} - \Phi_m\Sigma^n_{k|k,m}){A^n_{k-1,m}}' \tag{3.54}$$

for $k = K_n, \cdots, 2$, where

$$\Sigma^n_{K_n,K_n-1|K_n,m} = (I - K_{K_n,m}H_m)\Phi_m\Sigma^n_{K_n-1|K_n-1,m} \tag{3.55}$$

3.4.6 Updating ω

To update $\bar{\omega}^n_m$ according to Eqn.(3.15), $p(O^n|m, \bar{\Theta})$ must be calculated. It is calculated from the innovation sequence.

$$p(O^n|m, \bar{\Theta}) = \prod_{k=1}^{K_n}(2\pi)^{-\frac{d}{2}}\left|\Sigma^n_{\tilde{O}_{k,m}}\right|^{-\frac{1}{2}}exp\{-\frac{1}{2}(\tilde{O}^n_{k,m})'[\Sigma^n_{\tilde{O}_{k,m}}]^{-1}\tilde{O}^n_{k,m}\} \tag{3.56}$$

where $\tilde{O}_{k,m}$ and $\Sigma^n_{\tilde{O}_{k,m}}$ are the mean and covariance of the innovation at time K, respectively. They are computed directly from Kalman Filter described earlier. d is the dimension of $\tilde{O}_{k,m}$.

3.5 Likelihood-Scoring Algorithm

The speech model presented so far combines M different linear dynamic models (mixture models), according to the mixture weighting probabilities, to describe the

VTR dynamics. After the weighting probabilities and all other models' parameters are trained as described in the previous section, the likelihood of the model for each phone, given a sequence of observations, can be computed directly. We describe this computation below.

The likelihood $l(O|\Theta)$ is equal to

$$\begin{aligned} l(O|\Theta) &= \sum_X p(O,X|\Theta) = \sum_X p(O|X,\Theta)p(X|\Theta) \\ &= \sum_{m=1}^{M} \pi_m \cdot l_m(O|\Theta) \end{aligned} \tag{3.57}$$

Based on the estimation theory for dynamic systems ([94], [93], [54], etc.), the likelihood function for each individual mixture model is calculated from its innovation sequence $\tilde{O}_{k,m}$ according to

$$l_m(O|\Theta) = K_c \cdot exp\left[-\frac{1}{2}\sum_{k=1}^{K}\{\log|\Sigma_{\tilde{O}_{k,m}}| + \tilde{O}_{k,m}[\Sigma_{\tilde{O}_{k,m}}]^{-1}\tilde{O}'_{k,m}\}\right]. \tag{3.58}$$

where the innovation sequence $\tilde{O}_{k,m}$ and its covariance $\Sigma_{\tilde{O}_{k,m}}$ are computed from the Kalman filtering recursion. K_c is a constant.

Then log-likelihood for the entire mixture model becomes:

$$L(O|\Theta) = log\left[\sum_{m=1}^{M} \pi_m \cdot l_m(O|\Theta)\right] \tag{3.59}$$

For a speech utterance which consists of a sequence of phones with the phones' dynamic regimes given, the log-likelihood for each phone in the sequence as defined in Eqn.(3.59) are summed to give the total log-likelihood score for the entire utterance.

3.6 Evaluation Experiments

As before, in all the experiments reported in this section, we use a N-best list re-scoring paradigm to evaluate the new recognizer on the Switchboard spontaneous telephony speech data.

3.6.1 Speech model trained with one speaker's data

In this set of experiments, the **"1/2 hour"** training set is used for model training. We gradually increase the number of mixtures in the MLDM. The results (WER) are listed in Table (3.2). From one mixture to two mixtures, there are large improvements in the performance. The WER drops from "55.7%" to "50.7%" (about 10% relative error reduction) for the "Ref+100" case. There is about two percents absolute error reduction for the "100-best" case (without references included).

When we further increase the number of mixtures to four, we don't observe further decrease in the WER from the results shown in Table (3.2). Two factors might account for this observation. First, the amount of training data is not enough for the increased number (4) of mixtures. We had some warning information during the training, which said some models suffer from under-training. Second, the confusability among the different phones are increased with the increasing number of mixtures. In order to pin down the more likely cause between the two possibilities, we used more training data to train the models in the next set of experiments.

Under identical conditions, compared with the **"HMM-baseline"** system, the MLDM with two mixtures achieves about 2% absolute WER reduction for "100 best" case and much more reduction for "Ref+100" case. As analyzed before, this situation is caused by the error propagation problem in the new model.

3.6.2 Speech model trained with multiple speakers' data

To investigate how the amount of training data and number of mixtures affect the recognizer performance, we extracted more data from the Switchboard **"train-ws97-**

systems	Ref+100	100-best
MLDM:1-mix	55.7	58.9
MLDM:2-mix	50.7	57.0
MLDM:4-mix	50.7	57.7
HMM-baseline	56.1	58.9

Table 3.2: Performance (WER) of mixture linear dynamic system model with half an hour of training data

a" training set for training the new model.

In Table 3.3, "1 hour" means that we added another half an hour data to the original half an hour of training set, where the new half an hour of the training data comes from 30 different speakers. We name it **"1 hour"** training set. "2 hour" means that we added one more hour of training data to the **"1 hour"** training set. The additional one hour of data comes from 50 different speakers. We name it **"2 hour"** training set. Both training sets will be used again in the next chapter.

The re-scoring results are listed in Table 3.3. With one hour training data, from two mixtures to four mixtures, improvement is observed. It testifies the conclusion that half an hour training data is not enough.

Comparing the results shown in Table (3.3) and Table (3.2), the MLDM with two mixtures does not make substantial performance differences by doubling the amount of training data. That is because the number of mixtures is not enough. the MLDM with four mixtures turns out better performance from half an hour training data to one hour data, about 1% absolute WER reduction for both the "100 best" and "Ref+100" cases, It produces further WER reduction when trained on two hour training data. Therefore, more improvement should be expected with more training data coming.

From this set of experiments, we conclude that using more mixtures is able to improve the model's performance even though it may have simultaneously increased the confusions among the phones. More training data is needed for the training of the models with an increasing number of mixture components.

systems	Ref+100	100-best
MLDM:2-mix (1 hour)	51.0	57.1
MLDM:4-mix (1 hour)	49.8	56.6
MLDM:4-mix (2 hour)	49.5	56.0
ws97-baseline (160 hour)	56.2	56.9

Table 3.3: Performance (WER) of mixture linear dynamic system model with increased amounts of training data

Compared with the **"ws97-baseline"** HMM system (not under identical conditions), given in Table (3.3) as well, the MLDM with four mixtures trained on 2 hour data outperforms the HMM system for all cases, with and without including the references in the N-best list. Especially, in the situation with references included, the new model gives more than 10% relative error reduction for the "Ref+100" case. This means that the new model is able to score the correct reference hypotheses with higher likelihoods than the HMM system. For the case without references included, the new model achieves about 1.0% absolute WER reduction.

3.7 An Analysis Experiment

Why is the new dynamic system model able to achieve the better performance than HMM system ? We believe the main reason is that some key aspects of true dynamic properties of speech has been explicitly incorporated into the new system. In order to examine this belief, we have performed a set of analysis experiments by deliberately modifying the dynamic property in the model. To do this, we set the "time-constant" parameter, Φ, for all models to zero. This changes the state equation to

$$Z(k) = T_m + W_m(k-1), \qquad m = 1, 2, ..., M. \tag{3.60}$$

Now, the hidden state dynamics is modified to be a flat one with noise added. All other parts of the speech model were kept identical to the system described earlier. We carried out the speech recognition experiments using the same training data (1-hour) and using the same N-best re-scoring paradigm. The linear dynamic system model with 4 mixture components was used.

The log-likelihoods during the model training are plotted in Fig(3.2), as a function of the EM iteration number. The solid line is associated with the model without the state dynamics being modified (i.e. with the trained parameter Φ), and the dashed line is associated with the model with the state dynamics modified (i.e. setting $\Phi = 0$). The log-likelihood of the model after modifying the state dynamics is observed to be uniformly lower than that of the original model. This is especially so at the early iteration of the EM algorithm.

The N-best re-scoring results using the speech model with the state dynamics modified are listed in Table (3.4). For comparison purposes, the results using the original model are also shown in the same table. We observe from Table (3.4) that when the dynamic property is modified by setting $\Phi = 0$, the system performance is worsened greatly.

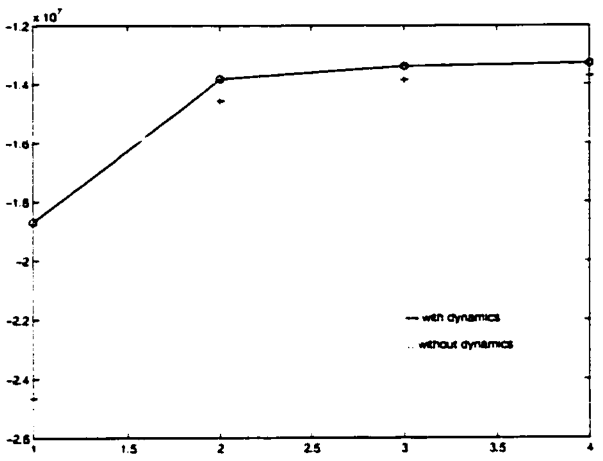


Figure 3.2: Comparison of log-likelihoods of the speech model with and without the state dynamics being modified

systems	Ref+100	100-best
MLDM:4-mix ($\Phi = 0$)	53.3	57.9
MLDM:4-mix (Φ trained)	49.8	56.6

Table 3.4: Comparison of recognizer WERs using the speech model with and without the state dynamics being modified

Compared with the **"ws97-baseline"** HMM system, the system with the state dynamics modified still performs better. It may be due to the use of Kalman filter. On one hand, it enables the new model to compute likelihood without the i.i.d. assumption of observations (which is a limitation for HMM). On the other hand, note that during the Kalman filtering process, due to the noises added to the state and observation equations, the estimated hidden states, $\hat{Z}_{k|k-1}$ is still varying with time k. This causes the estimated mean of observation, $\dot{H}\hat{Z}_{k|k-1}$, and its covariance, $H\Sigma_{k|k-1}H'$, to change with time as well. Such changes are more desirable than the constant means and variances associated with individual states in the conventional HMM.

3.8 Conclusion

In this chapter a new version of the target-directed dynamic model, a mixture linear dynamic model for speech recognition has been developed. This new version originates from the work in the previous chapter (more details in [3, 33]), where a nonlinear form of the observation equation was used to represent the nonlinear relationship between the hidden VTR space and the acoustic space (MFCC).

To improve the approximation accuracy by use of linear functions, we have developed a mixture linear dynamic model. The basic idea underlying this development is that within limited input and output spaces, a global nonlinear relation can be relatively accurately approximated by combining a set of linear regression functions. Applying this idea to the speech modeling problem, we approximate the nonlinear relation between the VTR space and the MFCC space for each separate phone by a mixture of static, linear regression functions. The division of the input-output spaces is achieved in two ways. First, phone-dependent linear mapping characterized by the regression matrix parameter $\dot{H}$ allows the VTR space to limit itself to only a narrow range of the variation specific to the phone. Second, further limitation of the VTR changes in a phone is achieved by using a set of regression matrices $\dot{H}_m$'s, each of which further narrows the range of the VTR variation.

A series of speech recognition experiments have been carried out on the Switchboard database to evaluate the new model. Under identical conditions, compared with the **"HMM-baseline"** system, With use of two mixture components and of half an hour for model training, the new model is able to achieve better performance, about 2% WER reduction for the case without references included and much more reduction for the case with reference included. When the number of mixtures is increased to four and the amount of training data increased to two hours (about 81 different speakers), the new model also outperforms the **"ws97-baseline"** HMM system trained on 160 hour data in all cases (with and without references included).

Those experiment results are consistent with the observation by other researchers [60]. Our new model is a segmental-level discrete mixture model as those in [53], [57], etc.. In [60] the authors observed that "*If the advantage of frame-level mixture distributions stems from systematic variation in speech, then segmental mixtures may be able to represent the systematic component via a framework that keeps the mixture mode constant across the segment. In contrast, the frame-level mixture model allows mixture modes to change randomly at each time step. Of course, if the advantage of frame-level mixtures is simply that Gaussian models do not fit the data well, then frame-based mixtures will be a more efficient representation than segmental mixtures. This question must be answered empirically and remains open at this point, although our intuition and preliminary experiments favor the systematic variation interpretation.*" Our experiments also favor the speech systematic variation interpretation.

It shows that the target-directed, mixture linear dynamic system model proposed in this chapter is a promising new approach to spontaneous speech recognition.

Chapter 4

A Mixture Linear Dynamic Model with Switching Parameters on the Measurement Equations

4.1 Introduction

In the previous chapter the mixture linear dynamic model was developed and evaluated. It turned out promising results. However, it used a linear function in the measurement equations to approximate the physically nonlinear relationship between the hidden dynamic space (VTR) and the observation space (MFCC). That approximation is not accurate, especially for some consonants. For example, stop consonants have closures followed by explosives, each period has different phonetic and acoustic property. Using a single linear function to represent those two different periods is apparently not accurate.

As analyzed in the previous chapter, linearity is of importance for the dynamic model because it gives rise to efficiency in the model training and likelihood computation processes. To overcome MLDM's weakness in representing the nonlinear relationship and keep the linearity of the model simultaneously, a more general version of the dynamic model is developed in this chapter . In this version the $\dot{H}_m$ in the measurement equation of the MLDM version (Eqn.(3.7)) is allowed to vary with time k, but not continuously, it is confined to take values from a set of different $\dot{H}$ values. Therefore, a piece-wise linear function is used to approximate the physically nonlinear relationship. Obviously, the new version is a more general case of the MLDM version.

Since a piece-wise linear function is able to approximate a nonlinear function more accurately than a single linear one, we hope to improve the system's performance further by developing this new version.

4.2 Model Formulation

In this new version each phone is still modeled by a combination of several linear dynamic models.

$$\sum_{m=1}^{M} \pi_m \; LDM_m \tag{4.1}$$

where M is the total number of linear dynamic models (or mixtures), π_m is the combining weights. However, the m-th linear dynamic model LDM_m becomes different from that in the MLDM version, it is formulated as follows:

$$\begin{aligned} Z(k) &= \Phi_m Z(k-1) + (I - \Phi_m)T_m + W_m(k-1) & (4.2) \\ O(k) &= \dot{H}_m(k)\dot{Z}(k) + V_m(k) & (4.3) \end{aligned}$$

where the state equation is identical to that in MLDM, but the measurement equation is different. Here $\dot{H}_m(k)$ changes with time k, but it is confined to take values from a set of $\{\dot{H}_m^l,\ l = 1, 2, ..., L\}$, L is the total number of $\dot{H}$ values. Note that the subscript m indicates different mixtures have different sets of $\dot{H}$ values.

We call this new version a mixture linear dynamic model with switching parameters on measurement equations (MLDM-SM). Each MLDM-SM has model parameters

$$\Theta = \{\pi_m, \Phi_m, T_m, Q_m, R_m, \dot{H}_m^l, \gamma_{m,l},\quad m = 1, 2, ..., M,\ l = 1, 2, ..., L\},$$

where π_m represents the mixture weight probability $P(m|\Theta)$ and $\gamma_{m,l}$ the $\dot{H}_m$ value weight probability $P(l|m, \Theta)$. How those parameters are estimated will be derived in the next section.

In this model, two levels of parameter switching have been designed. First, the mixture indexed by m switches at the segment (phone) level. Second, the $\dot{H}_m$ values indexed by l switch at the time frame level. The first level of switching corresponds to the target property of the VTR dynamics, which therefore must be at the segment level. The second level of parameter switching is designed to provide the flexibility for using multiple linear functions to approximate the nonlinear relationship between the VTR and the measurement variables. The parameter switching in the second level should happen at the frame level because the relationship can change at different time periods. We call the former switching *mixture switching* and the later one *H-value switching.*

As in MLDM, the mixture-path constraint must be imposed on the *mixture switching.* This constraint must be imposed on both the model training and likelihood computation algorithms.

4.3 Model Parameter Learning

Due to data incompleteness, Expectation-Maximization (EM) algorithm is adopted for model parameter estimation.

As in MLDM version. we define a discrete random variable X to indicate the observation-to-mixture assignment for a sequence of observation. For this new MLDM-SM version, we need one additional discrete variable to represent the H value switching on the measurement equation. We define it as $Y = \{y_1, y_2, ..., y_K\}$ (K is the length of the observation), $y_k(1 \leq k \leq K)$ is a discrete random variable which indicates which one of $\dot{H}^l_m(1 \leq l \leq L)$ is chosen (or is switched onto) at time frame k. For example, if $y_k = i$, it means the i-th value, $\dot{H}^i_m$, is chosen at time k. Finally, we define a discrete variable, $S = \{X, Y\}$, to represent the combination of X and Y.

As in MLDM, to impose the mixture-path constraint a joint variable must be defined at segment level. Suppose we have N training tokens for a phone, we define a joint variable as

$$\{O, Z, S\}^N = \{(O^1, Z^1, S^1), (O^2, Z^2, S^2), ..., (O^n, Z^n, S^n), ..., (O^N, Z^N, S^N)\},$$

where $O^n = \{O^n(1), O^n(2), \cdots, O^n(K_n)\}$ is the n-th observation sequence and Z^n the corresponding hidden state sequence. $S^n = \{X^n, Y^n\}$, where X^n indicates the observation-to-mixture assignment and $Y^n = \{y^n_1, y^n_2, \cdots, y^n_{K_n}\}$ describes the $\dot{H}$ value switching. Here, X^n and Y^n are working together to determine how the observation sequence is generated.

The following assumptions are made in the development of the learning algorithm:

- The N tokens are independent of each other. So, S^n $(1 \leq n \leq N)$ are independent of each other.

- The discrete random variables X^n ($1 \le n \le N$) have an identical distribution.
- y_1^n, y_2^n, $\cdots$, and $y_{K_n}^n$ are independent of each other. That is, the parameter switching does not depend on the history, nor on the future.
- y_k^n $(1 \le k \le K_n)$ have an identical distribution.

4.3.1 The PDF of the joint variable

With the assumptions made above, the conditional PDF of the joint variable $\{O, Z, S\}^N$ given Θ can be written as

$$\begin{aligned} p(\{O, Z, S\}^N|\Theta) &= \prod_{n=1}^{N} p(O^n, Z^n, S^n|\Theta) \\ &= \prod_{n=1}^{N} p(O^n, Z^n|S^n, \Theta)\ P(S^n|\Theta) \end{aligned} \tag{4.4}$$

In Eqn.(4.4) $p(O^n, Z^n|S^n, \Theta)$ is the conditional joint PDF of O^n and Z^n given the condition that model parameter Θ is known and the mixture and H value switchings designated. According to [39], the conditional PDF $p(O^n, Z^n|S^n, \Theta)$ is defined as

$$p(O^n, Z^n|S^n, \Theta) = p(Z_0^n|S^n, \Theta) \prod_{k=1}^{K_n} p(Z_k^n|Z_{k-1}^n, S^n, \Theta)\ p(O_k^n|Z_k^n, S^n, \Theta) \tag{4.5}$$

where $p(Z_0^n|S^n, \Theta)$ is the conditional distribution of the initial value for the hidden dynamics. We assume all tokens and mixtures have the same initial value distributions, so $p(Z_0^n|S^n, \Theta)$ is simplified to $p(Z_0|\Theta)$. At time k, only X^n in S^n affects the conditional PDF $p(Z_k^n|Z_{k-1}^n, S^n, \Theta)$ and only X^n and y_k^n affects $p(O_k^n|Z_k^n, S^n, \Theta)$, so Eqn.(4.5) can be further written as

$$p(O^n, Z^n|S^n, \Theta) = p(Z_0|\Theta) \prod_{k=1}^{K_n} p(Z_k^n|Z_{k-1}^n, X^n, \Theta)\ p(O_k^n|Z_k^n, X^n, y_k^n, \Theta) \tag{4.6}$$

On the other hand, in Eqn.(4.4), $P(S^n|\Theta)$ is equal to

$$\begin{aligned} P(S^n|\Theta) &= P(Y^n|X^n,\Theta)P(X^n|\Theta) \\ &= \prod_{k=1}^{K_n} P(y_k^n|X^n,\Theta)P(X^n|\Theta) \end{aligned} \tag{4.7}$$

where the independence between switchings at different time points has been used in the last step.

Then, substituting Eqn.(4.6) and (4.7) into (4.4), we obtain

$$\begin{aligned} & p(\{O,Z,S\}^N|\Theta) \\ = & \prod_{n=1}^{N} p(Z_0|\Theta)\left[\prod_{k=1}^{K_n} p(Z_k^n|Z_{k-1}^n,X^n,\Theta)\; p(O_k^n|Z_k^n,X^n,y_k^n,\Theta)P(y_k^n|X^n,\Theta)\right] \\ & \cdot P(X^n|\Theta) \end{aligned} \tag{4.8}$$

4.3.2 Several useful conditional PDFs

To calculate the Q-function in the E-step, we need several conditional PDFs, which we will derive here.

Firstly, $p(\{O,S\}^N|\Theta)$ can be factorized to

$$p(\{O,S\}^N|\Theta) = \prod_{n=1}^{N} p(O^n|S^n,\Theta)\; P(S^n|\Theta) \tag{4.9}$$

In the above, $p(O^n|S^n,\Theta)$ can be further decomposed to

$$p(O^n|S^n,\Theta) = \prod_{k=1}^{K_n} p(O^n(k)|O_{1,k-1}^n,X^n,y_k^n,\Theta) \tag{4.10}$$

where $O_{1,k-1}^n = \{O^n(1),O^n(2),\cdots,O^n(k-1)\}$ and we assume $p(O^n(1)|O_0^n,X^n,y_1^n,\Theta) = p(O^n(1)|X^n,y_1^n,\Theta)$.

Plugging Eqn.(4.10) and (4.7) into (4.9), we get

$$\begin{aligned} & p(\{O,S\}^N|\Theta) \\ = & \prod_{n=1}^{N}\left[\prod_{k=1}^{K_n} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta) \qquad (4.11) \end{aligned}$$

Secondly, to obtain $p(\{O,X\}^N|\Theta)$ we take integration of $p(\{O,S\}^N|\Theta)$ over $\{Y\}^N$.

$$\begin{aligned} p(\{O,X\}^N|\Theta) &= \sum_{\{Y\}^N} p(\{O,S\}^N|\Theta) \\ &= \sum_{Y^1}\sum_{Y^2}\cdots\sum_{Y^N}\prod_{n=1}^{N}\left[\prod_{k=1}^{K_n} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta) \\ &= \prod_{n=1}^{N}\left[\prod_{k=1}^{K_n}\sum_{y^n_k=1}^{L} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta) \qquad (4.12) \end{aligned}$$

where the independence between tokens and between switchings at different time points have been used.

Thirdly, to obtain $p(\{O\}^N|\Theta)$, we take integration of $p(\{O,X\}^N|\Theta)$ over $\{X\}^N$,

$$\begin{aligned} p(\{O\}^N|\Theta) &= \sum_{\{X\}^N} p(\{O,X\}^N|\Theta) \\ &= \prod_{n=1}^{N}\sum_{X^n=1}^{M}\prod_{k=1}^{K_n}\left[\sum_{y^n_k=1}^{L} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta) \qquad (4.13) \end{aligned}$$

Then, by PDF's property, $p(\{X\}^N)|\{O\}^N,\Theta)$ can be derived to be equal to

$$\begin{aligned} p(\{X\}^N)|\{O\}^N,\Theta) &= \frac{p(\{O,X\}^N|\Theta)}{p(\{O\}^N|\Theta)} \\ &= \frac{\prod_{n=1}^{N}\left[\prod_{k=1}^{K_n}\sum_{y^n_k=1}^{L} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta)}{\prod_{n=1}^{N}\sum_{X^n=1}^{M}\left[\prod_{k=1}^{K_n}\sum_{y^n_k=1}^{L} p(O^n(k)|O^n_{1,k-1},X^n,y^n_k,\Theta)\ P(y^n_k|X^n,\Theta)\right] P(X^n|\Theta)} \\ &= \prod_{n=1}^{N}\omega^n_m \qquad (4.14) \end{aligned}$$

where

$$\omega_m^n = \frac{\left[\prod_{k=1}^{K_n} \sum_{l=1}^{L} p(O^n(k)|O_{1,k-1}^n, m, l, \Theta)\ P(l|m,\Theta)\right] P(m|\Theta)}{\sum_{m=1}^{M} \left[\prod_{k=1}^{K_n} \sum_{l=1}^{L} p(O^n(k)|O_{1,k-1}^n, m, l, \Theta)\ P(l|m,\Theta)\right] P(m|\Theta)} \tag{4.15}$$

In the above, because X^n's have the identical distributions, they are replaced by a common variable m for notational simplicity. For the same reason, y_k^n's are replaced by a common l. Apparently, $\sum_{m=1}^{M} \omega_m^n = 1$. Because of the independence assumption between tokens, we can get

$$p(X^n|O^n, \Theta) = \omega_m^n \tag{4.16}$$

and

$$\sum_{\{X\}^N/X^n} p(\{X\}^N|\{O\}^N, \Theta) = \omega_m^n \tag{4.17}$$

where $\{X\}^N/X^n$ means the set of $\{X\}^N$ but X^n.

Finally, the conditional PDF $p(\{Y\}^N|\{X\}^N, \{O\}^N, \Theta)$ is equal to

$$\begin{aligned} p(\{Y\}^N|\{X\}^N, \{O\}^N, \Theta) &= \frac{p(\{O, S\}^N|\Theta)}{p(\{O, X\}^N|\Theta)} \\ &= \frac{\prod_{n=1}^{N} \left[\prod_{k=1}^{K_n} p(O^n(k)|O_{1,k-1}^n, X^n, y_k^n, \Theta)\ P(y_k^n|X^n, \Theta)\right] P(X^n|\Theta)}{\prod_{n=1}^{N} \left[\prod_{k=1}^{K_n} \sum_{y_k^n=1}^{L} p(O^n(k)|O_{1,k-1}^n, X^n, y_k^n, \Theta)\ P(y_k^n|X^n, \Theta)\right] P(X^n|\Theta)} \\ &= \prod_{n=1}^{N} \prod_{k=1}^{K_n} \xi_{k,m,l}^n \end{aligned} \tag{4.18}$$

where

$$\xi_{k,m,l}^n = \frac{p(O^n(k)|O_{1,k-1}^n, m, l, \Theta)\ P(l|m,\Theta)}{\sum_{l=1}^{L} p(O^n(k)|O_{1,k-1}^n, m, l, \Theta)\ P(l|m,\Theta)} \tag{4.19}$$

where as before X^n is replaced by m and y_k^n by l for notational simplicity. Apparently, $\sum_{l=1}^{L} \xi_{k,m,l}^n = 1$. Because of the independence among tokens, we can get

$$p(Y^n|X^n, O^n, \Theta) = \prod_{k=1}^{K_n} \xi_{k,m,l}^n. \tag{4.20}$$

Here, using the independence assumption among $\dot{H}$ value switchings at different time points, we obtain

$$p(y_k^n|X^n, O^n, \Theta) = \xi_{k,m,l}^n \tag{4.21}$$

and

$$\sum_{Y^n/y_k^n} p(Y^n|X^n, O^n, \Theta) = \xi_{k,m,l}^n \tag{4.22}$$

where Y^n/y_k^n denotes the full set of Y^n but y_k^n.

4.3.3 E-step

Given all the conditional PDF computations discussed above, we are now in a position to describe the EM algorithm.

Since both $\{Z\}^N$ and $\{S\}^N$ are missing, we take integrations over both of them to get the Q-function. The Q-function becomes

$$\begin{aligned} Q(\Theta|\bar{\Theta}) &= \sum_{\{S\}^N} \int \log p(\{O, Z, S\}^N|\Theta) \cdot p(\{Z, S\}^N|\{O\}^N, \bar{\Theta})\ d\{Z\}^N \\ &= \sum_{\{S\}^N} \int \log p(\{O, Z, S\}^N|\Theta) \cdot p(\{Z\}^N|\{O, S\}^N, \bar{\Theta})\ d\{Z\}^N \\ &\qquad \cdot p(\{S\}^N|\{O\}^N, \bar{\Theta}) \end{aligned} \tag{4.23}$$

where $\bar{\Theta}$ indicates the parameter value at the immediately previous step.

Substituting Eqn.(4.8) into Eqn.(4.23), we simplify the Q-function to (derivation is given in Appendix C.1):

$$\begin{aligned} & Q(\Theta|\bar{\Theta}) \\ = & \sum_{n=1}^{N} \sum_{S^n} \int \Big\{ \log p(Z_0|\Theta) \\ & \qquad + \sum_{k=1}^{K_n} \left[\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta) \right] \\ & \qquad \Big\} \cdot p(Z^n|O^n, S^n, \bar{\Theta})\ dZ^n \cdot p(S^n|O^n, \bar{\Theta}) \\ & + \sum_{n=1}^{N} \sum_{S^n} \left[\sum_{k=1}^{K_n} \log P(y_k^n|X^n, \Theta) \right] p(S^n|O^n, \bar{\Theta}) \end{aligned}$$

$$
\begin{aligned}
& + \sum_{n=1}^{N} \sum_{S^n} \log P(X^n|\Theta)\; p(S^n|O^n,\bar{\Theta}) \\
= \;& Q_Z + Q_Y + Q_X \qquad (4.24)
\end{aligned}
$$

where the Q-function is shown to be separated into three disjoint terms: Q_Z, Q_Y and Q_X. As before, we use m and l to denote X^n and y_k^n, respectively. These three terms can be simplified to (derivations are given in Appendix C.2)

$$
\begin{aligned}
Q_Z = \;& \sum_{n=1}^{N} \sum_{S^n} \int \Big\{ \log p(Z_0|\Theta) \\
& \quad + \sum_{k=1}^{K_n} \left[\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta) \right] \\
& \Big\} \cdot p(Z^n|O^n, S^n, \bar{\Theta})\; dZ^n \cdot p(S^n|O^n, \bar{\Theta}) \\
= \;& \sum_{n=1}^{N} \sum_{m=1}^{M} \int \left[\sum_{k=1}^{K_n} \log p(Z_k^n|Z_{k-1}^n, m, \Theta) \right] p(Z^n|O^n, m, \bar{\Theta})\; dZ^n \cdot \bar{\omega}_m^n \\
& + \sum_{n=1}^{N} \sum_{m=1}^{M} \int \left\{ \sum_{k=1}^{K_n} \sum_{l=1}^{L} \log p(O_k^n|Z_k^n, m, l, \Theta)\; \bar{\xi}_{k,m,l}^n\; p(Z^n|O^n, m, l, \bar{\Theta}) \right\} dZ^n \cdot \bar{\omega}_m^n \\
& + const. \qquad (4.25)
\end{aligned}
$$

$$
\begin{aligned}
Q_Y = \;& \sum_{n=1}^{N} \sum_{S^n} \sum_{k=1}^{K_n} \log P(y_k^n|X^n, \Theta)\; p(S^n|O^n, \bar{\Theta}) \\
= \;& \sum_{n=1}^{N} \sum_{m=1}^{M} \left[\sum_{k=1}^{K_n} \sum_{l=1}^{L} \log P(l|m, \Theta) \cdot \bar{\xi}_{k,m,l}^n \right] \bar{\omega}_m^n \qquad (4.26)
\end{aligned}
$$

$$
\begin{aligned}
Q_X = \;& \sum_{n=1}^{N} \sum_{S^n} \log P(X^n|\Theta)\; p(S^n|O^n, \bar{\Theta}) \\
= \;& \sum_{n=1}^{N} \sum_{X^n} \sum_{Y^n} \log P(X^n|\Theta)\; p(Y^n|X^n, O^n, \bar{\Theta})\; p(X^n|O^n, \bar{\Theta}) \\
= \;& \sum_{n=1}^{N} \sum_{m=1}^{M} \log P(m|\Theta)\; \bar{\omega}_m^n \qquad (4.27)
\end{aligned}
$$

where $\bar{\omega}_m^n$ and $\bar{\xi}_{k,m,l}^n$ have the same expressions as ω_m^n and $\xi_{k,m,l}^n$ respectively, except that the Θ in the earlier expressions is replaced by $\bar{\Theta}$.

By the definition of the speech model in Eqns. (4.2) and (4.3), $p(Z_k^n|Z_{k-1}^n, m, \Theta)$ is a Gaussian with mean: $\Phi_m Z^n(k-1) + (I - \Phi_m)T_m$ and covariance: Q_m. And $p(O_k^n|Z_k^n, m, l, \Theta)$ is also a Gaussian with mean: $\dot{H}_m^l \dot{Z}^n(k)$ and covariance: R_m. Therefore, Q_Z can be re-written as

$$\begin{aligned} Q_Z &= -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\left\{K_n \log|Q_m| + \sum_{k=1}^{K_n} E_m\left[e1_{k,m}^{n}{}'(Q_m)^{-1}e1_{k,m}^{n}\right]\right\}\cdot\bar{\omega}_m^n \\ &\quad -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M}\left\{K_n \log|R_m| + \sum_{k=1}^{K_n}\sum_{l=1}^{L} E_{ml}\left[e2_{k,m,l}^{n}{}'(R_m)^{-1}e2_{k,m,l}^{n}\cdot\bar{\xi}_{k,m,l}^n\right]\right\}\cdot\bar{\omega}_m^n \\ &\quad +const. \end{aligned} \tag{4.28}$$

where $e1_{k,m}^n$ and $e2_{k,m,l}^n$ are equal to

$$\begin{aligned} e1_{k,m}^n &= Z^n(k) - \Phi_m Z^n(k-1) - (I - \Phi_m)T_m, \\ e2_{k,m,l}^n &= O^n(k) - \dot{H}_m^l \dot{Z}^n(k). \end{aligned}$$

$E_m[\cdot]$ denotes the conditional expectation $E[\cdot|O^n, m, \bar{\Theta})]$ and $E_{ml}[\cdot]$ denotes the conditional expectation $E[\cdot|O^n, m, l, \bar{\Theta})]$. These conditional expectations, $E_m[\cdot]$ and $E_{ml}[\cdot]$, will be computed from the Kalman smoothing algorithm that will be discussed in detail later.

4.3.4 M-step

With the Q-function computed above, we now go to the M-step of EM-algorithm.

Re-estimate for π_m Let's first derive the re-estimate for the mixture probability $\pi_m = P(m|\Theta)$, where $m = 1, 2, ..., M$. Since in the Q-function only Q_X is related to $P(m|\Theta)$, we can derive the re-estimation formula by maximizing Q_X with respect to π_m subject to the constraint:

$$\sum_{m=1}^{M} \pi_m = 1.$$

Define the Lagrangian:

$$L_X = Q_X + \lambda(1 - \sum_{m=1}^{M} \pi_m)$$

and take derivative of L_X with respect to π_m to obtain

$$\frac{\partial L_X}{\partial \pi_m} = \sum_{n=1}^{N} \frac{1}{\pi_m} \bar{\omega}_m^n - \lambda$$

Setting the above derivative to zero, we have the re-estimate for π_m:

$$\hat{\pi}_m = \frac{1}{\lambda} \sum_{n=1}^{N} \bar{\omega}_m^n$$

Take $\sum_{m=1}^{M}$ over both sides of Eqn. (4.3.4) and use the property of $\sum_{m=1}^{M} \pi_m = 1$ to get

$$\lambda = \sum_{n=1}^{N} \sum_{m=1}^{M} \bar{\omega}_m^n$$

Then the re-estimate for π_m is

$$\hat{\pi}_m = \frac{\sum_{n=1}^{N} \bar{\omega}_m^n}{\sum_{n=1}^{N} \sum_{m=1}^{M} \bar{\omega}_m^n} \quad for \;\; 1 \leq m \leq M \tag{4.29}$$

Re-estimate for $\gamma_{m,l}$ Note in the Q-function, only Q_Y is related to the "switching" probability $\gamma_{m,l} = P(l|m, \Theta)$ and $\gamma_{m,l}$ has a constraint, $\sum_{l=1}^{L} \gamma_{m,l} = 1$. So the Lagrangian is defined as

$$L_Y = Q_Y + \lambda(1 - \sum_{l}^{L} \gamma_{m,l})$$

Take derivative of L_Y with respect to $\gamma_{m,l}$ and let it be zero,

$$\frac{\partial L_Y}{\partial \gamma_{m,l}} = \sum_{n=1}^{N} [\sum_{k=1}^{K_n} \frac{\bar{\xi}_{k,m,l}^n}{\gamma_{m,l}}] \bar{\omega}_m^n - \lambda$$

Solving the above for $\gamma_{m,l}$, we have

$$\hat{\gamma}_{m,l} = \frac{1}{\lambda} \sum_{n=1}^{N} [\sum_{k=1}^{K_n} \bar{\xi}_{k,m,l}^n] \bar{\omega}_m^n$$

Taking summation $\sum_l^L$ over both sides gives

$$\lambda = \sum_{n=1}^{N} K_n \, \bar{\omega}_m^n$$

It gives the final form of the re-estimation formula:

$$\hat{\gamma}_{m,l} = \frac{\sum_{n=1}^{N} [\sum_{k=1}^{K_n} \bar{\xi}_{k,m,l}^n] \, \bar{\omega}_m^n}{\sum_{n=1}^{N} K_n \, \bar{\omega}_m^n} \tag{4.30}$$

Re-estimates for Φ_m, T_m and Q_m Before going to estimate these parameters, we adopt the following notations for simplicity.

$$\begin{aligned}
A0_m^n &= \textstyle\sum_{k=1}^{K_n} E_m[Z^n(k-1)Z^n(k-1)'], & A1_m^n &= \textstyle\sum_{k=1}^{K_n} E_m[Z^n(k)Z^n(k)'],\\
A2_m^n &= \textstyle\sum_{k=1}^{K_n} E_m[Z^n(k)Z^n(k-1)'], & C_m &= (I - \hat{\Phi}_m)\hat{T}_m\\
B0_m^n &= \textstyle\sum_{k=1}^{K_n} E_m[Z^n(k-1)], & B1_m^n &= \textstyle\sum_{k=1}^{K_n} E_m[Z^n(k)]
\end{aligned}$$

Note that in the Q-function only the first term of Q_Z is related to these parameters in the state equation. The derivatives of Q_Z with respect to Φ_m and T_m are computed according to:

$$\begin{aligned}
\frac{\partial Q_Z}{\partial \Phi_m} &= Q_m^{-1} \sum_{n=1}^{N} (-A2_m^n + B1_m^n T_m' + T_m B0_m^{n\,\prime} - K_n T_m T_m') \cdot \bar{\omega}_m^n \\
&\quad + Q_m^{-1} \Phi_m \sum_{n=1}^{N} (A0_m^n - B0_m^n T_m' - T_m B0_m^{n\,\prime} + K_n T_m T_m') \cdot \bar{\omega}_m^n
\end{aligned} \tag{4.31}$$

and

$$\frac{\partial Q_Z}{\partial T_m} = -Q_m^{-1}(I - \Phi_m) \sum_{n=1}^{N} \{B1_m^n - \Phi_m B0_m^n - K_n (I - \Phi_m) T_m\} \cdot \bar{\omega}_m^n \tag{4.32}$$

Setting the derivatives to zero and solving the equations (note that Q_m must be full rank and $\Phi_m \neq I$ by definition), we obtain the estimates for Φ_m and T_m,

$$\hat{T}_m = \frac{(I - \Phi_m)^{-1} \sum_{n=1}^{N} \{B1_m^n - \Phi_m B0_m^n\} \cdot \bar{\omega}_m^n}{\sum_{n=1}^{N} K_n \cdot \bar{\omega}_m^n} \tag{4.33}$$

$$\hat{\Phi}_m = \{\sum_{n=1}^{N}(A2_m^n - B1_m^n \hat{T}_m' - \hat{T}_m B0_m^{n\,\prime} + K_n \hat{T}_m \hat{T}_m') \cdot \bar{\omega}_m^n\}$$
$$\cdot \{\sum_{n=1}^{N}(A0_m^n - B0_m^n \hat{T}_m' - \hat{T}_m B0_m^{n\,\prime} + K_n \hat{T}_m \hat{T}_m') \cdot \bar{\omega}_m^n\}^{-1}. \quad (4.34)$$

The derivative of Q_Z with respect to Q_m^{-1} is

$$\frac{\partial Q_Z}{\partial Q_m^{-1}} = \frac{1}{2}\sum_{n=1}^{N} K_n Q_m \bar{\omega}_m^n - \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[el_{k,m}^n el_{k,m}^{n\,\prime}] \cdot \bar{\omega}_m^n \quad (4.35)$$

Setting it to zero, we have the estimate for Q_m,

$$\hat{Q}_m = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K_n} E_m[el_{k,m}^n el_{k,m}^{n\,\prime}] \cdot \bar{\omega}_m^n}{\sum_{n=1}^{N} K_n \bar{\omega}_m^n} \quad (4.36)$$

where $\sum_{k=1}^{K_n} E_m[el_{k,m}^n el_{k,m}^{n\,\prime}]$ is calculated according to

$$\sum_{k=1}^{K_n} E_m[el_{k,m}^n el_{k,m}^{n\,\prime}]$$
$$= A1_m^n + \hat{\Phi}_m A0_m^n \hat{\Phi}_m' - A2_m^n \hat{\Phi}_m' - \hat{\Phi}_m (A2_m^n)' - B1_m^n (C_m)'$$
$$-C_m (B1_m^n)' + \hat{\Phi}_m B0_m^n (C_m)' + C_m (B0_m^n)' \hat{\Phi}_m' + K_n C_m C_m' \quad (4.37)$$

Re-estimates for $\dot{H}_m^l$ and R_m Finally, we derive the re-estimation formulas for the parameters, $\dot{H}_{m,l}$ and R_m, contained in the measurement equation. In the Q-function, only the second term in Q_Z is related to these parameters.

The derivative of Q_Z with respect to $\dot{H}_m^l$ is

$$\frac{\partial Q_Z}{\partial \dot{H}_m^l} = -R_m^{-1}\left\{\sum_{n=1}^{N}\left(\sum_{k=1}^{K_n} E_{ml}[(\dot{H}_m^l \dot{Z}^n(k) - O_k^n)(\dot{Z}^n(k))'] \,\bar{\xi}_{k,m,l}^n\right) \bar{\omega}_m^n\right\} \quad (4.38)$$

Setting the derivative to zero and solving it, we get the re-estimate for $\dot{H}_m^l$,

$$\hat{\dot{H}}_m^l = \left\{\sum_{n=1}^{N}\bar{\omega}_m^n \sum_{k=1}^{K_n} O^n(k) E_{ml}[\dot{Z}^n(k)]' \cdot \bar{\xi}_{k,m,l}^n\right\}$$
$$\cdot \left\{\sum_{n=1}^{N}\bar{\omega}_m^n \sum_{k=1}^{K_n} E_{ml}[\dot{Z}^n(k)(\dot{Z}^n(k))'] \cdot \bar{\xi}_{k,m,l}^n\right\}^{-1} \quad (4.39)$$

where $E_{ml}[\dot{Z}^n(k)] = [1, E_{ml}[Z^n(k)]']'$ and

$$E_{ml}[\dot{Z}^n(k)(\dot{Z}^n(k))'] = \begin{bmatrix} 1 & E_{ml}[Z^n(k)]' \\ E_{ml}[Z^n(k)] & E_{ml}[Z^n(k)(Z^n(k))'] \end{bmatrix}.$$

The derivative of Q_Z with respect to R_m^{-1} is

$$\frac{\partial Q_Z}{\partial R_m^{-1}} = \frac{1}{2}\sum_{n=1}^{N}\left\{\sum_{k=1}^{K_n}\sum_{l=1}^{L}\left(R_m - E_{ml}[e2^n_{k,m,l}(e2^n_{k,m,l})']\right)\cdot\bar{\xi}^n_{k,m,l}\right\}\cdot\bar{\omega}^n_m \tag{4.40}$$

Let it be zero, we obtain the re-estimate for R_m,

$$\hat{R}_m = \frac{\sum_{n=1}^{N}\left(\sum_{k=1}^{K_n}\sum_{l=1}^{L} E_{ml}[e2^n_{k,m,l}(e2^n_{k,m,l})']\cdot\bar{\xi}^n_{k,m,l}\right)\cdot\bar{\omega}^n_m}{\sum_{n=1}^{N} K_n\,\bar{\omega}^n_m} \tag{4.41}$$

where $E_{ml}[e2^n_{k,m,l}(e2^n_{k,m,l})']$ is calculated according to

$$\begin{aligned} & E_{ml}[e2^n_{k,m,l}(e2^n_{k,m,l})'] \\ = \; & O^n(k)O^n(k)' - E_{ml}[\dot{Z}^n(k)](\hat{\dot{H}}^l_m)' \\ & - \hat{\dot{H}}^l_m E_{ml}[\dot{Z}^n(k)] + \hat{\dot{H}}^l_m E_{ml}[\dot{Z}^n(k)\dot{Z}^n(k)'](\hat{\dot{H}}^l_m)' \end{aligned} \tag{4.42}$$

4.3.5 Calculation of the sufficient statistics

As we show earlier, in order to obtain the re-estimates for the model parameters, a set of conditional expectations, which form the sufficient statistics for the estimation problem, need to be calculated during the M-step of the EM algorithm. These sufficient statistics include $E_m[Z^n(k)]$, $E_m[Z^n(k)Z^n(k)']$, $E_m[Z^n(k)Z^n(k-1)']$, $E_{ml}[Z^n(k)]$ and $E_{ml}[Z^n(k)Z^n(k)']$.

The conditional expectation $E_m[\cdot] = E[\cdot|O^n, m, \bar{\Theta}]$ is the Kalman smoother of the m-th mixture(or LDM) for the n-th observation. However, the conventional Kalman smoother can not be directly applied here because the current model has parameter switching occurring on the measurement equation. This situation (parameter switching on measurement equation) is exactly the same as that presented in [40], where the filtering and smoothing algorithms were derived. Then the conditional expectation $E_{ml}[\cdot] = E[\cdot|O^n, m, l, \bar{\Theta}]$ becomes the smoother of the m-th mixture for the n-th observation under an extra condition that $\dot{H}_m(k) = \dot{H}_m^l$ (recall that we use l to represent $y_k^n = l$).

The basic theory about filtering and smoothing can be found in ([94], [84], [93], [77], etc.). In the following we list the filtering and smoothing algorithms (for one mixture) for our special model with parameter switching on the measurement equations.

Forward recursion (or filtering) :

$$
\begin{aligned}
\hat{Z}^n_{k|k-1,m} &= \Phi_m \hat{Z}^n_{k-1|k-1,m} + (I - \Phi_m)T_m && (4.43)\\
\Sigma^n_{k|k-1,m} &= \Phi_m \Sigma^n_{k-1|k-1,m} \Phi_m + Q_m && (4.44)\\
\tilde{O}^n_{k,m,l} &= O^n(k) - \dot{H}^l_m \dot{\hat{Z}}^n_{k|k-1,m} \quad l = 1, 2, \ldots, L && (4.45)\\
\Sigma^n_{\tilde{O}_{k,m,l}} &= H^l_m \Sigma^n_{k|k-1,m} H^{l\,\prime}_m + R_m && (4.46)\\
K_{k,m,l} &= \Sigma^n_{k|k-1,m} H^{l\,\prime}_m (\Sigma^n_{\tilde{O}_{k,m,l}})^{-1} && (4.47)\\
\hat{Z}^n_{k|k,m,l} &= \hat{Z}^n_{k|k-1,m} + K_{k,m,l} \tilde{O}^n_{k,m,l} && (4.48)\\
\Sigma^n_{k|k,m,l} &= \Sigma^n_{k|k-1,m} - K_{k,m,l} \Sigma^n_{\tilde{O}_{k,m,l}} K'_{k,m} && (4.49)\\
\psi_{k,m,l} &= \frac{\gamma_{m,l}\ \mathcal{N}(O^n(k) - \dot{H}^l_m \dot{\hat{Z}}^n_{k|k-1,m}, \Sigma^n_{\tilde{O}_{k,m,l}})}{\sum_{l=1}^L \gamma_{m,l}\ \mathcal{N}(O^n(k) - \dot{H}^l_m \dot{\hat{Z}}^n_{k|k-1,m}, \Sigma^n_{\tilde{O}_{k,m,l}})} && (4.50)\\
\hat{Z}^n_{k|k,m} &= \sum_{l=1}^L \psi_{k,m,l}\ \hat{Z}^n_{k|k,m,l} && (4.51)\\
\Sigma^n_{k|k,m} &= \sum_{l=1}^L \psi_{k,m,l}\ \Sigma^n_{k|k,m,l} && (4.52)
\end{aligned}
$$

where $\hat{Z}^n_{k|k-1,m}$ is the predictor and $\Sigma^n_{k|k-1,m}$ its error covariance. $\hat{Z}^n_{k|k,m}$ is the filter and $\Sigma^n_{k|k,m}$ its error covariance. $\mathcal{N}(O^n(k) - \dot{H}^l_m \dot{\hat{Z}}^n_{k|k-1,m}, \Sigma^n_{\tilde{O}_{k,m,l}})$ is a Gaussian density with mean $\dot{H}^l_m \hat{\dot{Z}}^n_{k|k-1,m}$ and covariance $\Sigma^n_{\tilde{O}_{k,m,l}}$, which is the density of the innovation sequence at time k.

Backward recursion (or smoothing) :

$$\begin{aligned} A^n_{k,m} &= \Sigma^n_{k|k,m}\Phi'_m(\Sigma^n_{k|k-1,m})^{-1} && (4.53)\\ \hat{Z}^n_{k|K_n,m} &= \hat{Z}^n_{k|k,m} + A^n_{k,m}[\hat{Z}^n_{k+1|K_n} - \hat{Z}^n_{k+1|k,m}] && (4.54)\\ \Sigma^n_{k|K_n,m} &= \Sigma^n_{k|k,m} + A^n_{k,m}[\Sigma^n_{k+1|K_n,m} - \Sigma^n_{k+1|k,m}]A'_k && (4.55) \end{aligned}$$

Note that the above smoothing is for the computation of $E[\cdot|O^n, m, \bar{\Theta}]$. For the computation of $E[\cdot|O^n, m, l, \bar{\Theta}]$ at the given time point k, $\hat{Z}^n_{k|k,m}$ and $\Sigma^n_{k|k,m}$ in the above smoothing algorithm are simply replaced by $\hat{Z}^n_{k|k,m,l}$ and $\Sigma^n_{k|k,m,l}$ respectively, and correspondingly, $\hat{Z}^n_{k|K_n,m}$ becomes $\hat{Z}^n_{k|K_n,m,l}$ and $\Sigma^n_{k|K_n,m}$ becomes $\Sigma^n_{k|K_n,m,l}$. At other points, the smoothing keeps unchanged.

Using the above Kalman smoothing results, the conditional expectations as sufficient statistics are computed by,

$$\begin{aligned} E_m[Z^n(k)] &= \hat{Z}^n_{k|K_n,m} && (4.56)\\ E_m[Z^n(k)Z^n(k)'] &= \Sigma^n_{k|K_n,m} + \hat{Z}^n_{k|K_n,m}(\hat{Z}^n_{k|K_n,m})' && (4.57)\\ E_m[Z^n(k)Z^n(k-1)'] &= \Sigma^n_{k,k-1|K_n,m} + \hat{Z}^n_{k|K_n,m}(\hat{Z}^n_{k-1|K_n,m})' && (4.58)\\ E_{ml}[Z^n(k)] &= \hat{Z}^n_{k|K_n,m,l} && (4.59)\\ E_{ml}[Z^n(k)Z^n(k)'] &= \Sigma^n_{k|K_n,m,l} + \hat{Z}^n_{k|K_n,m,l}(\hat{Z}^n_{k|K_n,m,l})' && (4.60) \end{aligned}$$

where $\Sigma^n_{k,k-1|K_n,m}$ is recursively calculated by [39]

$$\Sigma^n_{k,k-1|K_n,m} = \Sigma^n_{k|k,m}A^{n\,\prime}_{k-1,m} + A^n_{k,m}(\Sigma^n_{k+1,k|K_n,m} - \Phi_m\Sigma^n_{k|k,m})A^{n\,\prime}_{k-1,m} \qquad (4.61)$$

for $k = K_n, \cdots, 2$, where

$$\Sigma^n_{K_n,K_n-1|K_n,m} = \sum_{l=1}^{L}\psi_{K_n,m,l}(I - K_{K_n,m,l}H^l_m)\Phi_m\Sigma^n_{K_n-1|K_n-1,m,l} \qquad (4.62)$$

4.3.6 The calculation of $\bar{\omega}$ and $\bar{\xi}$

To compute $\bar{\omega}^n_m$ and $\bar{\xi}^n_{k,m,l}$, it suffices to know $p(O^n_k|O^n_{1,k-1}, m, l, \bar{\Theta})$. It is computed according to:

$$p(O^n_k|O^n_{1,k-1}, m, l, \bar{\Theta}) = (2\pi)^{-\frac{d}{2}}|\Sigma^n_{\tilde{O}_{k,m,l}}|^{-\frac{1}{2}} exp\{-\frac{1}{2}(\tilde{O}^n_{k,m,l})'[\Sigma^n_{\tilde{O}_{k,m,l}}]^{-1}\tilde{O}^n_{k,m,l}\} \quad (4.63)$$

where $\tilde{O}^n_{k,m,l}$ and $\Sigma^n_{\tilde{O}_{k,m,l}}$ are computed from the Kalman filter given earlier. d is the dimension of $\tilde{O}^n_{k,m,l}$.

4.4 Likelihood computation

We combine M different linear dynamic models (mixture models) according to different weights to describe a phone's VTR dynamics. After the weights and all models' parameters are trained as described in the preceding section, the likelihood of a phone for a sequence of observations is computed by,

$$\begin{aligned} l(O|\Theta) &= \sum_S p(O, S|\Theta) \\ &= \sum_X \sum_Y p(O|Y, X, \Theta)\, p(Y|X, \Theta)\, p(X|\Theta) \\ &= \sum_{m=1}^{M} \left\{ \prod_{k=1}^{K} \sum_{l=1}^{L} p(O^n_k|O^n_{1,k-1}, m, l, \Theta) \cdot \gamma_{m,l} \right\} \cdot \pi_m \end{aligned} \quad (4.64)$$

The log-likelihood is then

$$L(O|\Theta) = \log l(O|\Theta) \quad (4.65)$$

For a speech utterance consisting of a sequence of phones with the phones' dynamic regimes (boundaries) given, the log-likelihoods for all phones in the sequence as defined in Eqn.(4.65) are summed to give the total log-likelihood score for the entire utterance.

4.5 Evaluation on Switchboard Data

The same experimental paradigm (N-best list re-scoring paradigm) as in the previous chapters is used for the evaluation of this new version, which has been introduced in details in Chapter 2.

4.5.1 Experiment I: Models trained with one speaker's data

In these experiments, the **"1/2 hour"** training set (see Section 2.7.1) is used for model training.

First, we use a single mixture (1-mix) for the mixture-linear dynamic model (MLDM) with switching parameters and we increase the number of H-switching values from one to three. The percentage-WER results are tabulated in Table 4.1. It is observed that the use of H-switching (two or three H values) reduces errors compared with no use of H-switching (one H value only).

systems	Ref+100	100-best
MLDM-SM:1-mix, 1-H	55.7	58.9
MLDM-SM:1-mix, 2-H	55.0	57.7
MLDM-SM:1-mix, 3-H	55.1	57.2

Table 4.1: Performance (WER) of MLDM-SM with a single mixture and different numbers of $\dot{H}$ switching values

We then use two mixtures (2-mix) while again gradually increasing the number of H-switching values. The WER results are listed in Table 4.2. We observe a similar pattern of error reduction to the previous experiment while uniformly raising the overall recognizer performance level somewhat.

systems	Ref+100	100-best
MLDM-SM:2-mix, 1-H	50.7	57.0
MLDM-SM:2-mix, 2-H	50.4	56.6
MLDM-SM:2-mix, 3-H	50.5	56.8
HMM-baseline	56.2	58.9

Table 4.2: Performance (WER) of MLDM-SM with two mixtures and different numbers of $\dot{H}$ switching values (trained with **"1/2 hour"** training set)

Compared with the **"HMM-baseline"** trained on the same data, the new switching dynamic system model with 2 mixtures and 2 H switching values achieves 2.3% absolute WER reduction on the "100-best" case and more than 10% relative WER reduction on the "Ref+100" case.

4.5.2 Experiment II: Models trained with multiple speakers' data

In these experiments, the amount of training data is increased. First, the **"1 hour"** training data set is used. The results are listed in Table 4.3. For the two-mixture (2-mix) case, we observe a WER reduction from the use of one H value to the use of more than one H values. Similar observations are made for the four-mixture (4-mix) case, although the WER reduction is of less magnitude.

Then, for the four-mixture (4-mix) case, we further experimented with using **"2 hour"** training set. The WER results are shown in Table 4.4. A greater error reduction is observed moving from one H value to two H values when compared with the earlier result of Table 4.3 with use of fewer training data.

systems	Ref+100	100-best
MLDM-SM:2-mix, 1-H	51.0	57.1
MLDM-SM:2-mix, 2-H	50.1	56.4
MLDM-SM:2-mix, 4-H	50.0	56.4
MLDM-SM:4-mix, 1-H	49.8	56.6
MLDM-SM:4-mix, 2-H	49.6	56.5

Table 4.3: Performance (WER) of MLDM-SM with two mixtures and different H values (trained with "1 hour" training set)

systems	Ref+100	100-best
MLDM-SM:4-mix, 1-H	49.5	56.0
MLDM-SM:4-mix, 2-H	48.8	55.8

Table 4.4: Performance (WER) of MLDM-SM with two mixtures and different H values (trained with "2 hour" training set)

4.5.3 Some analyses of model behavior and experimental results

In the previous subsection I showed how the system performance is impacted by the increasing of the number of H switching values. Some limited improvements have been obtained. However, from those results in the previous subsection it is hard to see if our original design objective (improving the mapping accuracy on the measurement equation) is achieved or not. So, in this subsection I provide some analyses on this.

The analyses here are based on the fact that the noise covariance matrix R on the measurement equation (Eqn.(4.3)) (or the noise variance if R is treated as diagonal as in our model implementation) is estimated according to Eqn.(4.41), where $e2$ is the difference of the actual MFCC and the output of the $h(\cdot)$ function. Thus, if the function $h(\cdot)$ accurately describes the relation between the VTR space and the MFCC space, the estimated R will be small. Otherwise, the estimated R will be large. Therefore, the accuracy of approximating the physically nonlinear relation between the VTR space and the MFCC space using a piece-wise linear function as implementing by switching-H values can be assessed by examining the size of the estimated noise variance, R.

As typical examples, the diagonal values of the estimated R for phone models "aa", "d", and "n" are shown in Table 4.5, 4.6 and 4.7, respectively. (Column three lists the average values of these diagonal elements.) We observe that these estimated variance values are strictly decreasing with the increasing number of H-switching values. When the number of H-switching values is increased from one to four, the average R value is decreased from 22.1 to 18.6 for "aa", from 22.2 to 19.6 for "d", and from 20.0 to 17.6 for "n", respectively. This suggests that the accuracy of approximation to the nonlinear mapping on the measurement equation by using the piece-wise linear function is improved with the use of piece-wise linear functions. Therefore, our original design objective of improving the approximation to the nonlinear mapping is achieved.

no_of_H	diagonal elements of R	average
1 H	10.0 14.6 17.1 22.5 21.7 18.9 32.3 28.4 29.4 23.5 24.4 21.8	22.1
2 H	8.87 13.4 14.6 20.0 21.3 20.1 29.7 27.5 30.9 21.7 23.4 21.8	21.1
4 H	6.92 11.5 13.4 18.6 17.8 19.2 26.9 19.7 27.9 23.4 17.6 19.7	18.6

Table 4.5: Values of diagonal elements of R noise variance for the phone model "aa" as a function of the number of H switching values.

no_of_H	diagonal elements of R	average
1 H	8.9 17.5 18.8 16.9 19.8 25.6 27.4 27.2 30.2 27.2 26.7 20.7	22.2
2 H	5.6 16.8 16.4 15.1 18.8 24.9 26.4 23.4 31.4 23.6 24.4 19.8	20.6
4 H	5.2 17.0 16.7 15.0 17.0 24.8 24.9 22.9 27.4 21.8 24.7 19.6	19.6

Table 4.6: Values of diagonal elements of R noise variance for the phone model "d" as a function of the number of H switching values.

It is interesting to note that the improvement in the linear piece-wise approximation accuracy as reflected by the reduced R value is correlated with the mild WER reduction in the speech recognition results presented earlier in this section. However, the system's performance (WER) was not improved (less than 1%) as much as that in the noise covariances (more than 10/This phonemenon is consistent with our observation in Chapter 2 when we were trying to decrease prediction errors of the

no_of_H	diagonal elements of R	average
1 H	6.0 15.9 16.8 15.5 18.7 20.7 18.9 27.2 31.0 26.3 21.4 22.0	20.0
2 H	6.1 15.6 15.0 16.6 18.1 21.1 17.8 26.9 24.1 24.4 20.8 19.6	18.8
4 H	5.5 14.5 13.9 16.4 18.4 17.9 16.0 26.1 20.9 23.9 18.7 19.0	17.6

Table 4.7: Values of diagonal elements of R noise variance for the phone model "n" as a function of the number of H switching values

nonlinear function MLPs. From the observation obtained from both Chapter 2 and this chapter, we would like to make the following conclusion: *the relation between the hidden space (VTR) and acoustice space (MFCC) is highly complicated, not a qualitative improvement on the approximation accuracy doesn't impact the system's performance much.*

4.6 Conclusion

A more general version of the mixture linear dynamic model, the mixture linear dynamic model with switching parameters on the measurement equations, has been developed in this chapter. The novelty of the model is the use of piece-wise linear functions to approximate the physically nonlinear function between the partially observable VTR space and the observable MFCC space. This approximation is accomplished by introducing frame-dependent, discrete switching parameters (H parameters) in the observation equation in the state-space formulation of the model.

A series of speech recognition experiments have been carried out to evaluate this new model. The experimental results show that the approximation accuracy is improved with an increasing number of H-switching values (about a 10% reduction in the estimated measurement noise variances). The new version recognizer using the N-best rescoring evaluation paradigm also show some varying degrees of WER reduction compared with using no switching $\dot{H}$ parameters (number of $\dot{H}$ values = 1) in the otherwise identical speech model.

It has also been observed that the system's performance (WER) was not improved as much as that in the noise covariances or approximation accuracy. Because of the high complexity of the nonlinear relationship between VTR and MFCC spaces, it will not impact the system's performance much if the increasing on the approximation accuracy is not qualitative. This phenonemon is consistent with our observation in Chapter 2.

Chapter 5

Efficient Decoding Strategies for the New Dynamic Model

In the previous chapters we didn't touch the decoding problem for our new dynamic models, in all the experiments there we fixed the model dynamic boundaries sub-optimally from HMM phone boundaries during both the training and recognition processes. Due to the continuity constraint imposed on the hidden dynamics(VTR), local likelihood computations become dependent on past path histories. It doesn't allow any path deletion during search if the global optimality is needed. So the search space grows exponentially with time, it makes the search infeasible. That is why we avoided this problem in the previous chapters. But from experiment results we found that the dynamic boundaries are of importance to the dynamic model. So, in this chapter we deal with the decoding problem for the new dynamic model. An analysis of the search problem is first provided, then based on the analysis three efficient approximate decoding approaches are developed.

5.1 Introduction and Motivation

It is well known that search is a challenging problem for all those so-called "segment" models [60]. Because likelihood computations are based on segments for those models, to calculate models' likelihoods segmentations must be known first. This requirement is reflected on the dynamic programming (DP) search algorithm for "segment" models as that an additional set, segmentation set, must be added to the path deletion process. In [60] the DP search algorithm was given as follows,

Iterate: $t = 2, 3, \cdots, T; \forall i \in S_t, l_\tau = t - \tau$, *$\rho(t, i)$ is segmentation set.*

$$\begin{aligned} \delta_t(i) &= max_{j \in S_\tau, \tau \in \rho(t,i)}\{\delta_\tau(j) + log[p(y_{\tau+1}, \cdots, y_t|l_\tau, i)\; p(l_\tau|i)\; p(i|j)]\} && (5.1)\\ \psi_t(i) &= argmax_{j \in S_\tau, \tau \in \rho(t,i)}\{\delta_\tau(j) + log[p(y_{\tau+1}, \cdots, y_t|l_\tau, i)\; p(l_\tau|i)\; p(i|j)]\} && (5.2) \end{aligned}$$

In the above algorithm $\rho(t, i)$ is added to the most likely path choosing process, which is the set of all possible segmentations up to time t at node i. Apparently $\rho(t, i)$ becomes larger and larger with time increasing, it makes the search costly. For HMMs $\rho(t, i)$ shrinks to a single point, $t - 1$.

For our new dynamic model, the situation becomes different, it is even worsened. The DP search algorithm for the new dynamic model is as follows.

Iterate: $t = 2, 3, \cdots, T; \forall i \in \gamma_t$, *$\gamma_t$ is path set at time t.*

$$\begin{aligned} \delta_t(i) &= \{\delta_{t-1}(j) + log[p(y_t|j, S_{i,t})\; p(S_{i,t}|S_{j,t-1})]\}, \qquad j \in \gamma_{t-1} && (5.3)\\ \psi_t(i) &= argmax_{j \in \gamma_{t-1}}\{\delta_{t-1}(j) + log[p(y_t|j, S_{i,t})\; p(S_{i,t}|S_{j,t-1})]\} && (5.4) \end{aligned}$$

where $S_{i,t}$ is the state where path i stays at time t.

In the algorithm i and j become path indexes instead of node indexes in the earlier case. Because the local likelihood $p(y_t|j, S_{i,t})$ is dependent on the past path j, the path deletion is not allowed any more, the "*max*" term in Eqn.(5.3) has gone .

Obviously, the number of paths grows exponentially, which make the search for the new model impossible.

Why do we touch this challenging problem? The motivation is this: in the earlier investigation work on the nonlinear version of this new dynamic model given in Chapter 2 [67] and [3], we found that a very important factor affecting the new recognizer's performance is the boundaries of the dynamic regimes in the VTR dynamic model of speech. When the boundaries were manually [1] adjusted from those automatically (often with gross errors) determined from an HMM system so as to conform to the dynamic regimes expected from the model, drastic reduction of recognition error in the Switchboard task was consistently observed. This motivates the current work aiming to develop efficient (segmentation) algorithms which can automatically determine the optimal dynamic regimes in training the recognizer and in scoring spontaneous speech utterances.

The optimality criterion, chosen to be the likelihood on the observation data (MFCC sequences), is based on the dynamic VTR model used to represent the dynamic patterns of speech, rather than based on other inconsistent forms of model such as HMM. This achieves the desirable goal of consistency modeling in the entire speech recognition system.

5.2 Analysis of the Problem

Let's explain first in this section why the standard dynamic programming approach (Viterbi algorithm) can *not* be applied directly to search for optimal dynamic regimes in the hidden dynamic model of speech. This is so no matter how small the number of

[1]This work was done during the Workshop 1998 held in John Hopkins University. See the website *http://www.clsp.jhu.edu/ws98/* for details.

the dynamic regimes is. In short, the difficulty for the search efficiency arises because of the continuity constraint imposed across dynamic regimes in the hidden variable domain. A consequence of this continuity constraint is this: at any fixed node in the trellis search diagram, the local likelihood scores, into a fixed future node will in general be different depending on the past history arriving at the current node. [2]

As stated in Chapter 2 , the log-likelihood of the new model producing a sequence of observations is computed from the innovation sequence. Let's re-write Eqn.(2.54) here.

$$\begin{aligned} L(O_1^K|\Theta) &= \log p(O(1), O(2), \cdots, O(N)|\Theta) \\ &= -\frac{1}{2}\sum_{k=1}^{N}\{\log|\Sigma_{\tilde{O}_k}| + \tilde{O}_k'\Sigma_{\tilde{O}_k}^{-1}\tilde{O}_k\} + const. \end{aligned} \tag{5.5}$$

where $\tilde{O}_k$ and $\Sigma_{\tilde{O}_k}$ are the mean and covariance of the innovation at time k.

$\tilde{O}_k$ and $\Sigma_{\tilde{O}_k}$ are calculated from Kalman filtering process. Kalman filter is a recursive state estimation algorithm. It is illustrated in Table 5.1. At each time recursion, it runs prediction, innovation and filtering processes.

From the filtering algorithm one can see that the calculation of the mean and covariance of innovation sequence at current time k, used for the local score computation, is dependent on the prediction values $\hat{Z}_{k|k-1}$ and $\Sigma_{k|k-1}$, and the predictor is is initialized by the filtered state values $\hat{Z}_{k-1|k-1}$ and $\Sigma_{k-1|k-1}$ at the previous time $k-1$. So the local score computation depends on the previous recursion. Furthermore, the previous recursion is initialized by those filtered state values at the one further previous time point $k-2$, and so on. The continuity constraint of the

[2]It is the difficulty of such a type which prevents further development of a model that is also substantially different from the conventional HMM [64]. Note also that the computational complexity associated with most versions of stochastic segment models [60] arises from very different causes where no explicit constraints across segments are imposed.

$$\begin{tikzcd}
\text{time } k-1 \arrow[d] & \\
\boxed{\begin{array}{l}\text{Prediction:}\\ \hat{Z}_{k|k-1} = \Phi\hat{Z}_{k-1|k-1} + (I-\Phi)T \\ \Sigma_{k|k-1} = \Phi\Sigma_{k-1|k-1}\Phi' + Q\end{array}} \arrow[d] & \\
\boxed{\begin{array}{l}\text{Innovation:}\\ \tilde{O}_k = O(k) - h(\hat{Z}_{k|k-1}) \\ \Sigma_{\tilde{O}_k} = H_{k|k-1}\Sigma_{k|k-1}H'_{k|k-1} + R \\ K_k = \Sigma_{k|k-1}H'_{k|k-1}\Sigma^{-1}_{\tilde{O}_k}\end{array}} \arrow[d] \arrow[r] & \boxed{\begin{array}{l}\text{local-score at } k:\\ -\frac{1}{2}\{\log|\Sigma_{\tilde{O}_k}| + \tilde{O}'_k\Sigma^{-1}_{\tilde{O}_k}\tilde{O}_k\}\end{array}} \\
\boxed{\begin{array}{l}\text{Filtering:}\\ \hat{Z}_{k|k} = \hat{Z}_{k|k-1} + K_k\tilde{O}_k \\ \Sigma_{k|k} = \Sigma_{k|k-1} - K_k\Sigma_{\tilde{O}_k}K'_k\end{array}} \arrow[d] & \\
\text{time } k+1 &
\end{tikzcd}$$

Table 5.1: An illustration of Kalman filtering algorithm

hidden dynamics enforces this initialization process is carried out even at the phone boundaries, by setting the filtered values at the last time point of previous phone to be the initial values for the innovation calculation at first time point of current phone. Therefore, the likelihood computation at current time becomes dependent on the whole past path. Let's give an simple example to illustrate this.

In the trellis depicted in Fig.(5.1), we show three left-to-right dynamic regimes (S_1, S_2, and S_3) for simplicity reasons. At time frame $t = 3$, there are two possible paths entering regime S_2: "$b11 \rightarrow b22 \rightarrow b23$", and "$b11 \rightarrow b12 \rightarrow b23$". Note that at time $t = 2$ these two paths have an identical value, $\hat{Z}_{1|1}$, $\Sigma_{1|1}$, to initialize the Extended Kalman Filter (EKF). But since the two paths use distinct model parameters (T, Φ), one from S_1 and the other from S_2, they generate different likelihood scores, $L_1(t = 2)$ and $L_2(t = 2)$, as well as different filtered values, $\hat{Z}_{1,2|2}$, $\Sigma_{1,2|2}$ and $\hat{Z}_{2,2|2}$, $\Sigma_{2,2|2}$. At frame $t = 3$, these two paths will use the same model (S_2) parameters for the EKF (in likelihood computation) when they enter into regime S_2, but their initial points, $\hat{Z}_{1,2|2}$, $\Sigma_{1,2|2}$ and $\hat{Z}_{2,2|2}$, $\Sigma_{2,2|2}$, for the EKF are different. Therefore, these two paths will generate distinct regime-bound, local scores, $L_1(t = 3)$ and $L_2(t = 3)$, and distinct filtered values $\hat{Z}_{1,3|3}$, $\Sigma_{1,3|3}$ and $\hat{Z}_{2,3|3}$, $\Sigma_{2,3|3}$, at frame $t = 3$ and for regime (S_2).

In Viterbi algorithm for the conventional HMM, at this point the path with lower likelihood of among the two (i.e., the smaller one of $L_1(t = 1) + L_1(t = 2) + L_1(t = 3)$ and $L_2(t = 1) + L_2(t = 2) + L_2(t = 3)$) can be dropped for future path growth without losing global optimality. This is so because the two paths would give an identical score when entering node "b34" (or "b24") and would behave identically in the future path expansion.[3]

However, if we were to drop the low-score path at node "b23" as in the Viterbi algorithm for HMM, we would lose global optimality. This is so because both paths emanating from "b23" will generate different local scores while entering node "b34" (due to different initial values, $\hat{Z}_{1,3|3}$,$\Sigma_{1,3|3}$ and $\hat{Z}_{2,3|3}$, $\Sigma_{2,3|3}$ for EKF) and would hence behave differently in the future path growth.

The problem exemplified above associated with the dynamic model and with use of the KF algorithm applies to essentially all nodes in the trellis. That is, due to the continuity constraint in the VTR domain from one dynamic regime to the next,[4] no

[3]This is the essence of dynamic programming or Viterbi algorithm.

[4]This is implemented by forcing the end value of the earlier regime to initialize the KF for the

path drop shall be allowed without losing global optimality. In theory, the search space expands exponentially with time.

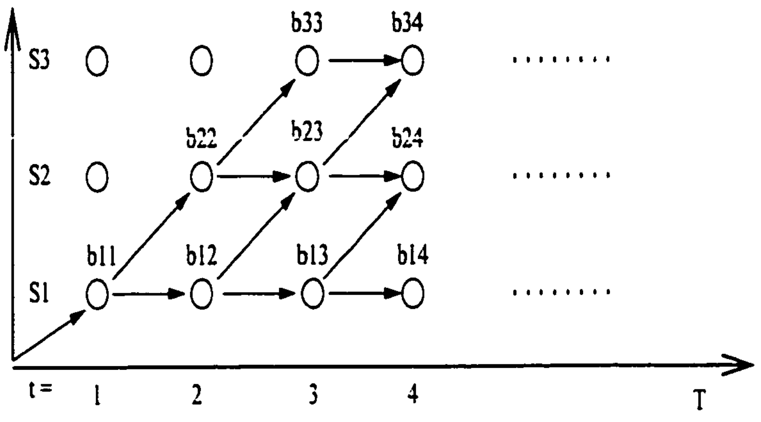


Figure 5.1: An example trellis diagram to show that the conventional trellis search is *not* applicable in theory because local scores into a future node depend on the history into the current node

Two strategies, path deletion and path merging, can be used to reduce the search space. Either of them makes the search not optimal. What we can do is to design efficient path deletion and merging strategies to make the search undermined to the least extent. Based on path deletion strategy, a path-stack decoding algorithm (PS-D) has been developed. Based on path merging strategy, two approaches, a second-order general pseudo-Bayesian decoding algorithm (GPB2-D) and an interacting multiple model decoding algorithm (IMM-D), have been developed. All the three approaches successfully convert the original problem which is exponential in time to one which is only linear in time. The PS-D algorithm will be elaborated in Section 5.3, the GPB2-D and IMM-D algorithms will be addressed in Section 5.4.

newly entered regime.

5.3 Path deletion strategy: the PS-D algorithm

The PS-D algorithm uses an intuitive way to do path deletion, the essence of this algorithm is to use a path-stack at each node in the trellis to maintain a sufficient but limited number of the promising paths (details can be found in [2]).

Essential to this "path-stack" algorithm is our discovery that the initial filter error covariance, $\Sigma_{k|k}$, is of much less importance in determining likelihood scores than the initial filter value, $\hat{Z}_{k|k}$. It was also discovered that if the filtered values of two paths are close to each other then the behavior of these two paths in the future will be very similar. In the example of Fig.(5.1), when the filtered values, $\hat{Z}_{1,3|3}$ and $\hat{Z}_{2,3|3}$ (corresponding to the two paths at node "b23" at frame $t = 3$), are not far apart from each other, then they will produce similar likelihood scores (using the same model parameters of S_3) when they enter node "b34" at time frame $t = 4$. Further, due to the asymptotic, target-directed property of the VTR dynamics modeled by Eqn.(2.1), the filtered values of the two paths at time $t = 4$, $\hat{Z}_{1,4|4}$ and $\hat{Z}_{2,4|4}$, will become more similar in value. This will guarantee the similarity of scores at future time frames associated with these distinct two paths occurring at an earlier time frame. Therefore, if we drop one (with a lower score) of these paths with similar filtered values at an early time frame, the loss of global optimality would be minimal.

In order to prevent the number of paths from increasing exponentially with time, we add a path stack at each node and time frame, and limit the size of the path stack to a fixed value. The size of the path-stack defines how many paths are kept for the node. If the size is small, the path dropping becomes heavy. By adjusting the stack size, we strike a balance between the degrees of optimization and of search efficiency.

Based on the above observation and computational constraint, we have developed the following operation on the path-stacks: *At time $t = k$ and node b, calculate the path likelihood of the m-th path ending at node b, $L(b, m, t = k)$, and calculate the filtered (by EKF) values, $\hat{Z}_{m,k|k}$ and $\Sigma_{m,k|k}$ (They are denoted by $Z(m, k)$ and $P(m, k)$ respectively in Fig.(5.2) for convenience). Then, compute the distance, D_{mi}, between $\hat{Z}_{m,k|k}$ and $\hat{Z}_{i,k|k}$ of the i-th path already in the path-stack of node b. Choose the path whose filter value is closest to $\hat{Z}_{m,k|k}$ (say j-th path). If the path-stack of node b is not full and the distance D_{mj} is greater than a preset threshold, insert the new path into the path-stack. Otherwise, if the new path has higher path score, substitute it for the early stored j-th path, if not, drop the new path.*

The entirety of the path-stack algorithm as we have implemented in this work for optimizing dynamic regimes and for (approximate) maximum likelihood scoring is shown in Fig.(5.2). K is the utterance length, N is the total number of nodes in a lattice or in an N-best list, and SS is the maximum number of paths allowed to be kept in a path stack.

The computation complexity of this algorithm is proportional to $SS * K$, where SS is the size of the path-stack (pre-determined) that need to be processed at each node in the search trellis. This algorithm turns the original search problem which is exponential in K (tree search) into one which is only linear in K.

The path-stack algorithm described above avoids the exponentially growing computation inherent in the exhaustive search by keeping only a limited but sufficient number of promising paths at each trellis node. It extends the earlier search algorithms and ideas developed mainly for the HMM-based recognizers, including the stack idea, the N-best search idea, and the idea of limiting the stack growth [22, 21, 20]. Our specific contribution lies in applying these ideas to the specific speech model where different types of information are used to determine whether to grow, to maintain, or to delete the stack entries during the trellis search. The

```
For each frame , O(k), in the observation squence of an utterance($0 < k < K+1$)
   best_score(k) = negative_infinity;
   For each node (phone) i in the lattice (or in a N-best list) of the utterance ($0 < i < N+1$)
      Let n_path_in_stack_of_node_i = 0;
      For each node j (not pruned) which can enter node i ($0 < j < N+1$)
         For each path m existing in the path stack of node j at time k-1 ( $0 < m < SS+1$)
            Calculate acoustic score L1(O(k)| i, m), filtered dynamics Z(m,k) and its error covariance P(m,k);
            Let min_distance = infinity and min_path_index = 0;
            For each path l existing in the path stack of node i at time k ( $0 < l < SS+1$)
               Let distance = | Z(i, l, k) - Z(m,k) | ;
               if ( distance < min_distance )
                  min_distance = distance;
                  min_path_index = l;
               endif
            End
            Let path_likelihood = L(j, m, k-1) + L1(O(k)| i, m);
            if ( (n_path_in_stack_of_node_i < SS ) and (min_distance/ | Z(m,k)| > Thres_deletion) )
               L( i, n_path_in_stack_of_node_i, k) = path_likelihood;
               Z( i, n_path_in_stack_of_node_i, k) = Z( m, k );
               P( i, n_path_in_stack_of_node_i, k ) = P( m, k );
               n_path_in_stack_of_node_i ++;
               Remeber the back pointer from ( i, n_path_in_stack_of_node_i ) to ( j, m );
            else if ( L(i, min_path_index, k) < path_likelihood )
               L ( i, min_path_index, k) = path_likelihood;
               Z ( i, min_path_index, k ) = Z ( m, k );
               P ( i, min_path_index, k ) = P ( m, k );
               Re-set the back pointer for node (i, n_path_in_stack_of_node_i) to (j, m)
            else
                Drop the new path and keep the stack untouched;
            endif
            if ( path_likelihood > best_score (k) )
                Let best_score (k) = path_likelihood;
            endif
         End
      End
   End
   Do the following path pruning if needed.
   For each node i in the latticeand each path k in the path stack of node i
      if ( | (best_score (k) - L( i, m, k ) ) / best_score(k) | > Thres_pruning )
         Prune path ( i, m) from the search space;
      endif
   End
End
Choose the path with maximum likelihoood. maxpathlikelihood = max L( i, m, K) over node i and path k in the stacks
Trace back to obtain the segmentation of dynamics and its associated speech units.
```

Figure 5.2: The Path-Stack Search Algorithm

algorithm has been kept within the time-synchronous, dynamic programming framework.

5.4 Path merging strategy: the GPB2-D algorithm and the IMM-D algorithm

The path merging strategies in the GPB2-D and the IMM-D algorithms are both borrowed from control area [77, 78] . So before going to elaborate them, I would like to give a brief background introduction.

5.4.1 Background Introduction

In control area there is a special model called a model with switching parameters. The model parameters are not time-invariant, neither do they vary continuously with time. The parameters switch among a number of sets of fixed values from time to time. So sometimes it is also called a switching model. One example of this type of model is given in Eqn.(5.6) and (5.7) [78], where $s(k)$ denotes the mode of the model at time k, which is assumed to be one of M possible modes, $s(k) \in \{s_i\}_{i=1}^{M}$. Apparently the parameter switching could happen at any time.

$$\begin{aligned} Z(k) &= F[s(k)]Z(k-1) + W(k-1, s(k)) \qquad (5.6) \\ O(k) &= H[s(k)]Z(k) + V(k, s(k)) \qquad (5.7) \end{aligned}$$

This switching phenomenon is similar to the node transition during the trellis search in speech recognition, where the paths could transit from one node to other nodes at any time point. Therefore, the merging strategies in the former case may be transplanted to the trellis search for the new dynamic model. In our case each node of the trellis represents a phone, which is described by one of the dynamic models.

$$\begin{aligned} Z(k) &= \Phi^{(i)}Z(k-1) + (I - \Phi^{(i)})T^{(i)} + W^{(i)}(k) & (5.8) \\ O(k) &= h^{(i)}(Z(k)) + V^{(i)}(k) & (5.9) \end{aligned}$$

The properties of this new dynamic model have been introduced in Chapter 2, they will not be repeated here. The only difference is the node (or phone) index i added to the model. The index i also indicates the dynamic regime which corresponds to node or phone i (or a unique set of model parameters). As the speech utterance traverses, during the trellis search, from left to right in time, phone-sized dynamic regimes switch from one to another, which induces the switching process among M parameter sets $\Theta^{(i)} = \{\Phi^{(i)}, T^{(i)}, R^{(i)}, Q^{(i)}, h^{(i)}\}$, where $i = 1, 2, ..., M$ and M is the total number of phones.

The model parameters are discrete at the dynamic regime boundaries (i.e.. no constraints), However, the underlying dynamics, as analyzed before, is *constrained* to be continuous at the dynamic regime boundaries.

5.4.2 State estimation for the model with switching parameters

In control area engineers are more interested in the hidden state estimation given observations. For a conventional state-space model with *fixed* parameters, the goal of optimal (in both the Bayesian sense [94] and the minimum mean square error sense [93]) state estimation (filtering) is to calculate the conditional mean and covariance of $Z(k)$ given the observations up to time k. Let's use O_1^k to denote the observation sequence $\{O_1, O_2, ..., O_k\}$, $\hat{Z}_{k|k}$ the conditional mean, and $\Sigma_{k|k}$ the conditional covariance, then

$$\hat{Z}_{k|k} = E[Z(k)|O_1^k]$$

and

$$\Sigma_{k|k} = Cov[Z(k)|O_1^k]$$

Generalizing from the fixed-parameter case to the case where the state-space model parameters switch with time, since how the model parameters switch is unknown, the mean and covariance of $Z(k)$ will be conditioned not only on the observations up to time k but also on the evolution history of the model parameter switching. Therefore, the conditional mean and covariance will become

$$E[Z(k)|s_k, s_{k-1}, ..., s_1, O_1^k] \tag{5.10}$$

and

$$Cov[Z(k)|s_k, s_{k-1}, ..., s_1, O_1^k] \tag{5.11}$$

where s_k is a discrete variable indicating which of the M modes (or nodes in the trellis search) is switched at time frame k.

If the switching evolution process were known, the conventional state estimation techniques would directly apply [93, 76]. Suppose $s_k = i_k, s_{k-1} = i_{k-1}, ..., s_1 = i_1$, where i_k $(1 \le i_k \le M)$ is the index to the mode which the dynamics in the system switches to at time k, then the state estimates become

$$E[Z(k)|s_k = i_k, s_{k-1} = i_{k-1}, ..., s_1 = i_1, O_1^k] \tag{5.12}$$

and

$$Cov[Z(k)|s_k = i_k, s_{k-1} = i_{k-1}, ..., s_1 = i_1, O_1^k] \tag{5.13}$$

Eqn.(5.12) and (5.13) is referred to as the elemental estimator [75, 85] in the overall state estimation procedure to be discussed.

However, either in the switching model or in the trellis search of speech recognition, the switching evolution process is unknown. In principle, all the possibilities of the evolution (or paths) have to be considered during the estimation. At each time point, any one of the M modes (or nodes) could be chosen, and such as, there are potentially as many as M^k paths for the switching evolution up to time k. Each of these paths forms an elemental estimator based on the conventional state estimation, and hence there are a prohibitively large number (M^k) of the elemental estimators at time k. Therefore, the overall state estimation has to sum up over all this possibilities according to their probabilities.

For simplicity purpose, we adopt the following notation:

Ψ_k: all paths of the switching evolution up to time k.

$\psi_k(n)$: the n-th (out of M^k) path of Ψ_k. This can be explicitly written as $\psi_k(n) = \{s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, \ldots, s_{n,1} = i_{n,1}\} = \{s_{n,k} = i_{n,k}, \psi_{k-1}(m)\}$ where $\psi_{k-1}(m)$ is the m-th path of Ψ_{k-1}, from which $\psi_k(n)$ stems. That is, $\psi_{k-1}(m) = \{s_{m,k-1} = i_{m,k-1} = i_{n,k-1}, \ldots, s_{m,1} = i_{m,1} = i_{n,1}\}$.

$P_k(n)$: the probability of $\psi_k(n)$ being true given O_1^k; that is, $P_k(n) = Pr(\psi_k(n)|O_1^k)$.

Using the above notation, the following state estimation algorithm, by applications of Bayes' rule, can be obtained (see the derivation in Appendix E).

Estimates for the state (continuous) and its covariance:

$$\hat{Z}_{k|k} = \sum_{n\in\Psi_k} P_k(n)\hat{Z}_{n,k|k} \tag{5.14}$$

$$\Sigma_{k|k} = \sum_{n\in\Psi_k} P_k(n)\{\Sigma_{n,k|k} + [\hat{Z}_{n,k|k} - \hat{Z}_{k|k}][\hat{Z}_{n,k|k} - \hat{Z}_{k|k}]'\} \tag{5.15}$$

where $\hat{Z}_{n,k|k}$ and $\Sigma_{n,k|k}$ is the elemental estimate for the fixed n-th evolution:

$$\hat{Z}_{n,k|k} = E[Z(k)|s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, \dots, s_{n,1} = i_{n,1}, O_1^k] \tag{5.16}$$

$$\Sigma_{n,k|k} = Cov[Z(k)|s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, \dots, s_{n,1} = i_{n,1}, O_1^k] \tag{5.17}$$

The probability $P_k(n)$ is recursively updated according to:

$$P_k(n) = \frac{p(O(k)|\psi_k(n), O_1^{k-1})}{p(O(k)|O_1^{k-1})} P(s_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1}) P_{k-1}(m) \tag{5.18}$$

where the normalizing factor $p(O(k)|O_1^{k-1})$ is computed from

$$\sum_{n \in \Psi_k} p(O(k)|\psi_k(n), O_1^{k-1}) P(s_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1}) P_{k-1}(m), \tag{5.19}$$

where $p(O(k)|\psi_k(n), O_1^{k-1})$ is the density of the innovation (or residual) process of the n-th elemental estimator and is calculated during the elemental estimation, and $P(s_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1})$ characterizes the evolution of the switching, which is assumed to follow a first-order Markov chain and to be known in advance. The first-order Markov chain can be expressed by the following transition probability matrix:

$$p = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1M} \\ p_{21} & p_{22} & \cdots & p_{2M} \\ \vdots & \vdots & \vdots & \vdots \\ p_{M1} & p_{M2} & \cdots & p_{MM} \end{pmatrix} \tag{5.20}$$

where $p_{ij} = P(s_k = j|s_{k-1} = i)$ and $\sum_{j=1}^{M} p_{ij} = 1$ for all i. $P(s_k = j|s_{k-1} = i)$ is the probability of the system dynamics which switches from discrete state (i.e., phone) i to discrete state j at time k. For this Markov chain, the current state depends only on the previous state, so $P(s_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1})$ becomes $P(s_{n,k} = i_{n,k}|s_{n,k-1} = i_{n,k-1})$. Therefore, Eqn.(5.18) can be drastically simplified to

$$P_k(n) = \frac{p(O(k)|\psi_k(n), O_1^{k-1})}{p(O(k)|O_1^{k-1})} P(s_{n,k} = i_{n,k}|s_{n,k-1} = i_{n,k-1}) P_{k-1}(m). \tag{5.21}$$

However, even under this Markovian assumption, the elemental estimator is still conditioned on the entire switching history, $\psi_k(n)$. As such, the number of the elemental estimators grows exponentially with time. To make the state-estimation algorithm computationally feasible, approximations have to be made which would make the estimation suboptimal as a price paid for alleviating computation burden. Two typical approximate approaches have been invented for this case. Both are based on merging the evolution hypotheses in a time-synchronous manner. The two merging strategies and the associated suboptimal state estimation algorithms will be discussed below.

5.4.3 Approximation I: A generalized pseudo-Bayesian approach

The general pseudo-Bayesian approach (GPB) has been used for state estimation in the target-tracking and econometrics literature for general dynamic systems [78, 75, 85, 82, 91]. We extended this approach, which has been developed specifically for speech recognition applications, to our specially constrained dynamic system model described in the previous chapters for the speech dynamics.

First-order GPB (GPB1) and second-order GPB (GPB2) for general linear dynamic systems have been available in the literature [78]. In the first-order GPB, the state estimate is carried out under each possible current model at each time k. In the switching evolution history $\psi_k(n) = \{s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, \ldots, s_{n,1} = i_{n,1}\}$, only the most recent term $s_{n,k} = i_{n,k}$ is kept, while the other terms are dropped. That is, $\psi_k(n)$ is approximated by $\{s_{n,k} = i_{n,k}\}$. Therefore, only M out of M^k paths are considered by merging all "older" paths for all models at time $k-1$. This approach is very efficient in computation, but the price for loss of accuracy would be too great to be used for implementing our speech recognition decoding algorithms because of the heavy merging.

In the second-order GPB which we use for implementing one of the speech recognition decoding algorithms, the state estimate is carried out under each possible pair of current and previous model at each time k. In the switching evolution history:

$$\psi_k(n) = \{s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, \ldots, s_{n,1} = i_{n,1}\},$$

the most recent two terms, $s_{n,k} = i_{n,k}$ and $s_{n,k-1} = i_{n,k-1}$, are considered and earlier terms are dropped. That is, $\psi_k(n)$ is approximated by $\{s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}\}$. Therefore, a total of M^2 paths for the path combination are considered at each time frame. For this case, all the paths for each model at time $k-1$ are merged. The merging in GPB2 is not as heavy as GPB1 and its estimation accuracy is substantially higher. We describe and derive GPB2 in detail for the new constrained switching dynamic model of speech.

Given the GPB2 approximation, the precise elemental estimator Eqns.(5.16) and (5.17) has now been approximated by

$$E[Z(k)|s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, O_1^k]$$

and

$$Cov[Z(k)|s_{n,k} = i_{n,k}, s_{n,k-1} = i_{n,k-1}, O_1^k].$$

For simplicity purposes, in the following derivation of the state estimate, we will use s_k, s_{k-1}, j and i to represent $s_{n,k}$, $s_{n,k-1}$, $i_{n,k}$ and $i_{n,k-1}$, respectively , and will adopt the following notation:

$$\begin{aligned}
\hat{Z}_{k|k-1}^{(i,j)} &= E[Z(k)|s_k = j, s_{k-1} = i, O_1^{k-1}] \quad (\textit{elemental predictor}) \\
\Sigma_{k|k-1}^{(i,j)} &= Cov[Z(k)|s_k = j, s_{k-1} = i, O_1^{k-1}]
\end{aligned}$$

$$\begin{aligned}
\hat{Z}_{k|k}^{(i,j)} &= E[Z(k)|s_k = j, s_{k-1} = i, O_1^k] \quad \textit{(elemental filter)} \\
\Sigma_{k|k}^{(i,j)} &= Cov[Z(k)|s_k = j, s_{k-1} = i, O_1^k] \\
\hat{Z}_{k|k}^{(j)} &= E[Z(k)|s_k = j, O_1^k] \quad \textit{(merged state estimate)} \\
\Sigma_{k|k}^{(j)} &= Cov[Z(k)|s_k = j, O_1^k].
\end{aligned}$$

Merging and state estimate

With use of the above notation, the state estimate, Eqn.(5.14) and (5.15), has become

$$\hat{Z}_{k|k} = \sum_{j=1}^{M}\sum_{i=1}^{M} P(s_k = j, s_{k-1} = i|O_1^k)\hat{Z}_{k|k}^{(i,j)} \tag{5.22}$$

$$\begin{aligned}
\Sigma_{k|k} &= \sum_{j=1}^{M}\sum_{i=1}^{M} P(s_k = j, s_{k-1} = i|O_1^k) \\
&\qquad \cdot \{\Sigma_{k|k}^{(i,j)} + [\hat{Z}_{k|k}^{(i,j)} - \hat{Z}_{k|k}][\hat{Z}_{k|k}^{(i,j)} - \hat{Z}_{k|k}]'\}.
\end{aligned} \tag{5.23}$$

In GPB2, the merging takes place at each model (discrete state), and the merged estimate at node j is calculated according to

$$\begin{aligned}
\hat{Z}_{k|k}^{(j)} &= E[Z(k)|s_k = j, O_1^k] \\
&= \sum_{i=1}^{M} P(s_{k-1} = i|s_k = j, O_1^k) \cdot E[Z(k)|s_k = j, s_{k-1} = i, O_1^k] \\
&= \sum_{i=1}^{M} P(s_{k-1} = i|s_k = j, O_1^k) \cdot \hat{Z}_{k|k}^{(i,j)}
\end{aligned} \tag{5.24}$$

$$\begin{aligned}
\Sigma_{k|k}^{(j)} &= Cov[Z(k)|s_k = j, O_1^k] \\
&= \sum_{i=1}^{M} P(s_{k-1} = i|s_k = j, O_1^k) \\
&\qquad \cdot \{\Sigma_{k|k}^{(i,j)} + (\hat{Z}_{k|k}^{(i,j)} - \hat{Z}_{k|k}^{(j)})(\hat{Z}_{k|k}^{(i,j)} - \hat{Z}_{k|k}^{(j)})'\}
\end{aligned} \tag{5.25}$$

That is, the merged quantities are obtained by summing up all the possibilities at each model according to their individual posterior probabilities, $P(s_{k-1} = i|s_k =$

$j, O_1^k)$. It is easy to show that the state estimate, Eqn.(5.22) and (5.23), is equivalent to

$$\hat{Z}_{k|k} = \sum_{j=1}^{M} P(s_k = j|O_1^k)\hat{Z}_{k|k}^{(j)} \tag{5.26}$$

$$\Sigma_{k|k} = \sum_{j=1}^{M} P(s_k = j|O_1^k)\{\Sigma_{k|k}^{(j)} + [\hat{Z}_{k|k}^{(j)} - \hat{Z}_{k|k}][\hat{Z}_{k|k}^{(j)} - \hat{Z}_{k|k}]'\} \tag{5.27}$$

The posterior probability $P(s_{k-1} = i|s_k = j, O_1^k)$ is called **merging probability** and $P(s_k = j|O_1^k)$ called **mode probability**. They are calculated recursively according to Bayes rule and by straightforward conditional probability manipulation:

$$P(s_{k-1} = i|s_k = j, O_1^k) = \frac{p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1})}{\sum_{i=1}^{M} p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1})} \tag{5.28}$$

and

$$P(s_k = j|O_1^k) = \frac{\sum_{i=1}^{M} p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1})}{\sum_{j=1}^{M}\sum_{i=1}^{M} p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1})}. \tag{5.29}$$

In Eqn. (5.28) and Eqn.(5.29), $p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1})$ is computed by

$$\begin{aligned} & p(O(k), s_k = j, s_{k-1} = i|O_1^{k-1}) \\ = \; & p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1}) \\ & \cdot P(s_k = j|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|O_1^{k-1}) \end{aligned} \tag{5.30}$$

where $p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1})$ is the density of innovation (residual) sequence obtained during the elemental estimation, $P(s_k = j|s_{k-1} = i, O_1^{k-1}) = P(s_k = j|s_{k-1} = i)$ is the transition probability and is known,[5] and $P(s_{k-1} = i|O_1^{k-1})$ is recursively calculated according to Eqn.(5.29) and (5.30) with the initial value given at time 0.

[5]This is the transition probability of the discrete Markov switching process. In our phone-based model construction, this probability gives the bi-phone "language model", which we fixed to be 0.5 in our recognizer implementation.

Elemental estimator

Many possible approaches are available for elemental estimate (i.e., conditional state estimation given the path history). The simplest approach we have adopted in the current recognizer implementation is the Kalman-type filter algorithms. In our recognizer implementation, the extended Kalman filter (EKF) algorithm ([94], [84], [93],[76], [77]) and the Kalman filter (KF) algorithm are used for the nonlinear versions and linear versions of our dynamic models, respectively. The Kalman-type filters have been adapted to suit the special structure of the speech model incorporating target-directed constraint and cross-regime continuity constraint. Use of the tailored Kalman-type filter algorithm for the elemental estimator here consists of the following steps:

$$\begin{aligned}
\hat{Z}^{(i,j)}_{k|k-1} &= \Phi^{(j)}\hat{Z}^{(i)}_{k-1|k-1} + (I-\Phi^{(j)})T^{(j)} && (5.31)\\
\Sigma^{(i,j)}_{k|k-1} &= \Phi^{(j)}\Sigma^{(i)}_{k-1|k-1}\Phi^{(j)'} + Q^{(j)} && (5.32)\\
\tilde{O}^{(i,j)}_{k} &= O(k) - h(\hat{Z}^{(i,j)}_{k|k-1}) && (5.33)\\
\Sigma^{(i,j)}_{\tilde{O}_k} &= H_{k|k-1}\Sigma^{(i,j)}_{k|k-1}(H_{k|k-1})' + R^{(j)} && (5.34)\\
K^{(i,j)}_{k} &= \Sigma^{(i,j)}_{k|k-1}(H_{k|k-1})'[\Sigma^{(i,j)}_{\tilde{O}_k}]^{-1} && (5.35)\\
\hat{Z}^{(i,j)}_{k|k} &= \hat{Z}^{(i,j)}_{k|k-1} + K^{(i,j)}_{k}\tilde{O}^{(i,j)}_{k} && (5.36)\\
\Sigma^{(i,j)}_{k|k} &= \Sigma^{(i,j)}_{k|k-1} - K^{(i,j)}_{k}\Sigma^{(i,j)}_{\tilde{O}_k}(K^{(i,j)}_{k})' && (5.37)
\end{aligned}$$

where, $K^{(i,j)}_k$ is the (elemental) Kalman gain, $H_{k|k-1}$ is the Jacobian matrix of function $h^{(j)}(\cdot)$ at point $\hat{Z}^{(i,j)}_{k|k-1}$.

While using the Kalman-type filters for the elemental estimator implementation, an assumption is made that the density $p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1})$ in Eqn.5.30 is Gaussian and computed by

$$p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1})$$

$$= (2\pi)^{-\frac{d}{2}} \left|\Sigma_{\tilde{O}_k}^{(i,j)}\right|^{-\frac{1}{2}} exp\{-\frac{1}{2}(\tilde{O}_k^{(i,j)})'[\Sigma_{\tilde{O}_k}^{(i,j)}]^{-1}\tilde{O}_k^{(i,j)}\} \quad (5.38)$$

where $\tilde{O}_k^{(i,j)}$ and $\Sigma_{\tilde{O}_k}^{(i,j)}$ are obtained from the Kalman filter given in Eqn.(5.31)-(5.37), d is the dimension of $\tilde{O}_k^{(i,j)}$.

Algorithm summary

Each recursion of the GPB2 algorithm, tailored for our specially constrained, nonlinear switching state-state model of speech, as detailed above in this sub-section can be summarized below for each time step:

1). Calculate elemental estimates, $\hat{Z}_{k|k}^{(i,j)}$ and $\Sigma_{k|k}^{(i,j)}$, for $i, j = 1, 2, ..., M$, using the EKF listed in Eqn.(5.31)-(5.37), using the merged state estimates at the previous time step, $\hat{Z}_{k-1|k-1}^{(i)}$ and $\Sigma_{k-1|k-1}^{(i)}$ as initial values.

2). Calculate the merging probabilities, $p(s_{k-1} = i|s_k = j, O_1^k)$, according to Eqn. (5.28), (5.30) and (5.38).

3). Merge states for each model at the current time step using the merging probabilities according to Eqn.(5.24) and (5.25). (These will be used for the next time-step recursion).

4). Calculate model probabilities, $P(s_k = j|O_1^k)$, according to Eqn.(5.29), (5.30) and (5.38) .

5). Calculate the state estimate, $\hat{Z}_{k|k}$ and $\Sigma_{k|k}$, according to Eqn.(5.26) and (5.27).

This algorithm is illustrated in Fig. 5.3

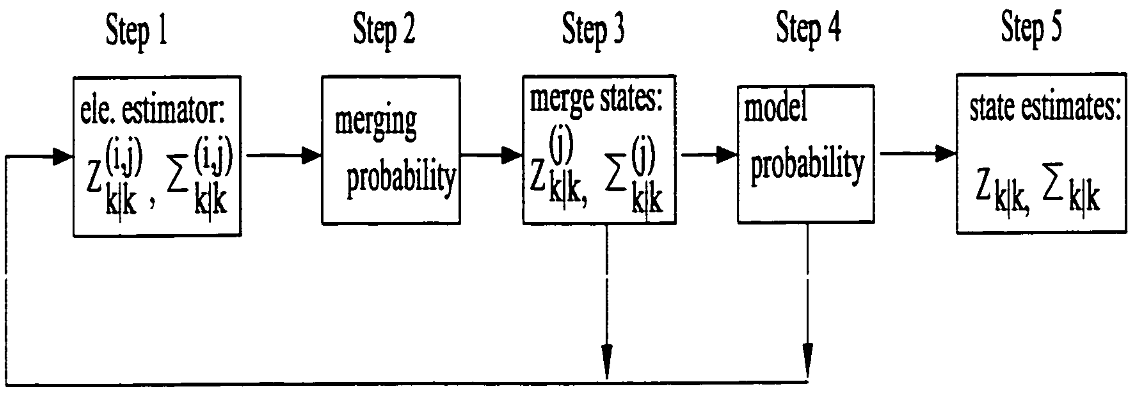

Step 1
Step 2
Step 3
Step 4
Step 5
ele. estimator: $Z^{(i,j)}_{k|k}, \Sigma^{(i,j)}_{k|k}$
merging probability
merge states: $Z^{(j)}_{k|k}, \Sigma^{(j)}_{k|k}$
model probability
state estimates: $Z_{k|k}, \Sigma_{k|k}$

Figure 5.3: A flow chart of GPB2 state estimation algorithm

5.4.4 Approximation II: An interacting multiple model approach

In the general interacting multiple model (IMM) approach [77, 78], the state estimate is carried out under each possible current model at each time step k. However, each possibility has its own initial value obtained by a weighted sum over all the estimates at the previous time $k-1$. This differs from the GPB1 approach [78] in which all possibilities are initialized by the same value. The merging in the IMM approach takes place just before the estimation begins at each current model, which differs from GPB2 in which the merging takes place after the estimation is completed at each current model. Like GPB1, the IMM approach has the same M possibilities considered at each time step, and hence have the same computational complexity. IMM is more accurate than GPB1. Compared with GPB2, the IMM approach has significantly lower computational complexity (a factor of M), but since the merging is more aggressive, it is expected to have a lower estimation accuracy.

In this subsection, we adapt the general IMM approach to suit the special need of our specially constrained state-space model of speech dynamics. In this approach, the elemental estimator is approximated by

$$E[Z(k)|s_k = j, O_1^k]$$

and

$$Cov[Z(k)|s_k = j, O_1^k]$$

. Different from the GPB2 approach, each elemental estimator in the IMM approach now has its own initial condition. The Kalman-type filter algorithm for this case becomes

$$\begin{aligned}
\hat{Z}^{(j)}_{k|k-1} &= \Phi^{(j)}\bar{Z}^{(j)}_{k-1|k-1} + (I - \Phi^{(j)})T^{(j)} && (5.39)\\
\Sigma^{(j)}_{k|k-1} &= \Phi^{(j)}\bar{\Sigma}^{(j)}_{k-1|k-1}\Phi^{(j)'} + Q^{(j)} && (5.40)\\
\tilde{O}^{(j)}_k &= O(k) - h(\hat{Z}^{(j)}_{k|k-1}) && (5.41)\\
\Sigma^{(j)}_{\tilde{O}_k} &= H_{k|k-1}\Sigma^{(j)}_{k|k-1}(H_{k|k-1})' + R^{(j)} && (5.42)\\
K^{(j)}_k &= \Sigma^{(j)}_{k|k-1}(H_{k|k-1})'[\Sigma^{(j)}_{\tilde{O}_k}]^{-1} && (5.43)\\
\hat{Z}^{(j)}_{k|k} &= \hat{Z}^{(j)}_{k|k-1} + K^{(j)}_k\tilde{O}^{(j)}_k && (5.44)\\
\Sigma^{(j)}_{k|k} &= \Sigma^{(j)}_{k|k-1} - K^{(j)}_k\Sigma^{(j)}_{\tilde{O}_k}(K^{(j)}_k)' && (5.45)
\end{aligned}$$

where, $\bar{Z}^{(j)}_{k-1|k-1}$ and $\bar{\Sigma}^{(j)}_{k-1|k-1}$ are initial values for the j-th elemental estimator. These initial values are obtained by merging according to:

$$\begin{aligned}
\bar{Z}^{(j)}_{k-1|k-1} &\equiv E[Z(k-1)|s_k = j, O_1^{k-1}]\\
&= \sum_{i=1}^{M} P(s_{k-1} = i|s_k = j, O_1^{k-1})\hat{Z}^{(i)}_{k-1|k-1} && (5.46)\\
\bar{\Sigma}^{(j)}_{k-1|k-1} &\equiv Cov[Z(k-1)|s_k = j, O_1^{k-1}]\\
&= \sum_{i=1}^{M} P(s_{k-1} = i|s_k = j, O_1^{k-1})\{\Sigma^{(i)}_{k-1|k-1}\\
&\quad + (\hat{Z}^{(i)}_{k-1|k-1} - \bar{Z}^{(j)}_{k-1|k-1})(\hat{Z}^{(i)}_{k-1|k-1} - \bar{Z}^{(j)}_{k-1|k-1})'\} && (5.47)
\end{aligned}$$

In the above Eqns.(5.46) and (5.47), $P(s_{k-1} = i|s_k = j, O_1^{k-1})$ is called the **mixing probability**, and is computed recursively from Bayes' rule:

$$\begin{aligned} P(s_{k-1} = i|s_k = j, O_1^{k-1}) &= \frac{P(s_k = j|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|O_1^{k-1})}{\sum_{i=1}^M P(s_k = j|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|O_1^{k-1})} \\ &= \frac{P(s_k = j|s_{k-1} = i)P(s_{k-1} = i|O_1^{k-1})}{\sum_{i=1}^M P(s_k = j|s_{k-1} = i)P(s_{k-1} = i|O_1^{k-1})} \qquad (5.48) \end{aligned}$$

where the model probability $P(s_{k-1} = j|O_1^{k-1})$ is updated as in GPB2 (given by Eqn.(5.29)). However, the density of the innovation sequence $p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1})$ has been changed to

$$p(O(k)|s_k = j, s_{k-1} = i, O_1^{k-1}) = (2\pi)^{-\frac{d}{2}}\left|\Sigma_{\tilde{O}_k}^{(j)}\right|^{-\frac{1}{2}} exp\{-\frac{1}{2}(\tilde{O}_k^{(j)})'[\Sigma_{\tilde{O}_k}^{(j)}]^{-1}\tilde{O}_k^{(j)}\} \quad (5.49)$$

where $\tilde{O}_k^{(j)}$ and $\Sigma_{\tilde{O}_k}^{(j)}$ are obtained from the Kalman filter given in Eqns.(5.39) - (5.45).

Algorithm summary

The entire IMM algorithm for state estimation can be summarized as follows (each time step):

1). Calculate mixing probabilities, $P(s_{k-1} = i|s_k = j, O_1^{k-1})$, using Eqn.(5.48).

2). Calculate merged initial conditions, $\bar{Z}_{k-1|k-1}^{(j)}$ and $\bar{\Sigma}_{k-1|k-1}^{(j)}$, for each current model according to Eqn.(5.46) and (5.47), given the estimates at the previous time step $k-1$.

3). Calculate the estimates, $\hat{Z}_{k|k}^{(j)}$ and $\Sigma_{k|k}^{(j)}$, by running the KF listed in Eqn.(5.39) - (5.45).

4). Update model probabilities, $P(s_k = j|O_1^k)$, according to Eqn.(5.29), (5.30) and (5.49).

5). Calculate the state and its covariance estimates, $\hat{Z}_{k|k}$ and $\Sigma_{k|k}$, according to according to Eqn.(5.26) and (5.27).

This algorithm is illustrated in Fig. 5.4.

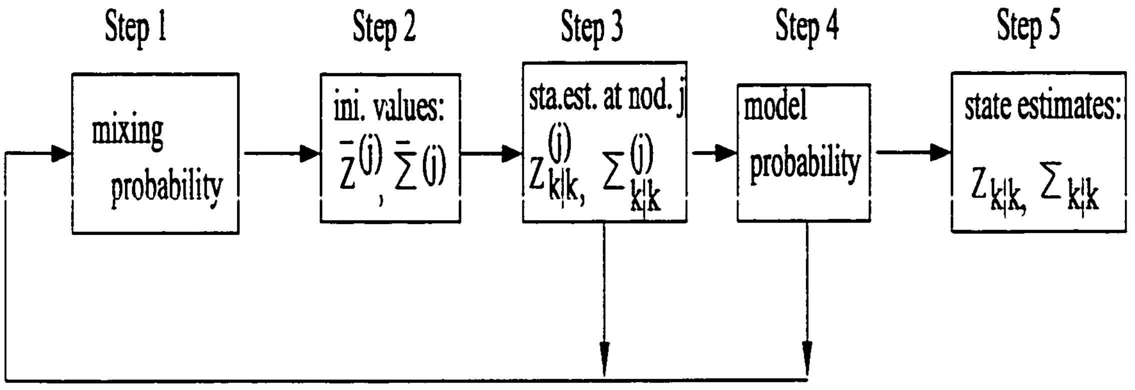

Step 1
Step 2
Step 3
Step 4
Step 5
mixing probability
ini. values: $\bar{Z}^{(j)}, \bar{\Sigma}^{(j)}$
sta.est. at nod. j $Z^{(j)}_{k|k}, \Sigma^{(j)}_{k|k}$
model probability
state estimates: $Z_{k|k}, \Sigma_{k|k}$

Figure 5.4: A flow chart of IMM state estimation algorithm

5.4.5 Analysis and comparison of GPB2 and IMM approximations

The essence or the commonplace of the merging in GPB2 and IMM is to approximate a Gaussian sum using a single Gaussian. The difference of them is the places they take place. The merging precedes KF in IMM but follows KF in GPB2. This is elaborated in a brief analysis in this subsection below.

To obtain the state estimates at node j, $\hat{Z}^{(j)}_{k|k}$ and $\Sigma^{(j)}_{k|k}$ in Eqn.(5.26) and (5.27), which are used for both GPB2 and IMM but calculated according to different ways in each approach, is eventually to calculate the conditional PDF $p(Z(k)|s_k = j, O_1^k)$. This conditional PDF can be shown to be equal to [94]:

$$\begin{aligned} p(Z(k)|s_k = j, O_1^k) &= \frac{p(O(k)|s_k = j, Z(k), O_1^{k-1})}{p(O(k)|s_k = j, O_1^{k-1})} p(Z(k)|s_k = j, O_1^{k-1}) \\ &= \frac{p(O(k)|s_k = j, Z(k))}{p(O(k)|s_k = j, O_1^{k-1})} p(Z(k)|s_k = j, O_1^{k-1}) \qquad (5.50) \end{aligned}$$

where $p(Z(k)|s_k = j, O_1^{k-1})$ can be expressed as (see proof in Appendix):

$$\begin{aligned} & p(Z(k)|s_k = j, O_1^{k-1}) \\ = & \int p(Z(k)|Z(k-1), s_k = j, O_1^{k-1}) \\ & \cdot [\sum_{i=1}^{M} p(Z(k-1)|s_{k-1} = i, O_1^{k-1}) P(s_{k-1} = i|s_k = j, O_1^{k-1})] \, dZ(k-1) \end{aligned} \tag{5.51}$$

Let's examine Eqns.(5.51) and (5.50) in some detail. First, it is not hard to find out that the PDF $p(Z(k)|s_k = j, O_1^k)$ is recursively computed because its value at the previous time point $p(Z(k-1)|s_{k-1} = i, O_1^{k-1})$ is needed in Eqn.(5.51). Second, in Eqn. (5.51) $p(Z(k)|Z(k-1), s_k = j, O_1^{k-1})$ and $p(Z(k-1)|s_{k-1} = i, O_1^{k-1})$ are Gaussians and $P(s_{k-1} = i|s_k = j, O_1^{k-1})$ is a constant. So the component, $\sum_{i=1}^{M} p(Z(k-1)|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|s_k = j, O_1^{k-1})$, is a Gaussian sum, and this makes the entire integral of Eqn.(5.51) to be also a Gaussian sum. In Eqn.(5.50) $p(O(k)|s_k = j, Z(k))$ is a Gaussian and the denominator $p(O(k)|s_k = j, O_1^{k-1})$ is a constant independent of $Z(k)$. Hence, this further makes $p(Z(k)|s_k = j, O_1^k)$ in Eqn.(5.50) to be a Gaussian sum.

Usually the initial value at time 0, $Z(0)$, is assumed to follow a Gaussian distribution, or $p(Z(0)|s_0 = i)$ is a Gaussian. At the first time point $p(Z(1)|s_1 = j, O(1))$ becomes a Gaussian sum with M mixtures after the calculation in Eqn. (5.51) and (5.50) (actually filtering process). At the second time point, $p(Z(1)|s_1 = j, O(1))$ will be plugged into Eqn.(5.50) to compute $p(Z(2)|s_2 = j, O(2))$. Because $p(Z(1)|s_1 = j, O(1))$ is a mixture Gaussian with M mixtures, it will make $p(Z(2)|s_2 = j, O_1^2)$ a mixture Gaussian with $M * M$ mixtures after the filter processing. At the third time point, following the same procedure, $p(Z(3)|s_3 = j, O_1^3)$ will become a mixture Gaussian with $M * M * M$ mixtures. If it keeps going, the number of mixtures will grow exponentially with time. This also demonstrates, from another point of view, infeasibility of the original state estimation. To prevent the number of mixtures from expanding exponentially, the merging strategies are doing this: at each time point, the mixture Gaussian $p(Z(k)|s_k = j, O_1^k)$ is approximated by a single Gaussian, then it is used for the next recursion. By this way the number of mixtures of $p(Z(k)|s_k = j, O_1^k)$ is always kept to be M.

Therefore, the essence of the merging in GPB2 and IMM is the use of approximations of a Gaussian sum by a unimodal Gaussian (the Gaussian sum, $p(Z(k)|s_k = j, O_1^k)$, is approximated by a unimodal Gaussian). However, there are two places where the approximation can be completed. In the IMM approximation, the Gaussian sum in Eqn.(5.51),

$$\sum_{i=1}^{M} p(Z(k-1)|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|s_k = j, O_1^{k-1})$$

is approximated by a single Gaussian before KF, which results in the approximate unimodal Gaussian for $p(Z(k)|s_k = j, O_1^k)$. In contrast, for the GPB2, the Gaussian sum, $p(Z(k)|s_k = j, O_1^k)$, is approximated directly by a single Gaussian after KF.

Certainly, those two approaches result in different approximation accuracies and computational tradeoffs. Since the approximation is done after KF process in GPB2 and before KF process in IMM, GPB2 approach is more accurate than IMM. On the other hand, since the mixture Gaussian

$$\sum_{i=1}^{M} p(Z(k-1)|s_{k-1} = i, O_1^{k-1})P(s_{k-1} = i|s_k = j, O_1^{k-1})$$

is approximated by one single Gaussian before KF in IMM, IMM only need run one KF at each time step. On the contrary, GPB2 has to run M KFs at each time step. So IMM is faster than GPB2.

5.4.6 Decoding algorithms incorporating the merging strategies

The state estimation methods described in the preceding section have been adapted from the methods well established in the control engineering, econometrics, and time series analysis, where the interest has been mainly focused on the accuracy of the (continuous) state estimate confined within each individual and *local* discrete state. Our contribution therein has been to tailor the methods to suit our special speech model's structure. In this section, we incorporate these methods into a new speech recognition decoding strategy, aiming to search (time-synchronous) for the *global* optimal path through all discrete states (i.e., the entire parameter switching history). Once the global path is found, the recognizer produces the text output according to this global path.

Before we present the two global decoding algorithms (incorporating the GPB2 and IMM merging strategies, respectively), we briefly explain why a straightforward use of the state estimation methods described in Section 3 is *not* desirable. For both the GPB2 and the IMM, at each time step the posterior probability, $P(s_k = j|O_1^k)$, can be computed for each discrete state (j) or phone (Eqn.(5.29)). Therefore, at each time step, if we were to "decode" the discrete state based on the highest posterior probability ($argmax_{1 \leq j \leq M}\ P(s_k = j|O_1^k)$), we would have a global "maximum a posterior" path. However, this will not be consistent with the decoding criterion for the maximum joint likelihood of the observation and path: $max_{\psi_T \in \Psi_T}\ L(O_1^T, \psi_T)$.

We use an example to show inadequacy of the decoding criterion:

$$argmax_{1 \leq j \leq M}\ P(s_k = j|O_1^k)$$

. In our left-to-right structure for the discrete-state or phone sequence, we have a sequence of observations that belong to an earlier discrete state. If there is an outlier frame within the observations which makes the dynamics switch to a later discrete state, the left-to-right constraint will not allow the dynamics to switch back to the earlier (correct) discrete state. This is so even if the observations after the outlier are correctly consistent with the earlier state. That is, the earlier state would not be allowed to be chosen even if it had the highest posterior probability ($P(s_k = j|O_1^k)$ after the outlier. This is clearly not desirable.

To implement a desirable decoding rule, $max_{\psi_T \in \Psi_T} \; L(O_1^T, \psi_T)$, which embeds a desirable mechanism of imposing the left-to-right constraint, we have developed a new dynamic programming based strategy. For the new speech dynamic model the standard Viterbi algorithm used for HMM cannot be applied directly to search for optimal dynamic regimes. This is so because different paths entering one node j in the trellis bring different initial values (due to different path histories) to the local score calculation at node j. For example, in Eqn.(5.31)-(5.37), the filtered values and score calculation at state j and time k of the path coming from state i depend on the initial values $\hat{Z}^{(i)}_{k-1|k-1}$ and $\Sigma^{(i)}_{k-1|k-1}$. These differential initial values will produce different filtered values and scores, which will be again used, due to the dynamics' continuity imposed across the adjacent discrete states, as initial values for the expansion of those paths at the next time $k+1$.

However, the GPB2 and IMM state estimation algorithms have provided effective ways to prevent the above problem of exponential growth of paths. The GPB2 uses Eqn.(5.24) and (5.25) to merge the different filtered values of those paths entering node j at time k. $\hat{Z}^{(i,j)}_{k|k}$ and $\Sigma^{(i,j)}_{k|k}$ are merged to a single point, $\hat{Z}^{(j)}_{k|k}$ and $\Sigma^{(j)}_{k|k}$, according to their a posterior probabilities after the filtering at the current model. The IMM uses Eqn.(5.46) and (5.47) to merge the different initial values of those paths entering node j, $\hat{Z}^{(i)}_{k-1|k-1}$ and $\Sigma^{(i)}_{k-1|k-1}$, to a single initial point, $\bar{Z}^{(j)}_{k-1|k-1}$ and $\bar{\Sigma}^{(j)}_{k-1|k-1}$, according to a posterior probabilities before the filtering begins at the current model. Both the GPB2 and IMM force all those paths to start off from the same point (have identical initial values) for their expansion at next time $k + 1$. .

With incorporation of the GPB2 and IMM merging strategies into the dynamic programming search mechanism, two decoding algorithms are developed, one is the GPB2-D algorithm, the other is IMM-D algorithm. The GPB2-D algorithm is listed in Fig.(5.5) and the IMM-D algorithm listed in Fig. 5.6.

In GPB2-D algorithm the first step of GPB2 state estimation algorithm is inserted inside the previous node loop. So different local scores are calculated for different paths coming from previous nodes because of the different initial conditions brought in. Those local scores are added to their corresponding path scores, then path deletions are carried out. The second, third and fourth steps of GPB2 state estimation algorithm are inserted outside the previous node loop and inside the current node loop. So different paths ending at each of the current nodes are merged to a single one. At the end of each time step there is only one merged path coming out of each node.

In IMM-D algorithm the first four steps of IMM state estimation algorithm are all inserted before the previous node loop. So for each of the current node all different initial conditions brought by different paths coming from the previous nodes are merged to one initial condition. Then one common local score is computed for all different paths, the common score is added to different path scores and then path deletions are carried out. Similarly, after each time step only one merged path comes out of each of the current nodes.

Note that in both algorithms the last steps of GPB2 and IMM state estimation algorithms are discarded. That is because we are just borrowing GPB2's and IMM's merging strategies and we don't care about the state estimates at each local time step.

```
For each frame, O(k), in the observation sequence of an utterance ( $0 < k < K+1$ )
   Initialize Best_score(k) to be negative infinity.
   For each node (model or phone) j in the lattice (or n-best list ) ( $0 < j < M+1$ )
      Initialize max. path log-likelihood L(k,j ) to be negative infinity.
      For each node i which can enter into node j
         Run step 1 of GPB2 state estimation algorithm to obtain local score L(O(k) | j, i )
           and filtered values.
         Let path log-likelihood L(k,j,i ) = L(k-1, i) + L (O(k) | j,i );
         if ( L(k,j,i ) > L(k,j ) )
            L(k,j ) = L(k,j, i );
            Remeber the tracking-back pointer from node j to node i .
         endif
         if ( L(k,j ) > Best_score(k) )
            Best_score(k) = L(k,j );
         endif
      End
      Run step 2, 3 and 4 of GPB2 state estimation algorithm to obtain the merged states
        and other updates for the next time step.
   End
   Impose path constraints here if needed.
   Do path pruning here if needed.
    [ if ( | (L(k,n ) - Best_score(k)) / Best_score(k) | > Pruning_thres ), delete node n
        for next expansion. ( $0 < n < M+1$ ) ]
End
Find the path with the highest likelihood.
Backtrack to obtain segment boundaries if needed.
```

Figure 5.5: The GPB2-D algorithm

```
For each frame, O(k), in the observation sequence of an utterance ( 0 <k<K+1)
   Initialize Best_score(k) to be negative infinity.
   For each node (model or phone) j in the lattice (or n-best list) (0 < j < M+1)
      Initialize max. path log-likelihood L(k,j) to be negative infinity.
      Run step 1,2,3,4 of IMM state estimation algorithm to obtain local score L(O(k) | j )
        and other updates for the next time step.
      For each node i which can enter into node j
         Let path log-likelihood L(k,j,i ) = L(k-1, i) + L(O(k) | j ).
         if ( L(k,j,i ) > L(k, j ) )
            L(k,j ) = L(k,j,i ),
            Remeber the tracking-back pointer from node j to i.
         endif
         if ( L(k,j ) > Best_score(k) )
            Best_score(k) = L(k,j );
         endif
      End
   End
   Impose boundary constraints here if needed.
   Do path pruning here if needed.
    [ if ( | (L(k,n) - Best_score(k))/ Best_score(k) | > Pruning_thres ), delete node n
      for next expansion ( 0 < n < M+1).
End
Find the path with the highest likelihood.
Backtrack to obtain segment boundaries if needed.
```

Figure 5.6: The IMM-D algorithm

A trellis search example is depicted in Fig.5.7. Let's use this diagram to illustrate the path merging mechanisms in the two decoding algorithms. There are many paths entering each node at each time point (suppose time t and at node $M-1$), each path brings different dynamics (or initial conditions). At each node, the path merging mechanisms in GPB2-D and IMM-D algorithms are illustrated in Fig. 5.8. From the picture we can see clearly how the two types of merging take place. In GPB2 algorithm different dynamics and different local scores are calculated for different paths which bring different dynamics (they are denoted by $(Z,\Sigma)^i_{k-1,k-1}, 1 \leq i \leq M$ in Fig.5.8), then those dynamics are merged to a single one and path deletion is done simultaneously. Only the path with highest score is kept. In IMM algorithm all the different dynamics are first merged to a single one, then a common local score and dynamics are calculated for all those paths. So path deletion can be done before the local score computation. Similarly, only the most likely path is kept. Therefore, only one path comes out of each node after merging in the two decoding algorithms.

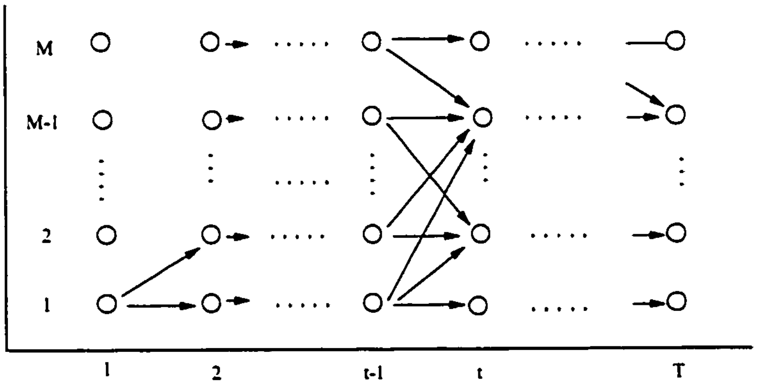
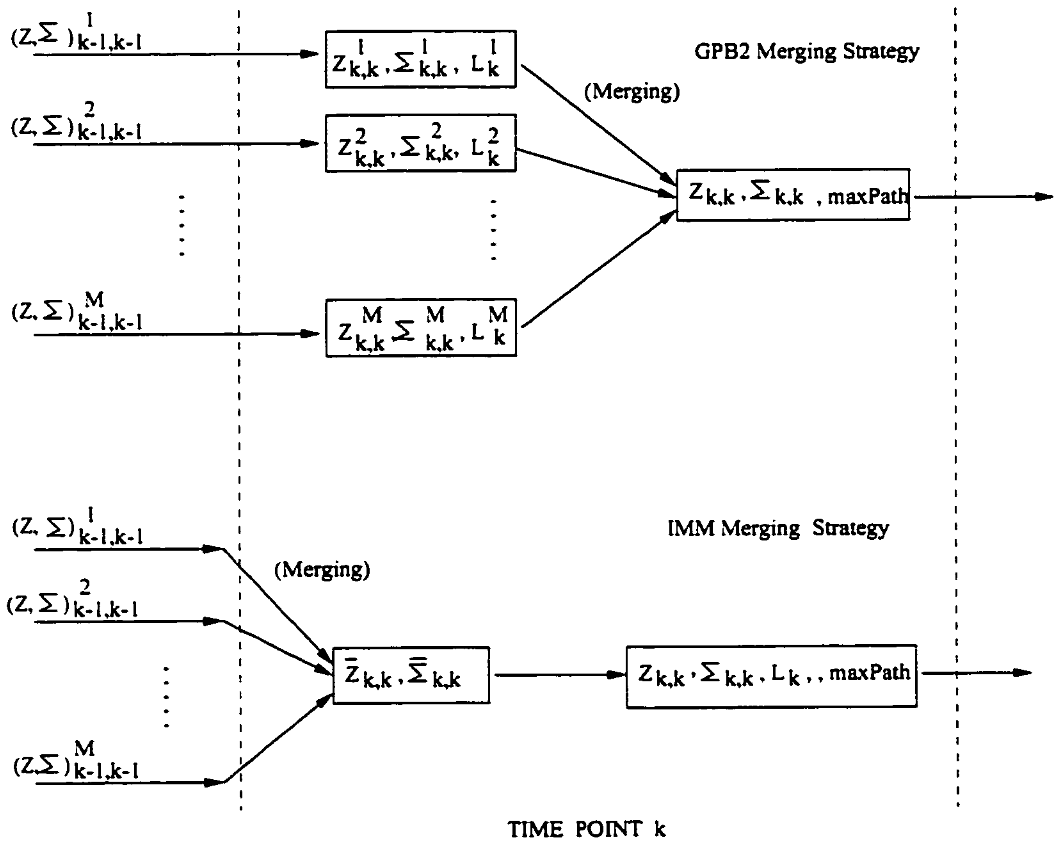


Figure 5.7: A trellis search diagram to show how the two merging strategies in GPB2-D and IMM-D algorithms work



GPB2 Merging Strategy
(Merging)
$(Z,\Sigma)^1_{k-1,k-1}$
$(Z,\Sigma)^2_{k-1,k-1}$
$(Z,\Sigma)^M_{k-1,k-1}$
$Z^1_{k,k}, \Sigma^1_{k,k}, L^1_k$
$Z^2_{k,k}, \Sigma^2_{k,k}, L^2_k$
$Z^M_{k,k}, \Sigma^M_{k,k}, L^M_k$
$Z_{k,k}, \Sigma_{k,k}$, maxPath
IMM Merging Strategy
(Merging)
$\bar{Z}_{k,k}, \bar{\Sigma}_{k,k}$
$Z_{k,k}, \Sigma_{k,k}, L_k$, , maxPath
TIME POINT k

Figure 5.8: A diagram to illustrate the GPB2 and IMM merging strategies

5.5 Comparison experiments on simulation data and a small amount of speech data

In this section, the three decoding algorithms, PS-D algorithm, GPB2-D algorithm and IMM-D algorithm, are tested and compared on both simulated data and real speech data. In all the experiments in this section all phones share a single global MLP.

The test results for the simulated data are presented as comparisons of accuracy in the estimated dynamic regime boundaries among the various methods. The test results for the real speech data (from the Switchboard corpus) are presented as the utterance-likelihood comparisons among the decoding algorithms based on the various approximate state estimation methods.

5.5.1 Experiments on simulated data

In these experiments, five different models are chosen, they share a common MLP. All the model parameters and MLP weights are fixed to generate simulation data. Then the simulated data is used to estimate the dynamic regime boundaries. Finally the estimated boundaries are compared with those real ones to check if the algorithms are able to pick up the optimal dynamic boundaries.

The results [1, 5] show that with a lower noise level, all three decoding algorithms are able to recover the true boundaries between the five models. This demonstrates that the algorithms are effective and that they have been implemented correctly. With a higher noise level, all the decoding algorithms began to produce some small errors in the optimized boundaries in comparison with the true ones. It appears that the level of noises (state noise and measurement noise) is crucial for the accuracy in the boundary estimation (equivalent problem to the estimation of discrete state sequence as required by speech recognition). More details can be found in [1, 5].

5.5.2 Comparison experiment on a sub-set of Switchboard database

In this simple experiment, all phone models share a common MLP (or $h^{(i)}$'s, $1 \leq i \leq M$, are implemented by a common MLP). The **"1/2 hour"** training data set is used for model training.

The test set consists of 240 utterances extracted randomly out of the male side of "test-ws97-dev-l" test set (a total of 23 male speakers, 24 conversation sides, and 50 minutes of speech).

Table (5.2) lists the re-scoring results (WERs) for four recognizers based on the same VTR state-space model with different methods of computing acoustic likelihoods. The "Baseline" recognizer computes the likelihoods of utterances using the sub-optimally fixed dynamic regimes provided by a separate HMM system. Recognizers labeled by "PS-D", "GPB2-D" and "IMM-D" represent the systems using the optimized dynamic regimes obtained by the PS-D, GPB2-D, and IMM-D methods, respectively. In these new recognizers, the computation of the acoustic likelihoods for each hypothesis transcription in the N-best list is carried out using the dynamic programming based decoding strategy for optimizing the dynamic regimes.

From the recognition accuracy results shown on Table (5.2), we observe that all three PS-D, GPB2-D, and IMM-D based recognizers outperform the baseline recognizer. It shows the effectiveness of the three decoding algorithms.

Comparing those three approaches, we find that the three algorithms turn out close performance. But the computational requirements for the three algorithms are

Recognizers	Ref+100	100-best
Baseline	53.2	60.6
PS-D	51.1	60.2
GPB2-D	50.1	60.1
IMM-D	49.5	59.7

Table 5.2: Word error rates (%) for four recognizers with different methods of likelihood computation using the N-best re-scoring paradigm

very different. The numbers of KFs needed at each time step for three algorithms are listed as follows:

$$\begin{aligned} PS-D &: S*M*M \\ GPB2-D &: M*M \\ IMM-D &: M \end{aligned}$$

where S is the path-stack size (≥ 1). So IMM decoding algorithm is the fastest one.

Based on those comparisons the IMM decoding algorithm is chosen the most efficient one. Then it is used on different versions of the dynamic model.

5.6 Evaluation experiments of the IMM decoding algorithm

In this section, the IMM decoding algorithm is used on three different versions of the new dynamic model. The first one is the "phone MLP version", the second one is the "MLDM-2mix" version, the mixture linear dynamic model with two mixtures, and the last one is the "MLDM-4mix" version, the mixture linear dynamic model with four mixtures. The first and second versions are trained with the **"1/2 hour"** training data set and the last version trained with the **"2 hour"** training data set.

The HMM phone boundaries are available (marked by the **"ws97-baseline"** HMM system). On the other hand. we know that physically the phone VTR dynamic boundaries must be located between the corresponding two consecutive HMM phone boundaries [6]. Therefore, this boundary constraint can be imposed on the search. In case that the HMM phone segmentations are not accurate sometimes, the HMM phone boundaries are relaxed by one or two frames (starting boundaries are moved forward and ending ones backward by one or two frames) when they are imposed on the search. By this constraint the search can be speeded up by 5-10 times.

The N-best re-scoring results (WERs) of the three versions and the two HMM systems are listed in Table 5.3. In the table "fix" means that the recognizers compute the likelihoods of utterances using the sub-optimally fixed dynamic regimes provided by the **"ws97-baseline"** HMM system, "IMM-D" means that the recognizers are using the dynamic regime boundaries optimized by the IMM decoding algorithm.

From the results listed in Table 5.3, we observe that by using the IMM-D algorithm to optimize the dynamic regime boundaries for different versions of the dynamic model consistent improvements (WER reductions) are obtained, especially for the cases with references included (about 5% - 10% WER reduction relatively).

Under identical conditions, compared with the **"HMM-baseline"** HMM system, by using the IMM-D algorithm the "MLDM-2mix" version achieves more than 15% relative WER reduction for the "Ref+100" case and more than 3% absolute

[6]Physically a phone's VTR dynamic boundary is ahead of its HMM (acoustic) boundary. So in an utterance the starting point of VTR dynamics of the current phone must be within the HMM segment of the previous phone

systems	Ref+100	100-best
phone MLP version: fix (1/2 hour)	55.6	58.3
phone MLP version: IMM-D (1/2 hour)	51.5	57.1
MLDM-2mix: fix (1/2 hour)	50.0	56.9
MLDM-2mix: IMM-D (1/2 hour)	47.7	55.7
HMM-baseline (1/2 hour)	56.1	58.9
MLDM-4mix: fix (2 hour)	49.5	56.0
MLDM-4mix: IMM (2 hour)	47.4	55.8
ws97-baseline (160 hour)	56.2	56.9

Table 5.3: WER of different systems using IMM for their dynamic regime optimization

WER reduction for the "100-best" case. Compared with the **"ws97-baseline"** HMM system, even with only two hour training data, the "MLDM-4mix" version decrease the WER by over 15% relatively for the "Ref+100" case and over 1% absolutely for the "100-best" case. It demonstrates that the IMM-D decoding algorithm is an efficient approach to optimize the dynamic regime boundaries for the new dynamic model.

5.7 Lattice Re-scoring

With the efficient IMM decoding algorithm at hand, we can move to a more realistic evaluation paradigm, lattice re-scoring. Eventually, each hypothesis in the N-best lists can be treated as a simple lattice, where paths are only allowed to either stay in the current phone or enter the next phone at each time point. So the IMM-D decoding algorithm can be directly used for lattice re-scoring. The only difference is that the number of possible extensions for each path are greatly enlarged. This will be our future work.

5.8 Conclusion and Discussion

5.8.1 Conclusion

In this chapter three decoding algorithms, the PS-D algorithm, the GPB2-D algorithm and the IMM-D algorithm, have been developed for the optimization of the dynamic regime boundaries of our dynamic model. The PS-D algorithm is using a path deletion strategy and GPB2-D and IMM-D algorithms using path merging strategies to reduce the search space. All the three algorithms have successfully overcome the formidable exponential growth in the original search and turned it into a linear one.

All the algorithms are first tested on simulated data, it is shown that they are all efficient in finding out the real dynamic regime boundaries, then they are compared on a small amount of Switchboard data and the IMM-D algorithm is chosen the most efficient one. Finally the IMM-D algorithm is further evaluated on different versions of the dynamic model, consistent improvements on different versions are observed. Compared with the **"HMM-baseline"** system, the MLDM with two mixtures achieves about 20% relative WER reduction for the case exposed to references and more than 3% absolute WER reduction for the case not exposed to references when the IMM-D algorithm is used for optimizing the dynamic regime boundaries.

5.8.2 Further discussion

The PS-D algorithm attaches a path stack to each node during the trellis search to maintain a limited number of promising paths. It use an intuitive way to do path deletion.

Both the GPB2-D and IMM-D algorithms use merging strategies to limit the exponential growth of the search space. The two mergings take place at different time, the GPB2-D method merges for each node all paths after the local score computation is complete for each path. In contrast, the IMM-D method merges for each node all paths first to create the combined initial values before the local score computation. The merging formulas for both the GPB2 and IMM methods are theoretically motivated, being derived from Bayes' rule. The consequence of the merging is direct applicability of the dynamic programming principle to recognizer decoding, which would otherwise be impossible due to the essential VTR continuity constraint imposed across speech units in the very construction of the speech model. A specific contribution of this work is establishment of a solid Bayesian strategy for the decoding problem of the new dynamic model.

The GPB2-D and IMM-D algorithms developed for our specialized, switching state-space model for the speech dynamics have been partly motivated by some earlier work on the various versions of the switching state-space model developed in control engineering, neural network, time series analysis, and in econometrics [75, 85, 83, 40, 82, 91, 78, 95, 81]. Such earlier work typically dealt with several simplified cases of the model, and focused only on the state estimation but not the decoding problem. For example, in [40] and [85], the state estimation problem for only linear state-space models was considered. In the former, switching only happens on the linear measurement equation (to trace the trajectories of multiple targets), while in the latter the switching is allowed for both linear state and measurement equations.

Both pieces of the work appear to be special cases of a more general approach to state estimation given in [75], where Bayes' rule is used when the parameter switching of the state-space model follows an arbitrary and in particular Markovian processes. The work of [83] allows the parameter switching to happen in the error covariances.

The physical motivations for constructing our switching state-space model for the speech dynamics, which determine the special structure of the model and the needed decoding algorithm, are also different from those of the earlier work in the target-tracking and econometrics literature. The earlier motivations arise from the difficulty with which exact mathematical models can be derived for the physical dynamic processes. One main cause of the difficulty is the random, unknown change of the model structures or parameters over time. However, the structure or parameter variations can often be narrowed down to where only a finite number of possibilities for the variations are allowed to be chosen. For example, sensors tracking multiple targets and plants having multiple modes of behavior can be dealt with in this way. For solving the multiple-target tracking problems, use of the switching state-space model is appropriate and a comprehensive review has been provided in [77, 78]. Another example for which the switching state-space model is appropriate is the statistical description of recession and booming stages in the economy, which has been modeled in the work of [82, 85]. A detailed review of the application of the switching state-space model to econometrics has been provided in [91].

Moving beyond the applications in target tracking and in econometrics, the research reported in this chapter represents our novel contribution of applying the specially constructed switching state-space model to functional modeling of speech production and speech recognition. The speech production process can be well fitted into the switching state-space model since each phone in a finite number of phones in a language can be associated with a largely distinct target vocal-tract shape and its related acoustic resonance structure. When a speech utterance is produced, the vocal tract shape or resonance (continuous state in the model) changes relatively smoothly from one target phone to another, where the target shapes determined by the model parameters are modeled to be switching from one target phone to its temporally adjacent one.

Taking into account the special temporal-flow properties in speech production and special requirements in speech recognition, we have developed two innovative ways of applying the switching state-space model, both distinct from the traditional approaches developed in the control-engineering and econometrics applications.

First, in the traditional applications, no constraints were imposed on the model switching process; i.e., the model parameters can arbitrarily switch from one discrete state to another with non-zero probability at any time. For example, in the models of [82] and [85], the modeled economy may relatively freely switch from recession to booming and vice versa. In the model of [40], the measurement of the sensor may be the trajectory of any one of the multiple targets at each time. For these cases, the transition matrix for the discrete state switching is a full matrix. However, for our applications to speech production and recognition only the left-to-right structure in the transition matrix is sensible. That is, once the model switches to a new stage, it will not return to the old stages. In this case, the transition matrix becomes constrained to be an upper-triangle one.

Second, as a direct consequence of the left-to-right structural constraint for the switching process, the search for the "global" optimal discrete-state sequence (i.e., the switching history) based on the observation data becomes significantly more complex than the case without such a structural constraint. In the traditional applications [75, 85, 83, 40, 82, 91, 78], exclusive attention was paid to obtain the optimal "local" state estimation under the switching condition. In contrast, for the speech recognition application as our subject of study, the interest is in how the optimal switching process develops or how to find the global optimal discrete-state transition path (i.e., the optimal phone or word sequence). It is interesting to note that if there were no structural constraints for the switching process, the objectives in both of the above cases would be identical because the "global" optimal solution must be achieved by selecting the best estimate at each "local" point. However, once the constraints are imposed as essential for our current speech recognition application, the traditional approaches must be extended because the constraints may prevent the "global" solution from choosing the best estimates at some "local" points. One main contribution of the work presented in this chapter is to accomplish such an extension based on the dynamic programming principle.

Chapter 6

Summary and Future Work

6.1 Summary of this thesis

In this thesis, a new statistical coarticulatory dynamic model for speech recognition is developed and evaluated. As analyzed, this new model can be treated as an extension of the work done in [54, 60], but there are some fundamental difference from them. The main novelty of this new model is the introduction of the vocal tract resonance as the internal, structured model hidden state (continuous-valued) for representing phonetic reduction and target undershoot in human production of spontaneous speech and the incorporation of knowledge about the dynamic, target-directed behavior in the vocal tract resonance into the model design, training, and likelihood computation.

In Chapter 2, the earliest version of the dynamic model, which was proposed by Deng [33], was presented and evaluated. In this version the model is formulated mathematically as a nonlinear dynamic system, where different neural networks (MLPs and RBFNNs) are used to approximate the physically nonlinear relationship between the hidden dynamic (VTR) space and observation (MFCC) space in the measurement equation. A version of the generalized EM algorithm was developed and implemented for automatically learning the model parameters. Evaluation experiments were done using the spontaneous speech data from the Switchboard corpus. The promise of the new model was demonstrated by showing its superior performance over a benchmark HMM system under identical experimental conditions. It was found that different neural networks (or different nonlinear versions) did not produce much difference.

Therefore, in Chapter 3, the nonlinear versions were first compared with a linear version. it was observed that there was not much benefit from the use of nonlinear models. To keep the efficiency of a linear model in model training and likelihood computation, and to be responsible for the systematic variations in speech, a mixture linear dynamic model was developed in which several linear dynamic models are combined to describe a phone. A version of the generalized EM algorithm was derived for the model parameter learning and likelihood computation, where an important constraint, "mixture-path" constraint, was imposed. A series of experiments were carried out to evaluate this version. It was shown that, with the use of multiple mixture components, the model achieved a significant improvement. It was also demonstrated that the model performance was gradually improved with the increasing of the number of mixtures and the amount of training data. Further to this, an analysis experiment was carried out. It indicated that the incorporation of speech dynamics into the new model design contributes to a major part of the improvement, and hence the dynamic property is of importance for speech recognition. Those experiment results are consistent with the observations from other researchers working on the "segment" models [60]. That is, the increasing of the number of mixtures at the segment level is due to speech systematic variations.

In Chapter 4, a more general version of the MLDM, a mixture linear dynamic model with switching parameters on measurement equations, was developed. This version aims at alleviating the inefficiency of the mixture linear dynamic model in representing the physically nonlinear relation between the hidden dynamic space and acoustic space. It uses piece-wise linear functions, rather than linear functions in the MLDM version, to approximate the nonlinear relation on the measurement equation, which was realized by allowing parameters on the measurement equation to switch among a fixed set of values. The corresponding model training and likelihood computation algorithms were derived as well. Furthermore, a series of evaluation experiments were performed. The results showed that the accuracy in modeling the nonlinear relation of hidden state space and acoustic space was indeed increased and the model performance was also improved.

In Chapter 5, the challenging decoding problem for the dynamic model was dealt with. Because of the continuity constraint imposed on the hidden VTR dynamics, no path deletion is allowed during the trellis search if a global optimality is required. It makes the search infeasible. By analyses, three approximate solutions were provided: PS-D, GPB2-D and IMM-D algorithms. The first one is based on a path deletion strategy while the other two are based on path merging strategies. All the three algorithms successfully turn the original search problem exponential in time into one linear in time. A series of experiments were done on both simulated data and Switchboard data. It was observed that all the three decoding algorithms are efficient in searching for the real dynamic boundaries. The IMM-D algorithm is the most efficient one. Consistent improvements for different versions of the dynamic model can be obtained when the IMM-D algorithm was used to optimize their phone dynamic regime boundaries.

6.2 Summary of contributions

The contributions of this thesis are summarized as follows:

- The introduction of the VTR dynamics as the hidden state dynamics:
  The use of VTR dynamics renders the model with not only the incorporation of pre-knowledge about target-directed behavior in the VTR dynamics into the model design, training, and likelihood computation, but also the implementation feasibility because of the lower dimensionality in the VTR dynamics. It is also helpful in the model learning and diagnosis because of the partial observability of the VTR dynamics.

- The development of the mixture linear dynamic model:
  The mixture concept at segment level has been used by other researchers [53]. An important contribution in this dissertation is the rigorous derivation of the parameter learning and likelihood computation equations for this mixture linear model, where the important "mixture-path" constraint (VTR targets switch at segment level) is imposed.

- The development of the mixture linear dynamic model with switching parameters on the measurement equations:
  This model is totally new. There are two levels of switching in this version, one is at segment level while the other one is at frame level. The state equation, Eqn.(4.2), switches at segment level because of the physical property of VTR targets (they are defined at segment level), the measurement equation, Eqn.(4.3), switches not only at segment level (synchronous with the switching of state equation) but also at frame level because physically the relation between the hidden and acoustic spaces can change from frame to frame. Even in a control area this situation has not been met. Again, the parameter learning and

likelihood computation algorithms are rigorously derived with the "mixture-path" constraint imposed on the mixture switching at segment level.

- The development of three efficient decoding algorithms (PS-D, GPB2-D and IMM-D):
  Search is a challenging problem for all "segment" models [60], and no researcher has looked into this problem seriously before. The path stack concept in the PS-D algorithm is not new (it has been used in [21],etc.), the new thing is the design of an efficient operation algorithm on the maintenance of path stacks. The two path merging strategies in GPB2-D and IMM-D algorithms, borrowed from the control area, are first used in the speech recognition area. Both the merging strategies efficiently reduce the search space to the same level as Viterbi algorithm does. They enable the dynamic model to do real speech recognition, which is their most important contribution.

6.3 Future work

6.3.1 Error propagation problem

In all the experimental results, much better results have been observed when the references were added to 100-best hypothesis lists. This phenomenon is caused by the error propagation problem in the new dynamic model. When one error occurs it can propagate into the future, which is caused by the continuity constraint imposed on the hidden VTR dynamics. As illustrated in Fig. 6.1, the solid line is the true VTR dynamics and the dashed line the estimated dynamics. If phone "phn2" is incorrectly recognized as phone "phn2"', then it pulls the dynamics to a wrong direction. Even if the following phones are recognized correctly, they must start from a wrong place due to the continuity constraint on the dynamics. So the error propagates into the following phones as the dashed line shows.

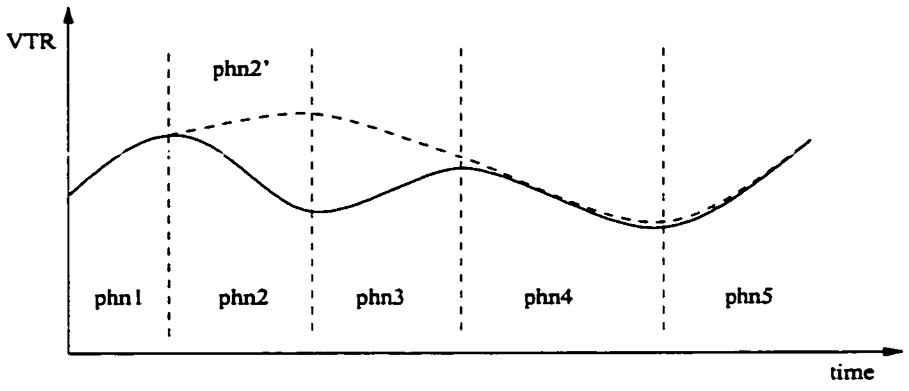


Figure 6.1: An example of error propagation problem in the new dynamic model

This constraint is physically required, and it can not be simply got rid of. Therefore, how to overcome this problem will be an important issue in the future work.

All the experiment results showed that the new dynamic model is able to pick up the correct hypotheses much more frequently than a HMM system if the correct hypotheses exist in the 100-best hypothesis lists (the "Ref+100" case). Usually lattices have much higher chances to include the correct hypotheses than 100-best lists. Hence, by doing lattice re-scoring the error propagation problem would be alleviated and more improvements be expected. Fortunately, the developed decoding algorithms enable the dynamic model to do lattice re-scoring.

6.3.2 Distribution of model parameters

Different speakers have somewhat different targets in the VTRs and different speaking rates (due to vocal tract geometric differences, for example) or even the same speaker may have different speaking rates at different times. To account for the systematic variations in speech, like speaker, gender, speaking style, and speaking rate variations, it is natural to use multiple points or a distribution, rather than a single point, for the model parameters when multiple-speaker data are used (such as the switchboard data). That is why big improvements were obtained from a single linear dynamic model to mixture linear dynamic model in Chapter 3 and 4. From multiple points to a continuous distribution, more improvements should be expected. To do it, appropriate parametric forms for the distribution have to be designed and the corresponding effective and efficient algorithms should be developed for parameter estimation.

One possible way is to augment the state space and treat all model parameters as hidden states as well, then apply Kalman filter to estimate them, like what was done in [6]. However, some constraints have to be imposed on the model parameter distributions.

6.3.3 Non-uniqueness of phone target

In all the versions developed, It is assumed that for each phone segment there is an unique target. That is true for vowels and semi-vowels, but not true for some consonants. For example, a stop consonant has a closure followed by an explosive. Each period has a different target. How to deal with those special cases would be another issue in the future.

One possible way is to concatenate multiple dynamic models, where each model is responsible for a different period.

6.3.4 Left context dependency

During all the experiments, it was found that the estimated noises (Q and R) were at a high level. More work is apparently required to improve the accuracy of the new model. One way is to use left-context dependent phones. At the current stage, only context-independent phones are used. However, a phone's VTR dynamics is influenced by its left context, because different left contexts make the VTR dynamics go along different curves due to different time constants, Φ (although it has the identical target). It is expected that the use of left-context dependent phones produces more improvements.

6.3.5 Combination with HMM

Another more practical consideration is to combine the new dynamic model with the conventional HMM. HMM has its strong advantages, for instance, its efficient training and recognition algorithms. From the evaluation and comparison experiments, the new model also show superiority in some aspects, especially the incorporation of speech dynamic property. Another advantage of HMM is its dominance in the speech recognition area after more than two decades of development. Currently almost all the best speech recognition systems are developed on this basis. Those facts make it reasonable to consider combining the two approaches.

As analyzed in Section 2.1.3, if we set special values for the model parameters, the new dynamic model becomes a special HMM. Actually the dynamic model can be treated as a HMM with continuous states (the conventional HMM has discrete states). There exist commonplaces between the new dynamic model and the HMM, so the combination of them might be able to be done at a deeper level, not just at a surface level (like voting).

Appendix A

Derivations for Chapter 2

A.1 Derivation of Eqn.(2.28)

The derivation of Eqn.(2.28):

$$
\begin{aligned}
& E_O[y_j(Z(k))] \\
= & \int exp\{-\frac{(Z_k-\mu_j)'\Sigma_j^{-1}(Z_k-\mu_j)}{2}\}\cdot p(Z_k|O,\bar{\Theta})\ dZ_k \\
= & \int exp\{-\frac{(Z_k-\mu_j)'\Sigma_j^{-1}(Z_k-\mu_j)}{2}\} \\
& \cdot (2\pi)^{-\frac{d}{2}}|\Sigma_{k/K}|^{-\frac{1}{2}} exp\{-\frac{(Z_k-\hat{Z}_{k/K})'\Sigma_{k/K}^{-1}(Z_k-\hat{Z}_{k/K})}{2}\}\ dZ_k
\end{aligned}
\tag{A.1}
$$

Let's just consider the exponents in the above equation,

$$
\begin{aligned}
& (Z_k-\mu_j)'\Sigma_j^{-1}(Z_k-\mu_j)+(Z_k-\hat{Z}_{k/K})'\Sigma_{k/K}^{-1}(Z_k-\hat{Z}_{k/K}) \\
= & Z_k'(\Sigma_j^{-1}+\Sigma_{k/K}^{-1})Z_k - 2Z'(\Sigma_j^{-1}\mu_j+\Sigma_{k/K}^{-1}\hat{Z}_{k/K})+\mu_j'\Sigma_j\mu_j+\hat{Z}_{k/K}'\Sigma_{k/K}^{-1}\hat{Z}_{k/K} \\
= & (Z_k-Vb)'V^{-1}(Z_k-Vb)+\mu_j'\Sigma_j\mu_j+\hat{Z}_{k/K}'\Sigma_{k/K}^{-1}\hat{Z}_{k/K}-b'V^{-1}b
\end{aligned}
\tag{A.2}
$$

where

$$V=(\Sigma_j^{-1}+\Sigma_{k/K}^{-1})^{-1} \quad and \quad b=\Sigma_j^{-1}\mu_j+\Sigma_{k/K}^{-1}\hat{Z}_{k/K}.$$

Therefore,

$$\begin{aligned}
& E_O[y_j(Z(k))] \\
= \ & (2\pi)^{-\frac{d}{2}}|\Sigma_{k/K}|^{-\frac{1}{2}} \int exp\{-\frac{(Z_k - Vb)'V^{-1}(Z_k - Vb)}{2}\} \cdot exp(\delta_{j,k})\ dZ_k \\
= \ & |\Sigma_{k/K}|^{-\frac{1}{2}}|V|^{\frac{1}{2}} exp(\delta_{ik}) \qquad (A.3)
\end{aligned}$$

where

$$\begin{aligned}
\delta_{jk} = \ & -\frac{1}{2}[\mu_j'\Sigma_j^{-1}\mu_j + \hat{Z}_{k/N}'\Sigma_{k/N}^{-1}\hat{Z}_{k/N} \\
& - (\Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N})'(\Sigma_j^{-1} + \Sigma_{k/N}^{-1})^{-1}(\Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N})]
\end{aligned}$$

A.2 Derivation of (2.29)

$$\begin{aligned}
& E_O[y_r(Z(k))\ y_j(Z(k))] \\
= \ & \int exp\{-\frac{(Z_k - \mu_r)'\Sigma_r^{-1}(Z_k - \mu_r)}{2}\} \\
& \cdot exp\{-\frac{(Z_k - \mu_j)'\Sigma_j^{-1}(Z_k - \mu_j)}{2}\} p(Z_k|O.\bar{\Theta})\ dZ_k \\
= \ & exp\{-\frac{(Z_k - \mu_r)'\Sigma_r^{-1}(Z_k - \mu_r)}{2}\} \cdot exp\{-\frac{(Z_k - \mu_j)'\Sigma_j^{-1}(Z_k - \mu_j)}{2}\} \\
& \cdot (2\pi)^{-\frac{d}{2}}|\Sigma_{k/K}|^{-\frac{1}{2}} exp\{-\frac{(Z_k - \hat{Z}_{k/K})'\Sigma_{k/K}^{-1}(Z_k - \hat{Z}_{k/K})}{2}\}\ dZ_k \qquad (A.4)
\end{aligned}$$

Again, let's just consider the exponents in the above equation.

$$\begin{aligned}
& (Z_k - \mu_r)'\Sigma_r^{-1}(Z_k - \mu_r) + (Z_k - \mu_j)'\Sigma_j^{-1}(Z_k - \mu_j) \\
& + (Z_k - \hat{Z}_{k/K})'\Sigma_{k/K}^{-1}(Z_k - \hat{Z}_{k/K}) \\
= \ & Z_k'(\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})Z_k - 2Z_k'(\Sigma_r^{-1}\mu_r + \Sigma_j^{-1}\mu_j + \Sigma_{k/K}^{-1}\hat{Z}_{k/K}) \\
& + \mu_r'\Sigma_r\mu_r + \mu_j'\Sigma_j\mu_j + \hat{Z}_{k/K}'\Sigma_{k/K}^{-1}\hat{Z}_{k/K} \\
= \ & (Z_k - C_{rjk}b)'C_{rjk}^{-1}(Z_k - C_{rjk}b) + \mu_r'\Sigma_r\mu_r + \mu_j'\Sigma_j\mu_j \\
& + \hat{Z}_{k/K}'\Sigma_{k/K}^{-1}\hat{Z}_{k/K} - b'C_{rjk}^{-1}b \qquad (A.5)
\end{aligned}$$

where

$$C_{rjk} = (\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})^{-1} \quad and \quad b = \Sigma_r^{-1}\mu_r + \Sigma_j^{-1}\mu_j + \Sigma_{k/K}^{-1}\hat{Z}_{k/K}$$

Therefore,

$$\begin{aligned} & E_O[y_r(Z(k))\, y_j(Z(k))] \\ = \; & (2\pi)^{-\frac{d}{2}}|\Sigma_{k/K}|^{-\frac{1}{2}} \int exp\{-\frac{(Z_k - C_{rjk}b)'C_{rjk}^{-1}(Z_k - C_{rjk}b)}{2}\} \cdot \delta_{rjk}\, dZ_k \\ = \; & |\Sigma_{k/K}|^{-\frac{1}{2}}|C_{rjk}|^{\frac{1}{2}} exp(\delta_{rjk}) \end{aligned} \tag{A.6}$$

where

$$\begin{aligned} C_{rjk} &= (\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})^{-1} \\ \delta_{rjk} &= \frac{1}{2}(\mu_r'\Sigma_r^{-1}\mu_r + \mu_j'\Sigma_j^{-1}\mu_j + \hat{Z}_{k/N}'\Sigma_{k/N}^{-1}\hat{Z}_{k/N} - M_{rjk}'C_{rjk}^{-1}M_{rjk}) \\ M_{rjk} &= (\Sigma_r^{-1} + \Sigma_j^{-1} + \Sigma_{k/K}^{-1})^{-1}(\Sigma_r^{-1}\mu_r + \Sigma_j^{-1}\mu_j + \Sigma_{k/N}^{-1}\hat{Z}_{k/N}) \end{aligned}$$

Appendix B

Derivations for Chapter 3

B.1 Derivation of Eqn.(3.19)

From Eqn.(3.18), we have

$$
\begin{aligned}
Q(\Theta|\bar{\Theta}) &= \sum_{\{X\}^N} \{ \sum_{n=1}^{N} \int \{ \log p(Z_0^n|m^n) \\
&\qquad + \sum_{k=1}^{K_n} [\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n) + \log p(O_k^n|Z_{k,m^n}^n)] \\
&\qquad \} \cdot p(Z^n|O^n, m^n, \bar{\Theta})\, dZ^n \\
&\quad + \sum_{n=1}^{N} \log p(m^n|\Theta) \} \cdot p(\{X\}^N|\{O\}^N, \bar{\Theta}) \\
&= \sum_{n=1}^{N} \sum_{\{X\}^N} \{ \int \{ \log p(Z_0^n|m^n) \\
&\qquad + \sum_{k=1}^{K_n} [\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n) + \log p(O_k^n|Z_{k,m^n}^n)] \\
&\qquad \} \cdot p(Z^n|O^n, m^n, \bar{\Theta})\, dZ^n \\
&\quad \} \cdot p(\{X\}^N|\{O\}^N, \bar{\Theta})
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{n=1}^{N}\sum_{\{X\}^N}\log p(m^n|\Theta)\cdot p(\{X\}^N|\{O\}^N,\bar{\Theta})\\
=\ &\sum_{n=1}^{N}\sum_{X^n}\sum_{\{X\}^N/X^n}\{\int\{\ \log p(Z_0^n|m^n)\\
&\qquad\qquad+\sum_{k=1}^{K_n}[\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n)+\log p(O_k^n|Z_{k,m^n}^n)]\\
&\qquad\qquad\}\cdot p(Z^n|O^n,m^n,\bar{\Theta})\ dZ^n\\
&\qquad\}\cdot p(\{X\}^N|\{O\}^N,\bar{\Theta})\\
&+\sum_{n=1}^{N}\sum_{X^n}\sum_{\{X\}^N/X^n}\log p(m^n|\Theta)\cdot p(\{X\}^N|\{O\}^N,\bar{\Theta})\\
=\ &\sum_{n=1}^{N}\sum_{X^n}\{\int\{\ \log p(Z_0^n|m^n)\\
&\qquad+\sum_{k=1}^{K_n}[\log p(Z_{k,m^n}^n|Z_{k-1,m^n}^n)+\log p(O_k^n|Z_{k,m^n}^n)]\\
&\qquad\}\cdot p(Z^n|O^n,m^n,\bar{\Theta})\ dZ^n\\
&\quad\}\sum_{\{X\}^N/X^n}p(\{X\}^N|\{O\}^N,\bar{\Theta})\\
&+\sum_{n=1}^{N}\sum_{X^n}\log p(m^n|\Theta)\sum_{\{X\}^N/X^n}p(\{X\}^N|\{O\}^N,\bar{\Theta})\\
=\ &\sum_{n=1}^{N}\{\sum_{m=1}^{M}\int\{\log p(Z_0^n|m)+\sum_{k=1}^{K_n}[\log p(Z_{k,m}^n|Z_{k-1,m}^n)\\
&\qquad+\log p(O_k^n|Z_{k,m}^n)]\}\cdot p(Z^n|O^n,m,\bar{\Theta})\ dZ^n\}\cdot\bar{\omega}_m^n\\
&+\sum_{n=1}^{N}\sum_{m=1}^{M}\log p(m|\Theta)\cdot\bar{\omega}_m^n
\end{aligned}
\tag{B.1}
$$

B.2 Derivation of Eqn.(3.51), (3.52) and (3.53)

Eqn.(3.51) is straight-forward,

$$
E_m[Z^n(k)]=E[Z^n(k)|O^n,m,\bar{\Theta}]=\hat{Z}_{k|K_n,m}^n \tag{B.2}
$$

Eqn.(3.52) is derived as follows,

$$\begin{aligned} E_m[Z^n(k)Z^n(k)'] &= E_m[\{Z^n(k)-E_m[Z^n(k)]\}\{Z^n(k)-E_m[Z^n(k)]\}'] \\ &\quad + E_m[Z^n(k)]E_m[Z^n(k)]' \\ &= \Sigma^n_{k|K_n,m} + E_m[Z^n(k)]E_m[Z^n(k)]' \end{aligned} \tag{B.3}$$

For Eqn.(3.53), it is not so easy.

$$\begin{aligned} E_m[Z^n(k)Z^n(k-1)'] &= E_m[\{Z^n(k)-E_m[Z^n(k)]\}\{Z^n(k-1)-E_m[Z^n(k-1)]\}'] \\ &\quad + E_m[Z^n(k)]E_m[Z^n(k-1)]' \\ &= \Sigma_{k,k-1|K_n} + E_m[Z^n(k)]E_m[Z^n(k-1)]' \end{aligned} \tag{B.4}$$

where, $\Sigma_{k,k-1|K_n} = E_m[\{Z^n(k)-E_m[Z^n(k)]\}\{Z^n(k-1)-E_m[Z^n(k-1)]\}']$. From the smoothing equation (2.40),

$$Z^n(k-1)-E_m[Z^n(k-1)] = Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m}-A^n_{k-1,m}[\hat{Z}^n_{k|K_n,m}-\hat{Z}^n_{k|k-1,m}] \tag{B.5}$$

$Z^n(k)-E_m[Z^n(k)]$ can be re-written as

$$Z^n(k)-E_m[Z^n(k)] = Z^n(k)-\hat{Z}^n_{k|k-1,m}-(\hat{Z}^n_{k|K_n,m}-\hat{Z}^n_{k|k-1,m}) \tag{B.6}$$

$Z^n(k)-\hat{Z}^n_{k|k-1,m}$ depends only on the innovation sequence up to time $k-1$ and , on the other hand, $\hat{Z}^n_{k|K_n,m}-\hat{Z}^n_{k|k-1,m}$ depends only on the innovation sequence after time $k-1$. so they are uncorrelated. Then we have

$$\begin{aligned} \Sigma_{k,k-1|K_n} &= E_m[(Z^n(k)-\hat{Z}^n_{k|k-1,m})(Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m})'] \\ &\quad + E_m[(\hat{Z}^n_{k|K_n,m}-\hat{Z}^n_{k|k-1,m})(\hat{Z}^n_{k|K_n,m}-\hat{Z}^n_{k|k-1,m})']A^n_{k-1,m}{}' \end{aligned} \tag{B.7}$$

The first term in Eqn.(B.7) is equal to

$$\begin{aligned} & E_m[(Z^n(k)-\hat{Z}^n_{k|k-1,m})(Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m})'] \\ =\ & E_m[(\Phi_m Z^n(k-1)-\Phi_m\hat{Z}^n_{k-1|k-1,m}+W_n(k-1))(Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m})'] \\ =\ & \Phi_m E_m[(Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m})(Z^n(k-1)-\hat{Z}^n_{k-1|k-1,m})'] \\ =\ & \Phi_m \Sigma^n_{k-1|k-1,m} \end{aligned} \tag{B.8}$$

In the second term of Eqn.(B.7) $\hat{Z}^n_{k|K_n,m} - \hat{Z}^n_{k|k-1,m}$ is equal to

$$[Z(k) - \hat{Z}^n_{k|k-1,m}] - [Z(k) - \hat{Z}^n_{k|K_n,m}] \tag{B.9}$$

By smoothing Eqn.(2.41),

$$\begin{aligned} & Z(k) - \hat{Z}^n_{k|K_n,m} \\ = \ & Z(k) - \hat{Z}^n_{k|k,m} - A^n_{k,m}[\hat{Z}^n_{k+1|K_n} - \hat{Z}^n_{k+1|k,m}] \\ = \ & Z(k) - \hat{Z}^n_{k|k-1,m} - K(k)\tilde{O}(k) - A^n_{k,m}[\hat{Z}^n_{k+1|K_n} - \hat{Z}^n_{k+1|k,m}] \end{aligned} \tag{B.10}$$

Notice that $Z(k) - \hat{Z}^n_{k|k-1,m}$ depends on innovation up to time $k-1$ and remaining part on innovation from time k to the end. $Z(k) - \hat{Z}^n_{k|k-1,m}$ is uncorrelated with the remaining part. Therefore, the second term of Eqn.(B.7) becomes

$$\begin{aligned} & [\Sigma^n_{k|K_n,m} - \Sigma^n_{k|k-1}]A^{n\,'}_{k-1,m} \\ = \ & [\Sigma^n_{k|K_n,m} - \Sigma^n_{k|k-1}](\Sigma^n_{k|k-1})^{-1}\Phi_m\Sigma^n_{k-1|k-1} \\ = \ & \Sigma^n_{k|K_n,m}(\Sigma^n_{k|k-1})^{-1}\Phi_m\Sigma^n_{k-1|k-1} - \Phi_m\Sigma^n_{k-1|k-1} \end{aligned} \tag{B.11}$$

Plug Eqn.(B.8) and (B.11) into Eqn.(B.7), we have

$$\Sigma_{k,k-1|K_n} = \Sigma^n_{k|K_n,m}(\Sigma^n_{k|k-1})^{-1}\Phi_m\Sigma^n_{k-1|k-1} \tag{B.12}$$

Then,

$$E_m[Z^n(k)Z^n(k-1)'] = \Sigma^n_{k|K_n,m}(\Sigma^n_{k|k-1})^{-1}\Phi_m\Sigma^n_{k-1|k-1} + E_m[Z^n(k)]E_m[Z^n(k-1)]' \tag{B.13}$$

Appendix C

Derivations for Chapter 4

C.1 Derivation of Eqn.(4.24):

Using the assumption of independence between different tokens, we have

$$
\begin{aligned}
& Q(\Theta|\bar{\Theta}) \\
= & \sum_{\{S\}^N} \int \sum_{n=1}^{N} \{ \log p(Z_0|\Theta) + \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) \\
& \quad + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta) + \log P(y_k^n|X^n, \Theta)] + \log P(X^n|\Theta)\} \\
& \quad \cdot p(\{Z\}^N|\{O, S\}^N, \bar{\Theta}) \; d\{Z\}^N \cdot p(\{S\}^N|\{O\}^N, \bar{\Theta}) \qquad \text{(C.1)} \\
= & \sum_{\{S\}^N} \{\sum_{n=1}^{N} \int \{\log p(Z_0|\Theta) + \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) \\
& \quad + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta) + \log P(y_k^n|X^n, \Theta)] + \log P(X^n|\Theta)\} \\
& \quad \cdot p(Z^n|O^n, S^n, \bar{\Theta}) \; dZ^n \cdot\} \; p(\{S\}^N|\{O\}^N, \bar{\Theta}) \qquad \text{(C.2)} \\
= & \sum_{n=1}^{N} \sum_{\{S\}^N} \int \{\log p(Z_0|\Theta) + \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) \\
& \quad + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \cdot p(Z^n|O^n, S^n, \bar{\Theta})\} \; dZ^n \cdot p(\{S\}^n|O^n, \bar{\Theta}) \\
& + \sum_{n=1}^{N} \sum_{\{S\}^N} \sum_{k=1}^{K_n} \log P(y_k^n|X^n, \Theta) \; p(\{S\}^N|O^n, \bar{\Theta})
\end{aligned}
$$

$$+\sum_{n=1}^{N}\sum_{\{S\}^N} \log P(X^n|\Theta)\, p(\{S\}^N|O^n,\bar{\Theta}) \tag{C.3}$$

$$= Q_Z + Q_Y + Q_X \tag{C.4}$$

where

$$\begin{aligned} Q_Z = \sum_{n=1}^{N}\sum_{\{S\}^N} \int \{ & \log p(Z_0|\Theta) \\ & + \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \\ & \} \cdot p(Z^n|O^n, S^n, \bar{\Theta})\, dZ^n \cdot p(\{S\}^N|O^n,\bar{\Theta}) \end{aligned} \tag{C.5}$$

$$Q_Y = \sum_{n=1}^{N}\sum_{\{S\}^N}\sum_{k=1}^{K_n} \log P(y_k^n|X^n,\Theta)\, p(\{S\}^N|O^n,\bar{\Theta}) \tag{C.6}$$

$$Q_X = \sum_{n=1}^{N}\sum_{\{S\}^N} \log P(X^n|\Theta)\, p(\{S\}^N|O^n,\bar{\Theta}) \tag{C.7}$$

C.2 Derivation of Eqn.(4.25):

$$\begin{aligned} Q_Z = & \sum_{n=1}^{N}\sum_{\{S\}^N} \int \{\log p(Z_0|\Theta) \\ & \quad + \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta) + \log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \\ & \quad \} \cdot p(Z^n|O^n, S^n, \bar{\Theta})\, dZ^n \cdot p(\{S\}^N|O^n,\bar{\Theta}) \\ = & \sum_{n=1}^{N} \log p(Z_0|\Theta) \\ & + \sum_{n=1}^{N} \int \sum_{\{X\}^n}\sum_{\{Y\}^n}\sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta)] \cdot p(\{Y\}^n|\{X\}^n, O^n, \bar{\Theta}) \\ & \quad \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \cdot p(\{X\}^n|O^n,\bar{\Theta}) \\ & + \sum_{n=1}^{N} \int \sum_{\{X\}^n}\sum_{\{Y\}^n}\sum_{k=1}^{K_n} [\log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \end{aligned}$$

$$
\begin{aligned}
& \cdot p(\{Y\}^n|\{X\}^n, O^n, \bar{\Theta}) \cdot p(\{X\}^n|O^n, \bar{\Theta})\} \\
= & \sum_{n=1}^{N} \log p(Z_0|\Theta) \\
& + \sum_{n=1}^{N} \int \sum_{\{X\}^n} \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta)] \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \\
& \qquad \cdot p(\{X\}^n|O^n, \bar{\Theta}) \\
& + \sum_{n=1}^{N} \int \sum_{\{X\}^n} \sum_{Y^n} \sum_{k=1}^{K_n} [\log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \\
& \qquad \cdot p(Y^n|X^n, O^n, \bar{\Theta}) \cdot p(\{X\}^n|O^n, \bar{\Theta}) \\
= & \sum_{n=1}^{N} \log p(Z_0|\Theta) \\
& + \sum_{n=1}^{N} \int \sum_{X^n} \sum_{\{X\}^n/X^n} \sum_{k=1}^{K_n} [\log p(Z_k^n|Z_{k-1}^n, X^n, \Theta)] \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \\
& \qquad \cdot p(\{X\}^n|O^n, \bar{\Theta}) \\
& + \sum_{n=1}^{N} \int \sum_{X^n} \sum_{\{X\}^n/X^n} \sum_{y_k^n} \sum_{Y^n/y_k^n} \sum_{k=1}^{K_n} [\log p(O_k^n|Z_k^n, X^n, y_k^n, \Theta)] \\
& \qquad \cdot p(Z^n|O^n, X^n, Y^n \bar{\Theta})\, dZ^n \\
& \qquad \cdot p(Y^n|X^n, O^n, \bar{\Theta}) \cdot p(\{X\}^n|O^n, \bar{\Theta}) \qquad \text{(C.8)} \\
= & \sum_{n=1}^{N} \sum_{m=1}^{M} \int [\sum_{k=1}^{K_n} \log p(Z_k^n|Z_{k-1}^n, m, \Theta)]\, p(Z^n|O^n, m, \bar{\Theta})\, dZ^n \cdot \bar{\omega}_m^n \\
& + \sum_{n=1}^{N} \sum_{m=1}^{M} \int \{\sum_{k=1}^{K_n} \sum_{l=1}^{L} \log p(O_k^n|Z_k^n, m, l, \Theta)\, \bar{\xi}_{k,m,l}^n\, p(Z^n|O^n, m, l, \bar{\Theta})\}\, dZ^n \cdot \bar{\omega}_m^n \\
& + const. \qquad \text{(C.9)}
\end{aligned}
$$

From step (C.8) to (C.9), the properties in Eqn.(4.17) and (4.22) are used. On the other hand, $p(Z_0|\Theta)$ is fixed for all tokens, so it is treated as a constant.

Appendix D

Derivations for Chapter 5

D.1 Derivation of Eqn.(5.14) and (5.15)

By total probability theorem,

$$p(Z(k)|O_1^k) = \sum_{n\in\Psi_k} P_k(n)p(Z(k)|\psi_k(n), O_1^k) \tag{D.1}$$

Hence,

$$\hat{Z}_{k|k} = E[Z(k)|O_1^k)] = \sum_{n\in\Psi_k} P_k(n)E[Z(k)|\psi_k(n), O_1^k] = \sum_{n\in\Psi_k} P_k(n)\hat{Z}_{n,k|k} \tag{D.2}$$

and

$$\begin{aligned}
\Sigma_{k|k} &= E[(Z(k) - \hat{Z}_{k|k})(Z(k) - \hat{Z}_{k|k})'|O_1^k] \\
&= \sum_{n\in\Psi_k} P_k(n)E[(Z(k) - \hat{Z}_{k|k})(Z(k) - \hat{Z}_{k|k})'|\psi_k(n), O_1^k] \\
&= \sum_{n\in\Psi_k} P_k(n)E[((Z(k) - \hat{Z}_{n,k|k}) + (\hat{Z}_{n,k|k} - \hat{Z}_{k|k})) \\
&\qquad \cdot ((Z(k) - \hat{Z}_{n,k|k}) + (\hat{Z}_{n,k|k} - \hat{Z}_{k|k}))'|\psi_k(n), O_1^k] \\
&= \sum_{n\in\Psi_k} P_k(n)\{\Sigma_{n,k|k} + [\hat{Z}_{n,k|k} - \hat{Z}_{k|k}][\hat{Z}_{n,k|k} - \hat{Z}_{k|k}]'\}
\end{aligned} \tag{D.3}$$

D.2 Derivation of Eqn.(5.18)

By Bayes' rule,

$$\begin{aligned}
& P_k(n) \\
= \; & P(\psi_k(n)|O_1^k) \\
= \; & \frac{P(O(k), S_{n,k} = i_{n,k}, \psi_{k-1}(m)|O_1^{k-1})}{P(O(k)|O_1^{k-1})} \\
= \; & \frac{P(O(k), S_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1})\, P(\psi_{k-1}(m)|O_1^{k-1})}{P(O(k)|O_1^{k-1})} \\
= \; & \frac{P(O(k)|S_{n,k} = i_{n,k}, \psi_{k-1}(m), O_1^{k-1})\, P(S_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1})}{P(O(k)|O_1^{k-1})} P_m(k-1) \\
= \; & \frac{P(O(k)|\psi_k(n), O_1^{k-1})\, P(S_{n,k} = i_{n,k}|\psi_{k-1}(m), O_1^{k-1})}{P(O(k)|O_1^{k-1})} P_m(k-1) \qquad \text{(D.4)}
\end{aligned}$$

D.3 Derivation of Eqn.(5.28), (5.29) and (5.30)

By Bayes' rule,

$$\begin{aligned}
p(S_{k-1} = i|S_k = j, O_1^k) &= \frac{p(O(k), S_{k-1} = i|S_k = j, O_1^{k-1})}{p(O(k)|S_k = j, O_1^{k-1})} \\
&= \frac{p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})}{p(O(k), S_k = j|O_1^{k-1})} \\
&= \frac{p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})}{\sum_{i=1}^{M} p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})} \qquad \text{(D.5)}
\end{aligned}$$

$$\begin{aligned}
p(S_k = j|O_1^k) &= \sum_{i=1}^{M} p(S_{k-1} = i, S_k = j|O_1^k) \\
&= \frac{\sum_{i=1}^{M} p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})}{p(O(k)|O_1^{k-1})} \\
&= \frac{\sum_{i=1}^{M} p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})}{\sum_{j=1}^{M}\sum_{i=1}^{M} p(O(k), S_{k-1} = i, S_k = j|O_1^{k-1})} \qquad \text{(D.6)}
\end{aligned}$$

and

$$\begin{aligned}
& p(O(k), S_{k-1} = i, S_k = j | O_1^{k-1}) \\
= \; & p(O(k), S_k = j | S_{k-1} = i, O_1^{k-1})\, p(S_{k-1} = i | O_1^{k-1}) \\
= \; & p(O(k) | S_{k-1} = i, S_k = j, O_1^{k-1})\, p(S_k = j | S_{k-1} = i, O_1^{k-1})\, p(S_{k-1} = i | O_1^{k-1}) \\
= \; & p(O(k) | S_{k-1} = i, S_k = j, O_1^{k-1})\, p(S_k = j | S_{k-1} = i)\, p(S_{k-1} = i | O_1^{k-1}) \qquad \text{(D.7)}
\end{aligned}$$

D.4 Derivation of Eqn.(5.51)

By total probability theorem.

$$\begin{aligned}
& p(Z(k) | S_k = j, O_1^{k-1}) \\
= \; & \sum_{i=1}^{M} p(Z(k) | S_k = j, S_{k-1} = i, O_1^{k-1}) P(S_{k-1} = i | S_k = j, O_1^{k-1}) \\
= \; & \sum_{i=1}^{M} [\int p(Z(k) | Z(k-1), S_k = j, S_{k-1} = i, O_1^{k-1}) \\
& \quad \cdot p(Z(k-1) | S_k = j, S_{k-1} = i, O_1^{k-1} dZ(k-1)] P(S_{k-1} = i | S_k = j, O_1^{k-1}) \\
= \; & \int p(Z(k) | Z(k-1), S_k = j, O_1^{k-1}) \\
& \quad \cdot [\sum_{i=1}^{M} p(Z(k-1) | S_{k-1} = i, O_1^{k-1}) P(S_{k-1} = i | S_k = j, O_1^{k-1})] \cdot dZ(k-1)
\end{aligned}$$

(D.8)

Bibliography

[1] J. Z. Ma and L. Deng, "Optimization of dynamic regimes in a statistical hidden dynamic model for conversational speech recognition", *Eurospeech 1999* . Vol.3. pp1339-1342. 1999.

[2] J. Z. Ma and L. Deng, "A path-stack algorithm for optimizing dynamic regimes in a statistical hidden dynamic model of speech". *Computer. Speech and Language.* vol. 14. pp101-114, 2000.

[3] L. Deng and J. Z. Ma. "A statistical coarticulatory model for the hidden vocal-tract-resonance dynamics", *Eurospeech 1999* . vol.4, pp. 1499-1502. 1999.

[4] L. Deng and J. Z. Ma. "Spontaneous speech recognition using a statistical coarticulatory model for the vocal-tract-resonance dynamics", submitted to *The Journal of the Acoustical Society of America*, July, 1999.

[5] J. Z. Ma and L. Deng, " Bayesian decoding strategy for conversational speech recognition using a constrained nonlinear state-space model for vocal-tract-resonance dynamics", submitted to *IEEE Transactions on Speech and Audio Processing*. June 1999.

[6] R. Togneri. J. Z. Ma and L. Deng, "Statistical Estimation of Dynamic System Parameters by EKF: Comparison with Maximum Likelihood and Simulation

Results with State-Space Switching", submitted to "Signal Processing", Jan. 2000.

[7] F. Jelinek, "Continuous speech recognition by statistical methods", *IEEE Proceeding*, vol.64, pp.532-556, Apr. 1976.

[8] B.-H. Juang and L. R. Rabiner, "Mixture autoregressive hidden Markov models for speech signals", *IEEE Trans. Acoust. Speech Signal Processing*", vol. ASSP-33. pp.1404-1413, Dec. 1985.

[9] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications", *IEEE Proceeding*, vol.77, pp.257-285, Feb.1989.

[10] J. K. Baker, "The dragon system - An overview", *IEEE Trans. on Acoust. Speech ans signal proc.*, Vol. ASSP-23(1), pp. 24-29, 1975.

[11] R. M. Schwartz, Y. L. Chow, O. A. Kimball, S. Roucos, etc., "Context-dependent modeling for acoustic-phonetic recognition of continuous speech", *ICASSP* , pp.1205-1208, Mar. 1985.

[12] K.-F. Lee, "Automatic Speech Recognition - The Development of the SPHINX-system", Kluwer Academic Publishers, Boston, 1989.

[13] L. E. Baum and J. A. Egon, "An Inequality with Applications to Statistical Estimation for Probabilistic Functions of a Markov Process and to a Model for Ecology", *Bull. Amer. Meteorol. Soc.*, 73:360–363, 1967.

[14] L. E. Baum, T. Petrie, G. Soules and N. Weiss, "A maximization technique in the statistical analysis of probabilitic function of finite state Markov chains", *Ann. Math. Stat.*, vol. 44, pp164-171, 1970.

[15] L. E. Baum and G. R. Sell, " Growth Functions for Transformations of Manifolds". *Pac. J. Math.*, 27(2):211–227, 1968.

[16] L. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, 3, 1-8, 1972.

[17] R. A. Boyles, "on the convergence of the EM algorithm", *J.Roy Statist. Soc. B*, vol. 45, 1983. pp47-50

[18] L. Deng, "Articulatory Features and Associated Production Models in Statistical Speech Recognition," in K. Ponting (ed.) *Computational Models of Speech Pattern Processing — NATO Advanced Study Institute*, 1997.

[19] L. Deng and S. Shen, "Maximum likelihood in statistical estimation of dynamic systems: Decomposition algorithm and simulation results", *Signal Processing*, vol.57, pp. 65-79, 1997.

[20] L. Bahl, F. Jelinek , R. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983, pp. 179-190.

[21] P. Gopalakrishnan, L. Balh, and R. Mercer, "A tree search strategy for large vocabulary continuous speech recognition," *Proc. IEEE ICASSP*, 1995, pp. 572-575.

[22] R. Schwartz and S. Austin, "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses," *Proc. of the IEEE ICASSP*, 1991, pp. 701-704.

[23] S. Haykin, *Neural Networks — A Comprehensive Foundation*, Maxwell Macmillan, Toronto. 1994.

[24] L. Devroye, "Non-uniform random variate generation", 1986.

[25] G. D. Forney, "The Viterbi Algorithm" *Proceedings of the IEEE*, 61:268–278, 1973.

[26] J. Jazwinski, "Stochastic process and filtering theory", New York, Academic Press.

[27] R. Kent, S. Adams and G. Turner, "Models of speech production," in *Principles of Experimental Phonetics*, Ed. N. Lass, Mosby: London, pp. 3-45, 1995.

[28] K. F. Lee, "Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognitio", *IEEE Trans. on Acoust. Speech and Signal Proc.*, vol. ASSP-39(4), pp599-609, 1990.

[29] S. Levinson, "Structual methods in automatic speech recognition", *Proceeding of the IEEE*, Vol.73(11), pp.1625-1650, Nov. 1985.

[30] R. McGowan, "Recovering articulatory movement from formant frequency trajectories using task dynamics and a genetic algorithm: Preliminary model tests," *Speech Communication*, 14, pp. 19-48, 1994.

[31] P. Rubin et al, "CASY and extensions to the task-dynamic model." *Proc. 4th European Speech Production Workshop*, Autrans, France, pp. 125-128, 1996.

[32] E. Saltzman and K. Munhall, "A dynamical approach to gestural patterning in speech production", *Ecological psychology*, pp.333-382, 1989.

[33] L. Deng, "A dynamic, feature-based approach to the interface between phonology and phonetics for speech modeling and recognition." *Speech Communication*, Vol. 24, No. 4, pp. 299-323, 1998.

[34] L. Deng, "Integrated-multilingual speech recognition using universal phonological features in a functional speech production model," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, 1997, Vol. 2, pp. 1007-1010.

[35] S. Dusan and L. Deng, "Recovering vocal tract shapes from MFCC parameters," *Proceedings of the International Conference on Spoken Language Processing* Sydney, Australia, Nov.30-Dec.4, 1998, pp. 3087-3090.

[36] A. Dempster, N. Laird and D. Rubin, "Maximum likelihood from incomplete data via the *EM* algorithm", *J. Royal Statist. Soc.*, Vol. B-39, 1977, pp. 1-38.

[37] C. F. J. Wu, "on the convergence properties of EM algorithm", *Ann. Statist.*, vol. 11, 1983, pp95-103

[38] V. Zue, "The use of speech knowledge in automatic speech recognition," *Proc. of the IEEE*, Vol. 73, 1985, pp. 1602–1550.

[39] R. H. Shumway, "An approach to time series smoothing and forecasting using the EM algorithm", J. of time series analysis, Vol.3, No.4, 1982.

[40] R. H. Shumway and D. S. Stoffer, "Dynamic linear models with switching", *J. of American Statistical Association*, pp763-769, vol. 86, 1991.

[41] R. L. Streit and T. E. Luginbuhl, "Probabilistic Multi-Hypothesis Tracking", *Studies in probabilistic multi-hypothesis tracking and related topics*, vol. SES-98-01, pp5-50, 1998.

[42] C. J. Wellekens, "Explicit time correlation in hidden Markov models for automatic speech recognition", *Proc. int. conf. Acoust. Speech Signal Processing*, pp384-387, 1987.

[43] P. F. Brown, "The acoustic modeling problem in automatic speech recognition", *Ph.D. Thesis, computer science dept., Carnegie Mellon Univ.*, 1987.

[44] M. Ostendorf and S. Roukos, "A stochasitc segment model for phoneme-based continuous speech recognition", *IEEE Trans. Acoust., Speech Signal Processing*, vol.37, no.12, pp1857-1869, 1989.

[45] P. Kenny, M. Lennig, and P. Mermelstein, "A linear predictive HMM for vector-valued observations with applications to speech recognition", *IEEE trans. Acoust. Speech. Signal Processing*, vol. 38, no.2, pp220-225, 1990.

[46] P. Woodland, "Hidden Markov models using vector linear prediction and discriminative output distributions", in *Proc. of ICASSP*, vol.1, pp509-512, 1992.

[47] S. Takahashi, T. Matsuoka, etc., "Phoneme HMMs constrained by frame correlations", *Proc. of ICASSP*, vol.2, pp219-222, 1993.

[48] L. Deng , "A generalized hidden Markov model with state-conditioned trend functions of time for the speech signal," *Signal Processing*, Vol.27, 1992, pp. 65-78.

[49] O. Ghitza and M. Sondhi, "Hidden Markov models with templates as nonstationary states: an application to speech recognition", *Computer, Speech and Language*, pp101-119, 1993.

[50] M. Russell, "A segmental HMM for speech pattern matching", *Proc. Int. Conf. Acoust. , Speech Signal processing*, vol. II, pp499-502, 1993.

[51] L. Deng, "A stochastic model of speech incorporating hierarchical nonstationarity," *IEEE Trans. Speech and Audio Processing*, Vol. 1, 1993, 471-474.

[52] M. Gales and S. Young, "Segmental HMM's for speech recognition", *Proc. Euro. conf. Speech Commun. Tech.*, pp1579-1582, 1993.

[53] H. Gish and K. Ng, "A segmental speech model with applications to word spotting". *Proc. Int. Conf. Acoust., Speech Signal Processing*, vol.II, pp447-450, 1993.

[54] V. Digalakis, J. Rohlicek and M. Ostendorf, "ML estimation of a stochastic linear system with the *EM* algorithm and its application to speech recognition." *IEEE Trans. Speech and Audio Processing*, Vol. 1, 1993, pp. 431-442.

[55] L. Deng, M. Aksmanovic, D. Sun and J. Wu, "Speech recognition using hidden Markov models with polynomial regression functions as nonstationary states," *IEEE Trans. Speech Audio Proc.*, vol. 2, 1994, pp. 507-520.

[56] O. Kimball, "Segment modeling alternatives for continuous speech recognition". *Ph.D.. dissertation*, Boston Univ., MA. 1994.

[57] Y.Gong and J.-P. Haton, "Stochastic trajectory modeling for speech recognition". *Proc. ICASSP*, vol.1, pp57-60, 1994.

[58] L. Deng and C. Rathuinavalu, "A Markov model containing state-condtioned second-order nonstationarity: Application to speech recognition", *Computer Speech Language*, vol.9, no. 1, pp.63-86, 1995.

[59] W. J. Holmes and M. J. Russell, "Speech recognition using a linear dynamic segmental HMM", *Proc. Eurospeech*, pp1611-1614. 1995.

[60] M. Ostendorf. "From HMMs to segment models: A unified view of stochastic modeling for speech recognition", *IEEE Trans. Speech Audio Proc.*, vol. 4, 1996, pp. 360-378.

[61] L. Deng and M. Aksmanovic, "Speaker-independent phonetic classification using hidden Markov models with state-conditioned mixtures of trend functions," *IEEE Transactions on Speech and Audio Processing*, Vol. 5, No. 4, July 1997, pp. 319-324.

[62] J. Goldberger, D. Burshtein, H. Franco, "Segmental modeling using a continuous mixture of non parametric models", *IEEE trans. on Speech and Audio processing*, vol.7, no. 3, pp262-271, 1999.

[63] W. Holmes and M. Russell, "Probabilistic-trajectory segmental HMMs," *Computer Speech and Language*, vol. 13, 1999, pp. 3-37.

[64] R. Bakis, "Coarticulation modeling with continuous-state HMMs," *Proc. IEEE Workshop Automatic Speech Recognition*, Arden House, N.Y., 1991, pp. 20-21.

[65] C. Blackburn and S. Young , "Towards improved speech recognition using a speech production model," *Proc. Eurospeech*, vol. 2, 1995, pp. 1623-1626.

[66] L. Deng, G. Ramsay and D. Sun, "Production models as a structural basis for automatic speech recognition," *Speech Communication* (special issue on speech production modeling), Vol. 22, No. 2, August 1997, pp. 93-112.

[67] J. Bridle, L. Deng, J. Picone , H. Richards, J. Ma, T. Kamm, M. Schuster, S. Pike, and R. Reagan, "An Investigation of Segmental Hidden Dynamic Models of Speech Coarticulation for Automatic Speech Recognition," Final Report for the 1998 Workshop on Language Engineering, Center for Language and Speech Processing at Johns Hopkins University, 1998, pp. 1-61.

[68] H. Richards and J. Bridle, "The HDM: A segmental hidden dynamic model of coarticulation," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1999, pp.

[69] L. Deng, "Computational models for speech production," *Computational Models of Speech Pattern Processing* (NATO ASI Series), Springer, 1999, pp. 214-224.

[70] J. Picone, S. Pike, R. Reagan, T. Kamm, J. Bridle, L. Deng, Z. Ma, H. Richards, M. Schuster. "Initial evaluation of hidden dynamic models on conversational speech," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1999, pp 109-112.

[71] L. Rabiner, B.-H. Juang and C.-H. Lee "An overview of automatic speech recognition," in *Automatic Speech and Speaker Recognition - Advanced Topics*, Kluwer Academic Publishers, 1996, pp. 1-30.

[72] K. Stevens, Course notes, "Speech Synthesis with a Formant Synthesizer", MIT, July 26-30, 1993.

[73] K. Stevens "On the quantal nature of speech," *Journal of Phonetics*, Vol.17, 1989, pp. 3–45.

[74] C. M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1994.

[75] C. B. Chang and M. Athans "State estimation for discrete systems with switching parameters", *IEEE Trans. on Aerospace and Electronic System*, Vol. AES-14, No. 3, pp. 418-423, 1978.

[76] Anderson, B. and Moore, J., *Optimal Filtering*, Prentice-Hall, Inc., 1979.

[77] Y. Bar-Shalom and T. E. Fortmann , *Tracking and Data Association*, Academic Press, Inc., 1988.

[78] Y. Bar-Shalom and X. R. Li, *Estimation and Tracking*, Artech House, Boston, MA., 1993.

[79] H. Bourlard, H. Hermansky, and N. Morgan, "Towards increasing speech recognition error rates," *Speech Communications*, Vol. 18, 1996, pp. 205-231.

[80] J. Cohen, "The summers of our discontent," *Proc. Addendum ICSLP*, 1996, pp. S9-10.

[81] Z. Ghahramani and G. Hinton, "Switching state-space models," submitted for publication, 1999.

[82] J. D. Hamilton, "A new approach to the economic analysis of nonstationary time series and the business cycle", *Econometrica*, Vol. 57, No. 2, pp. 357-384, 1989.

[83] P. J. Harrison and C. F. Stevens, "Bayesian forecasting", *Journal of the Royal Statistical Society*, Ser. B, 38, pp. 205-247, 1976.

[84] K. Kitagawa, "Non-Gaussian state-space modeling of nonstationary time series", *Journal of the American Statistical Association*, Vol 82, pp. 1032-1041, 1987.

[85] C. J. Kim, "Dynamic linear models with Markov-switching", Journal of Econometrics, Vol. 60, pp1-22, 1994.

[86] F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, MA, 1998.

[87] R. Moore, "Speech pattern processing" in *Computational Models of Speech Pattern Processing (NATO ASI)*, Springer, pp. 1-9.

[88] M. Siu, R. Iyer, H. Gish, and C. Quillen, "Parametric trajectory mixtures for LVCSR," *Proceedings of ICSLP*, Sydney, Australia, 1998, pp. 3269-3272.

[89] A. H. Jazwinski, "Stochastic Processes and Filtering Theory," Academic Press, New York, 1970.

[90] J. M. Mendel, "Discrete techniques of parameter estimation: The equation error formulation", New York, Marcel Dekker, 1973.

[91] H. Tsurimi, "Survey of Bayesian and non-Bayesian testing of model stability in econometrics", in *Bayesian Analysis of Time Series and Dynamic Linear Models*, Ed. J.C. Spall, New York: Marcel Deker, pp. 75-100, 1988.

[92] N. Merhav and Y. Ephraim, "A Bayesian classification approach with application to speech recognition", *IEEE trans. signal processing*, vol. 39(10), pp2157-2166, 1991.

[93] J. M. Mendel, *Lessons in estimation Theory for Signal Processing, Communications and Control*, Prentice Hall, 1995.

[94] H. Tanizaki, *Nonlinear Filters*, Second Ed., Springer Verlag, 1996.

[95] K. Watanabe, *Adaptive Estimation and Control — Partitioning Approach*, Prentice-Hall Inc., 1991.