Combination and Generation of Parallel Feature Streams for Improved Speech Recognition

Xiang Li

Thesis Committee:
Richard M. Stern, Chair
Tsuhan Chen
Hynek Hermansky
Rita Singh

Submitted to the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy at

Carnegie Mellon University
Pittsburgh, PA, 15213
February, 2005

Abstract

The combination of information from parallel features that provide complementary information about the speech signal generally improves speech recognition accuracy. There are two issues associated with parallel feature combination: the specific method of combining the parallel features, and the nature of the parallel features themselves. These two issues jointly determine the performance of an information combination system.

While a great deal of work has already been expended in the area of information combination, the techniques developed in previous studies are not without their drawbacks and could be further improved. First, most prior activity has focused on the combination method itself, with relatively little attention paid to the issue of designing parallel features for optimal combination performance. Because the benefit of combination ultimately derives from the different information provided by the parallel features, a more judicious way of designing parallel features that specifically tailors them to suit the type of combination procedure used should improve the combined recognition accuracy. In addition, conventional information combination methods can be further refined to better utilize the information that is contained in the parallel features.

The work in this thesis addressed the two issues cited above. We describe two new information combination schemes, lattice combination, which operates on the outputs of the individual recognition systems, and weighted probability combination, which operates at an earlier stage where recognition probabilities are evaluated. We also describe a new way to combine directly the probabilities of untied HMM states. Both the lattice combination method and the two new probability combination methods are shown to be effective in providing consistent improvements over conventional combination methods for various speech recognition tasks.

The second part of this thesis concerns a new way of generating parallel linearly-derived features that directly optimizes their performance when combined. These features are designed in a way that maximizes the normalized acoustic likelihood of the observed speech in the combined system, which can be directly related to speech recognition accuracy. Our new parallel feature generation algorithm consistently outperform conventional parallel feature sets in various speech recognition tasks. In addition, we can achieve even further improvement when apply multiple techniques developed in this thesis simultaneously.

Acknowledgements

This thesis would not have been possible without the support and encouragement of many people. To the following people, I owe an enormous debt of gratitude.

My advisor, Professor Richard Stern, who give me the help, support, direction and advice. Rich gave me the guidance I needed to become a capable researcher and the freedom to pursue my own ideas. He has been a role model of professionalism and integrity, as well as a good friend.

Dr. Rita Singh, I thank you for the countless fruitful discussions about my research and all of your helpful advice as I put this thesis together.

Professor Hynek Hermansky and Professor Tsuhan Chen, who made many valuable comments and suggestions for my work.

The members of the CMU Robust Speech Group, past and present, from whom I learned a great deal and with whom I shared many laughs. Bhiksha, Mike, Jon, Evandro never grew tired of answering my endless stream of questions. Thanks to Ziad, Lingyun, Govind, Murali, Wooil, and Professor Nam Soo Kim for their help, discussion and friendship.

My friend in CMU and in Pittsburgh. My life in CMU and Pittsburgh has been made much more comfortable because there were here.

Yuheng, who has given me more love and encouragement during this process than anyone can possibly ask for. Her boundless support kept me going when things were difficult.

Finally, there is no way I would be where I am today without the immeasurable love, support, and encouragement of my parents and grandparents. I cannot thank them enough for all the opportunities they have given me in my life and for always supporting me. They are an incredible inspiration to me.

To my parents, grandparents

To Yuheng

Table of Contents

List of Figures

List of Tables

1. Introduction

Contemporary automatic speech recognition technology has been based on statistics and probability rather than expert rules for some time now. As with all statistical pattern classification systems, the performance of a speech recognizer depends critically on two factors: the parametric features that characterize the input and the statistical models that are used to characterize these features. Because the models in a speech recognizer are actually trained on particular values of the features that are presented to the system, the choice of features is actually the fundamental issue that implicitly determines the fate of a speech recognizer.

Despite the fact that many reasonably good types of features have been developed over the years (*e.g.* MFCC [9], PLP [23], PCA [33], LDA [11][29], HLDA [42][68], Tandem [14][26], etc.) and have been successful for various tasks and conditions, none of them work perfectly over all conditions. In general, these features just represent different subsets of information that is contained in the speech signal, and at different levels of accuracy. It stands to reason that the proper combination of information from diverse feature sets could results in better recognition accuracy than the results that would be obtained using a single feature set alone. The advantages of combining multiple feature sets include the possibility that these feature sets can provide different information about the speech signal, and that the “union” of this information through the process of combination may result in a more complete description about speech data for different recognition classes than each individual feature sets can. In fact, many studies have shown the effectiveness of the combination, as no matter how well a single system performs, the appropriate combination of a best performing single system with some other systems will always result in a combined performance that is at least no worse than the single best individual system (*e.g.* [6][12][15][25][44][45][69]). Combination procedures are becoming a standard component of many state-of-the-art recognition systems because of their ability to improve recognition accuracy compared to what is obtained from a single system of features, no matter how well optimized it may be.

The degree of success of combination methods depends on two aspects of the system: (1) the different types of information sources that are combined together and (2) the specific method of combination that is used to fusing the different information sources together. These different information sources could in principle be either completely different recognition systems or different sets of features that are used to

train and test a single system. We will focus on the combination of different feature sets representing multiple information sources in this thesis.

There are generally three different stages in the recognition process at which parallel information sources such as parallel feature sets can be combined together. The first approach, which we call *feature combination*, refers to the concatenation of parallel features into a new single feature stream as the input to the speech recognition system, frequently followed by a reduction in dimensionality of the new single stream of features, with recognition performed on the basis of the observed values of these concatenated new features (*e.g.* [1][57][63][71]). In the second approach, referred to as *probability combination*, the normal decoding process is carried out on the individual feature streams to generate probabilities for each recognition class (which could be the specific state of a Hidden Markov Model, or a phoneme, syllable, or even word). The probabilities characterizing a given recognition class are then combined across the parallel decoding processes, and the final combined hypothesis is generated by searching through a network of these combined probabilities (*e.g.* [1][6][12][18][21][25][34][47][56][61][66][67] [71][72]). In the third approach, *hypothesis fusion*, complete recognition hypotheses are first generated in parallel from each individual recognition system, which then are combined together to generate the final combined hypothesis (*e.g.* [3][15][45][69][78]). These three stages of information combination are also referred to as *early integration*, *middle integration* and *late integration* respectively.

As noted above, the performance of a combined speech recognition system using parallel features depends critically on both the parallel features and the combination method used. Ideally, each of the feature streams to be combined should provide different information regarding the speech signal. These parallel feature streams need not to be able to recognize each subclass of speech perfectly, but the union of them should be more informative than the individual feature streams in representing individual recognition classes and in discriminating among different recognition classes. Currently, the most common ways of generating parallel feature streams for combination are the methods of feature selection and feature generation via splitting (*e.g.* [6][21][22][25]). Feature selection is used to describe methods that either select some individual features that are known to achieve reasonably good performance or that select from a predefined set of features based on some measure (such as the conditional mutual information criterion suggested by Ellis and Bilmes [13]). Splitting refers to a method of generating parallel features by dividing either the recognition task into several sub-tasks and designing features accordingly (such as the heteroge-

neous feature method developed by Halberstadt for phonetic recognition of the TIMIT database [21][22]), or by dividing the spectrum into several sub-bands and generating features accordingly (*e.g.* [6][25]). Clearly other types of "division of labor" are possible as well.

Despite the success of these previous attempts in generating and combining parallel features, they all have some drawbacks and further improvements can be achieved in both the generation and combination of parallel features for several reasons. First, current combination algorithms can be refined, especially at the stages of hypothesis fusion and probability combination. For example, in hypothesis fusion, most of the existing methods are based on either a single best list or an *N*-best list of hypotheses, which represent only a small fraction of the information that is available from the decoder. This will cause much other valuable information from the decoder to be discarded in the combination process, which will inevitably degrade the performance of the combined system. In probability combination, many of the current methods ignore differences in the reliability with which different feature streams represent the various recognition classes. While some current probability combination methods do place differing emphases in the contributions from parallel features, the way in which the weighting factors is determined may not be directly related to the objective of the combination process. In addition, most synchronous probability combination algorithms (the category that is most widely used in probability combination) require the parallel systems to share the same model structure (*i.e.* the HMM state identity) in order to carry the combination at the probability level. This constraint degrades the performance of the combined systems, which consequently reduces the benefit to be obtained by probability combination.

Second, little work has been done toward the joint design or generation of parallel features to achieve the maximum benefit from combination. While some current approaches simply select and combine the single streams of features that individually provide best performance, there is no guarantee that the combination of those best-performing individual feature sets will also result in the best possible combined performance, given that the objectives used in designing single-stream recognition features may be different from the criterion that should be used in designing parallel features for combination. Other methods generate parallel features by splitting, but there is still no guarantee that parallel features generated in this fashion will be the best possibilities for feature combination applications, since these parallel features are designed in isolation, taking no account of how they could interact in combination applications.

1.1 What this thesis is about

The work described in this thesis consists of two components. The first component concerns new refinement in combination methods. At the hypothesis fusion level we propose a lattice combination algorithm that directly combine the lattices generated from individual features together. Compared with conventional hypothesis-fusion methods that are based on either a single-best or N-best list, our new algorithm utilizes lattices (rather than the hypotheses derived from them) as the inputs to the combination process. Since lattices contain much more valuable information for decoding than the single best hypothesis or N-best hypotheses, the combined performance will also increase correspondingly. Another new combination algorithm is proposed at the probability combination level. Our new probability combination algorithm assigns a weight to each individual feature set that reflects the reliability of that individual feature set in representing each recognition class; this measure of reliability is estimated directly from training data. In contrast to conventional probability combination algorithms which either treat parallel features equally or assign them combination weights based on some measure which may not be directly related to combined performance, our new approach takes into account the differing abilities of the various features to represent each individual recognition class directly in a manner that will be most beneficial for combined performance. We also refine probability combination in a way that removes the conventional constraint that the systems to be combined have the same model structure. This provides much more flexibility in the design of the individual systems, which can enable them to be designed in a way that is more likely to maximize their individual performance, providing in turn the likelihood of obtaining better combined performance.

The second component of our thesis concerns a new approach to the design of the parallel features themselves, in a way that can provide the greatest benefit when they are used in combination. In contrast to conventional feature generation algorithms, our approach develops parallel features jointly and directly based on the objective of maximizing the likelihood of the correct recognition hypothesis, which is closely related to maximizing recognition accuracy. We update parallel features in a way that maximizes their normalized acoustic likelihoods over the likelihoods (or scores) of the incorrect recognition classes, consequently reducing the word error rate (WER) of the combined system. Instead of generating parallel features in an arbitrary format, we develop our parallel features as linear transformations from some original speech representation (such as the short-time log magnitude spectrum of speech), which transforms the process of generating parallel features into one of finding optimal linear feature transformation matrices.

Our new approach of generating parallel features has several advantages over current feature generation methods. First, by incorporating statistical models of the recognizer into the feature generation process, we ensure that the generated features contain that information which is most beneficial for recognition without placing undue emphasis on less important information. Second, our process of generating parallel features involves interaction among those parallel features, and these features are generated using the same interactions as they perform in the combination process. Third, our algorithm is also capable of generating single-stream features by simply switching the optimization based on the normalized acoustic likelihood term from the combined system to an individual system with a single stream of feature. The processes of generating a single stream feature and generating parallel streams of features are unified under one framework in our approach.

In addition to presenting new algorithms, we also compare the various combination schemes considered and the effects of different parallel features on combined performance. We evaluate the different choices of combining information at different stages, and present some guidance in choosing the appropriate combination scheme under the specific requirement.

1.2 Dissertation outline

This thesis is organized as follows:

Chapter 2 provides an overview of automatic speech recognition based on the Hidden Markov Model (HMM). We discuss some of the details of HMMs that are relevant to our proposed work.

Chapter 3 reviews the concept of combining parallel features in speech recognition. We first describe the three stages in speech recognition process where information from parallel features can be combined, and along with representative combination algorithms at each stage. Then we discuss conventional approaches to the design of parallel features for combination in speech recognition.

Chapters 4 and 5 describe our efforts to refine combination algorithms. In Chapter 4 we present the proposed lattice combination algorithm. We first describe the nature of word lattices used in our work, and then describe how these lattices are combined to produce a final hypothesis. We present our experimental results and discuss lattice combination both in terms of its performance and in comparison to previous hypothesis fusion algorithms.

In Chapter 5 we present our proposed algorithms for improving probability combination performance. The first algorithm introduces optimal weights to the individual feature streams. The second algorithm relaxes the constraint that all systems to be combined share the same model structure. This is accomplished through the use of a new definition of what it is that is combined.

In the second part of the thesis, our new algorithm for generating parallel features based on the maximum normalized acoustic likelihood is presented in Chapter 6. We begin with a discussion of the generation of parallel features for optimal probability combination performance, and then simplify the algorithm to the case of generating single-stream feature to provide the best recognition performance of an individual system. We then discuss how this approach can be extended to optimize the generation of parallel features to be used across multiple systems.

Finally, we compare and analyze the advantage and drawbacks of various combination schemes, including our proposed new combination algorithms and our new feature generating algorithms in Chapter 7. We also summarize the major results and contributions of this thesis and highlight some promising directions for future work.

2. Review of Automatic Speech Recognition Technologies

2.1 Introduction

This chapter presents some background information that is relevant to the thesis. As mentioned in the introductory chapter, the goal of our proposed work in both combination of feature streams and in parallel feature generation process is to improve speech recognition accuracy, so a basic understanding of how a speech recognizer operates is important for understanding this thesis. In this chapter, we provide a general overview of the whole speech recognition process from the point where a waveform is input to the recognizer to the point where the final recognition hypothesis is generated. Specifically, we describe the processes of developing a log-spectral representation [9][28] from the wavefile, transforming the log-spectral representation into feature vectors that are used by the speech recognizer (*e.g.* MFCC [9]), and building models and performing recognition based on the feature vectors of the incoming speech. we limit our discussion to systems that are based on Hidden Markov Models (HMM). We discuss work that has been done in using multiple information sources to improve speech recognition accuracy, closing this chapter with an analysis on recognition results from parallel information sources, which suggests potential improvements in recognition accuracy that may be obtained by combining multiple feature streams.

2.2 Automatic speech recognition based on hidden Markov models

Speech recognition systems are a special case of pattern classification systems [65], where sounds or subsequences of sounds (such as phonemes, syllables, and words) are modeled as distinct recognition classes. The goal of speech recognition is to estimate the sequence of these sound classes that make up the observed speech signal. As with all pattern classification systems, speech recognition systems include two stages. The first stage is called feature extraction and refers to the extraction of relevant parameters from a physical measurement (such as the sound pressure of a speech waveform), and the second stage is the process of making a decision based on the observed values of the features. Figure 2.1 is a diagram that summarizes these two processing stages.

2.2.1 Feature extraction

Speech recognition systems do not base their decisions on the speech signal directly, but rather on a set

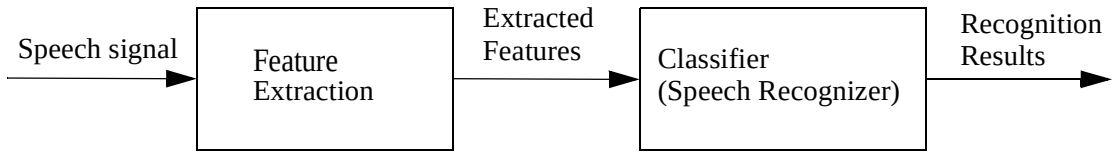

Figure 2.1. Block diagram of two-stage processing in speech recognition. The first stage is feature extraction, and the second stage is classification.

of feature vectors derived from the speech signal. As with most current speech recognition systems, the speech signal is first parameterized in this thesis into a *log-spectral representation*. We examine the derivation of the log-spectral representation in more detail since feature generation is an important component of the work in this thesis.

The original continuous speech waveform is first discretized via sampling and then segmented into a sequence of short overlapping sequences which are called frames. The frame is the basic unit that carries the information of the speech signal, and all processing described in this thesis is actually applied on a frame-by-frame basis. The duration of a frame is normally about 20 ms because the speech signal is quasi-stationary over this time interval.

Each frame of the speech signal is then windowed and transformed into the frequency domain using the short-time Fourier transform (STFT) [55]. The square of the magnitude is computed for each STFT component, and these coefficients are then passed through a bank of triangularly shaped weighting coefficients called Mel filters. This name reflects the Mel scale, which is one of several nonlinear scales that have been constructed to reflect the resolution of peripheral human auditory frequency analysis. The Mel scale is linear at low frequencies and logarithmic at frequencies above 1000 Hz. There is a 50% overlap between adjacent triangular “filters.” We take the log of the energy of each Mel-filter output. The set of these logs is the log-spectral representation of the speech signal used in subsequent processing. This process is illustrated in Figure 2.2.

In practice we usually apply a transformation that reduces the dimensionality of the log-spectral representation and build the model based on the transformed feature vector. This eliminates potentially-redundant information in the log spectra and also eliminates the confounding contribution of the pitch pulses in the representation. Some of the most commonly used transformations are the discrete cosine transform (DCT) as is used in MFCC features [9], a matrix derived using principal component analysis (PCA)

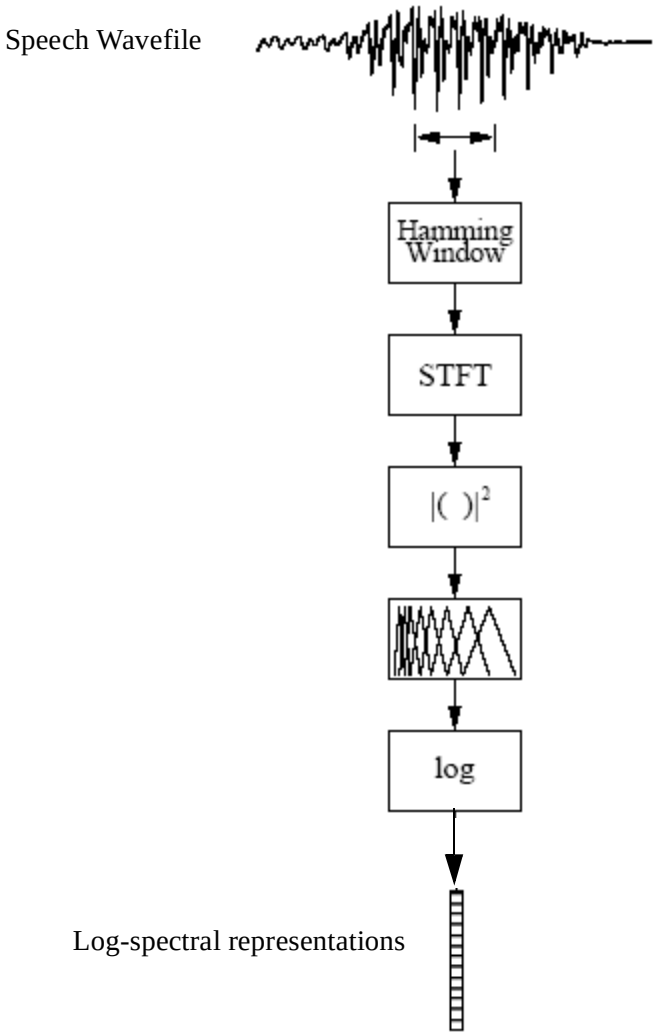

Figure 2.2. The process of extracting log-spectral features from a wavefile

[11][33] and a matrix derived using linear discriminant analysis (LDA) [11][29]. The feature-generation part of this thesis also discusses additional ways to generate these transformation functions.

2.2.2 Hidden Markov model based speech recognition

The representation of speech that is currently the most popular in automatic speech recognition is the Hidden Markov Model (HMM). In HMMs, the speech production mechanism is treated as a stochastic process which generates the observed speech signals. The speech production mechanism is assumed to operate in a fashion that can be characterized by a series of state transitions. These state transitions are assumed to be modelled by a Markov process in that the probability of moving to a given next state depends only on the identity of the current state. These state transitions are “hidden” in the sense that they are not directly

observable. As the underlying process transitions from an arbitrary state *i*, it is assumed to produce a vector of observations that are probabilistic and depend only the state that is being left. These are the vectors that are used by the speech recognition system in making classification decisions. In other words, the HMM is a doubly-stochastic process, generating first the unseen state transitions, and subsequently the observation vectors with probabilities that are conditioned on the identities of the sequence of states.

Most current speech recognition systems use a Gaussian mixture distribution for the observation probabilities because the Gaussian mixture can approximate many different distributions and because the parameters of the Gaussian mixtures can be estimated relatively easily. In addition, the speech feature vectors are closely approximated by Gaussian distributions. If we assume that $b_w(i, x)$ is the output probability distribution of the i^{th} state of recognition class *w* (which could be a word, syllable, or phoneme), then we can represent $b_w(i, x)$ as

$$b_w(i, x) = \sum_k \alpha_{i,k}^w N(x, \mu_{i,k}^w, \Sigma_{i,k}^w) \tag{2.1}$$

where $N(x, \mu_{i,k}^w, \Sigma_{i,k}^w)$ and $\alpha_{i,k}^w$ are the individual Gaussian distributions and mixture coefficients of the k^{th} Gaussian mixture component. The parameters $\mu_{i,k}^w$ and $\Sigma_{i,k}^w$ represent the mean and covariance of Gaussian component *k*. $\Sigma_{i,k}^w$ is usually assumed to be a diagonal matrix for computational simplicity.

The transitions of the underlying states of the HMM are modeled by the transition probability matrix A_w. Each element of this matrix $a_w(i, j)$ represents the probability of transiting from state *i* at time instance *t* to state *j* at time instance *t*+1. Clearly, all elements in the same row should sum up to 1:

$$\sum_j a_w(i, j) = 1 \tag{2.2}$$

If we use $B_w = \{\alpha_{i,k}^w, \mu_{i,k}^w, \Sigma_{i,k}^w\}$ and $A_w = [a_w(i, j)]$ to represent the state output probability distribution and transition probabilities, respectively, then the parameters of an HMM are completely defined

by $\lambda_w = \{A_w, B_w\}$. The model estimation process in HMM training is simply that of estimating λ_w for each recognition class *w*. Figure 2.3 is a simple example of a 3-state HMM. S4 in the figure is the exiting state, which can be treated as the first state (i.e. state S1) of another HMM. Once the transition enters the exiting state, it actually starts another transition to a new HMM.

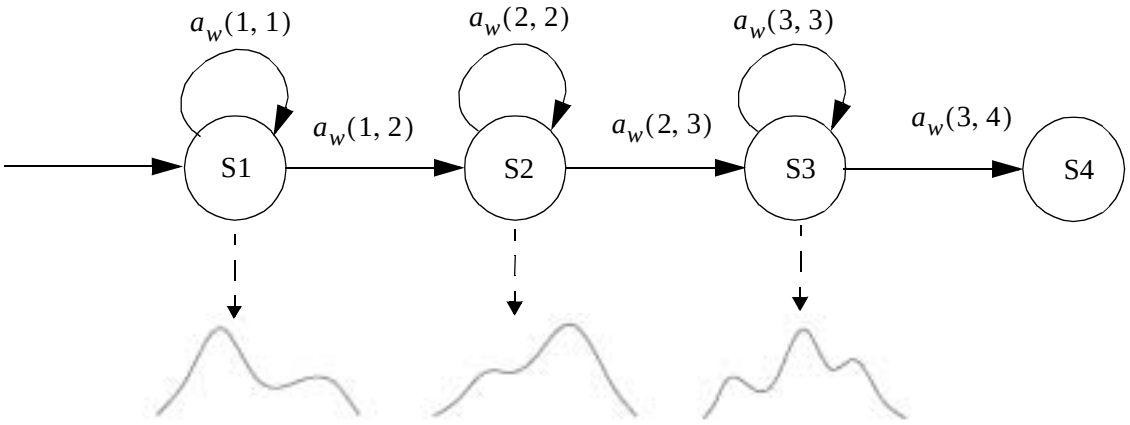

Figure 2.3. An example of an HMM. S4 is the exiting state, once the transition enters into it, it will start the new transition of another HMM.

In HMM modeling, the observation probabilities at different frames are treated as independent from each other. The likelihood of a feature vector sequence $X = \{x_1, x_2, \ldots, x_N\}$ given the HMM state sequence $S = \{s_1, s_2, \ldots, s_N\}$ and model parameters λ_w can be written as:

$$P(X|S, \lambda_w) = \prod_i P(x_i|s_i, \lambda_w) \tag{2.3}$$

The complete likelihood of a feature vector sequence given the model parameters is the summation of likelihoods of all the possible state sequence *S*:

$$P(X|\lambda_w) = \sum_S P(X, S|\lambda_w) = \sum_S \left[\prod_t P(x_t|s_t, \lambda_w)\right] P(S|\lambda_w) \tag{2.4}$$

where $P(x_t|s_t, \lambda_w)$ is the state observation probability as in Equation (2.1), $P(S|\lambda_w)$ is the state transition probability generated from the transition probability matrix A_w in Equation (2.2):

$$P(S|\lambda_w) = \prod_t a_s(s_t, s_{t+1}) p(s_1) \tag{2.5}$$

2.2.3 Viterbi alignment

Because the state sequence is hidden from the observations in HMMs, finding the specific state sequence which generates the highest likelihood of the feature vector sequence is one of the fundamental problems of using HMMs for speech recognition. Since in speech recognition we have only target information (*i.e.* the identity of the true recognition class) available at the sentence or word level, the state sequence can be treated as the target at the state level, which is frequently necessary for processing using HMMs.

Although in theory one could try all possible state sequences and pick the one that has the highest likelihood, an efficient algorithm such as *Viterbi alignment* [75] is typically used in practice to align the training transcript to the incoming speech signal to find the most likely state sequences. The process of Viterbi alignment can be described as follows. Let $X = \{x_1, x_2, \ldots, x_N\}$ be a sequence of feature vectors derived from the speech signal containing a particular word *W* modelled by the HMM parameters λ. Our goal is to find a state sequence $\hat{S} = \{\hat{s_1}, \hat{s_2}, \ldots, \hat{s_N}\}$ that maximizes the likelihood of the feature observations:

$$\hat{s_{t+1}} = Argmax\{P(x_{t+1}, s_{t+1}|\hat{s_t}, \lambda)\} = Argmax\{P(x_{t+1}|s_{t+1}, s_t, \lambda) P(s_{t+1}|s_t, \lambda)\} \tag{2.6}$$

The likelihood of this state sequence will be:

$$P(X, \hat{S}|\lambda) = max\{P(X, S|\lambda)\} = max\left\{\prod_t P(x_{t+1}, s_{t+1}|\hat{s_t}, \lambda)\right\} \tag{2.7}$$

Comparing Equation (2.7) with Equation (2.4), it is clear that the state sequence $\hat{S} = \{\hat{s_1}, \hat{s_2}, \ldots, \hat{s_N}\}$ is the candidate that has the greatest likelihood for the observed data out of many possible state sequences for the word *W*. This sequence will be treated as the target of the HMM state class for the future processing. We will return to the alignment issue in Chapter 6 in a discussion of the generation of parallel features.

2.2.4 Decoding and pruning the search space to generate the final hypothesis

In principle, the goal of the speech recognition process is to find the word $\hat{W}$ that has the highest posterior probability given the acoustic feature observations *X* as in Equation (2.8)

$$\hat{W} = Argmax\{P(W|X, \lambda_W)\} = Argmax\{P(X|W, \lambda_W)P(W)\} \quad (2.8)$$

where $P(X|W, \lambda_W)$ is the acoustic likelihood as in Equation (2.4), and $P(W)$ is the language model that describes the prior probability of the word *W*.

Up until now we have used the symbol *W* to denote a single isolated word. More generally we use the notation $W = \{w_1, w_2, \ldots, w_N\}$ to describe a sequence of words in a phrase or sentence. If individual words are modeled by unique HMMs in the manner described above, then HMMs corresponding to sequences of words can easily be made by concatenating the HMMs of the constituent words. The acoustic likelihood term $P(X|W, \lambda_W)$ can still be computed from the concatenated HMMs, and the language model $P(W)$ now becomes the probability of a word sequence.

In practical continuous speech recognition systems, however, processing as in Equation (2.8) becomes computationally demanding to the point of being impractical because Equation (2.8) would have to evaluated for every possible word sequence in the language. This issue is resolved typically by dividing the recognition process into two tasks. The first task, called “pruning” is to remove from consideration those word sequences that are less likely and to find a compact representation of the more likely word sequences given the observed speech feature vector. One of the most commonly used pruning algorithms is *“beam”* pruning ([46][76]). Pruning results in a compact search space which contains only those word sequences that have a higher probability of being the true word sequence. The *word lattice* is a convenient representation of the pruned search space, and it will be described in more detail in the Chapter 4. At the same time we do the pruning, we also simultaneously search for the specific word sequence that has the greatest posterior probability. This sequence will become the recognition output. This is referred to as the *“search”* processing. Some representative search algorithms include A* search [43] [76] and Viterbi search [75].

2.3 Effect of using different information sources on recognition accuracy

As with all pattern classification systems, the performance of a speech recognition system depends critically on the feature set that is used. Many different features have been proposed in speech recognition that have (unsurprisingly) provided different levels of recognition accuracy. Because of differences in the way features are generated, the recognition results provided by these different features could contain complementary information. If this complementary information can be combined together appropriately, better recognition accuracy should be expected. Generally speaking, research in feature combination for speech recognition can be divided into two sub-categories: approaches that combines acoustical features with visual features, and approaches that combine different types of acoustic features. We now discuss these two broad categories in a little bit more detail.

2.3.1 Recognition using acoustical and visual features

Audio visual speech recognition (AVSR) is currently a topic receiving a great deal of interest, using parallel information sources for speech recognition. In addition to extracting conventional acoustical information contained in the waveform of a speech signal, as is done by conventional speech recognition systems, AVSR also utilizes visual information related to movements of the lips and mouth in the speech production process. In fact, experimental results have shown that humans also rely on information like the shape of the mouth and lips in identifying sounds. For example, in the McGurk effect [48] it is demonstrated that most humans have a “fused response” in which they perceive a “DA” while they are actually hearing “BA” but seeing the lip movement of “GA”.

In noisy environments, which are a major impediment to the large scale commercial deployment of speech recognition systems, acoustical and visual feature streams are affected by environmental noise in strikingly different fashion. The visual feature stream remains basically unaffected by environmental noise, regardless of how low the signal-to-noise ratio (SNR) becomes. While the acoustical feature stream generally outperforms the visual feature stream at higher SNRs, the performance of acoustical features deteriorates rapidly as SNR decreases. When the SNR is extremely low (*e.g.* 5 dB or 0 dB), the acoustical features are so degraded by noise that the visual feature streams clearly outperform them. Combining information from visual and acoustical feature streams appropriately should result in better performance, especially in noisy environments. This is a major motivation for research in AVSR.

2.3.2 Different recognition performance using different acoustic features

We already know that acoustical and visual features perform differently in many conditions, and there is a potential benefit of combining them together. In this section we will argue that a similar situation exists in combining different types of acoustical features. While the fact that most current acoustical features are all derived from various types of log-spectral representations (*e.g.* MFCC, PLP, PCA and LDA features, as described above), they still perform differently from each other, largely because of different types of post-processing on the log-spectral representation, so the possibility of achieving benefit from combination still exists for these parallel acoustical streams.

As an example, in a series of pilot experiments we measured the recognition accuracy obtained using MFCC features [9] and PLP features [23] on the DARPA Resource Management (RM) task [64]. We compared the word error rate (WER) obtained using each feature set individually, and we also tested their performance when combined using the method of probability combination which will be described in detail later in this thesis. The individual WERs obtained using MFCC and PLP features were 10.29% and 11.49%, respectively, while the WER of the combined features could be reduced to 7.72%, a relative decrease of almost 30% compared to the single best performing set of MFCC features. Although this is just a simple example, it clearly shows the potential benefit that could be obtained from combining parallel information sources, even though these parallel information sources contain a large portion of similar information.

2.4 HMM implementations and speech corpora used

2.4.1The CMU SPHINX-III continuous speech recognition system

The CMU SPHINX-III continuous speech recognition system [60] has been used for all experiments in this thesis. Like most large vocabulary continuous speech recognition (LVCSR) systems, SPHINX-III is a phoneme-based system. Words are broken down into their constituent phonemes and each phoneme is modeled by an HMM. The HMMs for words are built by concatenating the corresponding phoneme HMMs, in the same manner as word-sequence HMMs are constructed from individual word HMMs. We use triphone context-dependent acoustical models, in which HMM states are modeled in terms of their left and right context. To reduce the total number of model parameters needed by the HMMs for modeling all

potential phoneme and context variations, the observation model parameters (*i.e.* the means, variances and mixture weights) of Gaussian distributions in the state observation probabilities can be shared by the states of different phonemes. States which share parameters in this manner are called “tied states” or “senones”. More information on this procedure can be found in [10], [30], and [31].

2.4.2 Speech corpora tested in the thesis

In order to evaluate the performance of the various algorithms considered, we need to evaluate different systems using a common database or databases. In this thesis, the following speech corpora are used: the DARPA Resource Management (RM) database, the Wall Street Journal database (WSJ0), the DARPA Speech in Noisy Environment databases I and II (SPINE), and the Telefónica Investigación y Desarrollo (TID) database. We will briefly describe the characters of each database.

The RM database is a collection of recordings of spoken sentences pertaining to a naval resource management. Subjects read the sentence from written prompts in low background noise. The speech data were recorded and down-sampled to 16 kHz and segmented into files corresponding to individual sentences. The normal duration of these sentences is 3 to 5 seconds. The training set used in our RM experiment contains 2800 sentences uttered by number of different speakers of both genders. The testing set contains 1600 utterances from different speakers. The RM database contains about 1100 words. The HMMs used in RM testing were 3-state continuous HMMs with triphone modeling. States of different context-dependent phones were tied together to produce a total number of 2000 tied states (or senones). The number of Gaussian mixtures used per HMM state (or senone) varies from 1 to 8 Gaussians.

The WSJ0 database is a much larger corpus than the RM database. It contains about 16,000 words in its dictionary, and has 7024 sentences in the training set, and 600 sentences in the testing set. The sentences both in the training set and test set are read from prepared text pertaining to North American business news by a number of different speakers of both genders. With its relatively large vocabulary size, the WSJ0 database is more challenging for the speech recognizer than the RM task. The HMM models used in the WSJ0 database are also 3-state HMMs using context dependent triphone modeling, with different HMM states tied together to share the same set of observation probabilities. The total number of tied states was 4000, and the number of Gaussians per state varied from 1 to 8.

The SPINE databases contain telephone-bandwidth speech data corrupted by a wide variety of noises at

various SNRs. In the SPINE 1 set, the training data consist of about 720 minutes of unsegmented audio, and the evaluation set has about 600 minutes of audio. The actual testing set we used in our experiment contains 2000 sentences. The SPINE 2 database has about 7 hours of audio data in the training set, 3 hours in the development set, and 7 hours in the evaluation set. We selected 2900 sentences from the testing corpus as our testing set. The models used in SPINE 1 and 2 were all continuous-context dependent triphones. States were tied together for a total number of 2600 senones in the SPINE 1 set and 1200 senones in the SPINE 2 set. We used 8 Gaussians per mixture in the SPINE 1and SPINE 2 sets.

The TID database is a 59-word Spanish-language telephone bandwidth speech corpus collected by Telefónica Investigación y Desarrollo in Spain. The training and testing sets that are used in our experiments contain 3458 and 700 sentences respectively, and they are uttered spontaneously. The average sentence is about 4.2 seconds long and contains 4.5 words on average. We used semi-continuous models in the TID database with 3 states per HMM, for a total of 400 tied states.

2.5 Summary

In this chapter, the basic operation of an HMM-based speech recognition system has been discussed. The feature extraction process, by which an incoming speech waveform is converted into the log-spectral representation, was described in detail. We then discussed how HMMs are used to model the distributions of speech feature vectors, and how such models are used to obtain an estimate of the words sequence given the speech feature vector. We then described Viterbi alignment which is used to find the most likely state sequence given the transcript, and also talked briefly about pruning and search in the search space to obtain the resulting recognition hypothesis.

In addition to providing some background in speech recognition, we also talked about the effects of using different features on speech recognition results. We described the general observation of using visual information to help (or confuse) humans in audio perception, and we also compared results obtained using parallel acoustic features with their performance in combination, showing in the latter case the potential benefit that may be derived from combination of parallel features. Finally, we described the specific recognition system used in our experiments, along with the speech databases that were used.

3. Combining Parallel Feature Streams to Improve Speech Recognition Accuracy

3.1 Introduction

As described in the previous chapter, different features that are derived from acoustic or visual representation of speech signal generally contain different types of information about the speech signal. They produce different recognition results, and appropriate combination of these features can improve recognition accuracy. As noted above, parallel feature combination refers to both the specific way of fusing the information contained in parallel features, and to the design of the parallel features themselves. Parallel feature combination and generation methods have received increased attention from researchers in areas such as speech recognition, statistics, and machine learning, and many effective algorithms have been proposed to utilize parallel information for better classification accuracy. In this chapter we review and discuss some of the successful methods in both parallel feature combination and parallel feature generation, particularly in the context of large vocabulary continuous speech recognition (LVCSR) systems that use Hidden Markov Models. We begin with a discussion of the specific combination methods to be used, and continue with a discussion about the selection and generation of individual feature streams to be used in combination. We conclude with an analysis of the advantages and drawbacks of these conventional methods.

3.2 Combination of information from parallel feature streams

Combination of information from parallel feature streams refers to the joint processing of information embedded in parallel features for better recognition accuracy. Depending on the stage in the recognition process at which that the information is merged, the combination is called hypothesis fusion, probability combination, or feature combination.

3.2.1 Hypothesis fusion

As illustrated in Figure 3.1, hypothesis fusion refers to those approaches that provide combination of the recognizer hypotheses arising from different feature streams after the search procedure is completed (*e.g.* [3][15][45][69][74][78]). The hypotheses to be combined from different feature streams can be represented as a single best hypothesis, an *N*-best list of hypotheses, or even a lattice of hypotheses.

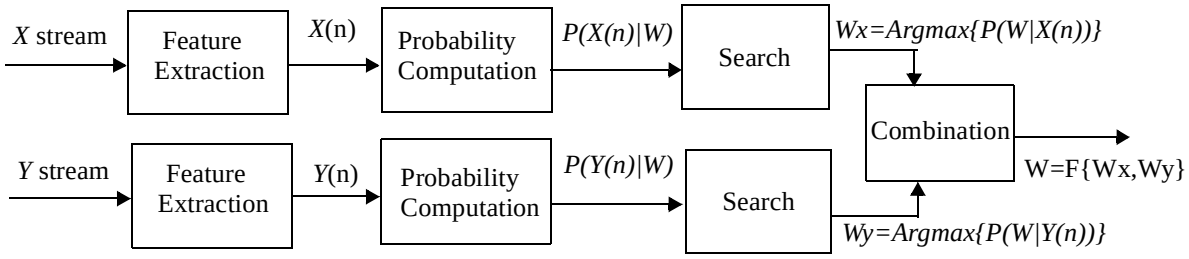

Figure 3.1. Block diagram of hypothesis fusion, where information from parallel feature streams *X* and *Y* is combined at the hypothesis stage after the final search in each system.

The most widely-used hypothesis fusion method is the *Recognizer Output Voting Error Reduction* (ROVER) method [15] developed by the speech group at the National Institute of Science and Technology (NIST). In ROVER, single hypotheses from different systems are first aligned to each other to form a Word Transition Network (WTN) as in Figure 3.2, whose basic unit is a segment called a *"correspondence set* (CS)*"*. Each CS aligns one word or silence region from individual feature streams. The final hypothesis that is generated is a concatenation of single words picked from each CS. A weighted voting scheme is used in each CS to pick the word with the highest score. The overall score of each word *W* from the i^{th} correspondence set is determined by both the frequency of occurrence of that word and the confidence score associated as weighted according to the parameter α as in Equation (3.1):

$$Score(W, i) = \alpha N_{W,i} + (1 - \alpha) C_{W,i} \tag{3.1}$$

where $N_{W,i}$ and $C_{W,i}$ represent the occurrence frequency and confidence score of word *W* in CS *i* respectively, and α is the weighting parameter that controls the relative contribution of these two quantities. The parameter α is trained from *a priori* data in the training set using a grid-search algorithm.

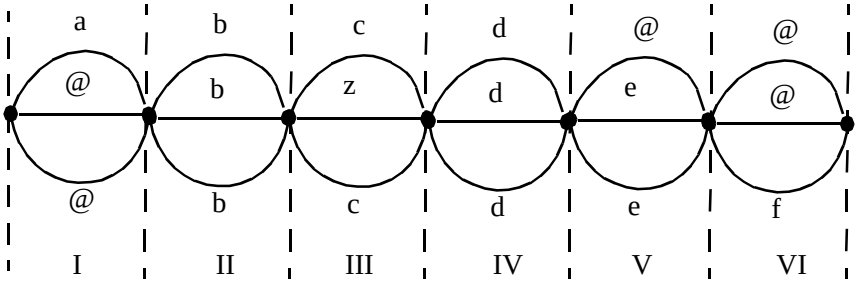

Figure 3.2. A Word transition network used in ROVER. Each letter represent a word and the symbol @ represent a silence region. Each Roman numeral represents a segment.

The ROVER method can be effective in improving recognition accuracy, especially when there is a large number of parallel feature streams to be combined. In addition, ROVER also has high flexibility in combining the outputs of systems with totally different structures since information from the parallel features or parallel systems is fused by merging their word sequences together. On the other hand, the ROVER utilizes only the information of the word identity at the time of combination, discarding other valuable information that could be relevant, and the benefit of parallel features or parallel systems may not be fully utilized.

The *Hypothesis Combination* [69] method developed at CMU is another effective method that combines single hypotheses. In this method, the combination of the hypotheses is based not only on the identities of the words to be combined but on their associated acoustic scores and durations as well. Each hypothesis begins with a starting node and ends with an ending node. During combination, the algorithm first builds a network by merging all the starting nodes and ending nodes from the parallel systems or features as in Figure 3.3

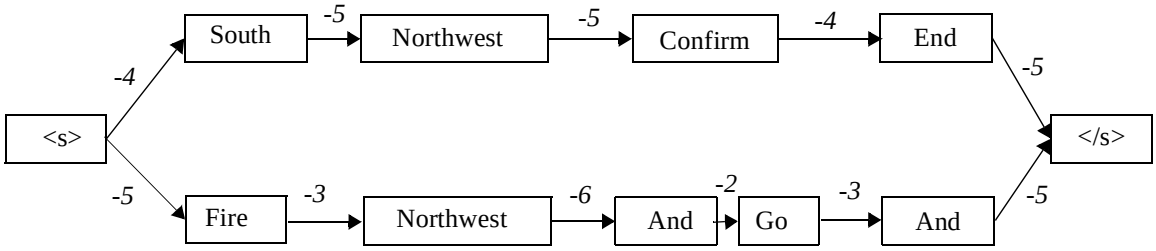

Figure 3.3. Merging hypotheses from parallel systems into a merged hypothesis network.

Once the merge is accomplished, new transitions between words from the different feature sets can be added if the words are time aligned. In addition, identical words from different hypotheses with similar start times and durations are merged into a single word with an updated score as in Figure 3.4.

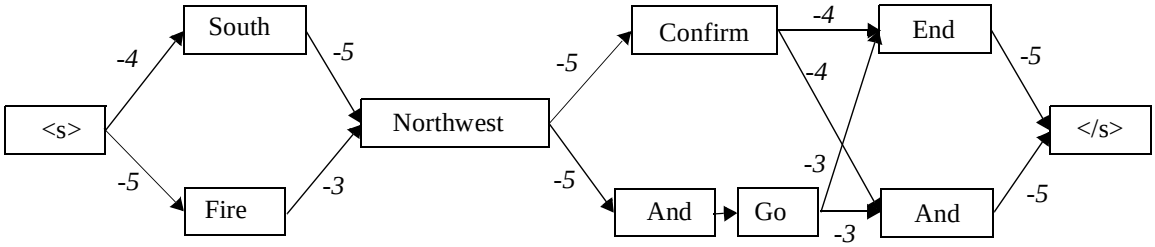

Figure 3.4. Building new transitions between words from different systems, and merging identical words from different hypotheses with similar time durations.

The final hypothesis is generated via a standard search procedure (*e.g.* A* search [43][76] or Viterbi

[75] search) through this mixed hypothesis network. Compared with ROVER, hypothesis combination utilizes the additional information of acoustic scores and durations of the word hypotheses from each individual feature set, which produces better recognition accuracy if this information can be utilized properly [69]. On the other hand, because hypothesis combination depends critically on the time alignment of the words in merging information and generating the combined result, and because the same words are frequently hypothesized with different time alignments by different features or systems, the advantage of hypothesis combination over ROVER may diminish as the number of features or systems to be combined become very large.

In addition to methods like ROVER and hypothesis combination that have as inputs the single best hypothesis from each feature stream, some researchers have developed methods that combine information from *N*-best lists of the most likely hypotheses emerging from each feature stream. For example, in the *discriminative model combination* (DMC) method developed by Beyerlien [3], an *N*-best list is first generated, and the score of each hypothesis in the list is updated using a log-linear combination of scores from different models. The scores from individual systems are combined with unequal weights, and the weighting coefficients are trained either numerically using the generalized probabilistic descent (GPD) approach [35] or derived as a closed-form solution. The final hypothesis is that which has the highest updated score. Wu combined the outputs of a recognizer that used syllable-oriented features with the outputs of second recognizer that used traditional phoneme-based features [78]. In this work, he first generated an *N*-best list from each system, performed Viterbi alignment on the decoding result from each system to obtain the acoustic and language scores of each utterance, combined the scores of the same utterance generated from different systems, and finally selected as the final output the utterance with the best combined score.

Hypothesis combination has the advantage of being able to combine the outputs of parallel systems with totally different model structures[1], but the fact that the result is based only on the outputs of individual recognizers creates its own problems and limitations. Let us refer to the space of all possible word sequences that could be generated from a speech recognizer and their corresponding scores as the *search space*. Ideally the goal of hypothesis combination should be to find the probabilities of each possible word sequence within the search space from all individual systems, combine the probabilities for each possible

1. Hypothesis fusion only combines the recognition outputs together, so it doesn't care how these recognition results are generated. Unlike probability combination, which will be described later, the process of hypothesis fusion is independent from the process of generating the individual hypotheses.

word sequence together to generate a new search space with updated probabilities, and determine the particular word sequence with the highest updated score. In practice this search space is very large, so the hypothesis combination methods must work with only a subset of hypotheses from each individual system rather than the complete search space. Because the different systems whose outputs are to be combined may have different properties (such as different pruning characters different probability distributions across different word sequences), the word sequence that appears at the output of one particular system may not also appear at the outputs of other systems. In general, hypothesis-fusion algorithms have limited ability to update the probabilities of word sequences that only appear at the outputs of some of the systems[2], which will inevitably affect the final performance of the combined system. This drawback of hypothesis fusion algorithms is more severe in speech recognition systems where two identical sequences of words could be treated as different hypothesis with one pruned from the output list and the other remaining on the list, based on different time alignments[3].

Two different approaches can be followed to improve the performance of the combined system. The first approach is to increase the number of word sequences to be included in the subset of the search space under active consideration, while the other is to find some representation other than a search space of final output hypotheses as the basis for information combination. In the next chapter we will describe our work directed toward increasing the search space subset in hypothesis fusion by using word lattices rather than lists of work hypotheses as the basis for hypothesis fusion. With regard to alternate representations, we will describe the probability combination method later in the thesis, including both already existing work in the next section and our own new work in the area in Chapter 5.

3.2.2 Probability combination

Probability combination refers to combination algorithms that combine the information from parallel features at the stage at which probabilities are evaluated in the search process. As described in Figure 3.5,

2. Some hypothesis fusion algorithms, such as Sing's hypothesis combination [69], are capable of generating the probabilities of certain words (or word sequences) that do not appear consistently in the outputs of all systems by concatenating words together. But these probabilities (or scores) are generated in a manner which replicates rather than updates the acoustic score of individual word sequences.

3. Two word sequences with identical word labels will be treated differently if their time alignments are different, as their scores depend on time alignment. It could happen that one of the word sequence will remain in the recognition output while the other is pruned out. Ideally we want to use the probabilities (or scores) of these two word sequences to generate an updated score, as they indeed represent the same hypothesis. But we will be unable to do that if only one of the word sequence remains in the recognition output.

information from parallel features is combined by first generating the probabilities of each recognition class from each individual system or feature set (which are generally the observation probabilities associated with each HMM state in an LVCSR), combining these probabilities via some combination functions (such as summation, multiplication, or maximization), and finally generating a combined hypothesis based on the updated probabilities using the normal search procedure.

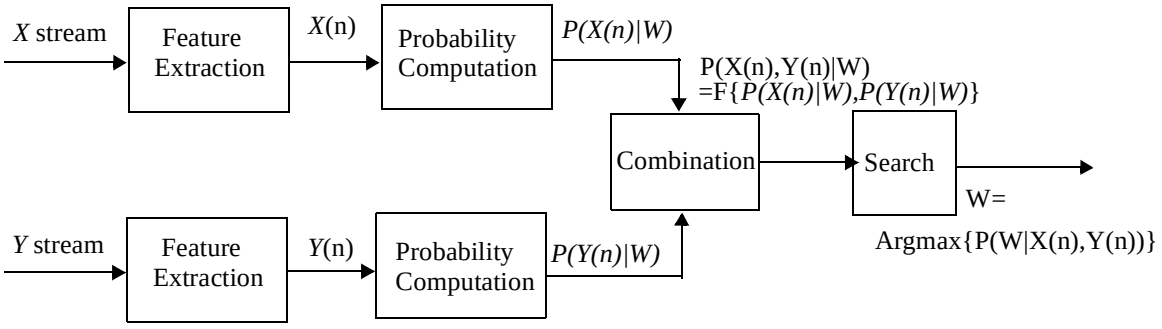

Figure 3.5. Block diagram of probability combination, where combination is done at the probability evaluation stage before the final hypothesis search.

In contrast to hypothesis fusion algorithms, which combine the information at the stage of recognition outputs, the merging of parallel information in probability combination is restricted to the combination of recognition class probabilities before the final search. This approach provides both advantages and constraints compared to hypothesis fusion.

On the positive side, the recognition class[4] probability is a complete and more compact representation of all possible word sequences in the search space than the recognition output representation[5]. In HMM-based speech recognition systems, each word sequence with a specific time alignment can be represented by a series of HMM chains, so the recognition class probability is a complete representation of the search space in the sense that every possible word sequence in the search space has its corresponding HMM chain. Because the number of HMM states in a practical speech recognition system is limited (typically from 100 to 10000 states), we can use a limited number of HMM states to represent the potentially unlimited number of possible word sequences in the search space. A complete representation is only possible at

4. Again, the recognition classes could be either HMM states, phonemes, syllables or even words which constitute a word sequence.

5. Here the term "recognition output representation" refers to the recognition output, which could be a single best hypothesis, an N-best hypothesis list, or a word lattice.

the probability stage, as the number of word sequences in an N-best list or even a lattice will increase as the size of the search space increases. As a simple comparison, the total number of HMM states in a conventional recognition task with a 1000-word dictionary and an average sentence length of three words[6] is approximately 2000 to 3000, while the possible number of word sequences could be up to $1000^3 = 10^9$. Using probability combination to merge the states requires only 2000 to 3000 probabilities from each individual system to update the score of all these 10^9 word sequences. This is a very small number compared to the 10^9 word sequences per system[7] that would be needed with hypothesis combination.

But probability combination is not without disadvantages. Compared with the flexibility of hypothesis combination in combining different features or systems with totally different structures, probability combination imposes more constraints on the model structures used by the parallel features or systems. In hypothesis combination, the parallel features or parallel systems to be combined can have totally different structures. Since only the resulting recognition hypotheses will be merged, it doesn't matter how these hypotheses are generated. In contrast, using probability combination the probability of each recognition class will be updated, which requires that the parallel features or systems use the same model structure or at least similar structures. Although it is true that we can force the individual features or systems to use similar model structures, the unique strengths of individual features or systems may not be fully utilized, which potentially may adversely affect the recognition accuracy of the final combined system.

Probability combination can be performed either synchronously (as in Figure 3.6), where the combination of observation probabilities of states is performed on a frame-by-frame basis, (*e.g.* [6][12][17][25][27][34][52][59][80]), or asynchronously (as in Figure 3.7 and Figure 3.8), where the probabilities are combined only at some higher-level unit (such as phonemes or syllables) instead of the HMM state (*e.g.* [5][6][7][19][47][73]). The state sequences of each individual feature stream are identical in synchronous combination, but they may be different when asynchronous combination is used.

Synchronous combination is more constrained than asynchronous combination, and one motivation for

---

6. Many practical speech recognition tasks have on average 10 to 15 words per sentence. Even the same word sequence may have different time alignments over multiple hypotheses, which makes the situation even more complex.

7. In hypothesis combination we need the scores of each possible word sequence to generate the updated scores.

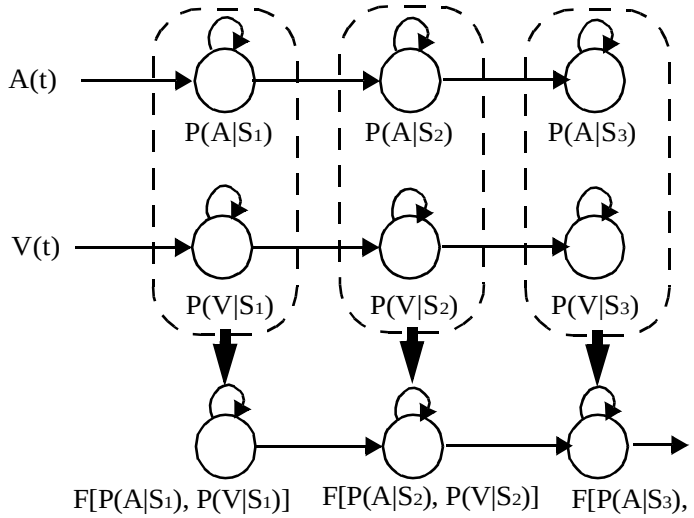

Figure 3.6. Synchronous probability combination of two feature streams A(t) and V(t). State sequences within feature streams are identical to each other. *F* is the combination function used to update probabilities.

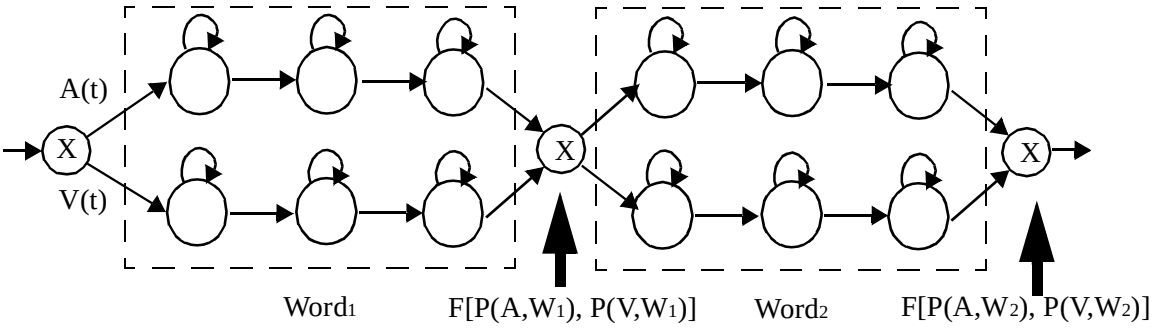

Figure 3.7. Asynchronous probability combination of two feature streams A(t) and V(t) using the multi-stream approach. The circle *X* is the synchronous unit, where probabilities from different feature streams are combined together. *F* is the combination function. The actual state sequences of the two feature stream can be different from each other.

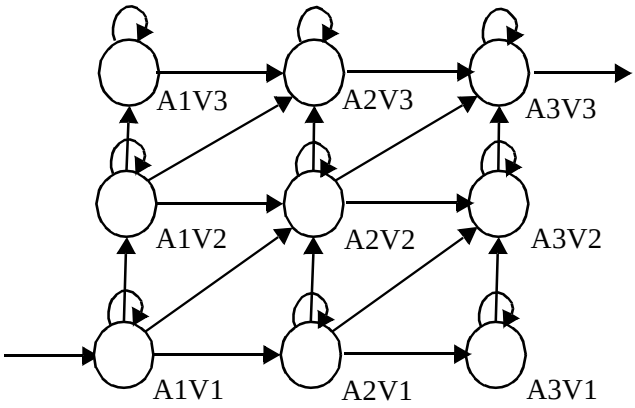

Figure 3.8. Asynchronous probability combination of two streams A and V using composite HMM[5][7][73].

using asynchronous combination is that the parallel feature streams to be combined together are frequently asynchronous with respect to each other. For example, when we combine an acoustical feature stream with a visual lip-reading feature stream in audio-visual speech recognition, the motion of the lips usually pre-

cedes the audio stream. Using asynchronous combination of the probabilities is more valid than forcing the parallel feature streams to be synchronous with each other.

There are at least two ways of implementing asynchronous combination: the multiple-stream approach (e.g. [6][47]) as in Figure 3.7 and the composite-HMM approach (e.g. [5][7][73]) as in Figure 3.8. (The composite HMM is sometimes also referred to as a "product HMM.") In the multiple-stream approach the likelihoods of higher-level synchronous units such as phonemes or words are first computed from each individual feature stream. The probability scores of these synchronous high-level units are then combined and a final hypothesis is generated directly based on the probabilities of the higher-level units. In contrast, the composite HMM, illustrated in Figure 3.8, is a multi-dimensional HMM with each dimension corresponding to one feature stream. Hence, each state of a composite HMM represents the joint status of all incoming feature streams, and the observation probability associated with that state is computed based on the observation probabilities of the parallel feature streams. The total number of states in a composite HMM is the product of the number of states for each single stream and the number of total feature streams. Some states (*e.g.* A1V3, A3V1 in Figure 3.8) that are far from the diagonal components (which represent states at which all feature streams synchronous with each other such as A1V1, A2V2, A3V3 in Figure 3.8) can be removed [5][7].

Multiple-stream methods are more appropriate (and more tractable) than composite HMM approaches when the number of parallel feature streams to be combined is very large, since the number of states in the composite HMM (which increases exponentially with the number of feature streams) prohibits the usage of this method in situations where the number of parallel feature streams is very large. Multiple-stream combination has an additional advantage over the composite HMM in that it is very similar to the synchronous combination approach, which means that we can easily transfer methods of synchronous probability combination to an asynchronous system based on multiple-stream probability combination with little additional work. In fact, the only difference between these two approaches is that synchronous probability combination combines the probabilities of states together on a frame-by-frame basis, while multiple-stream asynchronous fusion combines the probabilities of some high level units (such as phonemes, syllable or words).

Both synchronous combination and multi-stream asynchronous fusion require some specific combination function that is used to update the probabilities of the recognition class. Depending on the form of combination function, probability combination methods can be divided into the two categories of *unweighted combination* and *weighted combination*. Unweighted combination refers to those approaches in which each feature is considered to be equally important in updating probability scores (*e.g.* [38][40]). The most common methods of unweighted probability combination are *summation* as in Equation (3.2)

$$P_s = P_{1,s} + P_{2,s} + P_{3,s} + \ldots + P_{N,s}, \tag{3.2}$$

multiplication, (or *log-summation*) as in Equation (3.3)

$$LogP_s = LogP_{1,s} + LogP_{2,s} + LogP_{3,s} + \ldots + LogP_{N,s}, \tag{3.3}$$

and *maximization* as in Equation (3.4)

$$P_s = Max(P_{1,s}, P_{2,s}, P_{3,s}, P_{N,s}). \tag{3.4}$$

where P_s represents the updated probability for Class *S*, and $P_{i,s}$ is the probability of Class *S* from feature stream *i*.

A second type of combination operation is *weighted combination*, in which different feature streams are treated differently (*e.g.* [1][2][8][19][34][47][51][52][53][61][71]). One example of weighted combination is the weighted linear combination of log probabilities as in Equation (3.5):

$$LogP_s = a_{1,s}LogP_{1,s} + a_{2,s}LogP_{2,s} + a_{3,s}LogP_{3,s} + \ldots + a_{N,s}LogP_{N,s}$$
$$\sum_{i=1}^{N} a_{i,s} = 1 \tag{3.5}$$

where $a_{i,s}$ is a linear coefficient representing feature stream *i* for Class *S*, which can either be shared across all recognition classes for the same feature stream, or be made to depend on each recognition class as well as each feature stream. The coefficient $a_{i,s}$ is a measure of the contribution of each feature stream *i* to the final updated probability score of recognition Class *S*, and it can be obtained via many different ways.

Most weighted combination algorithms either require a training process that tunes the combination weights $a_{i,s}$ (*e.g.* [1][2][6][8][19][25][34][39][47][50][51][52][53][54][61][62][66][71]), or generate these weights adaptively based on a criterion such as the entropy of individual feature in predicting the probabilities of the recognition classes (*e.g.* [32]). As an example of the former category, combination weights can be generated based on either the recognition accuracy or the signal-to-noise ratio (SNR) of each individual feature stream (*e.g.* [1][6][8][51][71]). Other systematic approaches to weighting-function development include the use of discriminative training methods such as *minimum classification error* (MCE) (*e.g.* [19][50][52][53][61]), or *maximum mutual information* (MMI) (*e.g.* [34]) to derive the combi-

nation weights. Some researchers have used various confidence measures to determine the combination weights (*e.g.* [1][39][62][66]). In audio-visual speech recognition, the weighting of audio and video streams frequently depends critically on the ambient SNR. The ratio of voicing within each utterance or each frame, a measure that is correlated with SNR, can also be used to derive the weighting coefficients (*e.g.* [2]). Since there is no mathematical justification for linearly combining probabilities, some researchers have also proposed methods of updating probabilities using a multiple linear perceptron (MLP) network, with inputs that are the probabilities of all recognition classes from all feature streams and outputs that are the updated probabilities for all recognition classes (*e.g.* [6][25]). In certain situations where the number of weights is limited (*e.g.* when weights are shared by different recognition class within each individual system), an exhaustive search over a validation set may be feasible based on certain criteria (such as minimum classification error) to generate the coefficients (*e.g.* [52][47][20][56]).

In addition to these algorithms which generate the combination coefficients offline through some training process, algorithms have also been developed that generate the combination weights on line in an adaptive fashion. One example of such approaches is the entropy-based multi-stream full-combination method proposed by Ikbal *et al.* [32]. In that approach, the combination weight $a_{i,s}$ depends only on the feature stream *i* but not on the recognition class *s*. In contrast to schemes where the combination weights are fixed throughout the combination process, the weights a_i (which are no longer dependent on the class *s*) are adaptively changed from frame to frame. At each frame, the combination weight of the stream *i* is the normalized inverse value of its entropy as in Equation (3.6)

$$a_i = \frac{1/h_i}{\sum 1/h_j} \tag{3.6}$$

where h_i is the entropy of feature stream *i* at the current frame.

3.2.3 Feature combination

The third way of combining information from parallel features is to merge their information at the feature level, which is called feature combination. This method cannot be easily applied to situations where information is combined across different recognition systems.

As illustrated in Figure 3.9, feature combination refers to the extraction and combination of parallel feature sets prior to probability evaluation. As the name implies, feature combination is accomplished by concatenating all features together to form a new feature vector (*e.g.* [1][62][63][71]), and performing

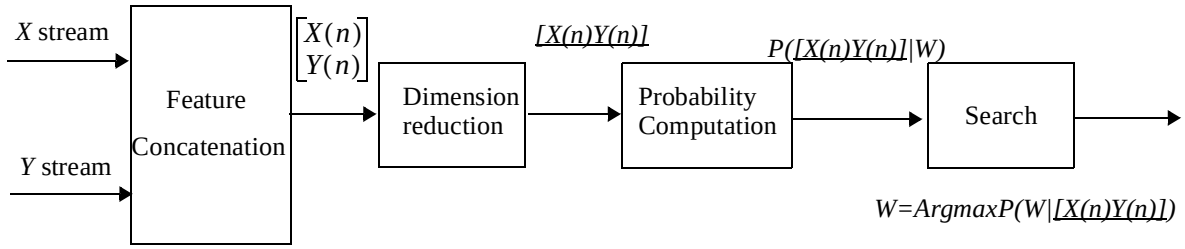

Figure 3.9. Block diagram of feature combination. [X(n)Y(n)] is a dimension reduced concatenated feature vector.

recognition directly on the basis of this new feature vector. To avoid problems associated with insufficient training data and the curse of dimensionality, it is common to reduce the dimensionality of the new concatenated feature vector using a linear transform such as principal component analysis (PCA) (*e.g.* [11][33]) or linear discriminant analysis (LDA) (*e.g.* [11][29]). Feature combination algorithms are intuitive and very simple in implementation since the combination can be done easily without any modification to the recognition and hypothesis search stages. Nevertheless, the effects of feature combination may be affected by noise in individual feature streams, since noise from a single feature stream may be spread to all other components of the concatenated feature vector through the linear transformations used in dimension reduction. In addition, concatenating parallel features may have a similar effect as multiplying their corresponding probabilities together, because we normally use diagonal covariance matrices for modeling state observation probabilities. While the dimensionality-reduction transformation may introduce some additional effects, concatenation of statistically-independent features is generally equivalent to multiplying their probabilities. This is just a subset of all the possible ways in which their probabilities could be combined using probability combination, so the use of probability combination rather than feature combination may provide greater flexibility. Finally, feature combination implicitly assumes synchronicity among the parallel features which are concatenated. When feature streams are not actually synchronized to each other, some approximations (such as linear interpolation) must be applied, which may also adversely affect the performance of the combined system.

3.3 Generating parallel feature streams to improve the recognition accuracy of the combined system

The performance of the combined system depends as critically on the parallel feature streams it uses as on the specific method of combination. Intuitively, the ideal set of parallel feature streams used in the com-

bination should provide complementary information regarding the speech signal. While the individual feature streams do not need to achieve perfect recognition by themselves on the whole recognition task, each stream should have its own "expertise" in the recognition task, and the union of these expertise should cover a much greater portion of the recognition task than each individual stream. The performance of the combined system depends jointly on the parallel features and the specific combination method used. For example, if we combine a feature stream with itself it is very likely that we will achieve no further benefit, no matter how we perform the combination, because the information in the "parallel" feature streams is identical. If the parallel features are not sufficiently different or complementary to one another, the implementation of combination methods will not make much difference in recognition accuracy.

While there has been a relatively large amount of work in parallel feature combination methods, particularly at the probability combination level, only a small amount of work has been done in the field of designing parallel features. Most researchers have used one of three different approaches to the design or choice of parallel feature streams for combination.

The first general approach is to pick some conventional individual features for use in the system to be combined according to some criterion. One such example is the work by Ellis and Bilmes [13] that explores the combination of PLP features [23] with features based on modulation-filtered spectrograms (MSG) [37] using different ranges of modulation frequencies. In that work, they first trained single systems based on each of the available individual feature stream using either PLP or MSG features, and then measured the conditional mutual information (CMI) between each pair of classifier outputs, selecting the pairs of streams with the lowest CMI measure for use as parallel features. Another example of combining well-performing conventional single streams of feature is work done by Kingsbury and Morgan [36], in which PLP features were combined with log-RASTA-PLP and J-RASTA-PLP features [24] in the recognition of reverberant speech. The two most commonly-used conventional single-stream features, MFCC and PLP features, are frequently combined as in the lattice combination work described in [45] and later in this thesis. In the audio-visual speech recognition field, many researchers simply extract features independently from the acoustical and visual streams and combine them together.

A second way of designing parallel feature streams is through *"splitting"*. The entire recognition task can be split into several sub-tasks, designing specific feature to accomplish each sub-task, and combining these features (or the decisions derived from them) together. For example, in Halberstadt's work on heterogeneous features [21][22] for phonetic recognition of the TIMIT database, he split the task of phonetic recognition into a hierarchical process that first identified the label of the class to which a phoneme belonged, and then determined the specific phoneme within that class. He divided the whole set of phones

into the classes of vowel/semivowel, nasal, stop, fricative, and chose different sets of features within each phoneme class by varying parameter values such as the duration of the analysis window, the dimension of the feature vector, the number of time derivative components used, and the inclusion or exclusion of pitch and energy information in the feature vector. The individual feature streams within each phoneme class were developed in an *ad hoc* manner based on general phonetic knowledge. They were combined together based on either hierarchical or voting methods as described in [21][22]. Another approach to splitting is the generation of parallel feature as in the "multi-band" speech recognition approach developed by Bourlard *et al.* [6] and Hermansky, *et al.* [25]. They divided the speech spectrum into several sub-bands, extracted feature streams from each band (such as the energies from critical bands included within each subband augmented by their first-order time derivatives), and combined these multi-band feature streams. Multi-band approaches have been proven effective in noisy environments, especially in situations where the speech signal is corrupted only in a subset of frequency bands.

Another approach to the generation of parallel features is to adjust specific system parameter values in the feature generation process to capture information at different scales. For example, in the BBN BYBLOS system [4], parallel features were generated by adjusting the analysis frame rate between 80 and 125 frames per second in order to capture information at different temporal scales. Halberstadt's heterogeneous features was similar in part, since he generated parallel features by adjusting the duration of the Hamming window from 10ms to 30 ms, among other feature manipulations.

Despite the substantial improvement achieved by combining the parallel features mentioned above, they still have two major drawbacks. First, most conventional parallel features sets had been designed independently of each other. Optimal performance in feature combination is likely to be obtained when the interaction of feature sets is taken into consideration as well. In addition, in current systems, the feature generation process is typically isolated from the feature combination process. We know that the combination performance depends both on the type of combination approach used and the parallel features that are used for the combination. Best performance is likely to be obtained if parallel features are designed in a way that takes account of the way in which they are combined. We will address this issue in our work in the generation of parallel features in Chapter 6.

3.4 Summary

In this chapter, we have presented an overview of algorithms that have been developed for combining parallel streams of information within a speech recognition system or across multiple recognition systems.

We began with a discussion of the hypothesis fusion algorithm, where information from parallel features or systems is merged at the level of the outputs of parallel recognition systems. We then considered probability combination, where parallel information streams are combined at the level of probability evaluation in the decoding process. Finally, we described feature combination algorithms, which combine the values of feature vectors at the beginning of the recognition process. We describe the advantages of combination at each stage, along with their drawbacks.

We also briefly described some of the existing work that has been done in the design of parallel features for better combination performance. This includes the selection of several streams of individual features for combination, as well as the splitting of a recognition task into sub-tasks and developing features that are optimal for each subtask. We also described efforts toward the generation of parallel features by adjusting feature-generation parameters such as the analysis frame rate.

As noted above, all information combination approaches have some drawbacks, and further improvements on them may be useful. In the next three chapters we will describe our efforts toward the improvement of various combination method and toward the more systematic design of parallel features. In Chapter 4 we describe a new hypothesis fusion algorithm called lattice combination, which combines word lattices developed from individual recognizers to improve hypothesis fusion performance. In Chapter 5 we discuss our efforts toward the refinement of probability combination methods. In Chapter 6, we focus on the issue of designing parallel features for improved recognition accuracy in the combination system. We discuss our new parallel feature generation algorithm which generates parallel features jointly to improve their combined performance. In Chapter 7 we summarize our work and the conclusions we draw from it.

4. Lattice Combination

4.1 Introduction

We noted in the previous chapter that hypothesis fusion is one way of combining the recognition results obtained from parallel systems or parallel features. In this chapter we present the new hypothesis fusion algorithm that we refer to as lattice combination, which combines the lattices generated from parallel features or systems. Lattice combination is different from conventional approaches such as hypothesis combination or the ROVER method in that it combines information from parallel processing at the lattice level rather than at the level of the hypotheses that are developed from the word lattices. We begin this chapter with a comparison of lattice combination to the combination of a single best hypothesis or a set of N-best hypotheses from parallel recognizers. We then describe the specific procedures used to combine lattices and generate the final resultant hypothesis from the combined lattice. Finally, we compare the performance of our lattice combination algorithm with that of previous algorithms, closing the chapter with an analysis of the comparative advantages of the various algorithms.

4.2 Combining recognition results at the lattice level

4.2.1 Organization of word lattices

A word lattice is a compact representation of the best-scoring word hypotheses developed by the recognizer, and it contains many more word hypotheses than that could be represented by an N-best list with the same amount of storage space. Lattices are directed acyclic graphs such as the one illustrated in Figure 4.1. Nodes in the lattice are associated with candidate words and their starting and ending frames, and the edges in the lattice represent the possible transitions of words in the hypothesis. A given word may occur in different hypotheses with different ending frames and/or be followed by different words. Since these variations may result in a different acoustic score for the same word, acoustic scores are stored in the edges rather than the nodes in a lattice. Each path that starts from the beginning node (“<s>”) and ends at the ending node (“</s>”) in the lattice represents a specific hypothesized word sequence. The likelihood of a hypothesized phrase or sentence is obtained by adding the log-scores of the included edges, which contain both the acoustic scores that are directly associated with the edges and the language model score based on the context information of that edge.

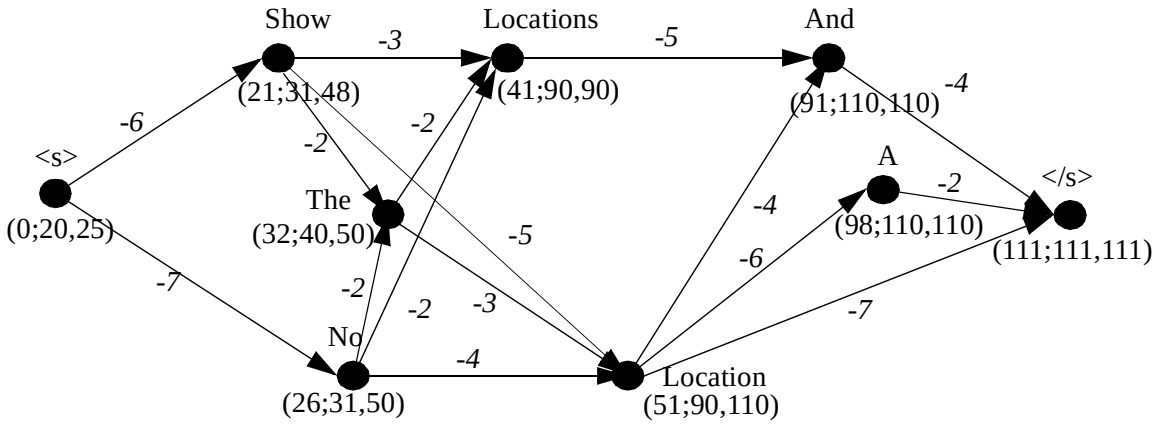

Figure 4.1. An example word lattice. All the hypothesized word sequences begin at the starting node <s> and end at the ending node </s>. Each node represents one word, as labelled around that node. The numbers (X;Y,Z) below each node represent the beginning time X, the first possible ending time Y and the last possible ending time Z of the node. The italic number associated with the edge represents the acoustic score of the corresponding word transition.

4.2.2 Rationale for combination of hypotheses at the lattice level

Fusion of information from parallel recognizers at the lattice level rather than at the hypothesis level (*i.e.* combination of a set of single best hypotheses or an N-best list of hypotheses) is advantageous for a number of reasons. A lattice contains many more word sequences than a single best hypothesis, but it is also a very compact representation which requires much less storage space than an N-best list. We will first talk about the advantage of combining multiple hypotheses over single best hypotheses, and then we will describe why we choose lattices rather than N-best hypothesis lists as the data structure within which the combination is achieved.

The advantage of using multiple hypotheses over a single best hypothesis in combination is obvious. In hypothesis combination, we generate the “union” of all the possible word sequences from the outputs of parallel systems, and determine the particular word sequence (or sequences) with the highest score from this union. When each individual system generates multiple hypotheses outputs, the union of these outputs will be much larger than the output that would have been generated with only the single best output from parallel systems. As a result, the union of multiple hypotheses is more likely to contain the correct word sequence than a union generated from the single best output of a set of parallel systems. This is the advantage of combining multiple hypotheses over the combination of single best hypotheses.

Both N-best lists and lattices can be used to represent multiple word sequences, but the lattice still has

the advantage of being a more compact representation of multiple hypotheses than an N-best list. For the same number of multiple hypotheses, lattice uses only a small portion of the storage space compared to what would be required by an N-best list. The combined performance of an N-best list representation or a lattice representation could be the same, as they both represent the same set of multiple word sequences. But the compactness of the lattice still makes it a better structure for hypothesis combination than the N-best list. As an example, the lattice shown in the Figure 4.1 contains 9 nodes and 16 edges. Each node would require 4 storage locations to store the word label, starting time, first ending time and last ending time, and each edge requires 3 storage locations for the outgoing node, ending node and corresponding acoustic score. The total number of storage locations required for the lattice in Figure 4.1 is just $9 \times 4 + 16 \times 3 = 84$. If we use an N-best list to represent all the possible word sequences in that lattice, there will be 16 word sequences in total, each of them contains at least 5 words (including the <s> and </s>), except for the word sequence "<s> No Location </s>", which contains only 4 words. Each word in a word sequence (except the last </s> word), requires 3 storage locations to store its identity, beginning time, and the corresponding acoustic score. So the total number of locations required for an N-best list representation would be at least $(15 \times 4 + 3) \times 3 = 189$, which is more than twice the amount of storage required for the lattice representation. Please keep in mind that the lattice shown in the Figure 4.1 is just a very simple example without many edges between different nodes. Actual lattices generated in real speech recognition are much more complex than this example[8], causing the motivation for the use of lattices rather than N-best lists for multiple word sequences representation in hypothesis combination to become increasingly more compelling.

4.3 The lattice combination procedure

The general process of combining lattices generated from parallel systems is as follows: we first merge the beginning and ending nodes of the utterances in the component lattices together to generate an initial combined lattice. Then we merge certain edges together if they satisfy certain constraints. We also add new edges between nodes that originated from different lattices based on a set of rules. Finally, we search for the path with highest overall probability from the merged (and modified) lattice to produce the final com-

8. As an example, for a small-vocabulary recognition task with around 1000 words and an average sentence length of 8 words, the average number of nodes in a lattice is about 500, and the average number of edges in a lattice is about 2500.

bination hypothesis. The whole process of lattice combination is illustrated in Figure 4.2. We will discuss each of these steps in more detail in this section.

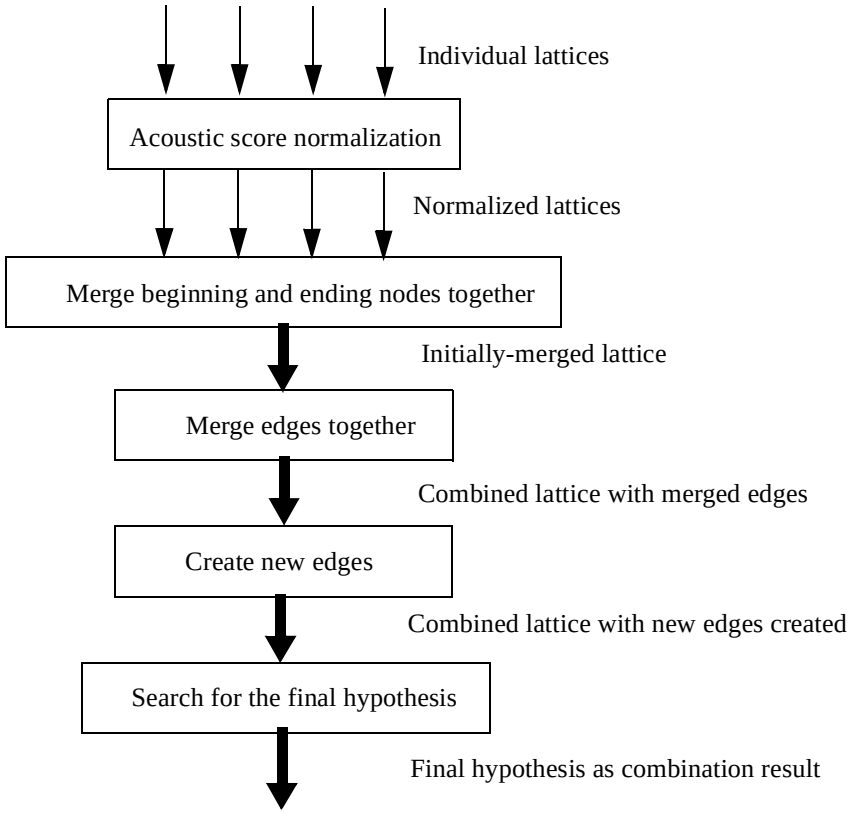

Figure 4.2. Diagram of the lattice combination process.

4.3.1 Merging the beginning and ending nodes

All paths in the lattice begin at the starting node (“<s>”) and end at the ending node (“</s>”). Although these starting and ending nodes don’t have specific meaning, they are treated as the silence regions at the beginning and end of an utterance. When parallel lattices are combined together, their corresponding starting and ending nodes will first be merged together to generate a larger lattice in which all the paths start from the merged starting node and end at the merged ending node.

4.3.2 Merging edges

Merging the lattices’ beginning and ending nodes will generate a very large lattice, and the goal of the next of processing is to reduce lattice size by merging certain edges together. In our lattice combination algorithm, edges originating from different lattices can be merged together if they satisfy the following

conditions:

- Their outgoing nodes have the same word label
- Their outgoing nodes have the same beginning and ending frame
- The word labels of the ending nodes of these edges have the same first phoneme

To merge these edges together, we first merge their outgoing nodes together into a new node, which shares the same information (*i.e.* word label, beginning frame and ending frame) with the previous outgoing nodes. The acoustic scores of these merged edges are then updated according to some functions to be discussed later.

The decision whether or not to merge is based on the three conditions above because most current speech recognition systems use context-dependent acoustic modeling. The acoustic score for a word depends on its context and is hence associated with an edge instead of a node in the lattice. If there are multiple edges originating from different lattices that have the same word label (for example, *W*) with the same duration (for example, from T_1 to T_2), and that are leading to the same first phoneme for their right context, then their acoustic scores should be the same in the merged lattice since they all represent the exact same acoustic transition. More commonly, these scores will differ for a variety of reasons (*e.g*. imperfect modeling of each individual systems, different features used in the parallel systems, etc.). Because of this, we need to combine their scores together into a new score that can be shared across all of the edges.

The combination functions considered include *maximization* as in Equation (4.1), *summation* (*i.e.* the algebraic mean) as in Equation (4.2), and *multiplication* (*i.e.* the geometric mean) as in Equation (4.3). The notation $P'_{A \to B}$ in these equations is the score of the merged edge from node *A* to *B*, and $P_{A \to B,\, i}$ is the individual score of that particular edge in lattice *i*. Figure 4.3 shows an example where the edges from the node *"LOCATION"* to the node "AND" are merged together.

$$P'_{A \to B} = Max\{P_{A \to B,\,1}, P_{A \to B,\,2}, \ldots, P_{A \to B,\,N}\} \tag{4.1}$$

$$P'_{A \to B} = \frac{1}{N}\left(\sum_{i=1}^{N} P_{A \to B,\, i}\right) \tag{4.2}$$

$$P'_{A \to B} = \left(\prod_{i=1}^{N} P_{A \to B, i} \right)^{\frac{1}{N}} \tag{4.3}$$

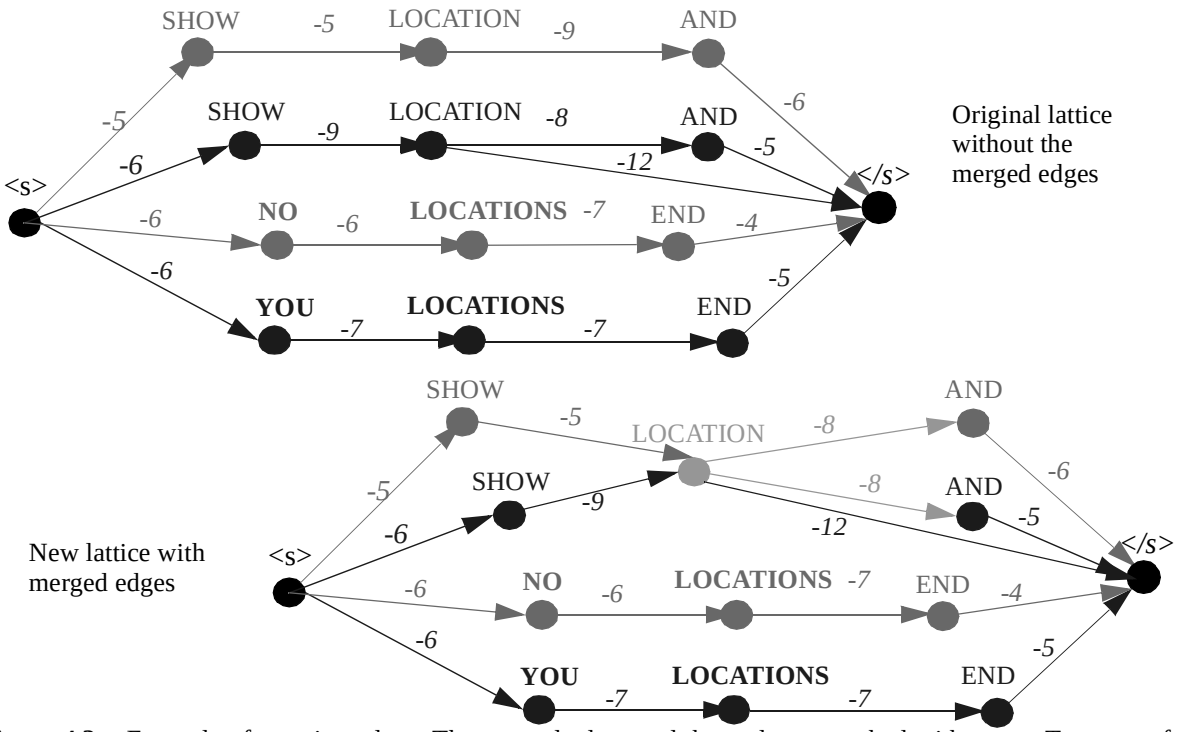

Figure 4.3. Example of merging edges. The merged edges and the node are marked with green. Two sets of parallel edges (from "LOCATION" to "AND", and from "AND" to "</s>") can be merged together. We only merge the first set for demonstration purpose. The parallel edges from "SHOW" to "LOCATION" and from "LOCATIONS" to "END" cannot be merged together because they either start at different time frames or end at different time frames.

4.3.3 Creating new edges

In addition to merging edges, we can also create new edges between nodes that originate from different lattices provided that the word label and time information of these nodes satisfy certain conditions. Specifically, we can create a new edge from Node A to Node B if the following conditions are satisfied:

- There is an edge from Node A to another Node C, and the difference in the beginning frame of Node B and Node C is within some threshold (*e.g.* 3-4 frames or 30-40 ms).
- The first phone of Node C is the same as the first phone of Node B.

The new edge will start from Node A and end at Node B. The acoustic score associated with this new edge can be estimated from the acoustic score and duration of Edge *A* to *C*, and the duration of the new

edge as in Equation (4.4):

$$S_{A \to B} \approx \frac{D_{A \to B}}{D_{A \to C}} \cdot S_{A \to C} \quad , \tag{4.4}$$

where $S_{i \to j}$ and $D_{i \to j}$ are the acoustic score (in log units) and the duration of the edge from Node *i* to Node *j* respectively.

Creating a new edge under these circumstances is reasonable, because if the existing edge from Node A to Node C specifies that Node A can end just before Node C at the time instance right before Node C, Node *A* should also be able to make a transition to Node B, provided that Node B has a similar beginning time to Node C. The second constraint is imposed because of the context-dependent acoustic modeling used in current speech recognition systems. When we created a new edge from Node A to *B*, we need to also assign an acoustic score for that edge. The acoustic score associated with an edge depends on the specific value of the feature vector in the specific time duration of that edge, and it also depends on the models, which are based on the label of the word of the outgoing node (*e.g.* Node A) and the first phoneme of the incoming node to that edge (*e.g.* Node B or *C*) if we use triphonetic context-dependent modeling. By constraining Nodes B and C to have the same first phoneme, the models used in computing the acoustic scores of the edges *A* to *B* and *A* to *C* become the same. As both edges span approximately the same duration, their acoustic scores should also be very similar to one another. In addition, since the acoustic score of an edge is the product of the acoustic scores for all the frames within the duration of that edge, linearly estimating the log-acoustic score of the new edge as in Equation (4.4) is a valid approximation.

Figure 4.4 shows an example of the creation of new edges, with the acoustic scores of the new edges assigned using Equation (4.4).

4.3.4 Score normalization

The lattice combination method also requires that acoustic scores be normalized across the different lattices before they are combined together. As with all other statistical pattern classification systems, the goal of the statistical speech recognizer is to pick the Class *C* (*i.e.* the hypothesized word sequence) with the maximum *a posterior* probability $P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$, where *X* represents the acoustic features.

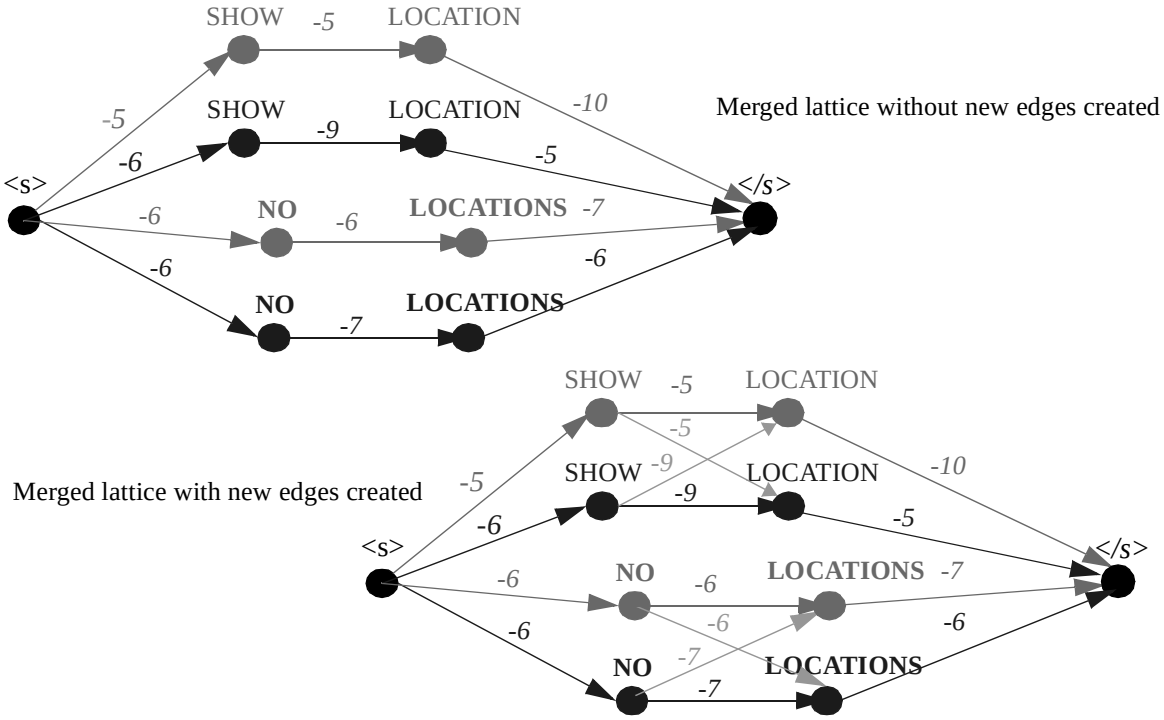

Figure 4.4. Example of creating a new edge. All newly-created edges are marked in green

When there is only one feature stream used in the recognition, $P(X)$ remains constant for all the recognition classes, and the classification decisions made on the basis of $P(X|C) \cdot P(C)$ or $P(C|X)$ will be identical to each other. It is possible that the acoustic scores $P(X|C)$ produced by the individual recognizers from which the combined lattice is generated from may not have comparable numerical ranges[9]. If we combine their scores without proper normalization, classification information embedded at certain individual feature or system may diminish. For example, if *summation* (i.e. algebraic mean) is used as combination function, the un-normalized acoustic scores from an individual system with very low absolute values will be overshadowed by the acoustic scores from systems with greater absolute values.

The specific score normalization scheme used in our lattice combination algorithms is based on the maximum acoustic score of states within each frame. The specific process can be described as follows: In generating lattices from individual recognizers that are later to be combined, for each feature vector X_i at

9. We use probability density functions instead of the actual probabilities in continuous HMM modeling, so the acoustic scores from different systems or features may have different dynamic ranges. In addition, the dimension of the features in these parallel systems may also be different from each other. This will also cause the acoustic scores of different systems to have different dynamic ranges.

frame *i*, $P(X_i|S_j)$, the acoustic score of HMM State S_j, is normalized based on the highest acoustic score among all the HMM states at that frame, $Max_j\{P(X_i|S_j)\}$. The normalization factor for any edge in a lattice is the sum of these maximum HMM state scores (expressed in log values) over the entire duration of that edge. Here is an simple example. Suppose we have an edge $E_{w_{1\to 2}}$ in the lattice that represents the transition from word W_1 to word W_2 starting at time instance T_1 and ending at time instance T_n. The state sequence associated with this edge is $S = S_{w_{1\to 2},1}, S_{w_{1\to 2},2}, \ldots, S_{w_{1\to 2},n}$, with State $S_{w_{1\to 2},i}$ occurring at frame *i*. The acoustic score of the edge before normalization is

$$P(X_{1\to n}|E_{w_{1\to 2}}) = \prod_i P(X_i|S_{w_{1\to 2},i}) \tag{4.5}$$

The normalized acoustic score associated with that edge $\tilde{P}(X_{1\to n}|E_{w_{1\to 2}})$ is

$$\tilde{P}(X_{1\to n}|E_{w_{1\to 2}}) = \prod_i \frac{P(X_i|S_{w_{1\to 2},i})}{Max_j\{P(X_i|S_j)\}} = \frac{P(X_{1\to n}|E_{w_{1\to 2}})}{\prod_{i=1}^{n} Max_j\{P(X_i|S_j)\}} \tag{4.6}$$

with $\prod_{i=1}^{n} Max_j\{P(X_i|S_j)\}$ the normalization factor of the edge $E_{w_{1\to 2}}$.

As reflected in Equation (4.6), the normalized acoustic score $\tilde{P}(X_{1\to n}|E_{w_{1\to 2}})$ will be invariant to the scale of the acoustic score of individual HMM state since

$$\tilde{P}(X_{1\to n}|E_{w_{1\to 2}}) = \prod_i \frac{P(X_i|S_{w_{1\to 2},i})}{Max_j\{P(X_i|S_j)\}} = \prod_i \frac{K\cdot P(X_i|S_{w_{1\to 2},i})}{Max_j\{K\cdot P(X_i|S_j)\}} \tag{4.7}$$

4.4 Searching for the final hypothesis from the combined lattice

After normalizing the acoustic scores, merging the appropriate edges, and creating new edges, the final hypothesis is generated from the combined lattice by searching for the path with the highest accumulated score from the combined lattice. This can be done using many different search algorithms such as A* or Viterbi search. We use Viterbi search in our lattice combination algorithm to obtain the final combination

hypothesis. We chose to just generate the single best hypothesis from the combined lattice for simplification, but N-best lists can also be easily generated from the combined lattice as well.

4.5 Evaluation of the performance of lattice combination

To test the performance of our lattice combination algorithm, experiments on several different tasks were performed using lattices that were generated from two sets of parallel features. We tested six different corpora with different vocabulary size and noise conditions. In each testing corpus, we compared the performance of our lattice combination algorithm with that of the conventional single best hypothesis fusion algorithms such as ROVER [15] and Hypothesis Combination [69]. The performance measure in terms of both the Word Error Rate and the statistical significance [16] were tested. In our lattice combination experiments, we first obtained lattices with normalized scores from the individual systems, then merged them together by merging their edges and creating new edges to generate the combined lattice as described in the previous sections in this chapter. The Viterbi search algorithm was then applied to the combined lattice with an appropriate language model (most commonly a bigram model) to generate the final combination hypothesis. Within each testing set, the models used for the individual features were all phoneme-based HMMs, but the topology of the HMM varied for the different recognition tasks.

The first testing corpus was the Resource Management (RM) database described in the previous section. We tested RM sentences using clean speech (without added noise), by combining hypotheses (single best hypotheses and lattices) derived using the MFCC and PLP features. Results were obtained using the ROVER, hypothesis combination and our own lattice combination algorithms. Tri-state context-dependent triphone continuous HMMs were used, with 13-dimensional feature vectors augmented by delta and double delta components (producing a total of 39 dimensions). The observation probability of each HMM state was modeled using a single Gaussian distribution. The *maximization* function was used in updating the score of the merged edges. Table 4.1 lists the performance in terms of word error rate of the various hypothesis fusion algorithms.

Results were also obtained again using the RM database, but using LDA and PCA/KLT features rather than MFCC and PLP features for generating the parallel lattices. All other aspects of the experimental procedures were identical to the first RM experiment using MFCC and PLP features. Table 4.2 lists the combined performance of the various combination schemes using the second RM database.

WER(%)	MFCC	PLP	ROVER	Hyp-Comb	Lat-Comb
RM	10.29	11.49	11.36	9.68	9.12

Table 4.1. Performance of various hypothesis fusion algorithms on the RM database. MFCC and PLP features were used. Hypo-Comb represents the hypothesis combination algorithm, Lat-Comb is our lattice combination algorithm.

WER(%)	LDA	PCA	ROVER	Hyp-Comb	Lat-Comb
RM	8.36	8.96	8.95	7.54	7.21

Table 4.2. Performance of various hypothesis combination algorithms on the RM database. LDA and PCA/KLT features were used. Hypo-Comb represents the hypothesis combination algorithm, Lat-Comb is our lattice combination algorithm.

In both RM databases, the statistical significance of measurements of differences between our lattice combination result and the conventional hypothesis combination result was evaluated using the matched pairs method [16]. The improvement of lattice combination over hypothesis combination was statistically significant (based on a 0.05 significance threshold as suggested in [16]) when combination results were compared for the full testing set of 1600 sentences.

The experimental results indicate that lattice combination produces better combined performance than the conventional previous best algorithm, hypothesis combination [69]. Although the actual relative improvement in WER over hypothesis combination varied from 8% to 4% for the two RM databases, our lattice combination algorithm consistently outperforms the hypothesis combination algorithm. In addition, the relative decrease in WER of our lattice combination algorithm over the single best individual feature stream (MFCC features in the first RM database and LDA features in the second RM database) was consistently around 14%.

We also note that the relative improvement of lattice combination over hypothesis combination algorithm varies from database to database. This may be because an advantage of combining lattices rather than single-best hypotheses is that the correct word sequence is more likely to be included in the lattice than in the single best hypothesis. Of the two RM databases, the second one produced better individual recognition accuracy using LDA and PCA features, so the correct hypothesis is more likely to be included in the single-best hypothesis than it would have been in the first experiment using MFCC and PLP features.

As the single-best hypothesis becomes more likely to contain the correct word string, the comparative advantage to be obtained using lattice combination diminishes, and the benefit to be expected from using lattices as structure for combination may also decrease.

The third set of comparisons was performed using the TID database. In these comparisons traffic noise was artificially added to the speech in the database to get a sense of how the performance of our lattice combination algorithm compared with other combination algorithms in noisy environments. We tested on two conditions with different noise levels, 5 dB and 10 dB SNR. The parallel features used in these two comparisons were the same 39-dimensional MFCC-derived and PLP-derived features as in the first RM experiment, but semi-continuous HMMs rather than fully-continuous HMMs were used because of the relatively small vocabulary size of the TID database. Table 4.3 describes the performance obtained using the various hypothesis fusion algorithms for these experiments. The statistical significance of differences between the lattice combination and hypothesis combination results using the matched pairs method was observed to be 0.18 and 0.05, respectively, for the 10-dB and 5-dB data.

WER (%)	MFCC	PLP	ROVER	Hyp-Comb	Lat-Comb
TID 10 dB	12.5	13.0	13.7	11.9	11.3
TID 5 dB	25.5	26.6	26.1	25.6	24.7

Table 4.3. Word Error Rates obtained for various hypothesis combination algorithms using the TID database for two SNRs, 5 dB and 10 dB. Hypotheses were combined from results obtained using standard MFCC and PLP features. Hypo-Comb represents the hypothesis combination algorithm, Lat-Comb is our lattice combination algorithm.

The experimental results on the TID databases revealed some other factors about our lattice combination algorithm. First of all, in addition to working well in “clean” environments, lattice combination is also capable of improving the combined performance in noisy environments. Indeed, among all hypothesis fusion algorithms that we have tested, lattice combination was the only combination scheme that improved the combined performance in the TID 5-dB condition. Second, noise affects not only the individual recognition performance, but also the performance of the combined hypotheses. This is even more obvious as we consider the 4% relative reduction in WER obtained using lattice combination compared to the error rate of the single best-performing individual system, which uses MFCC features, in the TID 5-dB experiment. In contrast, the use of lattice combination produced a corresponding 14% relative decrease in WER for clean RM sentences, using the same set of parallel features.

The last set of comparisons were obtained using the Speech In Noisy Environments (SPINE) 1 and 2 corpora. The SPINE 1 and 2 databases contain telephone-bandwidth speech data corrupted by a wide variety of noises at various SNRs. The features used in these experiments were different from those used previously. Specifically, the SPINE 1 experiments used two versions of MFCC features based on different DCT implementations, producing feature sets that were not very different from one another. The SPINE 2 performance measurements were obtained using different sets of LDA features that were designed to discriminate among two different sets of phoneme classes. Intuitively we can expect that the parallel features used in SPINE 2 set are more complementary than the features used in the SPINE 1 comparisons. The dimensions of the parallel features were still 39 including all delta and double delta components. Context-dependent fully-continuous triphonetic HMMs wee used. Results for the two SPINE tasks are summarized in Table 4.4. The statistical significance of differences between the lattice combination result and hypothesis combination was 0.12 in the SPINE 1 database and 0.04 in the SPINE 2 database.

WER(%)	Feature 1	Feature 2	ROVER	Hyp-Comb	Lat-Comb
SPINE 1	35.1	36.2	35.4	34.2	33.2
SPINE 2	17.5	16.6	17.8	15.9	15.0

Table 4.4. Word Error Rate performance of various hypothesis combination algorithms using the SPINE 1 and SPINE 2 databases. The parallel features used for SPINE 1 were MFCC features with two different DCT implementations. The parallel features used for SPINE 2 were two different LDA features that were designed at discriminating among different phoneme classes. Hypo-Comb represents the hypothesis combination algorithm, Lat-Comb is our lattice combination algorithm.

The experimental results using the two SPINE databases revealed, at least partially, an interesting relationship between the property of parallel features and their combined performance. Compared with the corresponding improvements in the SPINE 1 task, lattice combination[10] in the SPINE 2 task, for which parallel features used were designed intentionally to increase the complementarity or difference between individual streams, achieved better improvement over both the single-best performing individual system and the hypothesis-combination result, both in terms of word error rate and statistical significance. In fact, the most statistically significant improvement of lattice combination over hypothesis combination was achieved on the SPINE 2 database. Although this may not be a strictly fair comparison as there are many

10. Even for hypothesis combination, the improvement in WER over the single best performing individual system is greater in the SPINE 2 database than in the SPINE 1database.

differences between the SPINE 1 and SPINE 2 data, it does suggest the potential of designing complementary parallel features for improving the accuracy of the combined system. We describe our work in the design of parallel features to improve combined performance in Chapter 6.

4.5.1 The effect of using other functions in merging edges and normalizing acoustic scores

In our default implementation of the lattice combination algorithm, we used the maximization function to update the acoustic score of the merged edges and generate the normalization factors for the acoustic score of individual lattices as in Equation (4.1) and Equation (4.6). Nevertheless, other functions could also be used to generate the updated scores for the merged edges including the *summation* (*i.e*. the algebraic mean) function as in Equation (4.2), and the *multiplication* (*i.e.* the geometric mean) function as in Equation (4.3). There are also other ways to generate the normalization factors for acoustic score normalization in addition to computing the product of the maximum HMM state scores across the duration, as was done above. For example, one could base the normalization factors on the summation (rather than the maximum) of all HMM state acoustic scores at each frame as described in Equation (4.8), where the meaning of all symbols are the same as they are in Equation (4.6).

$$\tilde{P}(X_{1\to n}\big|E_{w_{1\to 2}}) = \prod_i \frac{P(X_i\big|C_{w_{1\to 2},i})}{\sum_j P(X_i\big|C_j)} = \frac{P(X_{1\to n}\big|E_{w_{1\to 2}})}{\prod_{i=1}^{n}\sum_j P(X_i\big|C_j)} \tag{4.8}$$

We also compared the performance obtained using summation and multiplication to update the acoustic scores of the merged edges, as well as the effects of using summation as in Equation (4.8) instead of maximization as in Equation (4.6) for normalizing the acoustic scores. These experiments were all carried out on the RM database using MFCC and PLP as parallel features, and the experimental results are listed in Table 4.5:

One of the interesting observations revealed from Table 4.5 is the variation in recognition accuracy obtained using the different combination functions that were used to update the acoustic scores of the merged edges. Although the multiplication function performs best in the RM database, using MFCC and PLP features and lattice combination, other combination functions (*e.g.* maximization, summation) were sometimes the best-performing function in other recognition tasks for which combination was performed

WER	Max Norm Max Merge	Max Norm Sum Merge	Max Norm Prod Merge	Sum Norm Max Merge	Sum Norm Sum Merge	Sum Norm Prod Merge
	9.12	9.47	8.91	9.46	9.72	8.95

Table 4.5. Performance in terms of word error rate of using other functions in updating acoustic score of the merged edge and generating normalization factor in RM database with MFCC and PLP as parallel features. "Max", "Sum" and "Prod" refer to the maximization, summation (algebraic mean) and multiplication (geometric mean) respectively. The top row of each column specifies the normalization function and how the scores of the merged edges were updated.

at other stages in the recognition process (*e.g.* probability combination using different parallel features)[11]. This may suggest that the best combination function may be depend on not only which type of combination procedure is used, but also on specific properties of the testing corpus and the parallel features used for combination.

We also observed that the proper normalization of acoustic scores has a critical effect on the combined performance. In addition to the experimental results reported in Table 4.5, we also performed other experiments using both hypothesis combination and lattice combination to combine un-normalized single best hypotheses and lattices. Using hypothesis combination with un-normalized scores, the WER was 10.85% compared with the corresponding WER of 9.68% using maximization normalization and 9.48% using summation normalization. For lattice combination, the performance degradation was even worse. These results, together with those in Table 4.5, clearly indicate that the acoustic scores from parallel systems need to be normalized properly in order to observe an improvement of performance in the combined system. The particular choice of maximization or summation in normalization did not appear to make too much difference in our experimental results. One reason for this could be the fact that both maximization and summation can normalize scale differences in acoustic scores between sets of parallel features[12]. On the other hand, the individual WER of MFCC and PLP features always remained at 10.29% and 11.49% regardless of how their scores were normalized for combination. This observation is no surprise, as nor-

11. In one example which will be described in the next chapter, maximization provides the best combined performance while multiplication provides the worst performance for the same RM database, when we combine MFCC and PLP features together at the state level rather than the hypothesis level.

12. As one example, assuming there is a set of acoustic scores P1, P2,...Pn. Feature A outputs these scores as Factor_A*P1, Factor_A*P2,..., Factor_A*Pn, while the corresponding scores from feature B are Factor_B*P1, Factor_B*P2,...,Factor_B*Pn. The scaling factors Factor_A and Factor_B can be removed when the summation or maximization functions are used as the normalization function.

malization affects only the relative scores of the same class from different systems. The relative scores of different classes from the same system (*i.e.* the same feature) will remain the same no matter how their scores are normalized.

4.6 Recognition accuracy and computational load

In the previous section, we have shown that the proposed lattice combination method is capable of achieving improvement over the conventional hypothesis-fusion algorithms using single-best hypotheses for various recognition tasks. Our proposed lattice combination method consistently outperforms the conventional hypothesis combination and ROVER methods. On certain recognition tasks like the TID task at 5 dB SNR, the lattice combination method is the only combination scheme that reduces the word error rate at all.

We also note from the experimental results that the amount of improvement varies for the different recognition tasks, and not all improvements are statistically significant. One of the reasons for these differences could be the different properties of the testing corpus, as they vary from each other in terms of the noise conditions, number of phonemes, total number of the HMM states used in the modeling, and their model structures as well (*e.g.* experiments with the TID database used semi-continuous HMMs, while experiments with the RM and SPINE databases used continuous HMM). Other reasons could be the intrinsic similarity of the parallel features used in the combination. It can be seen from the experimental results that one of the biggest improvements and also the most statistically significant result was achieved on the SPINE 2 database, for which the parallel features had been developed specifically to discriminate between different sub-word classes. This is expected: by forcing each individual feature to best discriminate different sub-word classes, the algorithm can select the locally-most-prominently-scoring words in its composite best path through the combined lattice.

Although lattice combination in general provides better performance than hypothesis combination, it incurs a greater computational load. Lattice combination needs to search the parallel edges that can be merged together and the nodes from which new edges can be created. But unlike hypothesis combination, which only operates on the single best hypotheses generated from parallel systems, lattice combination processes the lattices provided by individual systems. Since the number of edges and nodes contained in a lattice is much larger than the corresponding items in a single hypothesis, the time spent in the process of

searching for the available edges to merge and nodes to create new edge in lattice combination will be much larger than the corresponding time in the hypothesis combination. This will consequently make lattice combination slower than the hypothesis combination. As an example, if we simply count the amount of time spent in processing a same set of testing utterances, lattice combination takes about 60 to 80 times more processing time than is required by hypothesis combination for the same testing set. Although this comparison is not a fair comparison because the lattice combination code has not been optimized for speed, it provides a rough idea of the comparative execution speeds of lattice combination and hypothesis combination.

We suggest that lattice combination is likely to be preferred to hypothesis combination in situations for which 1) computational resources are not strained, 2) the performance of individual systems is not satisfactory, and 3) optimal recognition accuracy is more import than real-time execution.

5. Improving Synchronous Probability Combination

5.1 Introduction

In the previous chapter, a new hypothesis combination method was presented in which lattices generated from parallel systems are combined together to generate the final combination hypothesis. By utilizing the lattice as a more complete representation of all the possible hypotheses considered in the search space, we were able to achieve better hypothesis combination performance over the conventional single-best hypothesis-combination algorithms. The experimental results reported in the previous chapter showed that the approach of combining lattices is capable of delivering good performance for many different tasks under different conditions.

Although the lattice contains much more complete information about the hypotheses than a single-best hypothesis or an *N*-best list of hypotheses, it is still not a complete representation of all the word sequences that should be considered in the search space. Since many hypotheses in the original search space with relatively low scores will be pruned out from the lattice, they may still be excluded in lattice combination just as they would have been if the single-best hypothesis combination methods were used. This problem will limit the potential improvement of lattice combination over hypothesis combination of the single best recognition outputs.

The states of the HMM are a different stage in the decoding process where recognition hypotheses are implicitly represented. Since each individual hypothesis has its own corresponding state sequences, the set of HMM states can be considered to be a complete representation of all the word sequences that will be considered in the search space. The HMM state is also a compact representation because the total number of HMM states is very limited, on the order of 100 to 10000, depending on the specific task, many fewer than the total number of possible hypotheses in the search space. All this suggests that we can perform combination at the stage in the HMM decoder at which probabilities are evaluated, instead of combining hypotheses together in the output stage.

We begin this chapter with a general description of synchronous probability combination in Section 5.2. We then analyze two issues associated with synchronous probability combination as ways of weighting the contributions from parallel systems during the combination process and determining the spe-

cific attribute of speech class for which probabilities are merged together. We propose two algorithms that specifically address these two issues to improve the accuracy of a synchronous combination system. The first is a new way to generate the combination weights of parallel systems using *loss-based* training, and the second represents a new algorithm that releases the constraint of sharing the same HMM states tying patterns across the parallel systems. These two newly proposed algorithms are described in the Section 5.3. Finally, we present our evaluation results of the new synchronous probability combination algorithms for the various recognition tasks in Section 5.4.

5.2 Synchronous probability combination

As described above, information from parallel systems or features can be combined together at the probability level. One simple but effective way to accomplish this is to combine the acoustic scores of HMM states from parallel systems synchronously at each frame. The contribution from each individual system to the combined score can vary from system to system, which can be characterized by a combination coefficient that weights the contribution (or reliability) of each individual system in generating the final combined probability (or acoustic score) of the HMM state. This method has been called *weighted synchronous probability combination* (*e.g.* [6][12][17][25][27][34][47][52][59]), as described in the previous chapter that reviews information combination. Some examples of weighted synchronous probability combination include the weighted summation as in Equation (5.1), weighted multiplication or log-summation as in Equation (5.2), and weighted maximization as in Equation (5.3):

$$P_j = a_{1,j}P_{1,j} + a_{2,j}P_{2,j} + a_{3,j}P_{3,j} + \ldots + a_{N,j}P_{N,j} \tag{5.1}$$

$$P_j = Max\{a_{1,j}P_{1,j}, a_{2,j}P_{2,j}, a_{3,j}P_{3,j}, \ldots, a_{N,j}P_{N,j}\} \tag{5.2}$$

$$LogP_j = a_{1,j}LogP_{1,j} + a_{2,j}LogP_{2,j} + a_{3,j}LogP_{3,j} + \ldots + a_{N,j}LogP_{N,j} \tag{5.3}$$

In the equations above, P_j is the combined acoustic score of HMM state *j* at the current frame, and $P_{i,j}$ and $a_{i,j}$ are the acoustic score (*i.e.* acoustic likelihood) and combination weight, respectively, of the i^{th} individual combination component. We generally force the combination weights to sum to 1 as in Equation (5.4):

$$\sum_i a_{i,j} = 1 \tag{5.4}$$

The combination weights have a great effect on synchronous probability combination performance. They represent the relative confidence or contribution of individual combination components in predicting the score (or likelihood) of each recognition class (*e.g.* HMM state), and an appropriate way of generating these weights is necessary to obtain good synchronous probability combination performance. In the following section we will describe our proposed algorithm of generating combination weights, in which the weight assigned to an individual system is based on the performance of that system within each recognition class. We will also compare our new algorithm with some of the existing algorithms (*e.g.* [6][19][34][50][52][53][61]) in determining combination weights later in this chapter.

Another potential source of improvement for synchronous probability combination is in the definition of the speech class *j*. As reflected in the Equation (5.1) through Equation (5.3), the speech class *j* must be the same in all the combination components in order to generate the corresponding updated score (or likelihood). In other words, all the individual systems (or features) used for combination must share the same set of recognition classes in order for scores to be updated. Although this assumption is likely to be valid for high-level classes such as words, phonemes, or even syllables, it may not remain valid for classes defined at the level of the HMM state, which is the stage at which pattern classifications are most commonly made in speech recognition. This is because most current LVCSR systems use context-dependent acoustic modeling (*e.g.* triphone-based context-dependent acoustic modeling). A given phoneme will have many variations depending on its left and right context. Each of these variations requires a specific HMM model, so the total number of possible HMM states is generally too large to be fully trained with a limited amount of training data. (For example, if there are 50 phonemes, the total number of context-dependent triphones could be up to $50^3 = 125000$ [13], and if each triphone use a 3-state HMM, the total number of HMM states will be three times 125000, or 375000!) As described in Chapter 2, we can use *state tying* (*e.g.* [10][30] [31]) to reduce the total number of free model parameters by forcing different HMM states to share the same observation probability distributions. State tying works well for individual recognition systems, but it is problematical for synchronous probability combination of parallel systems. Since parallel systems combined using synchronous probability combination will be trained independently from each other based on maximum likelihood criteria, the state-tying patterns (*i.e.* the identities of the specific con-

13. The actual number of triphone also depends on the dictionary, which constrains the possible phoneme sequences.

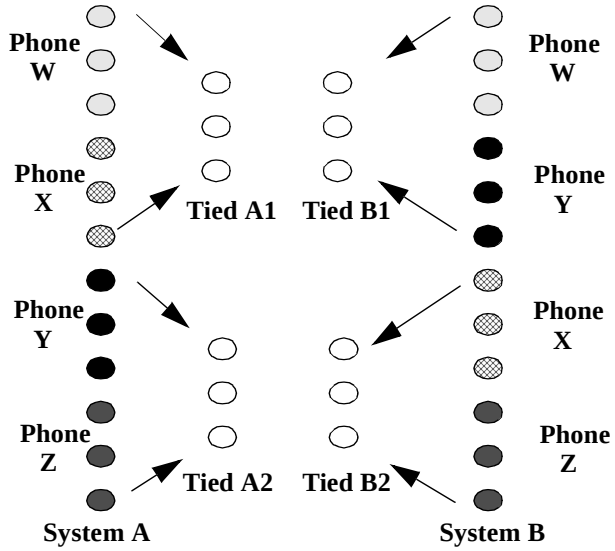

Figure 5.1 An example of different tying patterns within individual systems. Two systems, A and B, each has four different phones as W, X, Y and Z. The HMM of each phone has three states marked with different patterns. In System A, the states of the phones W and X tied together as tied states A1, while states of phones Y and Z are tied together as tied states A2. In System B, W and Y are tied together as states B1, while X and Z are tied together as states B2. If we directly combine the probabilities of the tied states A1, A2, B1 and B2, we will be unable to determine whether we should combine the probabilities of tied state A1 with B1 together or with the tied state B2.

text-dependent states that are tied together) of each individual system may differ from one another. In other words, context-dependent HMM states that are tied together in one system may not also be tied together in the other systems. If we perform probability combination directly based on these tied states, we may be unable to obtain a one-to-one mapping of the tied HMM states of the different parallel systems, in which case we would be unable to update their probabilities. Figure 5.1 illustrates this problem with a simple example.

One way of solving the problem is to force the parallel systems to share one tying pattern. This can be accomplished by first training the models for each individual system, picking the state-tying pattern from one system and re-training the models of all the other systems based on the selected state-tying pattern. For example, we could select one system (*e.g.* System A in the above example) with the best individual performance, and force all other systems (*e.g.* systems B) to use the same state-tying pattern as in System A. This will generate a one-to-one mapping between the tied states of parallel systems in synchronous probability combination, since all systems share one single state-tying pattern. This could be problematical for System B which imports the tying pattern from System A. While the tying pattern from System A is best for System A, it may not also be best for System B. As a result, the performance of System B will deteriorate if it

is trained based on the state-tying pattern of System A instead of being trained its own state-tying pattern, and this will consequently affect the probability combination performance of these systems. In other words, we must sacrifice the performance of individual system in order to enable them to be combined, and we hope that an overall gain in recognition accuracy in the combined system can offset the loss in individual performance. Although the overall effect in most cases is indeed improved performance, it would definitely be beneficial if we can avoid this sacrifice of individual system performance but still be able to benefit from probability combination. In the next section, we describe a proposed algorithm that enables us to combine directly the states of parallel systems for synchronous probability combination without sacrificing the performance of the individual systems.

5.3 Improving synchronous probability combination

5.3.1 Weighted synchronous combination with loss-based training of combination weights

The properties of the multiple systems to be combined should be different from one another, at least to some extent, in order to achieve benefit from combination. In practice, each of the parallel systems used in the combination may have its own "speciality" for certain recognition classes, in the sense that the scores provided by that system on these recognition classes will be closer to the "oracle" scores than the scores from other systems. It would be beneficial to be able to utilize the "speciality" of each individual system by putting more emphasis on predictions from each individual system for the recognition classes that belongs to its "speciality". One way of utilizing the "speciality" of each individual combination system is to assign a combination weight to each individual systems (*e.g.* [2][6][8][19][25][34][47][51][52][53][61][71]). This weight could either be shared across the different recognition classes within each individual system, or be dependent on the specific recognition class. The combination weight should reflect the "confidence" or "closeness" of each individual system in predicting the score of each recognition class. The more confident or closer an individual system can predict the score for a recognition class, the greater the weight of that system for that recognition class.

Our proposed approach of determining the combination weights is based on the difference between the predicted probability score for each recognition class from the individual systems to be combined and the "oracle" probability score that is observed from Viterbi alignment of the training data. Specifically, we will calculate the accumulated difference between the probability generated from each individual combination

component and the "oracle" probability from Viterbi alignment for each recognition class. We will consider this difference to be a measure of the "confidence" with which a given individual system can predict the correct acoustic score for a specific recognition class, and we will assign a combination weight based on the accumulated difference of each system to be combined for each recognition class (*i.e.* HMM state).

Let us consider an example in which we derive the combination weights of parallel feature streams from the individual systems to be combined. We first define an "oracle" state vector $\bar{S}_t^j$ for the j^{th} feature stream as an M_j -dimensional column vector with binary entries in each frame *t*. M_j is the total number of HMM states in the j^{th} feature stream[14]. The entry in $\bar{S}_t^j$ corresponding to the "true" state of the HMM at time t is 1 and all other entries are 0. The "true" state at time t is defined as the corresponding state of the most likely state sequence, which is generated from the Viterbi alignment of the correct transcription of the utterance.

Then for the same feature stream *j*, we compute the acoustic scores (*i.e.* acoustic likelihoods) of all HMM states at frame *t* together and form them into an M_j dimensional column vector $\bar{P}[S_t^j]$, whose k^{th} component is the acoustic likelihood of the corresponding HMM state *k*. We normalized $\bar{P}[S_t^j]$ according to Equation (5.5) to make its components to sum up to 1:

$$\bar{P}'[S_t^j] = \frac{\bar{P}[S_t^j]}{\sum_i \bar{P}[S_t^j]_i} \tag{5.5}$$

$\bar{P}'[S_t^j]$ defined as in Equation (5.5) can be treated as a prediction of $\bar{S}_t^j$ from feature stream *j*. We can define the *loss* for the feature stream *j* correspondingly, for each HMM state of the component system at each time instance. Specifically, the *loss* incurred by state i at time t from the j^{th} system is defined as the squared difference between the oracle probability and the observed (*i.e.* predicted) probability for each state i, and it can be formulated as:

---

14. M_j will be the same across different feature streams since these parallel features will be combined together synchronously. Their class identity should be the same to enable synchronous combination.

$$loss^{i}_{j,t} = \left| A_i(\bar{S}^{j}_{t} - \bar{P}'[S^{j}_{t}]) \right|^2, \tag{5.6}$$

where A_i is a row vector with the i^{th} element equal to 1 and all other elements equal to 0. The loss functions at each frame are then summed over all time frames *t* for each HMM state of each component system. This produces the accumulated loss L_{ij} for each feature stream *j* in predicting the acoustic score of HMM state *i* as:

$$L_{ij} = \sum_{t} loss^{i}_{j,t} \tag{5.7}$$

The combination weight $a_{i,j}$ of each feature stream *j* in predicting the probability score of state *i* is then generated from the accumulated loss:

$$a_{i,j} = \frac{\exp\left(\frac{-L_{ij}}{C}\right)}{\sum_{j=1}^{N} \exp\left(\frac{-L_{ij}}{C}\right)} \tag{5.8}$$

The parameter *C* in Equation (5.8) controls the extent to which a difference in the losses observed from different systems will affect their specific combination weights. The smaller the value of *C*, the greater the emphasis that is given to the scores or predictions from those more reliable systems. The specific value of *C* can be tuned using a development test set. Although the particular choice of using the exponential function in transferring *"loss"* to combination weight is kind of heuristic, the combination weights produced by the Equation (5.8) can be treated as the *"a posterior"* probabilities of choosing state *i given* feature *j* with flat *a prior* probabilities of the HMM states, if the negative *"loss"* can be interpreted as the log likelihood of choosing feature stream *j* given state *i* [41].

Figure 5.2 illustrates the training algorithm that develops the loss-based combination weights.

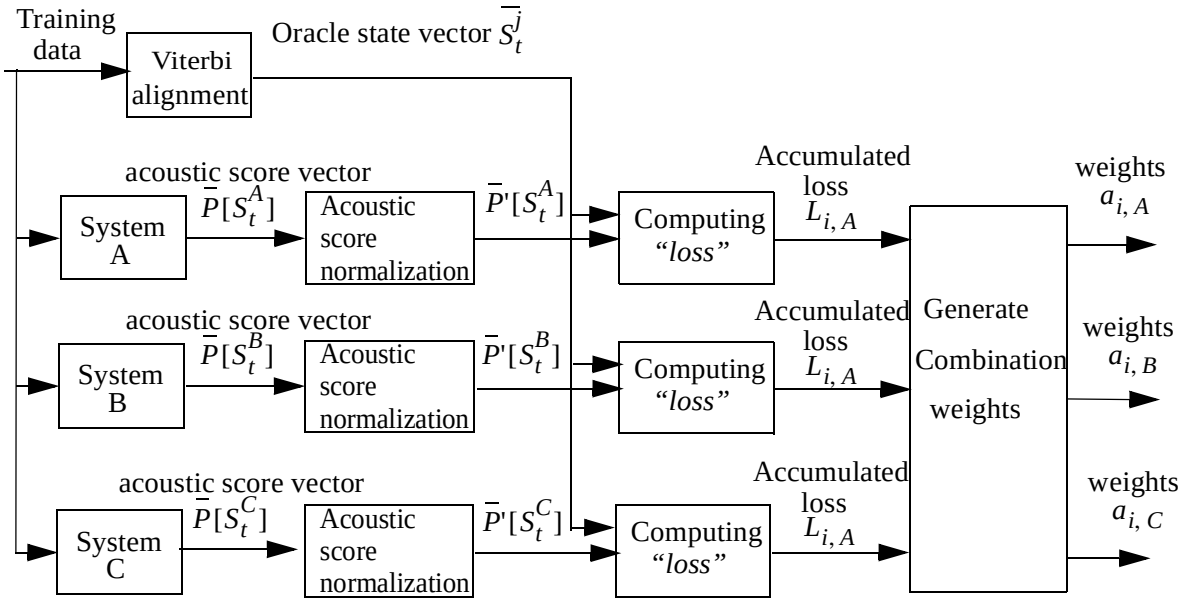

Figure 5.2 Diagram of generating loss-based combination weights.

As we already mentioned, there have already been many existing works in determine the combination weights. Here we will give some comparison of our approach with those existing methods. First of all, our approaches is similar to those works that generate the combination weights based on the recognition result of individual combination component (*e.g.* [6]). But rather than directly using the recognition accuracy and transform them into combination weights, we first compute the difference between the decoding result $\overline{P}[S_t^j]$ and the oracle probability vector $\overline{S}_t^j$, and then generate the combination weights based on these differences. As in the case of combination weighting algorithms based on neural networks (*e.g.* [6][25]), our algorithm requires the identity of the true recognition class (in the format of the oracle probability vector $\overline{S}_t^j$) to carry the training, but we are training the combination weights, which can be applied to different combination functions, rather than directly generating the updated probability as in MLP approaches (*e.g.* [6][25]). In addition, our "loss-based" weighting algorithm has more portability than those training based weighting algorithms. We can easily add some new combination components (*e.g.* new feature, new system) so long as we preserve the "loss" of the existing combination components. In adding new combination components, all we need to do is to compute the "loss" for each addition new combination component, then use Equation (5.8) to generate a new set of combination weights. While in those training based weighting algorithms as the MLP [6][25], MCE based training ([19][50][52][53][61]) or MMI based train-

ing ([34]), the combination weights need to be completely re-trained whenever there is a new combination component.

5.3.2 Releasing the constraint of sharing state-tying patterns across parallel systems

As described earlier, synchronous probability combination requires that parallel systems to share the same set of speech classes (*i.e*. tied states) in order to update their probabilities using updating functions as in Equation (5.1) through Equation (5.3). Most current speech recognition systems are HMM based with context-dependent acoustic modeling. We need to force these parallel systems to share the same state-tying pattern in order to combine their probabilities, which can degrade their individual performance and hence affect the final combination performance, as noted above. In this subsection we discuss a new algorithm that releases the constraint of sharing the state-tying patterns among parallel systems. As a result, the recognition accuracy of the individual systems will improve, which should improve the performance of the systems in combination.

Our proposed approach can be described as follows. We know that in most continuous speech recognition systems the final hypothesis, which is a word sequence, is the phoneme sequence that has the highest accumulated acoustical and language model scores. In generating such a phoneme sequence, the system carries out a search process, during which it determines the next possible states for each of the active states that are considered at each frame *t*, and it makes a transition into the one with the highest score. The system also tracks all the active states and their corresponding phoneme indices. In making the transition decision, the system has the two choices of either staying at the current state (*i.e.* making a self-transition), or transit to a new state which either represents a new position in the same phoneme[15] or the starting position of a new phoneme[16]. But no matter where the transition goes, the system must first obtain the corresponding scores associated with all the possible transitions, then transit to the particular state with the highest score. These scores are for the particular untied HMM states of the phoneme (*e.g.* the score of the second state of phoneme A, the score of the first state of the phoneme B, etc.). As most of the current systems use tied HMM state models and the score for each particular HMM state has been computed using the tied HMM

15. For example, if the current active state is the second state of Phoneme A, the system could transit to the third state of Phoneme A.
16. For example, if the current active state is the last state of Phoneme B, then the system could transit to the starting state of Phoneme C at the next time instance.

state model, they need to keep a mapping (or list) between the untied state of each particular phoneme and the index of the tied states. With this mapping, the system can easily find the index of the tied state corresponds to each particular untied state of the individual phoneme, and assign the score computed from the tied state model to the untied state of the individual phoneme. Figure 5.3 is an example in which the mappings between untied HMM states and tied HMM states in two systems are listed.

Label of the context-dependent phoneme (Basephone, left context, right context)	Index of the first state	Index of the second state	Index of the third state
AA, TD, R	156	165	174
AA, TH, N	160	163	175
AA, TH, R	160	163	175
AA, TS, N	162	166	174
AA, TS, R	162	166	172

System A

Label of the context-dependent phoneme (Basephone, left context, right context)	Index of the first state	Index of the second state	Index of the third state
AA, TD, R	156	167	174
AA, TH, N	161	165	171
AA, TH, R	161	165	173
AA, TS, N	161	165	174
AA, TS, R	161	165	174

System B

Figure 5.3 An example of the mapping between untied HMM states and tied HMM states of two systems (a and B) in the RM database. The HMM structure in each system is 3-state context-dependent. Each context-dependent phoneme is listed as “base phone, left context, right context”. The number in the table is the index of the tied state that corresponds to each untied state of the context-dependent phoneme.

Because of the different properties of different systems in tying untied HMM states, this mapping of tied states and untied states varies from system to system. In Figure 5.3, the state mappings in the two systems A and B are all different from each other. For example, in System A, the index of the tied state corre-

sponding to the second state of the phone “AA, TH, R” is 163, which is different from the corresponding index of 166 for the second state of phone “AA, TS, N”, as well as the corresponding tied state in System B which is 165. In combining HMM state probabilities, if we want to directly combine the probabilities of the tied states of different systems, we must force the parallel systems to share the same mapping (*i.e.* the same state tying pattern). In the example illustrated in Figure 5.3, we need to make the second state of phone “AA, TH, R” in System A share the same tied state as the second state of phone “AA, TS, N”. This will affect the performance of System A and consequently the combined performance of Systems A and B.

In addition to combining the tied HMM state probabilities, there is also the option of directly combining the probabilities of untied HMM states from parallel systems together. In this procedure, we first compute the probabilities of the tied HMM states from the individual systems, then update the probability associated with each untied HMM state based on the probability of the corresponding tied HMM state from the individual systems (using the mapping between tied and untied HMM state in the individual system). As a simple example in generating the combined probability of the second state of phoneme “AA, TH, R”, we can first compute the probability of the corresponding tied state 163 in System A as P(S163,A), then compute the probability of the corresponding tied state 165 in System B as P(S165, B), and combine it with the probability P(S163, A) to generate the final updated probability of the second HMM state of the phoneme “AA, TH, R”. We still use the combination functions described in Equation (5.1) to Equation (5.3) to generate the updated probability for each untied HMM state. Of course, the combination now must be processed synchronously in a frame-by-frame manner, in a sense that the parallel systems need to be synchronous with each other to provide their individual scores for updating the particular HMM state (untied) of every phoneme at each time frame. In basing the combination on the probabilities of untied HMM states rather than tied HMM states, the combination will be carried out at the search stage instead of the original likelihood computation stage[17]. In addition to the difference in the specific combination stage, we also need to keep the mapping between the untied tied HMM states from each individual system during combination, as we need this mapping to find the index of tied states corresponding to the particular untied state from individual system. This is a additional implementation requirement beyond that of combining tied HMM state probabilities, for which only one tied and untied HMM state mapping is necessary, as all

17. As we described earlier, the general decoding process contains several process. The first is the likelihood computation process, where the likelihood for each HMM states will computed. Then we perform a search, based on both the computed acoustic likelihoods for each HMM state and the language model, to generate the final decoding result.

the systems share the same mapping or state tying pattern. Figure 5.4 use the example listed in Figure 5.3 to illustrate (and compare) the idea of combining the tied HMM state probabilities with the direct combination of untied HMM state probabilities.

Updated probability	The first state	The second state	The third state
AA, TD, R	P(156,A) Comb P(156, B)	P(165, A) Comb P(167, B)	P(174, A) Comb P(174, B)
AA, TH, N	P(160, A) comb P(161, B)	P(163, A) Comb P(165, B)	P(175, A) Comb P(171, B)
AA, TH, R	P(160, A) Comb P(161, B)	P(163, A) Comb P(165, B)	P(175, A) Comb P(173, B)
AA, TS, N	P(162, A) Comb P(161, B)	P(166, A) Comb P(165, B)	P(174, A) Comb P(174, B)
AA, TS, R	P(162, A) Comb P(161, B)	P(166, A) Comb P(165, B)	P(172, A) Comb P(174, B)

Combination of untied HMM state probabilities

Updated probability	The first state	The second state	The third state
AA, TD, R	P(156,A) Comb P(156, B)	P(165, A) Comb P(167, B)	P(174, A) Comb P(174, B)
AA, TH, N	P(160, A) comb P(161, B)	P(163, A) Comb P(165, B)	P(175, A) Comb P(171, B)
AA, TH, R	P(160, A) Comb P(161, B)	P(163, A) Comb P(165, B)	P(175, A) Comb P(173, B)
AA, TS, N	P(162, A) Comb P(161, B)	P(166, A) Comb P(165, B)	P(174, A) Comb P(174, B)
AA, TS, R	P(162, A) Comb P(161, B)	P(166, A) Comb P(165, B)	P(172, A) Comb P(174, B)

Combination of tied HMM state probabilities by forcing System A to have the same state tying pattern as System B

Figure 5.4 An comparison of the combination of untied HMM state probabilities with the combination of tied HMM state probabilities. We use the example illustrated in Figure 5.3, and we assume that System A has been forced to share the same state tying pattern (*i.e.* the mapping between untied and tied HMM states) as System B. Each item in the table (*e.g.* P(172, A) Comb P(174, B) in the bottom right of the first table) is the combined probability. The first part of the item (*e.g.* P(172, A)) is the probability computed from System A, and the second is the one from System B. The numbers are the indices of the tied HMM states from individual systems.

5.4 .Evaluation of probability combination performance

To evaluate the performance of our new probability combination algorithms, experiments were carried out on several databases: The DARPA RM database, the TID database and the WSJ0 database. Evaluations using these corpora differed from one another both in terms of the parallel systems used for combination and the testing conditions. We carried out three separate experiments using the RM database, referred to as the RM_1, RM_2, and RM_3 experiments. For the RM_1 experiment, systems using MFCC and PLP features were combined together. In the RM_2 experiment, systems using two different LDA features were combined. The first LDA feature was designed to maximize the differences among all HMM states, while the second LDA feature was designed to discriminate the states belonging to the different broad phoneme classes of vowels, fricatives, silence, stops, and nasals. The RM_3 experiment also involved the combination of two different types of linear features. The first feature was an LDA feature developed to maximize the difference among all HMM states (as in Experiment RM_2), while the second feature was derived using PCA (which is also called Karhunen-Loeve Transformation (KLT)). This later choice was motivated by the fact that LDA and PCA are generally two of the best-performing conventional linear features. The experiments using the TID database followed a similar procedure to the corresponding lattice-combination experiments, with different noise conditions, 5-dB and 10-dB SNR, combining systems with standard MFCC and PLP features. In addition, experiments were run using the WSJ0 database, again combining LDA-based and PCA-based features as in RM_3. The WSJ0 database was used because it has a much larger vocabulary and more diverse testing conditions than the RM and TID databases, we want to get some feeling about performance of our algorithm on that condition as well.

For each database, word error rates obtained using synchronous probability combination and hypothesis fusion algorithms were compared. In the synchronous probability combination experiments we evaluated the accuracy obtained using both the tied HMM states as the location of combination (which require the parallel systems to share the same state-tying patterns) and using context-dependent phoneme states as the location of combination (which do not require the parallel systems to share the same state-tying patterns). In the former approach, all systems share the state-tying pattern from the best-performing individual system, which was MFCC in the RM_1, TID 5-dB and 10-dB experiments, and LDA in the RM_2, RM_3, and WSJ0 experiments. For each testing conditions, the combination functions used were *summation*, *maximization* and *multiplication* as in Equation (5.1) through Equation (5.3). The scores from parallel systems

were weighted both equally and unequally with combination weights. In the later case, the combination weights were obtained using the *loss-based* combination weights training algorithm described in Section 5.3.1 above. For hypothesis fusion, we compare the accuracy obtained with both the conventional *hypothesis combination* algorithm [69] and the proposed *lattice combination* algorithms described in the previous chapter of this thesis.

5.4.1 Summary of experimental results

Table 5.1 summarizes the performance in terms of word error rates obtained with various synchronous probability combination and hypothesis fusion schemes for all testing conditions. The synchronous probability combinations were performed using the tied HMM states as the combination class using both equal and unequal combination weights. (The state-tying patterns hence were shared across parallel combination systems.) We used two sets of unequal combination weights. The first was generated from our proposed *loss-based* combination weights training algorithm, and the other was generated from the entropy-based weighted combination algorithm as proposed by Ikbal *et al*. [32] for comparison. The hypothesis fusion was carried by combining the hypotheses (in the form of both the hypothesis in hypothesis combination, and the lattice in lattice combination) from the parallel systems together. These parallel systems were trained in a manner at which their HMM state-tying patterns were imported from the best performing individual system (*i.e.* MFCC on the RM_1, TID 5dB and TID 10dB setups, LDA on the RM_2, RM_3, and WSJ0 setups) instead of generated by themselves based on the maximum likelihood criterion.

We also tested the performance of our proposed synchronous probability combination algorithms using context-dependent phoneme HMM states as the entity to be combined, and the experimental results are listed in Table 5.2. The parallel systems were trained by themselves independently from each other without the constraint of sharing the same state-tying patterns. As had been done in the direct combination of tied HMM states, both equally and unequally weighted probabilities of the parallel systems were considered during the combination. In the latter situation, the combination weights were generated from our loss-based algorithm and the entropy-based algorithm by Ikbal *et al*. [32]. We list the corresponding performance of hypothesis combination algorithms using hypothesis combination and lattice combination on these parallel systems. The results in Table 5.2 can be compared to the results in Table 5.1.

The results in Table 5.1 and Table 5.2 indicate that weighted synchronous probability combination with

WERs (%)	RM_1	RM_2	RM_3	TID 5dB	TID 10dB	WSJ0
Feature 1	10.29	8.36	8.36	25.54	12.53	9.32
Feature 2	11.52	9.98	9.11	26.07	13.84	10.26
Hyp-Comb (Max)	10.45	8.90	7.86	24.47	12.12	8.26
Lat-Comb (Max)	9.84	8.21	7.51	23.94	11.64	8.17
Prob-Comb (Max)	9.10	7.66	7.27	23.33	10.73	8.07
Prob-Comb (MaxW)	8.66	7.48	6.95	22.72	10.47	7.71
Prob-comb (MaxW entropy)	9.13	7.69	7.17	23.80	10.93	7.75
Prob-Comb (Sum)	9.18	7.66	7.30	22.36	10.60	7.99
Prob-Comb (SumW)	8.73	7.32	7.02	21.61	10.28	7.82
Prob-Comb(SumW entropy)	9.07	7.60	7.27	21.99	10.60	7.94
Prob-Comb (Prod)	10.45	8.16	9.97	22.39	10.73	8.79

Table 5.1. Recognition accuracy of various hypothesis fusion and probability combination schemes for various experiments using the RM, TID and WSJ0 databases. Hyp, Lat, and Prob refer to hypothesis combination, lattice combination and synchronous probability combination, respectively. Max, Sum, Prod, MaxW, MaxW entropy, SumW, and SumW entropy refer to equal-weighted maximization, equal-weighted summation, multiplication, weighted maximization with loss-based training, weighted maximization with entropy based weights, weighted summation with loss-based training, and weighted summation with entropy based weights as the combination functions, respectively. Results were obtained by direct combining tied states of HMMs that share the same state tying patterns.

weights generated by our training algorithm outperforms conventional unweighted probability combination for all experiments, and most of the improvements were statistically significant. This clearly suggests the benefit of considering the difference in contribution from parallel features in combination by assigning different combination weights to the parallel system, and it also validate the effectiveness of our proposed *loss-based* combination weights algorithm. This is as expected, since unweighted probability combination can be treated as a special case of weighted probability combination. Weighted probability combination should always result in higher accuracy (or at least the same accuracy) compared to unweighted combination if the combination weights are assigned properly.

The comparison of our *loss-based* algorithm and the entropy-based combination weights algorithm proposed by Ikbal *et al*. [32] is also worthy of mention. As seen in Table 5.1 and Table 5.2, our *loss-based*

WERs (%)	RM_1	RM_2	RM_3	TID 5dB	TID 10dB	WSJ0
Feature 1	10.29	8.36	8.36	25.54	12.53	9.32
Feature 2	11.49	9.58	8.96	26.60	13.17	9.81
Hyp-Comb (Max)	9.68	7.98	7.54	25.62	11.93	8.07
Lat-Comb (Max)	9.12	7.62	7.21	24.76	11.32	7.91
Prob-Comb (Max)	8.09	7.37	7.06	23.96	11.32	7.72
Prob-Comb (MaxW)	7.72	7.16	6.88	22.96	10.99	7.44
Prob-Comb (MaxW entropy)	7.93	7.28	6.93	23.17	10.84	7.58
Prob-Comb (Sum)	8.12	7.15	6.90	23.16	11.24	7.75
Prob-Comb (SumW)	7.83	6.94	6.72	22.59	10.63	7.58
Prob-Comb (SumW entropy)	7.93	6.98	6.82	22.86	10.72	7.63
Prob-Comb (Prod)	10.12	7.42	7.91	22.46	10.49	9.22

Table 5.2. Same as Table 5.1, except that results were obtained by combining context-dependent phoneme untied HMM states (rather than tied states of individual recognition systems with the same HMM structure).

combination weights generally results in better combined WER than entropy-based combination weights. In most cases, however, the differences in results between the two algorithms were not statistically significant at the 5% level using the matched pairs method. While we do not know why this is the case, a possible reason could be the relatively small number of testing sentences used.

We also note in comparisons of results in Table 5.2 and Table 5.1 the performance gain obtained by releasing the constraint of sharing the same state-tying patterns among parallel combination systems, observed from the greatly improved recognition performance, especially in the combined system. For the same combination function, the performance obtained using the context-dependent phoneme state for combination (*i.e.* the results in Table 5.2) was consistently better than the performance obtained when parallel systems were forced to share the same model structure in directly combining their tied states together (as in the results in Table 5.1). For example, with the same the *multiplication* function (*i.e.* Prob-Comb), the WER of probability combination using context dependent phoneme state was 7.91 in the RM_3 setup, which was a 20% relative improvement from the WER obtained by forcing parallel systems to share the

same model structure during combination (*i.e.* the WER of 9.97 in Table 5.1). The improvements of combining the untied context-dependent phoneme HMM state over combination using conventional tied HMM states were also statistical significant using the matched pairs method. Indeed, from a baseline system that performs unweighted synchronous combination of tied HMM states, the improvement obtained by changing from combining tied HMM states to combining context-dependent untied phonemic HMM states was greater than the gain obtained by assigning combination weights to parallel systems. In the RM_1 experiments, changing the combination from tied states to context-dependent phonemic states reduces the relative WER by 11.5% (*e.g.* comparing WER of 8.12 using summation combination in Table 5.2 with WER of 9.18 using the same summation combination in Table 5.1), while assigning combination weights reduces the relative WER by 5% (*e.g.* comparing WER of 8.73 with weighted summation combination with WER of 9.18 with summation combination in Table 5.1). In addition, the improvements provided by these two approaches are at some extent independent, meaning that we can apply these two processes sequentially to provide further improvement. For the RM_1 experiments, applying the two approaches sequentially reduced the WER by about 14.7% compared to the baseline (*i.e.* comparing WER of 7.83 achieved by weighted combination of untied states of context-dependent phonemes in Table 5.2 with the WER of 9.18 obtained from equal combination of tied HMM states in Table 5.1). Given that the relative improvements observed when applying these two approaches individually are 11.5% and 5%, this 14.7% relative improvement is very close to the improvement to be expected if the effects of two approaches are statistically independent from each other, in which case the relative improvement would be: $1 - (1 - 0.05)(1 - 0.115) = 15.9\%$.

5.4.2 The impact of combination type on combined performance

In general, the WERs provided by the various probability combination schemes considered were consistently better than the corresponding WERs obtained using hypothesis combination algorithms, and these the differences were statistically significant for the most part. This is especially obvious in comparisons between *lattice combination* and *probability combination* using *summation* as the combination function. Although both methods use the same combination function, the WERs obtained from probability combination were consistently better than that obtained from the lattice combination for most experiments, and the differences were statistically significant.

We believe that the better representation of the search space in probability combination might be one of the major reasons for the superior performance of probability combination over hypothesis fusion. Since each word sequence can be represented by its corresponding HMM state sequences, using HMM states as the speech attribute to be combined enables us to represent all possible word sequences that will be considered in the decoding and search process, and to update their acoustic scores accordingly. While in hypothesis fusion, many word sequences have already been pruned out before their acoustic scores could be updated. The ability of generating the updated scores for all the possible word sequences during combination is an advantage of probability combination over the hypothesis fusion approach.

The recognition results also revealed dramatic differences in the effect of the various combination functions. The *maximization* function achieved the best combined performance for the RM_1 and WSJ0 experiments, while *summation* was best in the RM_2 and RM_3 experiments. Although *multiplication* was the worst performing combination function in the RM experiments, it became the best performing combination function in the TID 5-dB and 10-dB experiments. We believe that the specific form of the combination function which generates the best combined performance could depend on many different factors. The intrinsic nature of the parallel features, the specific combination stage at which information from parallel systems is combined together, and the particular testing task all can affect the choice of the best combination function.

5.4.3 The effect of parallel features on combined performance

As in the case of our observations in the previous chapter of lattice combination, different sets of parallel features have different effects on the performance of the combined system in the probability combination. Comparing word error rates obtained using *summation* with equal weights as the combination function for the RM_1, RM_2, and RM_3 experiments, the relative improvement of the combined system over the best single system was about 19% in RM_1 experiments, 15% in the RM_2 experiments, and 12% in the RM_3 experiments, all different from each other. For the *maximization* combination function, the variability in relative improvement of the combined system over the best single system in RM experiments was even more dramatic, as 21%, 11%, and 15% respectively. This clearly indicates the potential importance of the parallel features on the combined performance. We specifically address the issue of designing parallel features for optimizing the combined performance in the next chapter.

6. Generating Linear Features Based On The Maximum Normalized Acoustic Likelihood

6.1 Introduction

As seen in the experimental results in the previous two chapters, the performance obtained when combining speech recognition systems depends as critically on the features that are used as it does on the combination method. In both our experiments using hypothesis combination and probability combination described in Chapters 4 and 5, different sets of parallel features produced different results when used in combination, even when combined together using the same combination function and at the same stage in the recognition process. This clearly suggests the important role of the parallel features themselves on the performance of the combined systems.

But so far in most of the combined systems, the parallel features used were developed in some manner that was not directly related to the method of combination. Conventional parallel features were either pre-selected from some commonly-used single-stream features, or they were generated based on some simple criterion which may not be directly related to the combined performance. The most commonly-used single stream features for combination were variants on the classic MFCC, PLP, PCA, and LDA features. Some more recent approaches include the *heterogeneous features* proposed by Halberstadt ([21][22]), in which the complete set of the phoneme classes is divided into several subsets, and parallel features are developed from each subset; and the *Multi-stream features* by Bourlard (*e.g.* [6]) and Hermansky (*e.g.* [25]), in which the entire spectrum is separated into several subbands to extract parallel features. Some of the existing work on parallel feature can also be treated as a combination of the above two approaches, such as the work by Ellis and Bilmes [13] in which the mutual information criterion is used to select a set of single stream of features for combination.

Despite the success of the above methods, further improvements are likely to be possible, as few of the traditional approaches directly target improving the performance of the combined system. Some methods pick the conventional best-performing single-stream features for combination (*e.g.* [13][45]), but the use of the best single-stream feature does not also guarantee the best performance when features are combined together. Other methods generate parallel features based on some optimally criterion (*e.g.* [13]), but that criterion lacks a direct relation to the performance of the combined system. Some of the conventional

methods also require human phonetic knowledge (such as Halberstadt's heterogeneous features [21][22]),which may not transfer easily to other languages with different phonetic structures. In addition, for most of the conventional parallel feature generation algorithms, the manner in which the parallel features are generated is mostly independent from the manner in which they are combined (considering both the stage of combination and the combination function used), even though the performance of a combined system typically depends on the interaction between the parallel feature streams and the combination method used, as different combination function may require different sets of "optimal" parallel features for optimal combination performance.

Based on the above considerations, we propose a new parallel feature generation algorithm that directly targets the performance of the combined system. Compared with conventional parallel feature generating algorithms, our new algorithm will have the following properties: First, the parallel features will be generated in a manner that is directly related to the performance of the combination system. Specifically, we will use an objective function that directly relate to the Bayes classification accuracy of the combined system in the process of generating our parallel features. The generated parallel features will optimize this objective function and consequently maximize the combination performance. Second, the manner in which these parallel features are generated will be directly related to the specific combination stage and the combination function used. Our algorithm will generate different sets of parallel features for the different combination functions. This will ensure that our parallel features are specifically tuned for each combination function to achieve the maximum possible combined benefit for that function. Another property of our algorithm is that it can also be simplified into generating single-stream features that maximize the performance of an individual recognition system. As we will explain in more detail in the following sections, the process of generating parallel stream features and single-stream features can be united together under one single framework in our algorithm.

We describe the details of our algorithm in the following sections. In Section 6.2 we state the assumptions and simplifications we make in our parallel feature generation algorithm. One such assumption is the choice of the combination stage at which we attempt to optimize the combined performance. The second is the specific from of the objective that will be used to guide the parallel feature generation process. We will also discuss the format of the features that will be generated from our algorithm. As there are many possible ways of extracting information from the speech signal, we need to put some constraints on the specific

format of our features in order to carry out the analysis. Section 6.3 contains a detailed description of the process of generating parallel features to optimize the performance of the combined system. In Section 6.4 we discuss the simplified situation of generating single-stream features using our algorithm, illustrating how the processes of generating parallel features and single-stream features can be united together in our algorithm. Finally, we present our experimental results and analyses in Section 6.5.

6.2 Designing parallel features for improved combined performance

Before describing our new parallel feature generation algorithm in detail, we first must discuss some of the assumptions made in its development. We will talk about the specific stage at which we wish to optimize the combined performance using parallel features, the specific performance measure for the combined system that we will use in generating our parallel features, and the format of the parallel features generated from our algorithm.

6.2.1 Optimizing probability combination performance

We wish to obtain parallel features that optimize the performance of the combined system, but what specifically does this mean? As we have discussed, there are three major stages at which parallel features can be combined together: the feature-extraction stage at the input, the probability-evaluation stage in the middle of the decoding process and the hypothesizing stage at the output. The combination processes at each of these three stages are all different from one another: feature vectors are concatenated together in feature combination, the probabilities are merged together in probability combination, while hypotheses are fused in hypothesis combination. In addition, these three stages of combination may have different properties and impose different requirements on the optimal parallel features needed to obtain the best combined performance. It would be necessary and beneficial if we can specify the stage at which we want to optimize the combined performance before generating our parallel features.

For this algorithm we choose to generate parallel features that maximize the combined performance at the probability combination stage in the decoding process. More specifically, we seek to obtain the best synchronous probability combination performance using our parallel features. As a simple example, consider in Figure 6.1 two sets of parallel features to be combined as Feature Set A and Feature Set B. Each set contains two parallel feature streams. Set A is a set of conventional parallel features (*e.g.* MFCC and PLP

features), while Set B contains the parallel features generated from our algorithm. For each set of parallel features, we can combine them at the feature level, the probability level, and the hypothesis level as indicated in the figure. Our goal is to make the probability combination performance of parallel Feature Set B better than the probability combination performance of parallel Feature Set A. We don't care whether feature combination performance of Set B is better than that of Set A, or whether the hypothesis combination performance of Set B is better than that of Set A. We just want to optimize the probability combination performance using our parallel features, without regard to the performance of the other two combination schemes.

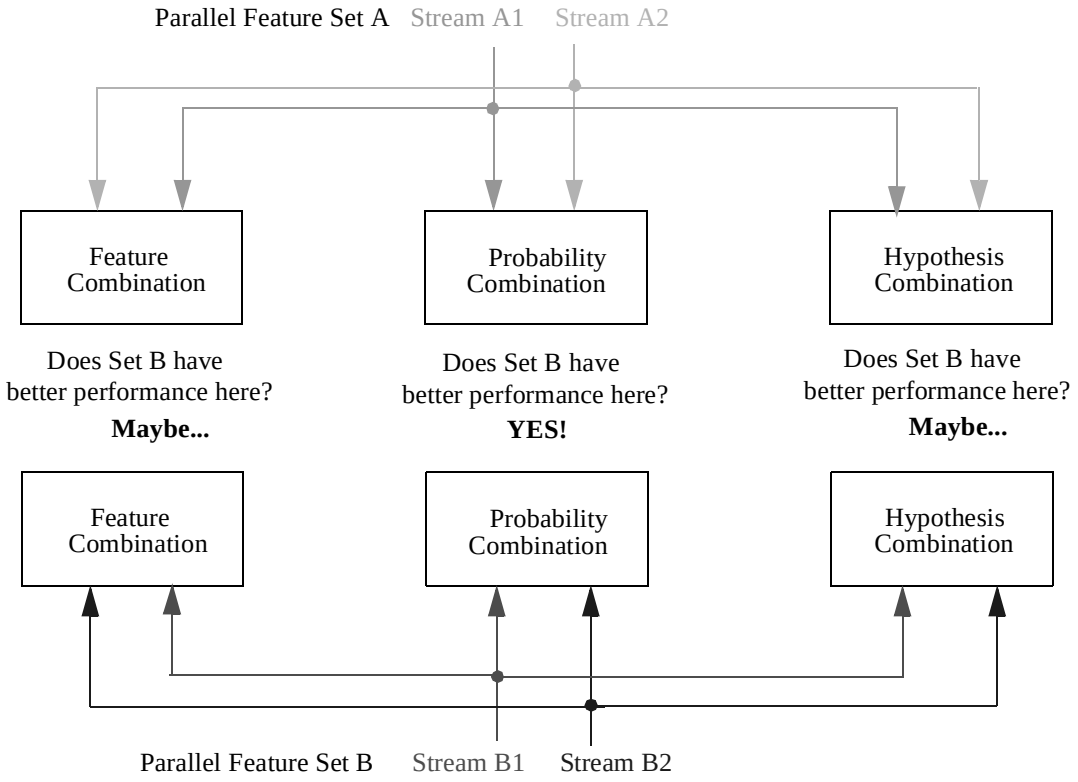

Figure 6.1 We strive to generate parallel features that optimize probability combination performance. Parallel Feature Set A is a conventional parallel feature set, while Set B represents the parallel features generated from our algorithm. Our goal is for Set B to have better probability combination performance than Set A.

There are two reasons why we wish to optimize probability combination performance using our parallel features. The first is simply that probability combination provides better recognition accuracy than combination at the other two stages. This is exemplified by the results in Table 6.1 which compare the WERs obtained by combined systems in which information is combined at the various different stages of the recognition process, and using different methods. The RM_1 database with MFCC and PLP features was used in all cases. As can be seen in Table 6.1, probability combination provides the lowest overall WER, which

is why we use that combination stage. We are also interested in using probability combination because it provides a great deal of analytical simplification. As will be elaborated in the following sections, the objective function used in generating our parallel features, which is called *"normalized acoustic likelihood"*, will be developed explicitly from the combined probability term at the probability combination stage. Focusing on the probability combination performance of the new parallel features will simplify the process.

MFCC (13)	PLP (13)	Feat Comb (20)	Feat Comb + PCA (13)	Hyp Comb (13)	Lat Comb (13)	Prob Comb Sum (13)	Prob Comb Max (13)	Prob Comb Prod (13)
10.29	11.49	10.62	8.54	9.68	8.91	7.83	7.72	10.12

Table 6.1. Comparison of WER for various combination schemes using RM data. MFCC and PLP features were used. The numbers in the parentheses in the headings represent the dimensionality of each individual feature. "Feat Comb" represents the concatenation of MFCC and PLP feature vectors. "Feat Comb + PCA" represents the concatenation of MFCC and PLP feature vector followed by a PCA transformation that reduces the dimensionality from 20 into 13. "Hypo Comb" represent the hypothesis combination result, and "Lat Comb" represents the lattice combination result. "Prob Comb Sum," "Prob Comb Max", and "Prob Comb Prod" represent the probability combination result using summation, maximization, and multiplication functions, respectively. The best combination performance is achieved from probability combination.

6.2.2 The normalized acoustic likelihood as the objective function for generating parallel features

We seek to generate parallel features that maximize probability combination performance. But what exactly does this mean? We know that the performance measure of a speech recognition system is the word error rate (WER), which is a summation of three kinds of errors made by the system: deletion, insertion, and substitution errors. Ideally we would use WER as the objective function in generating our parallel features, as it is the exact measure we use to judge the performance of a recognition system. But in practice we must choose some measure other than WER as the objective in generating our parallel features simply because a full decoding operation including all acoustic and language models involves too many other irrelevant components. For example, to the extent that WER reflects the effect of language models, it will not directly reflect the contribution of the features used. We prefer to use a performance measure that can be directly related to the acoustic features.

Since most of the current speech recognition systems are statistical systems in which recognition decisions are made based on the Bayes criterion by choosing the recognition class (such as the word sequence) with the highest posterior probability as in Equation (6.1)

$$\hat{C} = Argmax P(C|X), \tag{6.1}$$

where C is the recognition class (*e.g*. word, syllable, phoneme and HMM state) and X is the acoustic observation vector (*i.e*. the features), we choose to use the posterior probability of the true recognition class as a replacement for the WER measure as the objective function in generating our parallel features. In other words, we generate parallel features to maximize the *a posteriori* probability of the true recognition class of the (combined) recognition system instead of directly minimizing the word error rate. The two objectives of WER and posterior probability of the true recognition class are similar to each other, but using the latter as the performance measure is much more convenient for our analysis of acoustic features.

The definition of the recognition class C is another issue that need to be solved. Unlike other pattern classification systems, the recognition class for a speech recognition system could be defined at many different levels, such as HMM states, phonemes, syllables and words. We need to specify at which level we define our recognition class. In our feature generation process, we choose the HMM state as the recognition class because of the following two considerations. First, the HMM state is the most fundamental unit of modeling in speech recognition. Since most of the acoustic model parameters (such as the acoustic observation probabilities and state transition probabilities) are associated with the HMM state, use of the HMM state as the recognition class will expedite the analysis. The second reason, which is even more important, is the fact that our parallel features are designed to optimize the probability combination performance, in which specifically the probabilities of HMM states will be combined together. Using HMM states as the recognition class enables us to target directly the improvement of probability combination performance in generating our parallel features.

Nevertheless, using HMM states as the recognition class introduces its own problems. Unlike some of the other recognition classes (such as words), the true identity of the HMM state for training data is unknown. We need to first find out the identities of these true recognition classes in the training data before we can generate parallel features that optimize posterior probabilities of the true classes. We accomplish this in our algorithm by performing Viterbi alignment [75] on the training data, and then taking the most

likely HMM states generated from the alignment as the identity of the "true" HMM state recognition class. Henceforth we will refer to the most likely HMM states as identified by Viterbi alignment to the correct transcript as the "true" recognition class in each frame *t.*

If we assume the identity of the "true" HMM state in each frame *t* as $C^h{}_t$, then we can express the posterior probability of the true state in frame *t* as

$$P(C^h{}_t \mid X_t) = \frac{P(X_t \mid C^h{}_t) \cdot P(C^h{}_t)}{P(X_t)} = \frac{P(X_t \mid C^h{}_t) \cdot P(C^h{}_t)}{\sum_j P(X_t \mid C_j) \cdot P(C_j)}, \tag{6.2}$$

The above posterior probability is defined only for one frame *t.* We can accumulate this posterior probability expression across the frame *t* and write the accumulated posterior probability term P_c' as

$$P_c' = \prod_t P(C^h{}_t \mid X_t) = \prod_t \frac{P(X_t \mid C^h{}_t) \cdot P(C^h{}_t)}{P(X_t)} = \prod_t \frac{P(X_t \mid C^h{}_t) \cdot P(C^h{}_t)}{\sum_j P(X_t \mid C_j) \cdot P(C_j)} \tag{6.3}$$

The accumulated posterior probability P_c' in Equation (6.3) has the prior probability term $P(C_j)$ for each HMM state C_j. Since we don't know specifically how should we assign the prior probabilities to different HMM states[18], we make the simplification that the prior probabilities of all the HMM states are equal. This simplifies the accumulated posterior probability P_c' into a term we refer to as "*the accumulated normalized acoustic likelihood*" P_c as in Equation (6.4). We will use "accumulated normalized acoustic likelihood" as the objective function in generating our parallel features.

$$P_c = \prod_t \frac{P(X_t \mid C^h{}_t)}{\sum_j P(X_t \mid C_j)} \tag{6.4}$$

18. Although we could estimate the prior probabilities of HMM states by counting the relative frequencies of occurrence of each HMM state in the training data, we still assume that all prior probabilities are equal for simplicity.

6.2.3 Generating parallel features through linear transformation

Since there are many possible ways to extract acoustic features from the speech wavefile, we need to determine the specific form of our parallel features before we actually generate them to maximize their combination performance. We choose to generate linear features as the parallel features in our algorithm, which are indeed linear transformations from some original speech representation (such as the log magnitude spectrum of speech). Linear features are chosen for reasons of simplicity, as a linear feature stream can be completely characterized by a transformation matrix. In addition, linear features have been shown to provide superior performance. With the exception of PLP features, most of the commonly-used speech features that provide good recognition accuracy are linear transformations from log-spectral features, such as MFCC, PCA/KLT, LDA, and HLDA features.

If we use the symbol X to represent the original representation of the speech signal (such as the log magnitude spectrum vector), and A as the transformation matrix, then the transformed feature is AX. We will use these symbols throughout this chapter.

6.3 Generating parallel features for better probability combination performance: optimizing the normalized acoustic likelihood of the combined system

6.3.1 Parallel feature generation as an optimization process

It is now clear from the previous section that we wish to generate parallel linear features that maximize the accumulated normalized acoustic likelihood of the "true" HMM state in the combined system. In this section we will describe in details how this can be accomplished. As we will see, the process of generating parallel features in our algorithm is actually an optimization process, whose objective function is the normalized acoustic likelihood of the "true" (*i.e.* the most likely) HMM state in the combined system, and the variables for the optimization are the parallel transformation matrices.

As an illustration of this concept, assume that we wish to generate two linear feature streams as transformations of two original feature representations X and Y with matrices A_x and A_y respectively[19].

19. The original feature representations X and Y can either be identical to one another or they can be two different feature sets. Our feature generation algorithm is a transformation algorithm, so it doesn't matter what the original representations are.

The acoustic likelihoods for HMM state C_j at frame *t* with these two original representations are $P(X_t|C_j)$ and $P(Y_t|C_j)$ respectively. These acoustic likelihoods depend on the specific value of the feature vectors *X* and *Y* in each frame *t* and the parameters of the model of observation probabilities of each recognition class C_j . In the training data, feature vectors *X* and *Y* are drawn from the original feature space before transformation, and the statistical parameters characterizing the observation probabilities for each recognition class C_j are also estimated from training data, based on a partition of the training data into specific classes. (These partitions can be in the form of "soft partitions" of the training data as in the Baum-Welch training approach, or "hard-decision" partitions of the training data as in Viterbi training.) Hence, the acoustic likelihoods $P(X_t|C_j)$ and $P(Y_t|C_j)$ are dependent only on the partition of the training data. Applying the transformations A_x and A_y to the original representations *X* and *Y* will generate new partitions of the training data in the transformed feature spaces. But if we assume that the partitions of the transformed data A_xX and A_yY are the same as the partitions in their original representations (*i.e. X* and *Y*), then the acoustic likelihoods in the transformed feature spaces $P(A_xX_t|A_xC_j)$ and $P(A_yY_t|A_yC_j)$ are functions only of the transformation matrices A_x and A_y . When we combine their acoustic likelihoods together using probability combination with a specific combination function *F* (such as summation, maximization, or multiplication), we obtain their combined probability $P(A_xX_t \vee A_yY_t|C_j)$ as

$$P(A_xX_t \vee A_yY_t|C_j) = F\{P(A_xX|A_xC_j), P(A_yY|A_yC_j)\} \tag{6.5}$$

The combined probability $P(A_xX_t \vee A_yY_t|C_j)$ is a function of the individual acoustic likelihoods $P(A_xX_t|A_xC_j)$ and $P(A_yY_t|A_yC_j)$. Since these two likelihoods are functions of the transformation matrices A_x and A_y respectively, the combined probability $P(A_xX_t \vee A_yY_t|C_j)$ is a function of transformation matrices A_x and A_y as well. The normalized acoustic likelihood P_c in the combined system at

the transformed feature space will then become:

$$P_c = \prod_t \frac{F\left\{P(A_x X_t | A_x C^h{}_t), P(A_y Y_t | A_y C^h{}_t)\right\}}{\sum F\{P(A_x X_t | A_x C_j), P(A_y Y_t | A_y C_j)\}}, \tag{6.6}$$

which is also a function of the transformation matrices A_x and A_y. The goal of our parallel feature generation algorithm is to find the transformation matrices A_x and A_y that maximize the normalized acoustic likelihood of the "true" HMM state of the combined system in the transformed feature space, which can be described as:

$$\{A_x, A_y\} = ArgmaxP_c = Argmax \prod_t \frac{F\left\{P(A_x X_t | A_x C^h{}_t), P(A_y Y_t | A_y C^h{}_t)\right\}}{\sum_j F\{P(A_x X_t | A_x C_j), P(A_y Y_t | A_y C_j)\}} \tag{6.7}$$

Clearly, the parallel feature generation process now becomes an optimization process, with normalized acoustic likelihood P_c as the objective function and the transformation matrices A_x and A_y as the variables. For computational simplicity, we optimize the log of the normalized acoustic likelihood $LogP_c$ instead of P_c as in Equation (6.8). The result will be the same because the log is a monotonic function.

$$\{A_x, A_y\} = ArgmaxLogP_c$$
$$= Argmax \sum_t \left\{ Log\left[F\left\{P(A_x X_t | A_x C^h{}_t), P(A_y Y_t | A_y C^h{}_t)\right\}\right] - Log\left[\sum_j F\{P(A_x X_t | C_j), P(A_y Y_t | C_j)\}\right]\right\} \tag{6.8}$$

6.3.2 Finding parallel transformation matrices that maximize the normalized acoustic likelihood

We have seen in the previous subsection that the process of generating parallel features for optimal probability combination performance is indeed an optimization process, with the normalized acoustic likelihood of the combined system as the objective function and the transformation matrices as the variables. We now develop the specific transformation matrices that optimize our objective function.

The process of finding the transformation matrices that optimize the combination performance depends on the specific form of the observation probability distribution for the HMM states. Since most of the current speech recognition systems use either a single Gaussian or a mixture of Gaussians as the probability distribution for the HMM states, we will describe how to find the optimal transformation matrices in each of these two situations.

6.3.2.1 Generating parallel transformation matrices with single Gaussian distributions as state observation probabilities

Consider again the previous example of generating two transformation matrices A_x and A_y with single Gaussian distributions as the models for the HMM state observation probabilities. Assume that the mean vectors and covariance matrices for each recognition class C_j in the original feature representation spaces (X and Y) are known. We will refer to these as $\mu_{j,x}$, $\mu_{j,y}$ and $\Sigma_{j,x}$, $\Sigma_{j,y}$ respectively. If we apply transformations to the original feature representations (X, Y) to generate the transformed features X' and Y', the new feature vectors will have means $\mu'_{j,x}$, $\mu'_{j,y}$, and covariance matrices $\Sigma'_{j,x}$, $\Sigma'_{j,y}$. These can be expressed as

$$\begin{aligned} X' &= A_x X \\ \mu'_{j,x} &= A_x \mu_{j,x} \\ \Sigma'_{j,x} &= A_x \Sigma_{j,x} A_x^T \\ Y' &= A_y Y \\ \mu'_{j,y} &= A_y \mu_{j,y} \\ \Sigma'_{j,y} &= A_y \Sigma_{j,y} A_y^T \end{aligned} \tag{6.9}$$

Using single Gaussian distribution as the HMM state observation probabilities, the acoustic likelihood of the individual streams $P(A_x X_t | A_x C_j)$ $P(A_y Y_t | A_y C_j)$ and their combined acoustic scores $P(A_x X_t \vee A_y Y_t | C_j)$ using combination function F in the transformed feature spaces X' and Y' then become:

$$P(A_x X_t \mid A_x C_j) = N(A_x X_t; A_x \mu_{j,x}, A_x \Sigma_{j,x} A_x^T) = \frac{\exp\left\{-\frac{1}{2}(A_x X_t - A_x \mu_{j,x})^T (A_x \Sigma_{j,x} A_x^T)^{-1} (A_x X_t - A_x \mu_{j,x})\right\}}{(2\pi)^{D/2} \left|A_x \Sigma_{j,x} A_x^T\right|^{1/2}}$$

$$P(A_y Y_t \mid A_y C_j) = N(A_y Y_t; A_y \mu_{j,y}, A_y \Sigma_{j,y} A_y^T) = \frac{\exp\left\{-\frac{1}{2}(A_y Y_t - A_y \mu_{j,y})^T (A_y \Sigma_{j,y} A_y^T)^{-1} (A_y Y_t - A_y \mu_{j,y})\right\}}{(2\pi)^{D/2} \left|A_y \Sigma_{j,y} A_y^T\right|^{1/2}} \tag{6.10}$$

$$\begin{aligned} P(A_x X_t \vee A_y Y_t \mid C_j) &= F\{P(A_x X_t \mid A_x C_j), P(A_y Y_t \mid A_y C_j)\} \\ &= F\left\{ \frac{\exp\left\{-\frac{1}{2}(A_x X_t - A_x \mu_{j,x})^T (A_x \Sigma_{j,x} A_x^T)^{-1} (A_x X_t - A_x \mu_{j,x})\right\}}{(2\pi)^{D/2} \left|A_x \Sigma_{j,x} A_x^T\right|^{1/2}}, \right. \\ &\qquad \left. \frac{\exp\left\{-\frac{1}{2}(A_y Y_t - A_y \mu_{j,y})^T (A_y \Sigma_{j,y} A_y^T)^{-1} (A_y Y_t - A_y \mu_{j,y})\right\}}{(2\pi)^{D/2} \left|A_y \Sigma_{j,y} A_y^T\right|^{1/2}} \right\} \end{aligned} \tag{6.11}$$

Incorporating the expression for the combined acoustic scores in Equation (6.11) into Equation (6.8), we obtain the specific expression for generating parallel linear transformations that optimize the probability combination performance under the assumption that single Gaussians are used as the HMM state observation probabilities. We skip the complete expression in this chapter and put it in the Appendix because of the space limitation.

To find the optimal transformation matrices that maximize the normalized acoustic likelihood P_c as in Equation (6.8) assuming single-Gaussian output distributions, we must compute the partial derivatives of the objective function P_c or $LogP_c$ with respect to the transformation matrices A_x and A_y. Then we use these partial derivatives to generate the transformation matrices either in closed form or through the use of iterative methods such as gradient ascent. The partial derivatives of $LogP_c$ with respect to the individual transformation matrices A_x and A_y can be formulated from the expression of $LogP_c$ as in Equation (6.6) as:

$$\frac{\partial LogP_c}{\partial A_x} = \sum_t \frac{\partial LogF\left\{P(A_xX_t|A_xC^h{}_t), P(A_yY_t|A_yC^h{}_t)\right\}}{\partial A_x} - \sum_t \frac{F\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\sum_k F\{P(A_xX_t|A_xC_k), P(A_yY_t|A_yC_k)\}} \cdot \frac{\partial LogF\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\partial A_x}$$

$$\frac{\partial LogP_c}{\partial A_y} = \sum_t \frac{\partial LogF\left\{P(A_xX_t|A_xC^h{}_t), P(A_yY_t|A_yC^h{}_t)\right\}}{\partial A_y} - \sum_t \frac{F\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\sum_k F\{P(A_xX_t|A_xC_k), P(A_yY_t|A_yC_k)\}} \cdot \frac{\partial LogF\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\partial A_y} \quad (6.12)$$

Equation (6.12) indicates that the partial derivatives of the $LogP_c$ can be treated as a weighted combination of the partial derivatives of the combined acoustic scores of each state C_j with respect to the individual transformation matrix (*e.g.* $\frac{\partial LogF\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\partial A_x}$). Since the combined acoustic score $F\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}$ is the output of the combination function *F* with the acoustic likelihood of each individual streams as input, its partial derivative is clearly dependent on the specific form of the combination function *F*. If *F* is a differentiable function as in the cases of summation or multiplication, we can simply use the chain rule to further decompose the partial derivative expressions (*e.g.* $\frac{\partial LogF\{P(A_xX_t|A_xC_j), P(A_yY_t|A_yC_j)\}}{\partial A_x}$) into the derivatives of each individual stream as in some of the examples given in Equation (6.13) (summation function) and Equation (6.14) (multiplication function):

$$
\begin{aligned}
&\frac{\partial Log\{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)\}}{\partial A_x} \\
&= \frac{P(A_x X_t|A_x C_j) \cdot \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_x} + P(A_y Y_t|A_y C_j) \cdot \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_x}}{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)} \\
&= \frac{P(A_x X_t|A_x C_j) \cdot \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_x}}{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)} \\
&\frac{\partial Log\{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)\}}{\partial A_y} \\
&= \frac{P(A_x X_t|A_x C_j) \cdot \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_y} + P(A_y Y_t|A_y C_j) \cdot \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_y}}{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)} \\
&= \frac{P(A_y Y_t|A_y C_j) \cdot \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_y}}{P(A_x X_t|A_x C_j) + P(A_y Y_t|A_y C_j)}
\end{aligned}
\tag{6.13}
$$

$$
\begin{aligned}
\frac{\partial Log\{P(A_x X_t|A_x C_j) \cdot P(A_y Y_t|A_y C_j)\}}{\partial A_x} &= \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_x} + \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_x} \\
&= \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_x} \\
\frac{\partial Log\{P(A_x X_t|A_x C_j) \cdot P(A_y Y_t|A_y C_j)\}}{\partial A_y} &= \frac{\partial LogP(A_x X_t|A_x C_j)}{\partial A_y} + \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_y} \\
&= \frac{\partial LogP(A_y Y_t|A_y C_j)}{\partial A_y}
\end{aligned}
\tag{6.14}
$$

When F is a non-differentiable function such as maximization, we must use differentiable approximations instead. One such example for maximization function is the use of the *P-norm* function in the limiting case as P approaches infinity to approximate the maximization function as in Equation (6.15):

$$
Max\{Z_1, Z_2, \ldots, Z_n\} = \lim_{p \to \infty} \left(\sum_{i=1}^{n} Z_i^p \right)^{1/p} \tag{6.15}
$$

No matter what's the specific form of the combination function F used, the partial derivative of the

combined acoustic likelihood with respect to the individual transformation matrix is always a weighted combination of the derivatives of the likelihood of individual acoustic stream with respect to its corresponding transformation matrix $\frac{\partial LogP(A_x X_t | A_x C_j)}{\partial A_x}$ $\frac{\partial LogP(A_y Y_t | A_y C_j)}{\partial A_y}$ as indicated in Equation (6.13) and Equation (6.14). Because the single Gaussian distribution is used for the HMM state observation probability modeling, we can easily obtain expressions for $\frac{\partial LogP(A_x X_t | A_x C_j)}{\partial A_x}$ $\frac{\partial LogP(A_y Y_t | A_y C_j)}{\partial A_y}$ as in Equation (6.16).

$$\begin{aligned}
\frac{\partial LogP(A_x X_t | A_x C_j)}{\partial A_x} &= -\frac{1}{2}\frac{\partial\left[(A_x X_t - A_x \mu_{j,x})^T (A_x \Sigma_{j,x} A_x^T)^{-1} (A_x X_t - A_x \mu_{j,x}) + Log\left|A_x \Sigma_{j,x} A_x^T\right|\right]}{\partial A_x} \\
\frac{\partial LogP(A_y Y_t | A_y C_j)}{\partial A_y} &= -\frac{1}{2}\frac{\partial\left[(A_y Y_t - A_y \mu_{j,y})^T (A_y \Sigma_{j,y} A_y^T)^{-1} (A_y Y_t - A_y \mu_{j,y}) + Log\left|A_y \Sigma_{j,y} A_y^T\right|\right]}{\partial A_y}
\end{aligned} \tag{6.16}$$

The detailed expressions for the derivatives in Equation (6.16) have been put in the Appendix for space saving purpose. With these expressions, we can obtain the derivatives of the combined acoustic scores with respect to the transformation matrices as in Equation (6.13) through Equation (6.15), depending on the specific form of the combination function *F*. We further accumulate the derivative of the normalized acoustic likelihood P_c or $LogP_c$ according to Equation (6.12).

Once we obtain the partial derivatives $\frac{\partial LogP_c}{\partial A_x}$ and $\frac{\partial LogP_c}{\partial A_y}$, we can either set them to zero to find a closed-form solution, or use the gradient approach to iteratively update the transformation matrices for the optimal solution. Since we currently cannot obtain a closed-form solution for the derivatives, we use the gradient ascent approach, in which we iteratively update the transformation matrices as in Equation (6.17):

$$\begin{aligned}
A_x^{i+1} &= A_x^i + \varepsilon_x \frac{\partial LogP_c}{\partial A_x^i} \\
A_y^{i+1} &= A_y^i + \varepsilon_y \frac{\partial LogP_c}{\partial A_y^i}
\end{aligned} \tag{6.17}$$

where A_x^i and A_x^{i+1} represent the transformation matrices at the current and next iteration respectively, ε_x is the step size for the updating, we will reduce it for every fixed number of iterations to guarantee the convergence of the iterative procedure. The choice of initial values for the transformation matrices is also a very important issue, and it will be discussed in more details later this chapter.

The complete process of generating parallel linear transformation features for optimal probability combination performance is illustrated in Figure 6.2, and the process of computing the derivative of the log-value of the normalized acoustic likelihood with single Gaussian distributions is described in Figure 6.3.

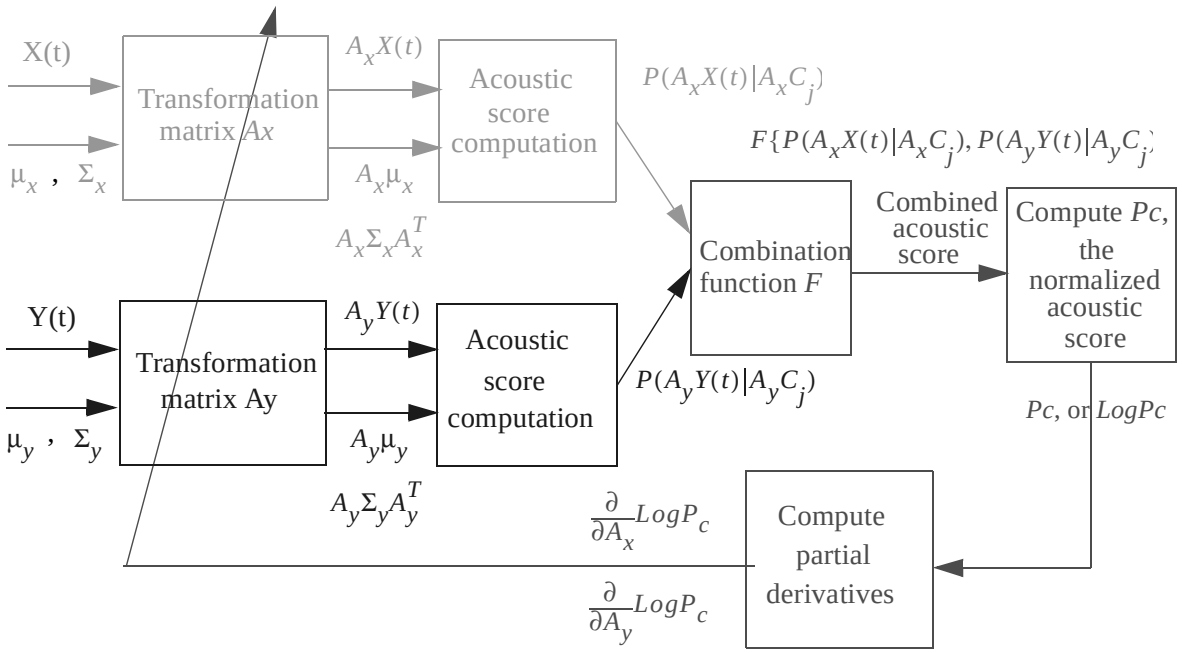

Figure 6.2 Diagram of generating parallel linear transformation matrices for optimal probability combination performance

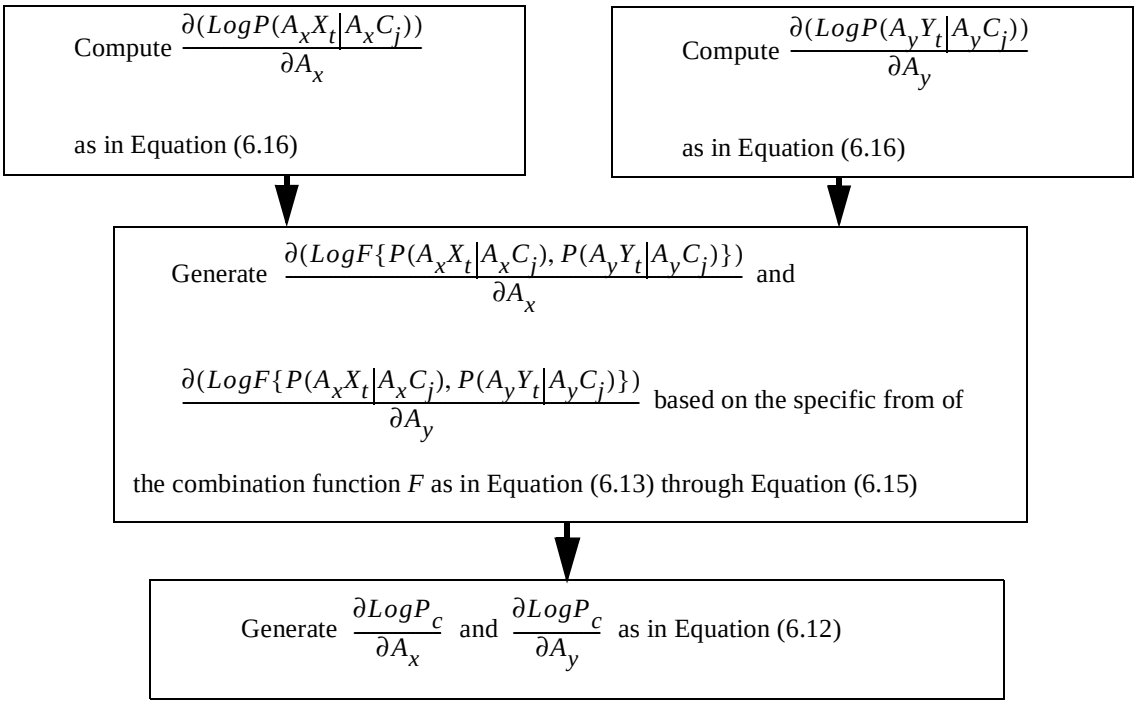

Figure 6.3 Diagram of computing the partial derivative of the normalized acoustic likelihood (log value) with respect to the transformation matrices. Assuming HMM states that are modeled by single Gaussian distributions.

6.3.2.2 Generating parallel transformation matrices with mixture of Gaussians distributions as HMM state observation probabilities

In many situations, speech recognition systems use mixtures of Gaussians as formulated in Equation (6.18) instead of a single Gaussian distribution as the HMM state observation probabilities. We describe in this subsection the process of generating parallel linear features for optimal probability combination performance when mixture of Gaussians distribution distributions are used for the HMM state observation probabilities.

$$P(X_t|C_j) = \sum_c w_c N(X_t;\mu_{j,c},\Sigma_{j,c}) = \sum_c w_c \frac{\exp\left\{-\frac{1}{2}(X_t-\mu_{j,c})^T\Sigma_{j,c}^{-1}(X_t-\mu_{j,c})\right\}}{(2\pi)^{D/2}|\Sigma_{j,c}|^{1/2}} \quad (6.18)$$

The process in general is very similar to that of generating optimal parallel features with single Gaussian distributions for the HMM state observation probabilities. As in the previous case, the partial derivative

of P_c or $LogP_c$ is computed with respect to the transformation matrices, which further depends on the derivatives of the combined likelihood for each recognition class (*i.e.* HMM state) C_j as in Equation (6.12), which in turn depends on the derivatives of the likelihoods of individual feature stream based on the specific form of the combination function.

The major difference between the processes of generating optimal parallel features using single Gaussian or mixtures of Gaussians as the output distributions is in the way that the derivatives of the acoustic likelihoods of individual feature streams are computed. Since the observation probabilities are now modeled as mixture of Gaussians, the expressions for the derivative of the individual likelihood $\frac{\partial LogP(A_x X_t | A_x C_j)}{\partial A_x}$ $\frac{\partial LogP(A_y Y_t | A_y C_j)}{\partial A_y}$ will change accordingly.

In developing these expressions, we begin with the expression of the transformed feature vector and model parameters of each HMM state C_j in Equation (6.19). The symbols in Equation (6.19) share the same meaning as those for single Gaussian distributions in Equation (6.9). Each HMM state C_j in Equation (6.19) now has multiple mean vectors, covariance matrices, and mixture coefficients rather than the single mean and covariance matrix in Equation (6.9). The parameters of each of the Gaussian components are indexed using the variable *c* in Equation (6.19), same as what we did in Equation (6.18).

$$\begin{aligned} X' &= A_x X \\ \mu'_{j,c,x} &= A_x \mu_{j,c,x} \\ \Sigma'_{j,c,x} &= A_x \Sigma_{j,c,x} A_x^T \\ w'_{c,x} &= w_{c,x} \end{aligned}$$
$$\begin{aligned} Y' &= A_y Y \\ \mu'_{j,c,y} &= A_y \mu_{j,c,y} \\ \Sigma'_{j,c,y} &= A_y \Sigma_{j,c,y} A_y^T \\ w'_{c,y} &= w_{c,y} \end{aligned} \qquad (6.19)$$

As can be seen in Equation (6.19), we assume that the mixture coefficients in the transformed feature space $w'_{c,x}$ are the same as in the original feature space $w_{c,x}$. This assumption enables us to formulate the model parameters in the transformed feature space as the result of the transformation as in Equation

(6.19).

With the transformed feature vector and model parameters as in Equation (6.19), the observation probabilities of state C_j in the transformed feature spaces can now be formulated as:

$$
\begin{aligned}
P(A_x X_t \left| A_x C_j \right.) &= \sum_c w'_{c,x} P(A_x X_t \left| A_x C_{j,c} \right.) \\
&= \sum_c w'_{c,x} \frac{\exp\left\{ -\frac{1}{2}(A_x X_t - A_x \mu_{j,c,x})^T (A_x \Sigma_{j,c,x} A_x^T)^{-1} (A_x X_t - A_x \mu_{j,c,x}) \right\}}{(2\pi)^{D/2} \left| A_x \Sigma_{j,c,x} A_x^T \right|^{1/2}} \\
P(A_y Y_t \left| A_y C_j \right.) &= \sum_c w'_{c,y} P(A_y Y_t \left| A_y C_{j,c} \right.) \\
&= \sum_c w'_{c,y} \frac{\exp\left\{ -\frac{1}{2}(A_y Y_t - A_y \mu_{j,c,y})^T (A_y \Sigma_{j,c,y} A_y^T)^{-1} (A_y Y_t - A_y \mu_{j,c,y}) \right\}}{(2\pi)^{D/2} \left| A_y \Sigma_{j,c,y} A_y^T \right|^{1/2}}
\end{aligned}
\quad , \qquad (6.20)
$$

where $P(A_x X_t \left| A_x C_{j,c} \right.)$ and $P(A_y Y_t \left| A_y C_{j,c} \right.)$ are the likelihoods of the Gaussian component *c* of the individual feature streams in the transformed feature space. The derivative expressions of the new likelihoods then become:

$$\frac{\partial LogP(A_x X_t | A_x C_j)}{\partial A_x} = \frac{\partial Log\left(\sum_c w'_{c,x} P(A_x X_t | A_x C_{j,c})\right)}{\partial A_x}$$

$$= \frac{\sum_c w'_{c,x} P(A_x X_t | A_x C_{j,c})}{\sum_k w'_{k,x} P(A_x X_t | A_x C_{j,k})} \cdot \frac{\partial LogP(A_x X_t | A_x C_{j,c})}{\partial A_x}$$

$$= \frac{\sum_c w'_{c,x} P(A_x X_t | A_x C_{j,c}) \cdot -\frac{1}{2} \cdot \frac{\partial\left[(A_x X_t - A_x \mu_{j,c,x})^T (A_x \Sigma_{j,c,x} A_x^T)^{-1} (A_x X_t - A_x \mu_{j,c,x}) + Log\left|A_x \Sigma_{j,c,x} A_x^T\right|\right]}{\partial A_x}}{\sum_k w'_{k,x} P(A_x X_t | A_x C_{j,k})}$$

$$\frac{\partial LogP(A_y Y_t | A_y C_j)}{\partial A_y} = \frac{\partial Log\left(\sum_c w'_{c,y} P(A_y Y_t | A_y C_{j,c})\right)}{\partial A_y}$$

$$= \frac{\sum_c w'_{c,y} P(A_y Y_t | A_y C_{j,c})}{\sum_k w'_{k,y} P(A_y Y_t | A_y C_{j,k})} \cdot \frac{\partial LogP(A_y Y_t | A_y C_{j,c})}{\partial A_y}$$

$$= \frac{\sum_c w'_{c,y} P(A_y Y_t | A_y C_{j,c}) \cdot -\frac{1}{2} \cdot \frac{\partial\left[(A_y Y_t - A_y \mu_{j,c,y})^T (A_y \Sigma_{c,j,y} A_y^T)^{-1} (A_y Y_t - A_y \mu_{j,c,y}) + Log\left|A_y \Sigma_{j,c,y} A_y^T\right|\right]}{\partial A_y}}{\sum_k w'_{k,y} P(A_y Y_t | A_y C_{j,k})} \quad (6.21)$$

As indicated by Equation (6.21), the derivatives of the acoustic likelihoods of individual feature streams using Gaussian mixtures as the state observation probabilities become a weighted summation of the derivatives of the individual Gaussian components. Hence, we must first compute the derivative of each Gaussian component within the mixture, then take a weighted summation of these derivatives to generate the derivative of the observation probability as in Equation (6.21).

The specific process of computing the partial derivatives of $LogP_c$ for Gaussian mixture distributions is illustrated in Figure 6.4, which is slightly different from the computation using a single Gaussian distribution illustrated in Figure 6.3. The process of generating parallel linear features with Gaussian mixtures for the state observation probabilities is the same as the process of combined feature generation using single Gaussian distributions as illustrated in Figure 6.2.

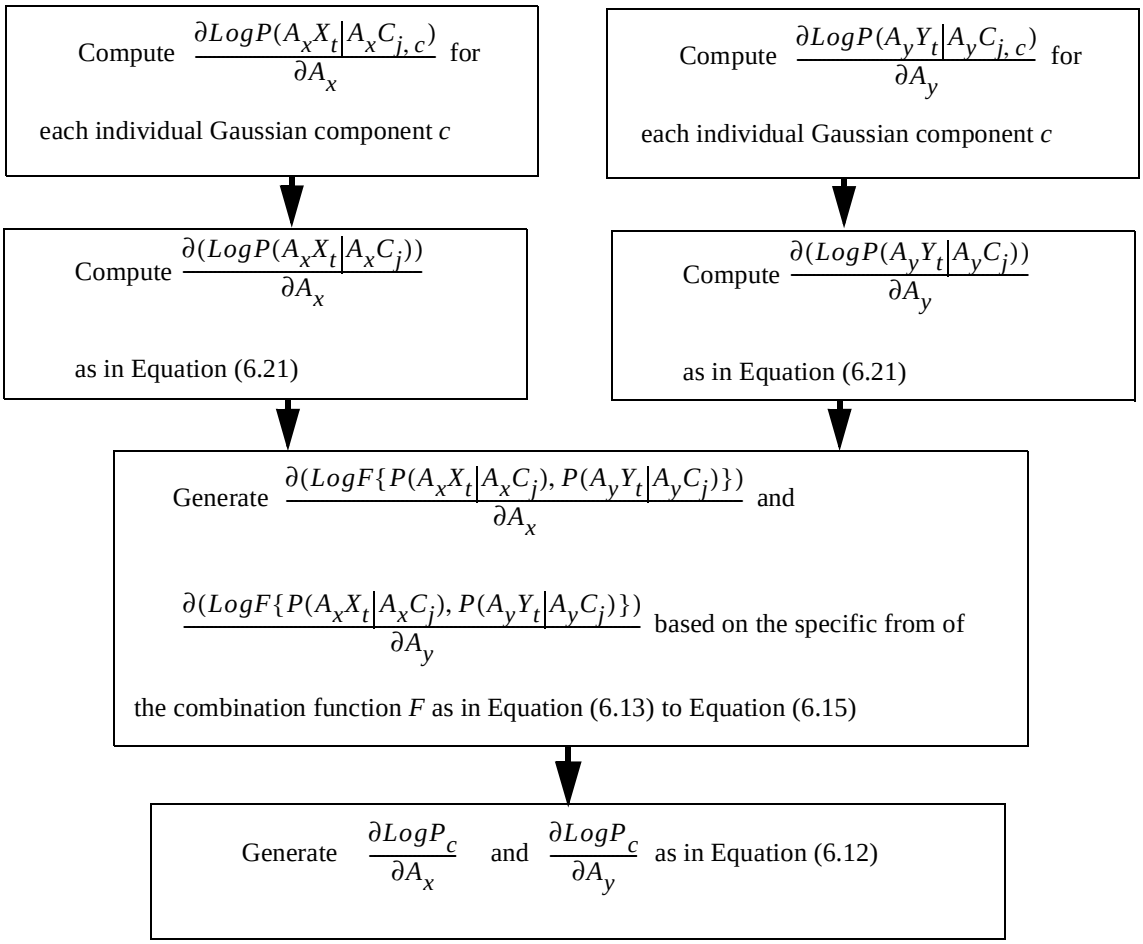

Figure 6.4 Computation of the partial derivative of the normalized acoustic likelihood (log value) with respect to the transformation matrices assuming HMM states that are modeled by Gaussian mixture distributions

6.4 Generating single-stream features for better individual recognition performance

Not only can our proposed algorithm generate parallel linear features for optimal performance when used in the context of probability combination, but it can also generate a single stream of linear feature that optimizes the individual recognition performance of a single system. In this section we will discuss the specific process of generating single-stream linear features for optimal individual recognition performance. We will show that the process of generating parallel linear features for optimal probability combination performance and single-stream features for optimal individual recognition performance using our algorithm can be united together under the single framework of optimizing the normalized acoustic likelihood of the true recognition class in the “system”. When this “system” actually refers to a combination of

individual recognizers, as in the previous section, parallel features with optimal combination performance can be generated. If we change the definition of the "system" to an individual recognition system, as will be described in this section, optimal single-stream features that maximize the individual recognition performance can be generated. From this point of view, the process of generating single-stream features is a simplification of the process of generating parallel linear features.

6.4.1 Generating single-stream linear features as an optimization process

In generating parallel feature streams that optimize probability combination performance, we exploited the normalized acoustic likelihood of the "true" recognition class in the combined system as an approximation of the combination performance measure, and we searched for the parallel transformation matrices that maximized this term. We approach the generation of optimal single-stream linear features in a similar fashion, seeking the transformation matrix which maximizes the normalized acoustic likelihoods of the "true" recognition classes in an individual system.

The specific process of generating optimal single-stream features can be described as follows: we adopt the same notation as in the previous section about the generation of parallel features to define the most-likely recognition class (*e.g.* HMM state) in each frame t as $C^h{}_t$, a single transformation matrix as A, and the original feature representation as X. We then represent the accumulated value of the normalized acoustic likelihood P_c in the transformed feature space AX as

$$P_c = \prod_t \frac{P(AX_t | AC^h{}_t)}{\sum_j P(AX_t | AC_j)} \tag{6.22}$$

Next, we search for the matrix A which maximizes the normalized acoustic likelihood term P_c to produce the optimal single-stream linear feature:

$$A = ArgmaxP_c = Argmax_A \left\{ \prod_t \frac{P(AX_t | AC_t^h)}{\sum_j P(AX_t | AC_j)} \right\} \tag{6.23}$$

where $P(AX_t | AC_j)$ is the acoustic likelihood of Class C_j. This could be computed assuming either single-

Gaussian output distributions as in Equation (6.24), or assuming Gaussian mixtures for the output distributions as in Equation (6.25):

$$P(AX_t|AC_j) = N(AX_t;A\mu_j, A\Sigma_j A^T) = \frac{\exp\left\{-\frac{1}{2}(AX_t - A\mu_j)^T (A\Sigma_j A^T)^{-1}(AX_t - A\mu_j)\right\}}{(2\pi)^{D/2}\left|A\Sigma_j A^T\right|^{1/2}} \tag{6.24}$$

$$P(AX_t|AC_j) = \sum_c w_{j,c} P(AX_t|AC_{j,c}) = \sum_c w_{j,c} \frac{\exp\left\{-\frac{1}{2}(AX_t - A\mu_{j,c})^T (A\Sigma_{j,c} A^T)^{-1}(AX_t - A\mu_{j,c})\right\}}{(2\pi)^{D/2}\left|A\Sigma_{j,c} A^T\right|^{1/2}} \tag{6.25}$$

The parameters μ_j and Σ_j in Equation (6.24) represent the Gaussian mean and covariance of state C_j respectively, while $\mu_{j,c}$, $\Sigma_{j,c}$ and $w_{j,c}$ in Equation (6.25) are the corresponding mean, covariance and mixture coefficient of Gaussian component c of state C_j. By simply changing the normalized acoustic likelihood from the combined system to an individual system, our algorithm is capable of generating single-stream features that optimize the recognition accuracy of an individual system.

Before describing the detailed process of generating our single stream linear feature based on the maximum normalized acoustic likelihood, we'd like to make a brief comparison of our new feature with some conventional linear feature generation schemes like PCA[11][33], LDA[11][29], HLDA[42][68], Maximum Representation and Discrimination Feature (MRDF)[70]. PCA seeks the transformation matrix which best represents the original data in the transformed feature space in the minimum mean square error sense. LDA seeks the transformation matrix which generates the maximum ratio of between class covariance determinant and within class covariance determinant. HLDA extends the LDA by incorporating multiple clusters into each class, while the MRDF method seeks a transformation that keeps the representation of each class and increases the separation among different classes at the same time. Through the optimization of an objective function defined as a mixture of the representation objective (*i.e.* the transformed covariance criterion as used in the PCA) and separation objective (which is the ratio between average inter-class data square distance and average within class covariance), MRDF generates its transformation matrix. In contrast to all of the above approaches, our single stream linear feature seeks the transformation matrix which directly maximizes the normalized acoustic likelihood of the true recognition class (*i.e.* the

most likely HMM state) in the transformed feature space.

6.4.2 Finding the single linear transformation that maximizes normalized acoustic likelihood

The process of finding a single-stream linear optimal transformation matrix that maximizes the normalized acoustic likelihood as in Equation (6.23) is very similar to the process of finding parallel transformation matrices that optimize the probability combination performance as in Equation (6.7) or Equation (6.8). We still need to first compute the derivative of the normalized acoustic likelihood term (or log value $LogP_c$) with respect to the transformation matrix A as in Equation (6.26):

$$\frac{d}{dA}LogP_c = \sum_t \frac{d}{dA}LogP(AX_t|AC^h{}_t) - \sum_t \frac{P(AX_t|AC_j)}{\sum_k P(AX_t|AC_k)} \frac{d}{dA}LogP(AX_t|AC_j) \ , \tag{6.26}$$

and then use the gradient ascent approach to iteratively update the transformation matrix as in Equation (6.27)

$$A^{i+1} = A^i + \varepsilon \frac{\partial LogP_c}{\partial A^i} \ , \tag{6.27}$$

where the meanings of the symbols in Equation (6.27) are the same as in Equation (6.17).

Since there is no combination function F or combined acoustic likelihood involved in single-stream feature generation, the derivative of the normalized acoustic likelihood $\frac{d}{dA}LogP_c$ can be accumulated directly from the derivative of the acoustic likelihood of individual feature stream $\frac{d}{dA}LogP(AX_t|AC_j)$. Depending on whether we use a single Gaussian or a mixture of Gaussians for the HMM state observation probabilities, the derivative of the acoustic likelihood of individual feature stream $\frac{d}{dA}LogP(AX_t|AC_j)$ can be formulated as:

$$\frac{d}{dA}LogP(AX_t|AC_j) = -\frac{1}{2} \cdot \frac{d}{dA}\left\{ (AX_t - A\mu_j)^T (A\Sigma_j A^T)^{-1} (AX_t - A\mu_j) + Log\left|A\Sigma_j A^T\right| \right\} \tag{6.28}$$

$$\frac{d}{dA}LogP(AX_t|AC_j) = \frac{\sum_c w_{j,c}P(AX_t|AC_{j,c}) \cdot \frac{d}{dA}LogP(AX_t|AC_{j,c})}{\sum_k w_{j,k}P(AX_t|AC_{j,k})}$$

$$= \frac{\sum_c w_{j,c}P(AX_t|AC_{j,c}) \cdot -\frac{1}{2} \cdot \frac{d}{dA}\left\{(AX_t - A\mu_{j,c})^T(A\Sigma_{j,c}A^T)^{-1}(AX_t - A\mu_{j,c}) + Log\left|A\Sigma_{j,c}A^T\right|\right\}}{\sum_k w_{j,k}P(AX_t|AC_{j,k})} \quad (6.29)$$

The specific expressions of the derivatives in Equation (6.28) and Equation (6.29) are the same as the partial derivative expressions in Equation (6.16) and Equation (6.21). We put them in the Appendix to save space here.

With the above equations for the likelihoods and their derivatives, the process of computing the derivative of the normalized acoustic likelihood in the single-stream feature generation can be characterized in Figure 6.5. The whole process of generating single-stream linear features that optimize the individual system recognition performance is illustrated in Figure 6.6.

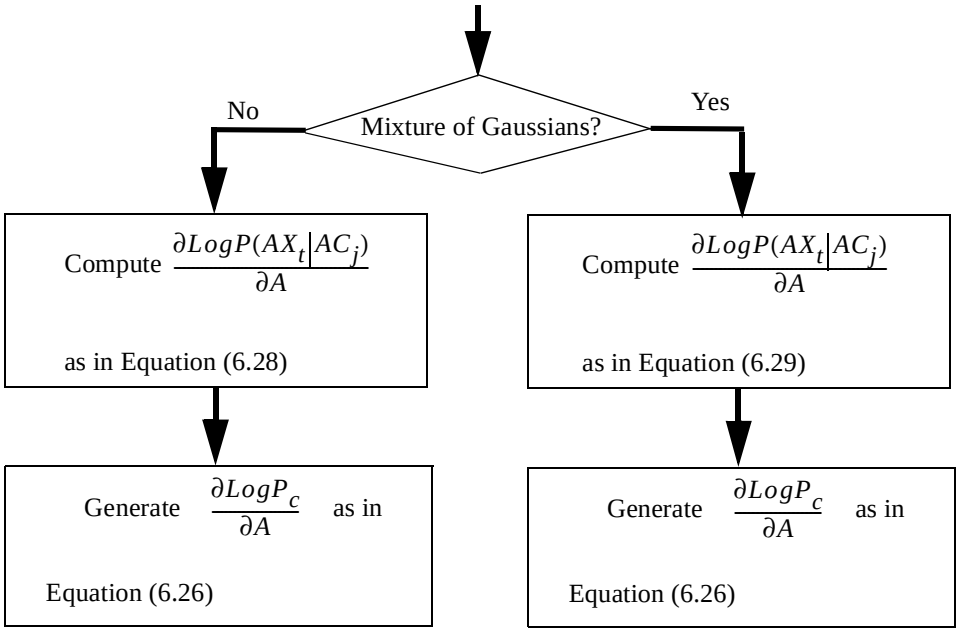

Figure 6.5 Computation of the derivative of the normalized acoustic likelihood with respect to the transformation matrix for single-stream feature generation.

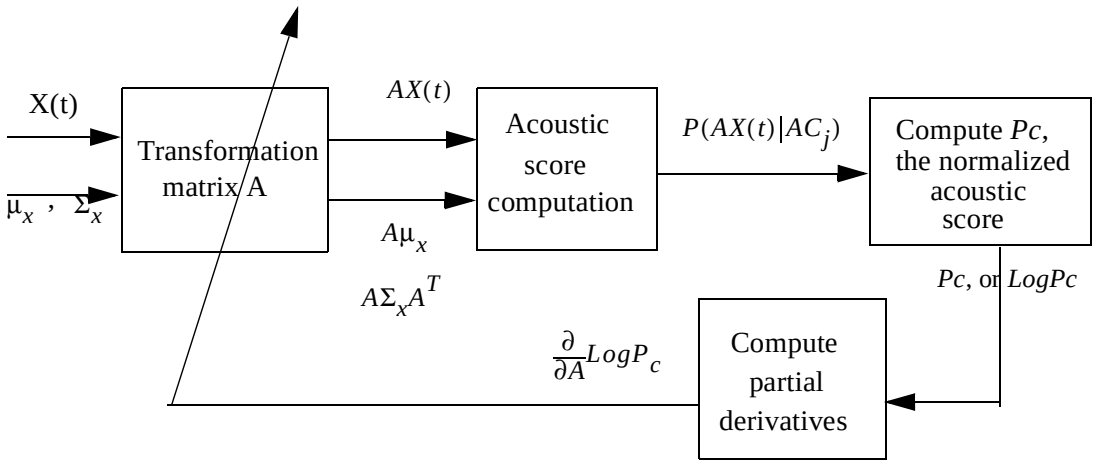

Figure 6.6 Generation of single-stream linear feature for optimal individual recognition performance.

6.5 Evaluating the performance of parallel features

6.5.1 Experimental procedures

Two databases, RM and WSJ0, were used to evaluate the performance of our linear feature generation algorithm. The experiment procedures were the same as described in previous chapters on lattice combination and probability combination, except that some of the conventional features used in recognition and combination were replaced by features generated using our algorithm.

In each testing dataset, both the parallel linear features that are targeted at improving probability combination performance and the single-stream linear features that are designed to optimize the recognition performance of an individual system were generated from our algorithm. The iterative gradient ascent approach as described in Equation (6.17) was used to generate the transformation matrices in both situations. The iterative procedure terminates when the results (*i.e* the normalized acoustic likelihood of the system) converge. To assure convergence, we gradually reduced the step size in the iterative procedure every 10 to 15 iterations depending on the specific recognition task. When there is oscillation (*i.e.* when the value of the objective function at the current iteration is less than it was at the previous iteration), we also reduce the step size.

All the features that were tested and combined in our experiments were generated as linear transformations from the original log-spectral speech representation. The original dimension of log-spectral feature was 40. We augmented this log-spectral feature with its delta and double-delta components before trans-

formation. Hence, the total dimensionality of the original features was 120. This 120-dimensional feature vector was transformed into a 39-dimensional new feature space, so the dimension of the transformation matrices was 39 by 120.

Both the new features generated from our algorithm and the conventional linear features such as MFCC, PCA/KLT, LDA, and HLDA, were evaluated using both the RM and WSJ0 corpora. For each corpus we evaluated both the individual recognition accuracy provided by the various features and the recognition accuracy obtained when parallel linear features were used in combination.

For the remainder of this chapter, we will first describe the experimental results for the RM and WSJ0 databases. We will then describe some other issues involved in the experiment such as the effect of initial points on the recognition accuracy.

6.5.2 Experimental results using the RM database

We evaluated individual recognizer performance using both conventional linear features such as MFCC, PCA/KLT, LDA, and HLDA as well as our new linear features. Our linear features were generated according to the diagram in Figure 6.6, using the LDA matrix as the initial point in the gradient ascent process. Four different linear features were generated from our algorithm with different number of Gaussians per mixture used in the HMM state observation probability distribution. These four features were labeled as A1, A2, A4 and A8, which correspond to the use of 1, 2, 4 and 8 Gaussians per state for the HMM state observation probabilities. These four new features, together with the four conventional linear features (MFCC, PCA/KLT, LDA and HLDA) were tested for their individual performance and compared against one another. Acoustical models with 1, 2, 4 and 8 Gaussians per state as observation probabilities were trained and tested for each of these eight features, producing a total of 32 individual experimental results for the RM corpus. Table 6.2 describes the WER obtained from these linear features on the RM corpus under the different training/testing conditions. Each column in the table represents the performance of various features under one training and testing situation with the number of Gaussians used per mixture in the state observation probability listed at the top of that column. Each row in the table represents the performance of one feature across the various training and testing conditions.

There are a number of observations that can be drawn from Table 6.2. The first is that our new linear features provide better recognition accuracy than conventional linear features. In each column of Table 6.2,

WERs(%)	1 Gau	2 Gaus	4 Gaus	8 Gaus
MFCC	10.29	8.68	7.66	8.31
PCA/KLT	8.96	7.86	7.30	7.85
LDA	8.36	7.67	6.93	7.51
HLDA	8.66	7.57	6.74	7.38
A1	**7.57**	6.82	6.98	7.46
A2	7.62	**6.69**	6.50	7.33
A4	8.34	7.11	**6.34**	7.09
A8	8.37	7.30	6.48	**6.88**

Table 6.2. WERs obtained using various linear features for the RM data set with different training and testing conditions. The number at the top of each column indicates the number of Gaussians per mixture used for the HMM state observation probabilities in training and testing. A1 through A8 are the new linear features generated from our algorithm with correspondingly 1, 2, 4, and 8 Gaussians per mixture used as observation probabilities.

which describes results for a specific number of Gaussians per mixture, the best performing feature is always in the lower part of that column, *i.e.* the region that corresponds to our new linear features. Most of the improvements in recognition accuracy of our new linear feature over the conventional one were statistical significant, as indicated by the significance measures in Table 6.3. We also note that the best-performing linear feature is that for which the number of Gaussians used to generate the feature-transformation matrix is identical to the number of Gaussians used in the actual system evaluation. This is not surprising, as matching conditions in generating linear features to the training and testing conditions should result in better performance. We believe that generating linear features under conditions that resemble the model training and testing process contribute (at least partially) to the better recognition accuracy obtained by our new linear features compared to conventional linear features.

	A1 vs LDA	A2 vs LDA	A4 vs LDA	A8 vs LDA
Statistical Significance measure P	0.015	0.005	0.043	0.060

Table 6.3. Statistical significance of differences between the WERs obtained using our new linear features and using conventional LDA feature for the RM set. Significance measures were generated using the “matched pairs” method [16].

In the parallel feature combination experiments, both the individual recognition accuracy and the com-

bined performance of the parallel features were tested. We compared three pairs of parallel linear features: (1) the conventional linear features of LDA and PCA, (2) parallel features generated from our algorithm using summation as the combination function *F*, and (3) parallel features generated from our algorithm using multiplication as the combination function *F*. These two sets of new parallel features were iteratively generated using the gradient ascent approach as described in Equation (6.17). In the experimental results reported here, the iterative procedure was initialized using LDA and PCA transformation matrices. We also explored the effect of using other initial points on the performance of the final transformation matrices, as will be discussed later in this chapter. For each set of parallel features, both the individual recognition accuracy and the probability combination accuracy were tested. Probability combination was implemented using two different combination functions, summation and multiplication. Hence there were four different results for each set of parallel features: two for the individual recognition performance of each feature stream, and two more for combined performance using the summation and multiplication functions. These two sets of new parallel features were all generated based on using 1, 2, 4, and 8 Gaussians per mixture in the HMM state observation probability, and the models were also trained and tested (and combined) with 1, 2, 4, and 8 Gaussians per mixture in the HMM state observation probability. Because the effect of the number of Gaussians used per mixture in parallel feature generation is the same as was observed in single-stream feature generation, we only reported the recognition performance for matched conditions[20] in which parallel features were generated, tested and combined using four Gaussians per state for the HMM state observation probabilities. These experimental results are summarized in Table 6.4

20. The experimental results listed in Table 6.4 was a matched condition in terms of the number of Gaussians used per mixture in the HMM state observation probability. We will look at the effect of another matched conditions in terms of the combination functions used in generating and combining parallel features.

WERs(%)	LDA&PCA	Para_multiplication	Para_summation
First stream individually	6.93	7.14	7.14
Second stream individually	7.30	6.42	7.27
Multiplication combination	7.06	5.99	6.10
Summation combination	5.28	4.83	4.61

Table 6.4. Recognition and combination results for three pairs of parallel features. The feature sets are conventional LDA and PCA, parallel features generated using multiplication as the combination function (Para_multiplication), and the parallel features generated using summation as the combination function (Para_summation).

From Table 6.4 we can see that the use of parallel features provides significant improvements in recognition accuracy. We achieved 13% and 15% relative improvements using the summation and multiplication combination compared to the conventional parallel features of LDA and PCA, and these improvements are statistically significant using the matched pairs test. This clearly indicate the effectiveness of our parallel feature generation algorithm. We also note that for each combination function tested, the set of parallel features that yields the best combined performance was always the one that was generated using the same combination function. For example, the globally optimum summation combination performance was achieved by our parallel features which were generated using the same summation function. This observation is similar to what we found with the dependences on the terms of number of Gaussians per mixture used for the HMM state observation probabilities: matching the conditions used to develop the features with the conditions used for system training and testing produces the best performance. Again, we believe this observation confirms (at least partially) the value of generating parallel features for each specific combination function in order to fully utilize the benefit of that combination function, as suggested earlier as an improvement of our approach over conventional parallel feature generation algorithms.

It is also interesting to compare the recognition accuracy obtained using our parallel features with that obtained using conventional LDA and PCA features. Specifically, while the parallel features developed using the summation and multiplication functions provide the best performance when used in combination, the best-performing parallel-sum features are outperformed by the conventional LDA and PCA features when the features are considered in isolation. This reinforces the concept that while these features are

developed to optimize performance when used in combination, performance of the individual features may be suboptimal. Of course, we can still generate single-stream optimal features for better individual recognition accuracy, as described in Section 6.4, but those features will be different from the ones generated with the purpose of optimizing combination performance, since the criteria for these two situations are different from each other.

In general we believe that the maximum benefit from parallel feature combination arises when the features are most "complementary" or "different" from one another. From this point of view, a set of "good" parallel features should be those for which the individual features are as different from each other as possible. An increase in the "complementary" or "difference" between parallel features may result in a decrease in the individual recognition accuracy that is obtained when parallel features are used in isolation, but it will provide better performance when these parallel features are combined together. Indeed, if we adopt as a heuristic definition for "complementarity" the relative reduction in word error rate of a combined system over the single best individual system, this complementary measure in our new parallel feature set was higher than the one computed from the conventional LDA and PCA set, as we compare the 23.8% reduction in WER using the LDA and PCA set with the corresponding 24.7% obtained from parallel features generated with multiplication function, and the 35.4% relative reduction from parallel features generated with summation function.

6.5.3 WSJ0 experimental results and analysis

The experiments carried out using the WSJ0 database were the same as those using the RM database. We generated both single-stream linear features with the goal of improving individual recognition performance, and parallel linear features for optimal probability combination performance. The single-stream features for optimal individual recognition accuracy were generated using four Gaussians per mixture for the HMM state observation probabilities, while the parallel linear features were generated with each individual stream using two Gaussians per mixture as the HMM state observation probability distributions[21]. As before, the summation and multiplication combination functions were used to generate parallel linear

21. As we combine two feature streams together, the number of free model parameters in the combined system (2 streams with 2 Gaussians per stream) will be the same as in the individual system (4 Gaussians per stream). In this fashion we compare the best performance obtained using single stream features with the best combination performance obtained using parallel features developed with the same number of free model parameters.

features. Both the single-stream and parallel stream linear features were iteratively generated using the gradient ascent approach as in Equation (6.17) and Equation (6.27). The LDA matrix was used as the initial point for generating single-stream features, while LDA and PCA matrices were used to initialize the updating procedure in generating parallel features. Conventional linear features such as MFCC, PCA, LDA, and HLDA were also tested in the single-stream feature case for comparison purpose. The combined performance of our parallel features was compared with the combined performance of LDA and PCA features.

Table 6.5 summarizes the WER obtained using the individual feature streams in the WSJ0 database, with the acoustic models trained and tested using four Gaussians per mixture in the HMM state observation probabilities.

Feature	MFCC	PCA	LDA	HLDA	MaxProbFeat
WERs(%)	9.02	8.48	8.03	7.88	7.68

Table 6.5. Recognition accuracy provided by single-stream features for the WSJ0 database. "MaxProbFeat" represents the single-stream feature generated from our algorithm, which maximizes the value of the normalized acoustic likelihood of the "true" HMM state. It was generated using four Gaussians per mixture for the HMM state observation probabilities. All the systems were also trained and tested using four Gaussians per mixture.

Table 6.6 summarizes both the individual and combined performance of parallel features generated using our algorithm. The same three pairs of parallel features are shown that were used for the RM data: LDA and PCA features, and parallel linear features developed using the summation and multiplication combination functions. All the models were trained and tested using two Gaussians per mixture for the HMM state observation probability distributions.

WERs(%)	LDA&PCA	Para_multiplication	Para_summation
First stream individually	9.32	9.94	9.69
Second stream individually	9.81	10.09	10.48
Multiplication combination	8.79	8.28	9.0
Summation combination	7.99	7.81	7.57

Table 6.6. Combined performance of parallel feature streams in the WSJ0 database. The models for each individual feature stream were trained and tested using two Gaussians per mixture. The meaning of the symbols are the same as the one in Table 6.4

As seen in the data in Table 6.5 and Table 6.6, all overall trends observed in the RM experiments remain true for the WSJ0 experiments. Our single-stream linear features provided the best individual recognition accuracy and the new parallel features improved the combined performance. Matching conditions in the functions used to generate parallel features and to combine them in the recognition process still produces the best combined system performance. It is still the case that the parallel features that provide the best combined performance do not provide the best individual recognition accuracy. Nevertheless, the relative improvement achieved for the WSJ0 database from our algorithms was not that great as for the RM database, as we note a 3-5% relative improvement of our optimal single-stream features over the conventional best linear feature in WSJ0 compared to a relative improvement of 6-11% for the RM database. Similarly, the 5%~6% relative performance gain in the parallel feature combination experiment in WSJ0 is smaller than the corresponding 12%~15% in the RM set. Although the specific reason for this reduced benefit of our algorithm is unclear, we hypothesize that the complex testing situation in WSJ0 database might be the major reason. As reflected in the Equation (6.7) and Equation (6.23), our feature generation algorithm seeks the transformation matrices that optimize the normalized acoustic likelihood in the combination system or the individual system, which is in the broad sense a discriminative training algorithm. As many other people have observed [28], the effectiveness of a discriminative training algorithm diminishes as the testing situation get more and more diverse. In our situation of generating linear transformation matrices, it becomes increasingly difficult to find a global transformation matrix (or matrices) which is (are) significantly better than other matrices (*e.g.* the one from LDA, HLDA) in discriminating the data from different classes. The transformation matrices generated from our algorithm may still be better than those generated from conventional methods (as is confirmed by our experimental results), but their difference in recognition accuracy become smaller and smaller as the testing situation getting more and more complex.

6.5.4 Portability of our new linear features

Our new feature generation algorithm generates linear features through the maximization of the normalized acoustic likelihood of the training data, which is a data-driven approach. As different training set may have different characteristics, the optimal transformation matrices generated for one training set (*e.g.* RM) may be different from the other set (*e.g.* WSJ0). Although in theory we should apply our algorithm and generate a new set of transformation matrices for each new training set, it would be nice if we could still use the optimal transformation matrices generated from other sets to achieve reasonable performance

without the need to regenerate new transformation matrices. In this subsection, we reports our experimental results on the portability of our linear features.

The portability experiments were carried out in the following manner: We generated the optimal single stream transformation matrix for the WSJ0 database, and then applied the generated transformation matrix to the RM dataset. For comparison purposes, we also tested the portability of the LDA and PCA matrices by generating them for the WSJ0 database and training and testing them using the RM database. The experimental results in terms of WER are listed in Table 6.7.

	1 Gau	2 Gaus	4 Gaus	8 Gaus
PCA feature generated from WSJ0	9.35	8.21	7.84	8.04
LDA feature generated from WSJ0	9.06	7.97	7.94	7.62
Optimal linear feature from WSJ0	8.85	7.09	6.86	7.30
PCA	8.96	7.86	7.30	7.85
LDA	8.36	7.67	6.93	7.51
Optimal linear feature	7.57	6.69	6.34	6.88

Table 6.7. Portability of various linear features. Three linear features (PCA, LDA and our new optimal single stream linear feature) were tested and compared against each other. The top three rows represent the linear features that were generated from the WSJ0 database while tested on the RM dataset. The bottom three rows are the result of linear features which have been generated and tested on the same RM set.

Table 6.7 revealed some interesting observations. It is clear that generating linear features on one feature set and training and testing on another set will degrade recognition accuracy. This can be seen from comparisons of the experimental results in the bottom three rows with the corresponding results in the top three rows. We need to generate the transformation matrix for each specific training/testing set in order to achieve optimal performance. Nevertheless, our optimal linear features are more robust than the conventional LDA and PCA features in transferring between different databases. Although they all rendered degraded performance in the new RM dataset, our optimal linear feature still obtained the best absolute performance among the three linear features that we have tested.

6.5.5 What do the transformation matrices look like?

Our new transformation matrices have been generated in a manner that optimizes the normalized acoustic likelihood of the most likely HMM state sequence. The new transformation matrices produce improved performance in the combined system. In this subsection we carry out some analyses on the generated transformation matrices to see if we can make any interesting observations about the content of the transformation matrices themselves. As our parallel transformation matrices have been generated from the initial points of LDA and PCA matrices, our analysis have been focused on comparisons of our new transformation matrices with the original LDA and PCA matrices.

We first generated parallel transformation matrices using LDA and PCA matrices as the initial point (choosing the multiplication function for combination). In our comparison, we arbitrarily choose several rows of the transformation matrices, plotted them and compared against each other. Figure 6.7 is the comparison of the first row of LDA matrix and the new matrix initialized from LDA, and the first row of PCA matrix with the corresponding new matrix that had been initialized from PCA. Figure 6.8 and Figure 6.9 are corresponding comparisons of the 12^{th} row, and the 22^{nd} row of the transformation matrices.

One thing we noticed from the above figure is fact that the matrix initialized from PCA changed a lot compared to the LDA initialized matrix. Although we don't know the specific reason for this, possible explanations could include the following: as the LDA matrix provides better individual recognition performance than the PCA matrix, the position of the LDA matrix on the surface of $LogP_c$ could be closer to the optimal point than the position of the PCA matrix. Correspondingly, the derivative computed from the LDA-initialized matrix could be smaller than the derivative computed from the PCA-initialized matrix. According to Equation (6.12), the derivative of the objective function $LogP_c$ with respect to the individual transformation matrix $\frac{\partial}{\partial A_x} LogP_c$ and $\frac{\partial}{\partial A_y} LogP_c$ is proportional to the derivative of the combined likelihood with respect to the particular individual transformation matrix $\frac{\partial}{\partial A_x} \partial LogF\{P(A_x X_t \left| A_x C_j), P(A_y Y_t \right| A_y C_j)\}$ and $\frac{\partial}{\partial A_y} \partial LogF\{P(A_x X_t \left| A_x C_j), P(A_y Y_t \right| A_y C_j)\}$. As we use the multiplication function to combine probabilities, the derivative of the combined likelihood with respect to the individual transformation matrix is indeed the derivative of the individual likelihood with respect to the transformation matrix as indicated in Equation (6.14). As the result, the increment contributed to the updat-

ing of the LDA-initialized matrix, which is proportional to the derivative component, may be smaller than that of the PCA-initialized matrix, which consequently produces less change to the LDA-initialized matrix than to the PCA-initialized matrix.

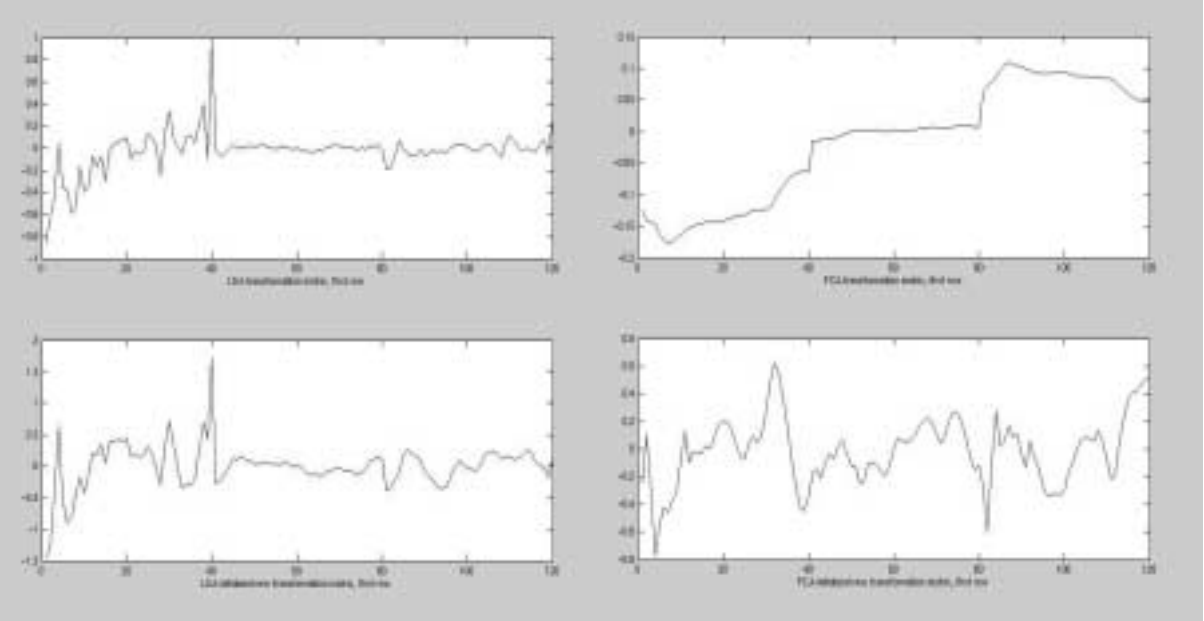

Figure 6.7 The first row of the transformation matrices. The matrices represent features obtained from LDA, PCA, the PCA-initialized new transformation matrix, and the LDA-initialized new transformation matrix, in clockwise order beginning with the upper left panel.

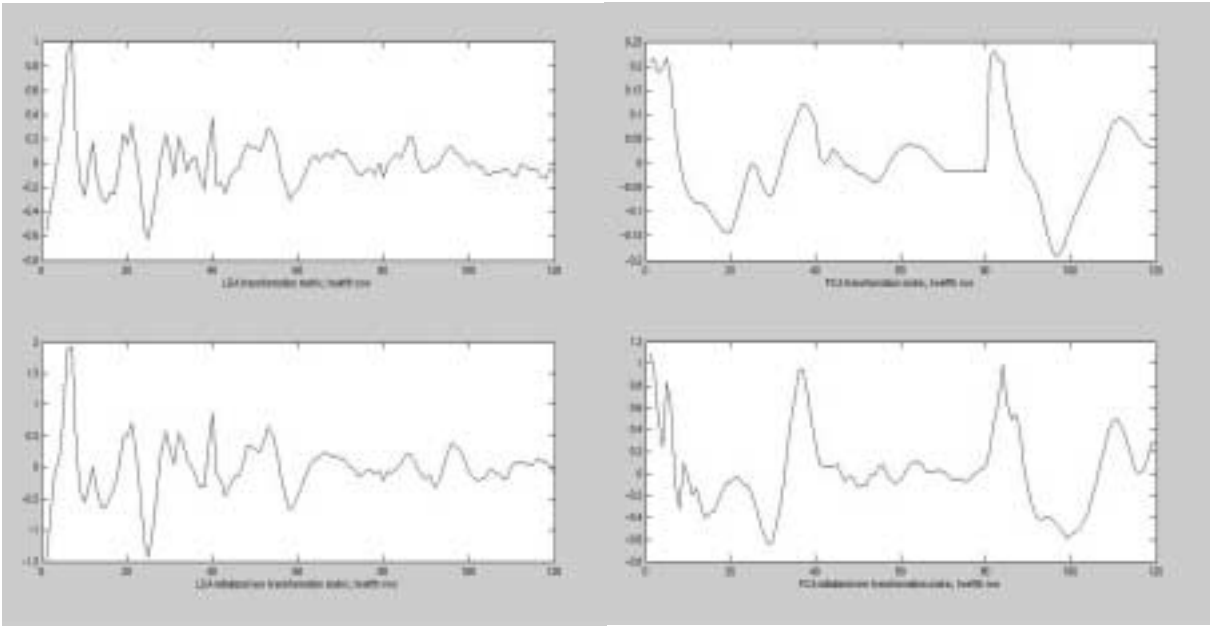

Figure 6.8 Same as Fig. 6.7, but representing data from the twelfth row of the transformation matrices.

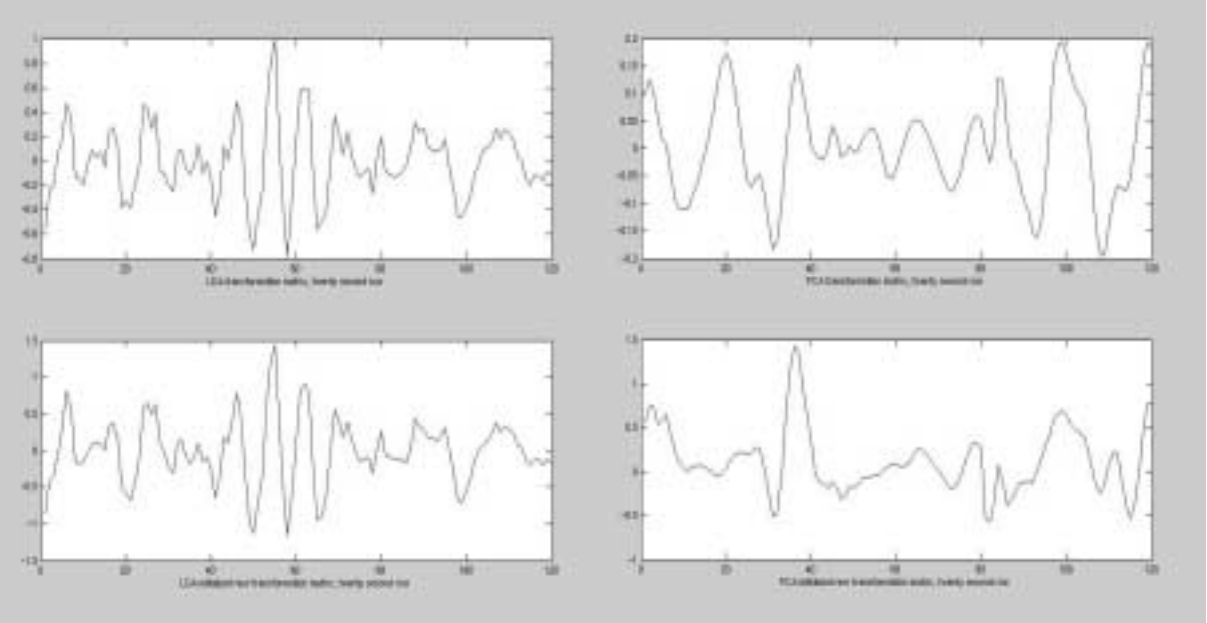

Figure 6.9 Same as Fig. 6.7, but representing data from the twenty-second row of the transformation matrices.

In addition to the "graphical" comparison of the transformation matrices, we also evaluated a heuristic measure T defined to be the average cosine distance between the row vectors of parallel transformation matrices:

$$T = \sum_i \sum_j Abs\left(\left\langle \frac{A_{x,i}}{|A_{x,i}|}, \frac{A_{y,j}}{|A_{y,j}|} \right\rangle \right) \tag{6.30}$$

where $A_{x,i}$ is row *i* of matrix A_x, "||" represents the norm of a vector, "*Abs*" represents its absolute value, and "<,>" represents the dot product of two vectors. We hypothesize that the parallel matrices could be more complementary if their individual components are more orthogonal to each other, which corresponds to a smaller value of the heuristic measure *T*. Our new transformation matrices (as shown in Figure 6.7 to Figure 6.9) reduced the *T* measure by 5% compared to the initial set of LDA and PCA matrices.

As our transformation matrices were generated using data-driven approach, we didn't obtain much additional meaningful information by direct subjective inspection of the transformation matrices.

6.5.6 The effect of the initial points

The transformation matrices in our algorithm were generated iteratively using the gradient ascent approach as described in the Equation (6.17). Since the surface of the normalized acoustic likelihood P_c (or $LogP_c$) over the transformation matrix values is not a convex surface, as illustrated in one simple example in Figure 6.10, the initial points in the iterative updating procedure will directly affect the final transformation matrices generated. Ideally, we want to start from initial points which will make the iterative procedure converge at the global optimal point, but often in practice we don't know from which point should we start to make the iterative procedure converge at the global optimal point. As the result, the iterative updating procedure can get trapped in some local optimal points, and the performance of the final transformation matrices will be affected.

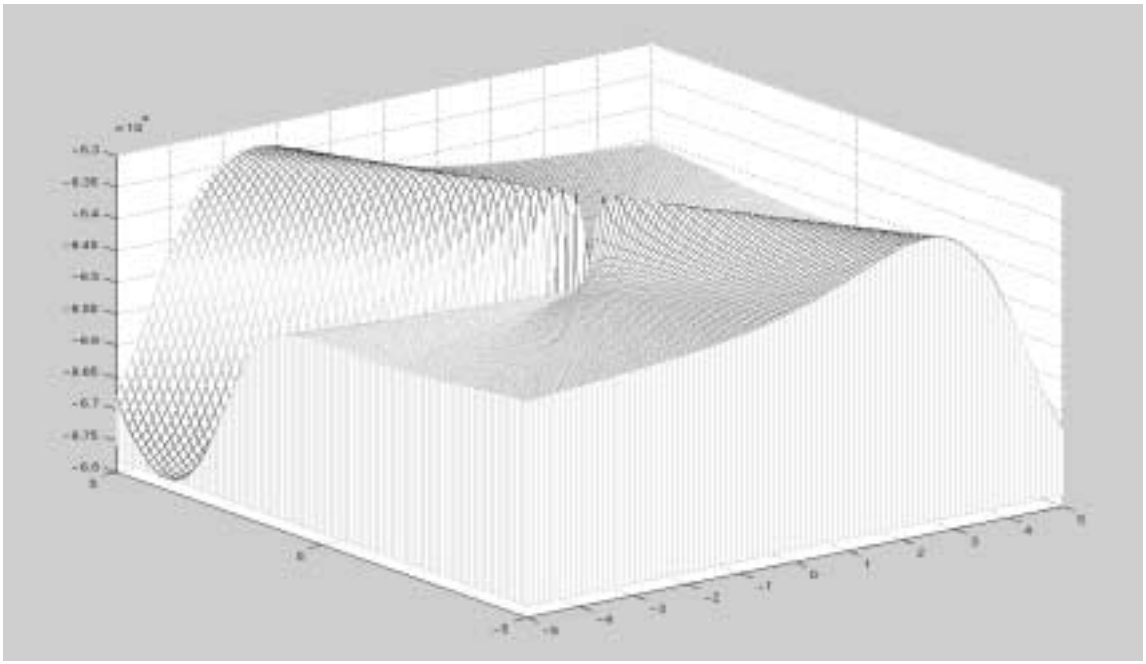

Figure 6.10 The surface of the normalized acoustic likelihood for a simple 1-by-2 transformation matrix. The horizontal axes represent the specific value of the two components of the transformation matrix, and the vertical axis represents the corresponding value of the normalized acoustic likelihood. It is clear from the figure that the surface of the normalized acoustic likelihood over the transformation matrix component is not convex.

In general two approaches can be used to generate initial points. One is to start from some other "optimal" points (such as the transformation matrices obtained using LDA, PCA, etc.) that are generated from other approaches based on different criteria; and the other is to start from some randomized initial points.

In this section, we compare the performance of these two initial-point selection approaches.

We choose the task of generating optimal parallel-stream linear features using the multiplication function for the RM corpus to compare the performance of these two approaches. We start the updating procedure from using the LDA and PCA matrices and some randomized points as the two initial points. At each iteration of the updating procedure we record the value of the objective function P_C (in log values) and the recognition accuracy of the combined system using the multiplication function. Figure 6.11 shows the log value of the objective function $LogP_C$ at each iteration with the two different ways of selecting initial points. The curve of the combined performance at each iteration is shown in Figure 6.12.

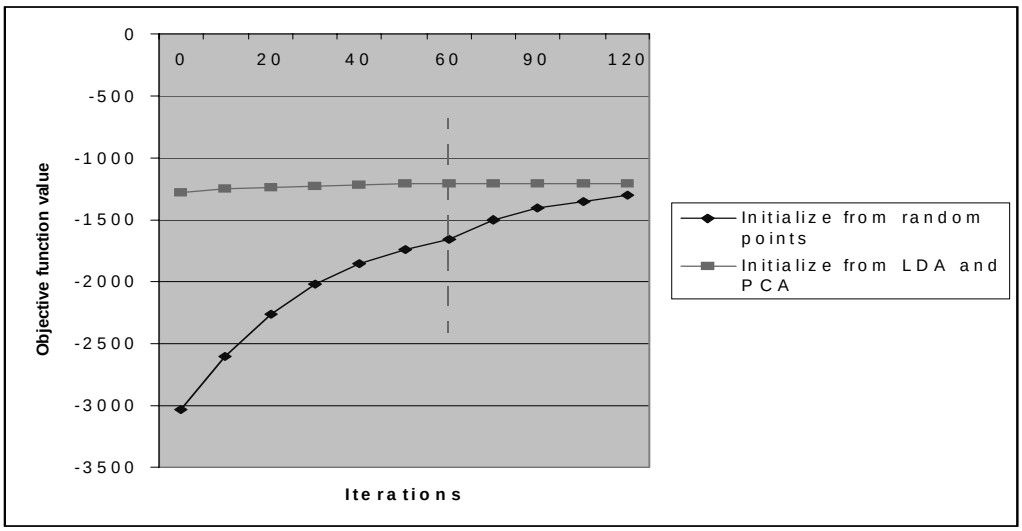

Figure 6.11 The value of the objective function at each iteration. The dashed line represents the point at which the iteration process that starts from the LDA and PCA converges.

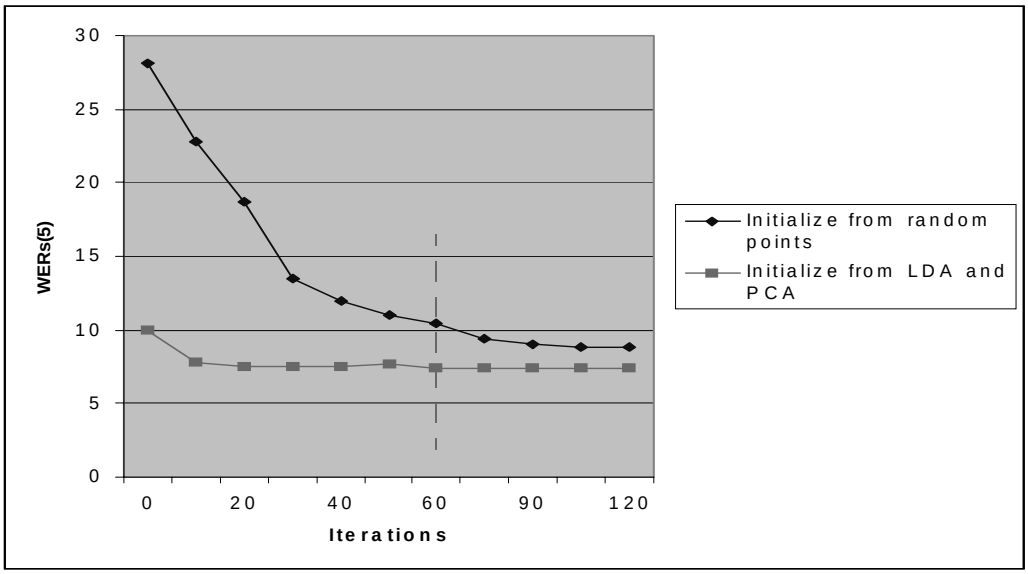

Figure 6.12 The recognition results at each iteration.

As illustrated in the above two figures, starting with the "optimal" points generated by other approaches (*e.g.* the LDA and PCA matrices) have proven to be a more effective approach in iterative updating procedure than the procedure of starting from randomized initial points, which is also consistent with the similar observations of others (*e.g.* Wu observed in his work of finding MCE based adaptation matrix [77] that initializing from the optimal point generated by other approaches (*e.g.* MLLR) performs better than starting from the point of using identity matrix). We hypothesize that although the optimal points for the objective of maximizing the normalized acoustic likelihood in our approach are different from the optimal points generated from other approaches based on some other objectives, they could be relatively close to each other, as they all have the goal of improving recognition performance of the system, either directly or indirectly. Since the surface of the normalized acoustic likelihood is very complicated, the iterative updating procedure is very likely to be finally trapped into some local optimal points rather than the global optimal point. Because the "optimal" points generated by other approaches (*e.g.* LDA and PCA) have in general higher values of the objective function than randomized initial points, the iterative updating procedure started from these more "optimal" points (*e.g.* LDA and PCA matrices) is more likely to end at a local optimal points which has higher value of the objective function than the ending points of those iterative procedures that start from randomized initial points. We believe that this could be the explanation for the better performance obtained from initializing from LDA and PCA matrices than initializing from randomized points.

6.5.7 Comparing the performance of combined systems with single systems with the same number of free model parameters

For each recognition task, two general approaches can be applied. One is to extract a single-stream feature, train the model, and test it with single-stream features. The other is to extract multiple features from speech data, train the models separately and combine them together. In this section we compare the recognition accuracy obtained with single-stream feature to the performance obtained using a combination of multiple feature streams, under conditions in which the number of free model parameters used in both situations is equal.

The experiments were carried out again using the RM and WSJ0 database. The individual recognition accuracy using single-stream features was compared to the performance obtained by combining two parallel features. The number of Gaussians per mixture used for the observation probabilities in the single-stream feature model was twice the number of Gaussians per mixture used in each individual system in the parallel feature set, so that the total number of free model parameters was equal for the two systems. Two comparisons were made: a comparison of individual recognition accuracy of LDA features with the combined performance of LDA and PCA, and a comparison of the recognition accuracy obtained for a single optimal feature generated according to Equation (6.23) and the combination of parallel optimal features as generated by Equation (6.7) and Equation (6.8). The parallel features were generated and combined together using the same summation function in order to obtain a matched conditions in terms of the function used in generating and combining the parallel features.

The experimental results are summarized in Table 6.8. These results clearly show the advantage of combining parallel features over the direct use of single-stream features, as the performance of the former approach was consistently equal to or better than the performance of the latter approach when using the same number of free model parameters. The advantage of probability combination over single-stream recognition was even more obvious at certain points at which the acoustic model get over-trained. For example, with eight Gaussians per mixture in the RM database, the single-stream feature model showed signs of over-fitting as its performance was worse than that obtained using four Gaussians per mixture. But the performance obtained by combining two feature streams with four Gaussians per mixture still showed an improvement compared to the combination of two streams with two Gaussians each. Even though the number of free model parameters in both situations (*i.e.* single-stream features using 8 Gaussians per mix-

ture and two parallel streams using 4 Gaussians per mixture individually) was the same, the performance of the single-stream model showed signs of being over-fitted, while the combined system still showed an improvement. This may suggest another advantage of the combination is a "robustness" to over-fitting: at the point at which a single feature stream becomes over-fitted by training data, combining parallel systems may provide a further improvement in recognition accuracy.

WER (%)	Single-stream, 2*n* Gaussians per mixture	Two parallel streams, each with *n* Gaussians per mixture
RM: LDA vs LDA and PCA comb; n=1	7.67	7.30
RM: LDA vs LDA and PCA comb; n=2	6.93	5.76
RM: LDA vs LDA and PCA comb; n=4	7.51	5.28
RM: optimal single stream vs optimal parallel streams; n=1	6.69	6.16
RM: optimal single stream vs optimal parallel streams; n=2	6.34	5.11
RM: optimal single stream vs optimal parallel streams; n=4	6.88	4.61
WSJ0: LDA vs LDA and PCA comb; n=2	8.03	7.99
WSJ0: optimal single stream vs optimal parallel streams; n=2	7.68	7.57

Table 6.8. Comparison of single-stream recognition accuracy with parallel-stream combined performance under the constraint that the number of free model parameters is the same in both situations. The parameter *n* is the number of Gaussian components used per mixture in each individual stream of the parallel feature. The number of Gaussian components used in the single-stream features is 2*n*.

6.6 Comparison of different methods for parallel feature generation and combination

So far we have proposed four different algorithms to improve the performance of combined systems from different perspectives. Three of them are the parallel feature combination algorithms: the lattice com-

bination method described in Chapter 4 and the two probability combination methods described in Chapter 5. The fourth approach is the parallel linear feature generation method based on maximum normalized acoustic likelihood that is described in this chapter. These algorithms are all described in an isolated manner without taking into account the joint effect of applying them together. In this section, we consider the combined performance when these algorithms are applied together. Specifically, we will compare the recognition accuracy obtained by combining lattice combination and probability combination with the parallel features described in this chapter. These comparisons are intended to be more suggestive than comprehensive, and only a limited number of experimental conditions are employed.

Again, we use the RM database for these experiments. We compare in these experiments:

- The individual recognition accuracy obtained using LDA and PCA features
- Lattice combination performance of LDA and PCA feature
- Probability combination performance of LDA and PCA features with tied HMM states
- Probability combination performance of LDA and PCA features with context-dependent untied HMM states
- Loss-based weighted probability combination performance of LDA and PCA features
- Individual performance obtained using parallel linear features (with the summation combination function and LDA and PCA matrices as initial points)
- Lattice combination performance of our new parallel features
- Probability combination performance of our new parallel feature (using both tied HMM states and context-dependent phoneme untied HMM states for combination)
- Loss-based weighted probability combination, as above.

These conditions were selected because they provided a representative comparison of all of the algorithms discussed in this thesis.

Recognition results for the conditions above are summarized in Table 6.9, obtained using a single Gaussian for the output distribution for generating parallel features and for training / testing the speech recognition system[22]. These experimental results are consistent with our previous observations when the proposed techniques as lattice combination, probability combination and parallel feature generation were applied

22. We also performed the experiments using mixture of Gaussians. The observations were very similar to the results obtained using single Gaussians that are reported here, so they are omitted from here.

WERs(%)	LDA &PCA	Parallel features
First individual feature stream	8.36	8.82
Second individual feature stream	8.96	8.82
Hypothesis combination of parallel streams	7.54	7.38
Lattice combination of parallel streams	7.21	7.01
Probability combination of parallel streams (Tied HMM state, equal weights)	7.30	6.16
Probability combination of parallel streams (Tied HMM state, loss-based weights)	7.02	5.97
Probability combination of parallel streams (Context dependent phoneme HMM state, equal weights)	6.90	5.86
Probability combination of parallel streams (Context dependent phoneme HMM state, loss-based weights)	6.72	5.74

Table 6.9. Comparison of the joint effects of applying parallel feature generation and combination algorithms in the RM set. Parallel features are generated using LDA and PCA matrices as the initial points.

and tested in isolation. Lattice combination provides better combination performance than hypothesis combination on various sets of parallel features; probability combination in general outperforms the hypothesis combination algorithms, and weighted combination (with *loss*-based weights) using context-dependent phoneme HMMs state provides the best[23] combination performance among the various combination methods that we have tested. When the specific combination stage and combination function have been fixed, combination performance can be further improved by using parallel features that are specifically designed to optimize the combination performance using the specific combination function at that combination stage. This effect of the parallel features and the combination method on combined performance also confirms our original expectation that the task of improving the combined performance should be approached from both the directions of refining the combination method and designing parallel features for the specific type of combination to be employed.

23. While this is true in general, for the TID set context-dependent phoneme HMM states performed a little bit worse than tied HMM states.

7. Summary and Conclusions

7.1 Introduction

In this thesis we have sought to improve the performance of speech recognition systems through the combination of parallel acoustic features. Combination of information has been proven to be an effective approach for improving the speech recognition accuracy in many situations, as the performance of the combined system consistently outperforms the best performance of individual feature streams, provided that the combination is done appropriately.

Research in information combination can generally be divided into two categories, focussing either on the specific algorithm by which information combination is accomplished, or the selection of the features of the systems to be combined. Our work considers both of these issues. We describe a new lattice combination algorithm that combines the lattices from parallel features (or systems) together, and two new probability combination algorithms, as *loss-based* weighted probability combination and untied context dependent phoneme HMM state combination, which fuse the probabilities of parallel features or systems at the stage at which probabilities are evaluated in the HMMs prior to the search process. These two kind of combination algorithms operate in rather different ways and have different properties. The lattice combination algorithm is more flexible in that it can combine parallel systems or features with any arbitrary model structure, while the new probability combination algorithms provide greater accuracy performance but assumes that the systems to be combined have similar model structures (i.e. all parallel systems should be HMM based). In our work on the extraction of parallel acoustic features, our *maximum normalized acoustic likelihood based linear feature generation* algorithm generates parallel linear features directly in a fashion that optimizes the performance of a combined system that is based on the generated parallel features. The process of generating parallel features has been reformulated as an optimization process with the objective of maximizing the accuracy of the combined system, and the task of generating parallel features for optimal combined performance has been transformed into a process of finding parallel features that optimize an objective function. In addition to generating “optimal” parallel features for each specific combination algorithm, this approach can also generate a single stream feature that optimizes the recognition accuracy of an individual system, just by switching the definition of the objective function from the accuracy of a combined system to the accuracy of an individual speech recognizer.

In the remainder of this chapter we summarize the findings and the contributions of this thesis, discuss some of the remaining open questions in this area, and we point out possible directions for future research.

7.2 Summary of the major contributions of the thesis

Lattice combination: We have developed a new lattice combination algorithm that directly combines the lattices generated from parallel features (parallel systems) together. Lattices contain much more information about the word sequences that should be considered in a search process than a single best hypothesis, and they provide a much more compact and efficient representation than an N-best list of hypotheses. Combining lattices yields better combined performance than hypothesis-based combination algorithms that operate on the best outputs of individual recognizers.

Probability combination with loss-based combination weights: This new algorithm combines the probabilities (acoustic likelihoods) of each recognition class (HMM state) from parallel features, and produces a combination result based on the combined probabilities. This procedure weights unequally the contributions of individual features or systems, based on the "loss" of that particular feature or system in predicting the acoustic likelihood for each recognition class. This loss measure is based on the difference between the predicted probability of an individual feature or system and the oracle score for the individual recognition class. This loss-based weighted probability combination provides better probability combination performance than equally weighted probability combination algorithms.

Probability combination based on untied context-dependent phoneme HMM states: Probability combination can operate on either the probabilities emerging from tied HMM states or untied states of context-dependent phonemes. We described a new probability combination algorithm that combines directly the probabilities of untied HMM states of context-dependent phonemes. This enables the HMMs in the systems to be combined to be developed freely without the need for all systems to share a common model structure that is a tied HMM across all systems. According to our experimental results, probability combination based on untied context-dependent phoneme-HMM states typically provides a relative decrease of about 10 percent in WER compared to the direct combination of tied-HMM state probabilities.

Acoustic score normalization for improved probability combination: The probabilities or acoustic scores of individual systems generally have different dynamic ranges. Properly normalization of these

scores across systems generally improves the recognition accuracy of the combined system.

Parallel linear feature generation for optimal performance of the combined system: The algorithms that generate optimal parallel linear features may be one of the most significant contributions of this thesis. We describe a new feature generation algorithm that generates parallel features directly by maximizing the accumulated value of normalized acoustic likelihood that is obtained after probability combination across multiple systems. This transforms the process of generating parallel linear features into an optimization of the linear transformation matrices according to the normalized acoustic likelihood criterion. The exact features that are generated depend on the specific function that is used to combine them, so, for example, a set of parallel features that maximizes performance under *summation* combination is likely to be different from a similar set of parallel features that maximize performance under *multiplication* combination.

Linear feature generation for optimal individual recognition performance: Our proposed parallel linear feature generation process can also be reduced to the generation of single stream features for optimal individual recognition performance. The process of generating a single-stream optimal feature set is very similar to the process of generating parallel streams of linear features for optimal combined performance, with the major difference being the shift of the definition of “accumulated value of normalized acoustic likelihood” from the combined system to the individual system. This approach is capable of up to a relative reduction of 15% in word error rate compared to the best-performing LDA or HLDA features.

7.3 Directions for future research

While the algorithms developed in this thesis have been quite successful at improving the performance of the combined system, there is still ample room for additional improvement. Additional research in the following areas may have the potential to provide additional improvement.

We made two assumptions in generating linear features for the optimal probability combination or individual recognition performance. The first was that the partitions of the training data in the original feature space and the transformed feature spaces are the same, and the second was that the mixture coefficients of individual Gaussian component in the HMM state observation probability distributions remain the same in the transformed feature space. We needed these two assumptions to formulate the model parameters of the

transformed feature to carry analysis and generate optimal linear features. But these two assumptions can be relaxed with an iterative updating procedure. At each iteration, we can partition the training data and estimate the mixture coefficients iteratively using transformation matrices generated from the previous iteration. This iterative approach for generating new partitions and new mixture coefficients may improve the quality of the final transformation matrices generated. As the iterative updating procedure proceeds, the partition generated by the current transformation matrices should also change accordingly. It would be better to change the training data partition iteratively rather than stick with the original partition. Of course, an iterative procedure that generates new partitions and mixture coefficients would impose additional computational requirements, but these operations do not need to take place after every iteration.

While the use of accumulated normalized acoustic likelihood has been successful in this work, it still may not be the optimal choice as the objective function because it is defined on the HMM state level, while word error rate (the real objective) is defined at the word level. In principle, performance could be improved if we can reduce the inconsistency between the stages at which we define the objective function and at which we measure the recognition and combination performance. One of several possible directions could be to use a “minimum phone error rate” (MPE) measure as the objective function. MPE is defined at the phoneme level, which is one level closer to the word level than our current accumulated normalized acoustic likelihood (at the state level), so the use of MPE as an objective function may lead to further improvements.

In generating optimal transformation matrices for linear features, the initial values used in the gradient ascent process do have an impact on the accuracy provided by the final sets of features. As experimental results indicate, it would be better to start the gradient ascent from initial values which already provide good accuracy than from random starting points. We currently use LDA and PCA matrices as the initial points in generating parallel feature streams, but as the number of parallel feature streams that we want to generate increases, the question of how to best select different sets of good initial points will become an issue that needs to be addressed properly. One of possible solution might be to start from random variations of some good initial points such as the LDA and PCA matrices.

Our current lattice combination algorithm requires an extensive search to find qualified edges for merging and qualified nodes for creating new edges. This process is very computationally costly, especially when the lattice is large. It would be valuable to develop a more efficient algorithm that searches for

matched edges that should be merged together and nodes that should be linked together.

Acoustic score normalization also plays an important role in the performance of the combined system. As we saw from the lattice combination experiments, normalization is necessary for combination to be successful. Although the specific form of the normalization function in our experiments does not have a great impact on the performance on the combined system, the current procedure is far from optimal, and better results are likely to be obtained with a more systematic approach to the problem.

8. Appendix

The derivative of the likelihood of the class C in the transformed feature space AX with respect to the transformation matrix A.

If we denote $P(AX|C)$ as the likelihood of class C in the transformed feature space, the transformation matrix as A, whose dimensions are N (Before transformation) and M (after transformation) respectively. The individual component of matrix A at the k^{th} row and p^{th} column is $a_{k,p}$. With the assumption of Gaussian distributions with diagonal covariance matrix, we have:

$$P(AX|C) = \prod_{k=1}^{M} \frac{1}{\sqrt{2\pi\sigma'^2_k}} \exp\left\{-\frac{(x'_k - m'_k)^2}{2\sigma'^2_k}\right\} \tag{8.1}$$

where x'_k, m'_k and σ'^2_k are the transformed feature vector, mean and variance of the k component:

$$x'_k = \sum_{p=1}^{N} a_{k,p} x_p \tag{8.2}$$

$$m'_k = \sum_{p=1}^{N} a_{k,p} m_p \tag{8.3}$$

$$\sigma'^2_k = \left\{ \sum_{i=1}^{N} \left(a_{k,i} \sum_{p=1}^{N} a_{k,p} \sigma^2_{p,i} \right) \right\} \tag{8.4}$$

As indicated by Equation (8.4), the variance of the k^{th} component in the transformed feature space is computed based on the entire components $\sigma^2_{p,i}$ of the covariance matrix in the original feature space. This implicitly requires that we compute the full covariance matrix in the original feature space; otherwise we will be unable to formulate the new variance according to Equation (8.4). Also please note that $\sigma^2_{p,i} = \sigma^2_{p,i}$, as we will need this to formulate the derivative expression later.

As described earlier, we compute the derivative of the log value of likelihood with respect to the transformation matrix. The derivative will be computed for each component of the transformation matrix. We first have the log likelihood expression formulated as:

$$LogP(AX|C) = \sum_{k=1}^{M}\left[-\frac{(x'_k - m'_k)^2}{2\sigma'^2_k} - \frac{1}{2}\log\sigma'^2_k - \frac{1}{2}\log 2\pi\right] \tag{8.5}$$

Its derivative with respect to the individual component (k, p) of the transformation matrix A, $a_{k,p}$, then become:

$$\frac{\partial}{\partial a_{k,p}} LogP(AX|C) = \frac{1}{2}\sum_{k=1}^{M} Log\frac{1}{\sqrt{2\pi\sigma'^2_k}}\exp\left\{-\frac{(x'_k - m'_k)^2}{2\sigma'^2_k}\right\} = \frac{1}{2}\left[-\frac{\partial}{\partial a_{k,p}}\frac{(x'_k - m'_k)^2}{\sigma'^2_k} - \frac{\partial}{\partial a_{k,p}}\log\sigma'^2_k\right] \tag{8.6}$$

Please note that with the diagonal covariance matrix assumption, the k^{th} row of the transformation matrix will only affect the k^{th} component of the likelihood in the transformed feature space. So the derivatives of other likelihood components with respect to $a_{k,p}$ are 0.

The derivative of the first item $\frac{\partial}{\partial a_{k,p}}\frac{(x'_k - m'_k)^2}{\sigma'^2_k}$ then becomes:

$$\frac{\partial}{\partial a_{k,p}}\frac{(x'_k - m'_k)^2}{\sigma'^2_k} = -\frac{2(x'_k - m'_k)(x_p - m_p)\sigma'^2_k - 2(x'_k - m'_k)^2\sum_{q=1}^{N} a_{k,q}\sigma^2_{p,q}}{(\sigma'^2_k)^2} \tag{8.7}$$

Note that the derivatives $\frac{\partial x'_k}{\partial a_{k,p}} = \frac{\partial}{\partial a_{k,p}}\sum_{l=1}^{N} a_{k,l}x_l = x_p$, $\frac{\partial m'_k}{\partial a_{k,p}} = \frac{\partial}{\partial a_{k,p}}\sum_{l=1}^{N} a_{k,l}m_l = m_p$,and $\frac{\partial\sigma'^2_k}{\partial a_{k,p}} = 2\sum_{q=1}^{N} a_{k,q}\sigma^2_{p,q}$ from Equation (8.2) to Equation (8.4), and $\sigma^2_{p,i} = \sigma^2_{p,i}$.

Similarly, we have the derivative of the second item $\frac{\partial}{\partial a_{k,p}} \log \sigma'^2_k$ to be:

$$\frac{\partial}{\partial a_{k,p}} \log \sigma'^2_k = \frac{2 \sum_{q=1}^{N} a_{k,q} \sigma^2_{p,q}}{\sigma'^2_k} \tag{8.8}$$

Incorporating Equation (8.7) and Equation (8.8) into Equation (8.6), we then have the complete form of the derivative expression as:

$$\frac{\partial}{\partial a_{k,p}} LogP(AX|C) = -\left[\frac{(x'_k - m'_k)(x_p - m_p)\sigma'^2_k - (x'_k - m'_k)^2 \sum_{q=1}^{N} a_{k,q} \sigma^2_{p,q}}{(\sigma'^2_k)^2} + \frac{\sum_{q=1}^{N} a_{k,q} \sigma^2_{p,q}}{\sigma'^2_k} \right] \tag{8.9}$$

where x'_k, m'_k and σ'^2_k are the transformed feature vector, mean, and variance as in Equation (8.2) to Equation (8.4).

References

[1] Adjoudani, A. and Benoit, C. (1996). “On the integration of auditory and visual parameters in an HMM-based ASR”, *Speechreading by Humans and Machines*. Berlin, Germany: Springer, pp. 461-471.

[2] Berthommier, F. and Glotin, H. (1999). “A new SNR-feature mapping for robust multistream speech recognition”, *Proc. International congress on phonetic sciences*, 1999, pp.711-715.

[3] Beyerlein, P., (1998). “Discriminative model combination”, *Proc. ICASSP 1998,* Vol. 1, pp 481-484.

[4] Billa, J., Colhurst, T., El-Jaroudi, A., Iyer, R., Ma, K., Marsoukas, S., Quillen, C., Richardson, F., Siu, M., Zavaliagkos, G., Gish, H. (1999). “Recent experiments in large vocabulary conversational speech recognition”, *Proc. ICASSP 1999, Vol. 1* pp. 41 - 44.

[5] Brand, M., Oliver, N., and Pentland, A. (1997). “Coupled hidden markov model for complex action recognition”, *Proc. computer vision and pattern recognition,* 1997, pp. 994-999.

[6] Bourlard, H. and Dupont, S. (1996). “A new ASR approach based on independent processing and recombination of partial frequency bands”, *Proc. ICSLP 1996*, pp. 422-425.

[7] Chu, S. and Huang, T. (2002). “Audio-visual speech modeling using coupled hidden Markov models”, *Proc. ICASSP 2002,* Vol. 2, pp. 2009-2012.

[8] Cox, S., Matthew, I., and Bangham, A. (1997). “Combining noise compensation with visual information in speech recognition”, *Proc. Eurospeech 1997*, Vol. 1, pp. 53-56.

[9] Davis, S.B.; Mermelstein, P. (1980). “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences” *IEEE Transactions on Acoustic, Speech and Signal processing,* Vol. 28, Aug. 1980 pp. 357 -366.

[10] Deller, Jr., J.R., Proakis, J.G., and Hansen, J.H.L. (1993). *Discrete-time processing of speech signals*, Englewood Cliffs, NJ, Macmillan Publishing Company.

[11] Duda, R.O., Hart, P.B., and Stork, D.G. (2001). *Pattern classification*, Wiley, 2001

[12] Dupont, S. and Luettin, J. (2000). “Audio-visual speech modeling for continuous speech recognition”, *IEEE transaction on multimedia,* 2(3), pp. 141-151.

[13] Ellis, D.P.W. and Bilmes, J. A. (2000). “Using mutual information to design feature combinations”, *Proc. ICSLP 2000*.

[14] Ellis, D.P.W., Singh, R. and Sivadas, S. (2001). “Tandem acoustic modeling in large vocabulary recognition”, *Proc. ICASSP 2001,* Vol. 1, pp. 7-11.

[15] Fiscus, J.G., (1997). “A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)”, *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding,* 1997, pp. 347-354.

[16] Gillick, L. and Cox, S.J. (1989). “Some statistical issues in the comparison of speech recognition algorithms”, *Proc. ICASSP 1989,* Vol. 1, pp. 532-535.

[17] Glotin, H. and Berthommier, F., (2000). “Test of several external weighting function for multiband full combination ASR”, *Proc. ICSLP 2000*, Vol. 1, pp. 333-336.

[18] Glotin, H.; Vergyr, D.; Neti, C.; Potamianos, G.; Luettin, J (2001). “Weighting schemes for audio-visual fusion in speech recognition”, *Proc. ICASSP 2001,* Vol. 1, pp.173 -176.

[19] Graver, G. Alexlrod, S., Potamianos, G., and Neti, C., (2002). “Maximum entropy and MCE based HMM stream weight estimation for audio-visual ASR”, *Proc. ICASSP 2002*, Vol. 1, pp. 853-856.

[20] Graver, G., Potamianos, G., and Neti, C. (2002). “Asynchronous modeling for audio-visual speech recognition”, *Proc. HLT,* 2002.

[21] Halberstadt, A.K. (1998). *Heterogeneous Acoustic Measurements and Multiple Classifiers for Speech Recognition*, Ph.D. Thesis, MIT 1998.

[22] Halberstadt, A. K. and Glass, J. R.(1998). “Heterogeneous measurements and multiple classifiers for speech recognition”, *Proc. ICSLP 1998,* Vol. 2, pp. 995-998.

[23] Hermansky, H. (1990). “Perceptually linear predictive (PLP) analysis of speech “, *Journal of the Acoustic Society of America*, Vol. 87, pp. 1738-1752.

[24] Hermansky H.; Morgan N. (1994). “RASTA processing of speech”, *IEEE Transactions on Speech and Audio processing,* Vol. 2, pp. 578-589.

[25] Hermansky H.; Tibrewala S.; Pavel M. (1996). “Towards ASR on Partially Corrupted Speech”, *Proc. ICSLP 1996,* Vol. 1, pp 462-465.

[26] Hermansky, H., Ellis, D.P.W. and Sharma, S. (2000). “Tandem connectionist feature extraction for conventional HMM”, *Proc. ICASSP 2000,* Vol. 3, pp. 1635-1638

[27] Hernando, J., Ayarte, J., and Monte, E. (1995). “Optimization of speech parameter weighting for CDHMM word recognition”, *Proc. EuroSpeech 1995*, Vol. 1, pp. 105-108.

[28] Huang, X.D., Acero, A. and Hon, H.W (2001). *Spoken language processing, a guide to theory, algorithm, and system development*, Prentice Hall, 2001

[29] Hunt, M. et al., (1991). “An Investigation of PLP and IMELDA Acoustic Representations and of their Potential for Combination”, *Proc. ICASSP 1991*, Vol. 2, pp. 881-884.

[30] Hwang, M. Y. (1993). *Subphonetic acoustic modeling for speaker independent continuous speech recognition*, Ph.D. thesis, Carnegie Mellon University.

[31] Hwang, M. Y. and Huang, X. D. (1993). “Shared-distribution hidden Markov models for speech recognition”, *IEEE Transactions on speech and audio processing*, Vol. 1, 1993, pp. 414-420.

[32] IKbal, S., Misra, H., Bourlard, H., and Hermansky, H. (2004). “Phase autocorrelation (PAC)

features in entropy based multi-stream for robust speech recognition", *Proc. ICASSP 2004,* Vol.1, pp. 17-21.

[33] Jolliffe, I. T, (1986). *Principal Component Analysis,* SpringerVerlag, New York, 1986

[34] Jourlin, P. (1997). "Word dependent acoustic-labial weights in HMM based speech recognition", *Proc. European tutorial workshops on audio-visual speech processing,* 1997, pp. 69-72.

[35] Katagiri, S., Lee, C.H. and Juang, B.H. (1993). "New discriminative algorithm based on the generalized probabilistic descent method", *Proc. IEEE workshop on neural network for signal processing, 1991,* pp. 299-309.

[36] Kingsbury, B.E.D.; Morgan, N (1997). "Recognizing reverberant speech with RASTA-PLP", *Proc. ICASSP 1997,* Vol. 2, pp. 1259 -1262.

[37] Kingsbury B. E.D. (1998). *Perceptually-inspired signal processing strategies for robust speech recognition in reverberant environments,* Ph.D. dissertation, Dept. of EECE, UC Berkeley, 1998.

[38] Kirchhoff K. (1998). "Combining articulatory and acoustic information for speech recognition in noisy and reverberant environments", *Proc. ICSLP 1998,* vol. 2, pp. 891-894.

[39] Kirchhoff K.; Bilmes J. (1999). "Dynamic classifier combination in hybrid speech recognition systems using utterance-level confidence values". *Proc. ICASSP 1999,* Vol. 2, pp. 693-696.

[40] Kittler J, Hatef M, Duin R.P.W. and Matas J. (1998). "On combining classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 20, 1998 pp. 226 -239.

[41] Kevinen, J. and Warmuth, M.K. (1999). "Averaging expert predictions", *Proc. 4th European conference on computational learning theory,* 1999, vol. 1572 of LNAI, pp. 153-167.

[42] Kumar, N. and Andreou, A.G. (1998). "Heteroscedastic discriminant analysis and reduced rank HMM for improved speech recognition", *Speech communication,* Vol. 26, 1998, pp. 283-297.

[43] Lee, K.F., and Alleva, F.A. (1990). "Continuous speech recognition", *Recent progress in speech signal processing,* Furui, S. and Sondhi M., eds., 1990, Marcel Dekker, Inc.

[44] Li, X. and Stern, R. (2003). "Training of stream weights for the decoding of speech using parallel feature streams", *Proc. ICASSP 2003.* Vol. 1, pp. 832-835.

[45] Li, X., Singh, R., and Stern, R. (2002). "Combining heterogeneous search space for improved speech recognition", *Proc. ICSLP 2002.* Vol. 1, pp. 405-408.

[46] Lowerre, B. T. (1976). *The HAPPY speech recognition system,* Ph.D. Thesis, Carnegie Mellon University, 1976.

[47] Luettin, J., Potamianos, G., and Neti, C. (2001) "Asynchronous stream modeling for large

vocabulary audio-visual speech recognition", *Proc. ICASSP 2001*, Vol. 1, pp.169 -172.

[48] McGurk, H.and MacDonald, J. (1976). "Hearing lips and seeing voice", *Nature*, 264, pp. 746-748.

[49] Mclachlan G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York, 1992.

[50] McMahon P.; McCourt P.;Vaseghi S. (1998). "Discriminative weighting of multi-resolution sub-band cepstral features for speech recognition", *Proc. ICSLP 1998*, Vol. 2, pp. 1055-1058.

[51] Meier, U., Hurst, W., and Duchnowski, P. (1996). "Adaptive bimodal sensor fusion for automatic speech reading", *Proc. ICASSP 1996*, Vol. 2, pp. 833-836.

[52] Miyajima, C., Tokuda, K., and Kitamura, T. (2000). "Audio-visual speech recognition using MCE based HMMs and model-dependent stream weights", *Proc. ICSLP 2000*, Vol. 2, pp. 1023-1026.

[53] Nakamura, S., Ito, H., and Shikano, K. (2000). "Stream weight optimization of speech and lip image sequence for audio-visual speech recognition", *Proc. ICSLP 2000*, Vol. 3, pp. 20-23.

[54] Nakumura, S. (2001). "Fusion of audio-visual information for integrated speech processing", *Audio-and video based biometric person authentication*, Bigun, J. and Smeraldi, F. Eds. Springer-Verlag, pp. 127-143.

[55] Nawab, S.H. and Quatieri, T.F. (1988). "Short Time Fourier Transform", *Advanced topics in signal processing*, Lim, J.S. and Oppenheim, A.V. Eds, Prentice Hall, pp. 289-337.

[56] Neti, C., Potamianos, G., Luettin, J., Matthews, I., Glotin, H., Vergyri, D., Sison, J., Mashari, A., and Zhou, J. (2000). *Audi-visual speech recognition. Final workshop 2000 report*, Center for Language and speech processing, The Johns Hopkins University.

[57] Neti, C., Potamianos, G., Luettin, J., Matthews, I., Glotin, H., and Vergyri, D. (2001). "Large-vocabulary audio-visual speech recognition: A summary of the Johns Hopkins Summer 2000 Workshop", *IEEE Fourth Workshop on Multimedia Signal Processing*, 2001, pp. 619 -624.

[58] Okawa S.; Bocchieri E.; Potamianos A. (1998). "Multi-band speech recognition in noisy environments" *Proc. ICASSP 1998*, Vol. 2, pp. 641-644.

[59] Okawa, S., Nakajima, T., and Shirai, K. (1999). "A recombination strategy for multi-band speech recognition based on mutual information criterion", *Proc. Eurospeech, 1999*, Vol. 2, pp. 603-606

[60] Placeway, P., Chen, S., Eskenazi, M., Jain, U., Parikh, V., Raj, B., Ravishankar, M., Rosenfeld, R., Seymore, K., Siegler, M., Stern, R., and Thayer, E. (1997). "The 1996 Hub-4 sphinx-3 system", *Proc. DARPA Speech recognition workshop.*

[61] Potamianos, G. and Graf, H.P. (1998). "Discriminative training of HMM stream exponents

for audio-visual speech recognition", *Proc. ICASSP 1998,* Vol. 6, pp. 3733-3736.

[62] Potamianos, G. and Neti, C. (2001). "Automatic speech reading of impaired speech". *Proc. International conference on audio-visual speech processing,* 2001, pp. 177-182.

[63] Potamianos, G, Luettin, J., and Neti, C. (2001). "Hierarchical Discriminant Features for audio-video LVCSR", *Proc. ICASSP 2001,* Vol. 1, pp. 165-168.

[64] Price, P., Fisher, W.M., Bernstein, J., and Pallet, D.S. (1988). "The DARPA 1000 word Resource Management database for continuous speech recognition", *Proc. ICASSP 1988,* Vol. 1, pp. 651-654.

[65] Rabiner, L. and Juang, B.H. (1993). *Fundamentals of speech recognition*, Englewood cliff, NJ, Prentice Hall.

[66] Rogozan, A., Deleglise, P., and Alissali, M. (1997). "Adaptive determination of audio and visual weights for automatic speech recognition", *Proc. European Tutorial workshop on Audio-visual speech processing,* pp. 61-64.

[67] Rogozan, A. (1999). "Discriminative learning of visual data for audiovisual speech recognition", *International journal on artificial intelligence tools,* 8(1), pp.43-52.

[68] Saon G., Padmanabhan, M, Gopinath, R. and Chen, S. (2000). "Maximum likelihood discriminant feature spaces", *Proc. ICASSP 2000,* Vol. 2, pp.1129-1132.

[69] Singh, R., Seltzer, M., Raj, B., and Stern, R. (2001). "Speech in noisy environments: robust automatic segmentation, feature extraction, and hypothesis combination", *Proc. ICASSP 2001,* Vol. 1, pp.273-276.

[70] Talukder. A., and Casasent, D. "A general methodology for simultaneous representation and discrimination of multiple object classes", *Optical Engineering, special issue on "Advances in Recognition techniques"*, March 1998.

[71] Teissier, P., Robert-Ribes, J., and Schwartz, J.L. (1999). "Comparing models for audiovisual fusion in noisy-vowel recognition task", *IEEE transaction on speech and audio processing* 7(6): pp.629-642.

[72] Tibrewala S. and Hermansky H. (1997). "Sub-band based recognition of noisy speech", *Proc. ICASSP 1997*, Vol. 2, pp. 1255-1258.

[73] Varga, P. and Moore, R.K. (1990). "Hidden Markov model decomposition of speech and noise". *Proc. ICASSP 1990*, Vol. 2, pp. 845-848.

[74] Vergyri, D. (2000). *Integration of multiple knowledge sources in speech recognition using minimum error training,* Ph.D. Thesis, Center for speech and language processing, The johns Hopkins University.

[75] Viterbi A. (1967). "Error Bound for conventional codes and an asymptotically optimum decoding algorithm", *IEEE Transactions on Information Theory,* Vol 13, 1967, pp.260 -269.

[76] Winston, P.H. (1992). *Artificial Intelligence*, 3rd, Ed. 1992, Reading, MA, Addison-Wesley.

[77] Wu, J. and Huo, Q., (2002). "Supervised adaptation of MCE-trained CDHMMs using minimum classification error linear regression", *Proc. ICASSP 2002*, Vol. 1, pp. 605-608.

[78] Wu, S. L.; Kingsbury, B.E.D.; Morgan, N.; Greenberg, S (1998). "Incorporating information from syllable-length time scales into automatic speech recognition", *Proc. ICASSP 1998*, Vol. 2, pp. 721 -724

[79] Xu, L., Krzyzak, A., and Suen, C.Y. (1992). "Methods of combining multiple classifiers and their applications in handwriting character recognition", *IEEE Transaction on systems Man and Cybernetics*, 22(3): pp.418-435.

[80] Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. (1999). *The HTK Book*, Enbtropic, ltd.