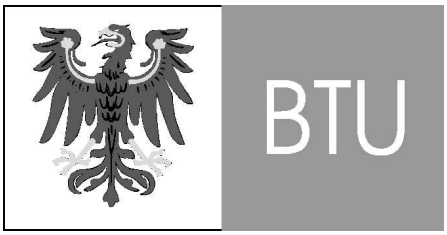
Discriminative Connectionist Approaches for Automatic Speech Recognition in Cars



Brandenburgische Technische Universität Cottbus

Von der Fakultät für Maschinenbau Elektrotechnik und Wirtschaftsingenieurwesen der Brandenburgischen Technischen Universität Cottbus zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigte Dissertation vorgelegt von

Enginyer de Telecomunicació

Joan Marí Hilario

geboren am 18.09.1973 in Maó, Spanien

Vorsitzender: Prof. Dr.-Ing. habil. C. Henschel
Gutachter: Prof. Dr.-Ing. K.-R. Fellbaum
Gutachter: Prof. Dr.-Ing. R. Hoffmann
Tag der mündlichen Prüfung: 31.08.2004

Danksagung

Diese Arbeit entstand während meiner Zeit als Doktorand beim DaimlerChrysler Forschungszentrum in Ulm. Die in dieser Arbeit vorgelegten experimentellen Untersuchungen wurden zum größten Teil im Rahmen der von EU-Projekten SPHEAR und RESPITE durchgeführt.

An dieser Stelle möchte ich allen danken, die die Durchführung dieser Arbeit ermöglicht und unterstützt haben. Besonders danken möchte ich meinen Betreuer bei der DaimlerChrysler AG, Herrn Dr. Udo Haiber, ohne dessen Ermutigung, kluge Anregungen und geduldiges Korrekturlesen dieser Arbeit nie zu Stande gekommen wäre. Ebenso möchte ich mich bei Herrn Dr. F. Class und Herrn Dr. A. Kaltenmaier für ihre Unterstützung und Diskussionbereitschaft ganz herzlich bedanken. Bei den anderen Kollegen und Kolleginnen in der Abteilung RIC/AD möchte ich mich ebenfalls bedanken, besonders bei meinem Kollegen Herr Dr. André Berton und bei meiner Lieblingssekretärin Gabriele Maier. Auch bei den Diplomanden und Werkstudenten J. Ferrer, V. Maier, S. Sakti und A. Macià möchte ich mich für ihre Beihilfe bei den Untersuchungen bedanken. Einen besonderen Dank auch an Herrn Prof. Dr. K.-R. Fellbaum, der die Betreuung und Begutachtung dieser Arbeit übernommen hat, und auch an Herrn Prof. Dr. R. Hoffmann für die Übernahme des zweiten Gutachtens.

Schließlich möchte ich mich ganz herzlich bei den Kollegen der Projekte RESPITE und SPHEAR bedanken, insbesondere bei Phil Green, Prof. Dr. J. Blauert, A. Morris, A. Hagen, C. Tsakostas, J. Buchholz, C. Ris, D. Ellis and J. Barker. Natürlich möchte ich mich an dieser Stelle bei allen bedanken, die zwar nicht direkt zu der Vollendung dieser Arbeit beigetragen haben, die aber ihre Freizeit mit mir geteilt haben. Der besondere Dank geht an Claudia, aber auch an Anette, Hans, Oriol, Ana, Billy, Carmen und Sandra, Rober, Chus, Fabrizio, Rino, Claudia, Stefi und Jochen, Ralf, Alex und an alle anderen, die aus reinen Platzgründen hier leider nicht erwähnt werden können.

Zusammenfassung

In der automatischen Spracherkennung wird die Erkennungsgenauigkeit sehr von den Umgebungsgeräuschen beeinflusst. Dies gilt insbesondere für Anwendungen in stark geräuschbehafteten Umgebungen, wie z. B. bei sprachgesteuerten Navigationssystemen im Fahrzeug, weil hier die Variabilität der Geräusche besonders groß und wenig vorhersehbar ist. Zur Verbesserung der Erkennungsgenauigkeit kommen unterschiedliche Methoden zum Einsatz, denen zumeist spezielle Annahmen über die Art der Störungen zugrunde liegen. Wenn die störenden Geräusche diese Eigenschaften nicht besitzen, können die Störungen nicht mehr kompensiert werden und die Erkennungsgenauigkeit sinkt. In den letzten Jahren sind jedoch neue Methoden vorgeschlagen worden, die lediglich auf der hohen Redundanz des Sprachsignals beruhen. Deshalb brauchen sie keine Annahmen über die Störungen selbst zu machen und sind daher zumindest in der Theorie unabhängig von speziellen Eigenschaften der Störung.

Im ersten Teil dieser Arbeit wird systematisch die Leistung einiger der auf der Ausnutzung der Redundanz basierender Ansätze miteinander und diese wiederum mit konventionellen Methoden zur robusten Spracherkennung verglichen. Die Basis der Arbeiten bildet dabei eine erste Evaluation der Redundanz-Methoden, die innerhalb der EU-Projekte RESPITE und SPHEAR durch die Projektpartner erfolgt ist. Dieser Evaluation wird auf der AURORA 2000 Datenbank durchgeführt, welche Handy-Sprachaufnahmen in unterschiedlichen Umgebungen, wie z.B. im Kraftfahrzeug, enthält. Auf der Grundlage dieser Evaluation sowie dem darauf basierenden Vergleich werden diejenigen beiden Ansätze weiter ausgearbeitet und detaillierter analysiert, die hierbei die besten Ergebnisse erzielt haben. Dabei ist ein Ziel der vorliegenden Arbeit, die Leistung dieser Methoden speziell auf einer im Kraftfahrzeug aufgenommenen umfangreichen Datenbank zum ersten Mal methodisch und quantitativ zu untersuchen.

Der erste dieser Ansätze verbindet die herausragende Klassifikationsleistung von neuronalen Netzen mit radialen Basisfunktionen (RBF) mit der Fähigkeit von Hidden-Markov-Modellen (HMM), Zeitveränderlichkeiten zu modellieren. In einem zweiten Ansatz werden NN zur nichtlinearen Dimensionsreduktion hochdimensionaler Kontextvektoren in unterschiedlichen Netzwerk-Topologien untersucht. In den experimentellen Untersuchungen konnte gezeigt werden, dass der erste dieser Ansätze für die AURORA-Datenbank eine ähnliche Leistungsfähigkeit wie semikontinuierliche HMM (SC-HMM) aufweist. Der zweite Ansatz erzielt auf einer umfangreichen im Kraftfahrzeug aufgenommenen Datenbank keine Verbesserungen gegenüber den klassischen linearen Ansätzen zur Dimensionsreduktion (LDA), erweist sich aber auf der AURORA-Datenbank als signifikant überlegen.

Abstract

Environmental noise is one of the major factors that affect the performance of automatic speech recognition in real world applications, such as in-car navigation systems. To improve the recognition accuracy in noisy environments, a number of noise robust techniques, e.g. spectral subtraction, are being used in speech recognition systems for commercial applications. All these approaches assume that the interfering noises have certain properties. If the interfering noises do not show these properties, the current robust approaches completely or partially fail to compensate for their deleterious effect. In the last years, however, new kinds of robust approaches have been proposed which do not make any assumptions about the interferences. Consequently, their performance is not affected by changes in the properties of noise. These approaches rely on the high redundancy of the speech signal to compensate for any changes in the input signal.

In the first part of this thesis we compare, for the first time, the performance of some redundancy-based approaches with each other and with conventional robust approaches. These redundancy-based approaches were proposed in the EU-projects SPHEAR and RESPITE by our project partners. This evaluation is carried out on the AURORA 2000 database, which is a digit recognition task in different noisy environments, typical for mobile phone applications. The results of this evaluation show that the best-performing approaches are those that combine the use of multiple streams of features with some kind of discriminative approach based on neural networks. The final goal of this thesis is to assess, for the first time, the performance of the previous approaches on an real in-car task.

The first of the previous approaches is the hybrid RBF/HMM, which is an attempt to combine the superior classification performance of radial basis functions (RBFs) with the ability of HMMs to model time variation. Moreover, RBFs can be discriminatively trained with low computational effort. The second of the mentioned approaches is to use neural networks for non-linear discriminative feature reduction, since this problem is closely related to classification. In this thesis, we propose the use of different MLP topologies to reduce the dimensionality of large feature vectors including context frames.

Experiments on the AURORA 2000 database reveal that the performance of the first approach is similar to the performance of semi-continuous HMMs. The second approach, however, cannot outperform the conventional linear discriminant analysis (LDA) on an in-car database, although its performance is superior on average to that of LDA on the AURORA 2000 database.

Contents

1. Introduction

In this chapter we introduce the field of automatic speech recognition (ASR) and the scope of this thesis within this field. In particular, we explain the concept of *variability* of the speech signal, and how this enormous variability makes automatic speech recognition a challenging task even with 'state-of-the-art' computers. The sources of this variability are various, but one that has attracted much attention in the last years is the environment in which the ASR system operates. This environment includes any noises and channel distortions that degrade the quality of the speech signal. The topic in ASR that deals with this problem is known as *noise robustness*, and constitutes the framework of this thesis. In the final point, the contents of the individual chapters are briefly summarized and a figure displaying the thesis' outline is also shown.

1.1 Automatic Speech Recognition

In a wide sense, the objective of ASR is to design and construct machines that can understand the information contained in the speech signal (speech understanding), and carry out a certain task according to the received information. In addition, those machines should be able to communicate with humans using synthetic speech. From this point of view, ASR is a part of the search for the intelligent machine that is able to 'hear', 'understand', 'act' and 'speak'. The paradigm of this kind of machine is the supercomputer HAL in Stanley Kubrik's famous film 2001, which was able to talk with humans as a normal human about any theme, and even take its own, albeit fatal, decisions and actions. From a different viewpoint, the objective of ASR is also to transcribe the linguistic contents of the speech signal into sentences and words. Nevertheless, this interpretation is inadequate for the case of spontaneous speech, where many extra-linguistic phenomena and incomplete sentences occur, and is restricted to the case of read speech or commands.

Unfortunately, our state-of-the-art ASR technology is still far away from this science-fiction paradigm, since the ASR systems used today in real world applications are always restricted to working just for the application concerned, i.e. to recognize a limited amount of possible sentences and words. In addition, current ASR systems are very sensitive to changes in the environment (noises, channel distortions, etc...), and usually perform much better if they are adapted to a particular speaker. In spite of the differences between reality and fiction, the progress made in the nearly fifty years of ASR has been enormous, and the number of application fields of this technology is still growing, although the bad

sales on the market of speech transcription systems has recently reduced the enthusiasm towards ASR.

The main reason for this contrast between reality and fiction is the extraordinary *variability* of the speech signal. As explained in [JH96], this variability comes from a variety of sources that fall into one of the following categories:

- **inter-speaker**, which groups all the variations due to differences between speakers in the physiological and articulatory habits, such as sex, age or vocal-tract anatomy.
- **context**, which includes co-articulation between phonemes, linguistic context such as syntax, semantics and pragmatics (context of the conversation), and social interaction.
- **environment**, which groups all the phenomena that affect the speech production, transmission and perception process. Typical phenomena are environmental noises, room acoustics or channel distortions.
- **intra-speaker**, which includes all the variations associated with the different speaking styles of a given speaker. In fact, a speaker can change his voice quality or his speaking rate depending on his physiological and psychological state.
- **linguistic**, which includes all the variations associated with the dialect or accent of the speaker.

The boundaries between these categories, however, are not always clear because there are mutual interactions between the different sources. Despite that, this classification is useful in showing the large number of variability sources that have an effect on the observed and perceived form of the speech signal.

Most of this variability is not relevant to speech recognition and must therefore somehow be suppressed or accounted for during the ASR process. For example, inter-speaker variability is of no linguistic relevance and must consequently be compensated for before utterance decoding begins.

In Fig. 1.1 we show a time/frequency representation of the uttered sentence '`switch the radio off`', to give an idea of the frequency composition of the speech signal. Note the non-stationary nature of the variations in the speech spectrum, which is a result of the different shapes assumed by the vocal tract to produce the speech sounds. For example, the initial `/s/` in the sentence has only components in the high frequency region (4000-8000 Hz) of the spectrum and has no 'striped' appearance, because it is an unvoiced sound and consequently the vocal chords do not vibrate. By contrast, the `/I/` in '`switch`' exhibits the typical striped appearance of voiced sounds, and has only components in the low frequency region. Note also that this last sound has high energy regions in the spectrum, which means that this sound has a formant structure.

Since the handling of this enormous number of sources of variability is impossible with the current knowledge and technology, we are forced to restrict the scope of current ASR systems to particular tasks. The complexity of a certain task can be measured using the following 'dimensions of difficulty' as were defined in [DPH93]:

- **Speaker dependence or independence**. An ASR system that is used by more than one speaker must be able to deal with the variability in the speech signal

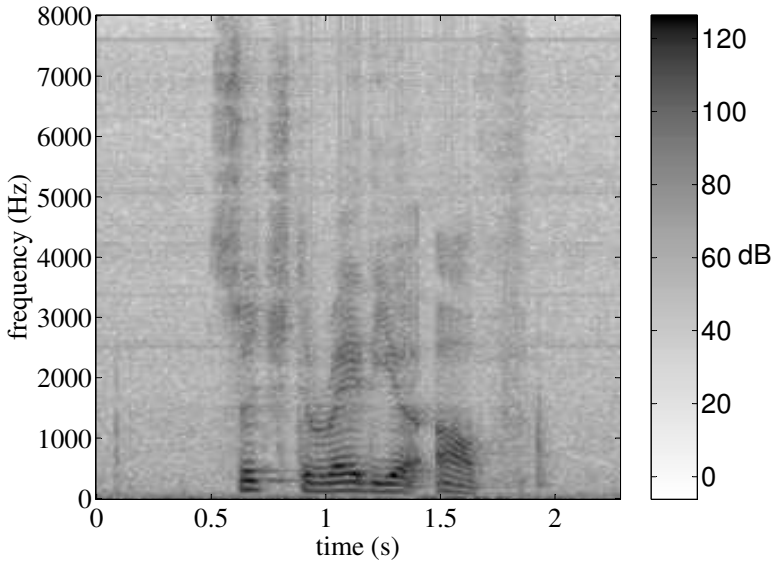


Fig. 1.1: A time/frequency plot of the utterance 'switch the radio off'. The dark regions represent the high energy levels whereas the light ones represent low energy levels.

induced by the differences between speakers. In fact, it is often the case that speaker dependent ASR systems (one speaker) perform noticeably better than their speaker independent counterparts. However, for a wide range of tasks, e.g. applications over the telephone net, speaker independency is a must since the number of potential users is huge, so that such kind of systems need large training data sets with many speakers to have an acceptable performance level. To improve the performance of speaker independent systems for single speakers, some type of *speaker adaptation* approach is often included in the speech recognition process. More about those approaches can be found in [Hai98].

- **Nature of the utterances**. Depending on the nature of the task in hand, there are basically two different kinds of recognition strategies: *isolated speech recognition* and *continuous speech recognition*. In the first case, the words in the utterance are pronounced with pauses between them, so that the speech segments passed to the decoder contain only one word. This simplifies enormously the decoding task, but requires some degree of cooperation from the speaker. In continuous speech recognition, by contrast, the words in the utterance are pronounced without pauses between them, and it is therefore impossible to establish, before decoding, where the words begin and end. This fact complicates the decoding stage of ASR because the number of hypotheses to keep track of is much larger, but no cooperation from the speaker is needed to successfully decode the utterances. Another important distinction is the difference between *read speech* and *spontaneous speech*. The latter contains many extra-linguistic phenomena (hesitations,etc...) and grammatically uncorrect sentences, and is therefore very difficult to transcribe into words.

- **Size of the lexicon**. As a general rule, the larger the lexicon, the more complex is the search space in the decoding stage (cf. Fig. 2.1), and therefore the longer the recognition time and the larger the memory needed. As a consequence, for

large vocabulary applications mechanisms such as lexicon trees (cf. Sec. 2.4.2) and language model look-ahead [NO00] are used to reduce the number of operations per frame. For large lexicons, the performance scores are also in general lower than for small lexicons, due to the higher confusability between the words. However, there are also small lexicons, such as the alphabet letters, which are also highly confusable.

- **Confusability and ambiguity of the lexicon**. The first term refers to the acoustic similarity of words in the vocabulary, e.g. 'B' and 'D' in a spelling task, whereas the second term refers to the amount of homophonous words in the lexicon, e.g. 'know' and 'no' or 'to' and 'two', which are not acoustically distinguishable. The more confusable and ambiguous words there are in a vocabulary, the lower in general is the performance on the task. As already mentioned, the larger the lexicon, the more probable it is to have confusable and ambiguous words, and therefore the lower is the performance.

- **Language complexity**. In continuous speech recognition it is usual to limit the number of possible utterances by using some kind of grammar or statistical language model (cf. Sec. 2.4.3). The complexity of these grammars or models determines also the size of the search space in the decoding step, and consequently the recognition time and the memory use.

- **Environmental factors**. These factors are very important in real world applications of ASR where the production, transmission and reception of the speech signal may be affected by distortions. In fact, this kind of factors can dramatically reduce the performance of an ASR system, especially when the environmental conditions during recognition are very different from those seen in the training stage of the speech models used in recognition [LMP87].

The complexity of any task can be characterized using these six dimensions, and it is therefore possible to deduce from them the requirements that our ASR system must fulfil. Unfortunately, this characterization of the task is not equal to a parametrization of the task, since most of the dimensions are difficult to quantify numerically. This parametrization would be very useful to estimate the performance of a given ASR system on a certain task, for which no speech database is available [PL95].

1.2 Noise Robustness in ASR

The topic in ASR that studies the improvement of ASR performance in noisy and distorted environments is known as *noise robustness*, and constitutes the framework of this thesis. An ASR technique is noise robust in the wide sense, if it is able to deal with a wide range of different environmental conditions.

The distortions of the speech signal can be additive (noise) or convolutive (channel responses). The first type of distortion is linear in the power spectral domain, whereas the second is linear in the log-spectral or cepstral domain. A combination of both types of distortion is often the case in practical ASR, which it is even more difficult to handle. Although in general environmental noise is additive to the speech signal, this is not always the case. In fact, the environmental noise affects the speech production process, and therefore changes the shape of the uttered speech signal. This phenomenon is known as

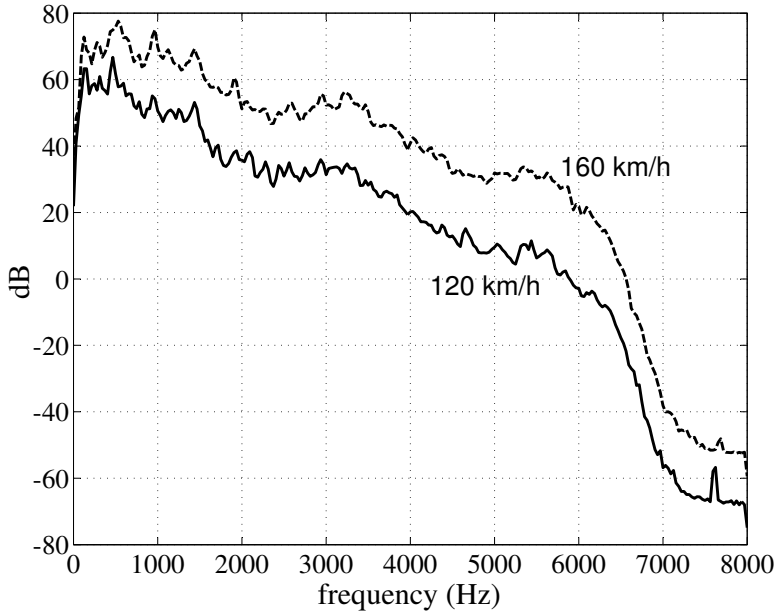


Fig. 1.2: Power spectrum of the noise at different speeds. Note that the average increase over all frequencies is about 20 dB. The increase in noise power is not linear with the speed, since the higher the speed the larger is the increase in the noise power. The power spectral density (PSD) has been computed using the Welch method, with 16 kHz sampling frequency, Hamming window of 32 ms, shifted every 10 ms.

Lombard reflex [LT71, Jun93], and it is the natural increase in our vocal level when the environmental noise level is high enough. This increase cannot be depicted as a simple loudness increase because to increase the vocal level we modify our articulatory movements, and consequently the shape of the generated speech signal. This phenomenon has a greater impact on the performance of speaker-dependent ASR than the additive noise itself [JW89]. However, for speaker independent ASR the impact of the Lombard reflex on the performance is not so large, since the variability introduced by this phenomenon is already covered by the variability introduced in the speech models by the different speakers [JH96].

In a car the sources of noise can be classified into two groups: those noises coming from outside and those coming from inside. Outside noises are caused by the car itself and its movement. Most of these noises are low frequency noises due to mechanical sources (engine, tires) and flat spectrum phenomena produced by aerodynamic phenomena. The resultant outside noise is depicted in Fig. 1.2 for two different speeds. In fact, the power of the noise increases in a non-linear way with the speed of the car. In contrast, inside noises come from a variety of sources such as the passengers, audio equipment, acoustic signals (turn signal) or windscreen wiper. The variety of resulting noises is depicted in Fig. 1.3, where we can see that some of these noises have a non-stationary nature.

During the nineties, a great number of approaches to improving the performance of ASR of noisy or distorted speech have been proposed. Most of these approaches can be roughly classified in one of the following categories:

- **Speech enhancement**. These techniques reduce the amount of noise or distor-

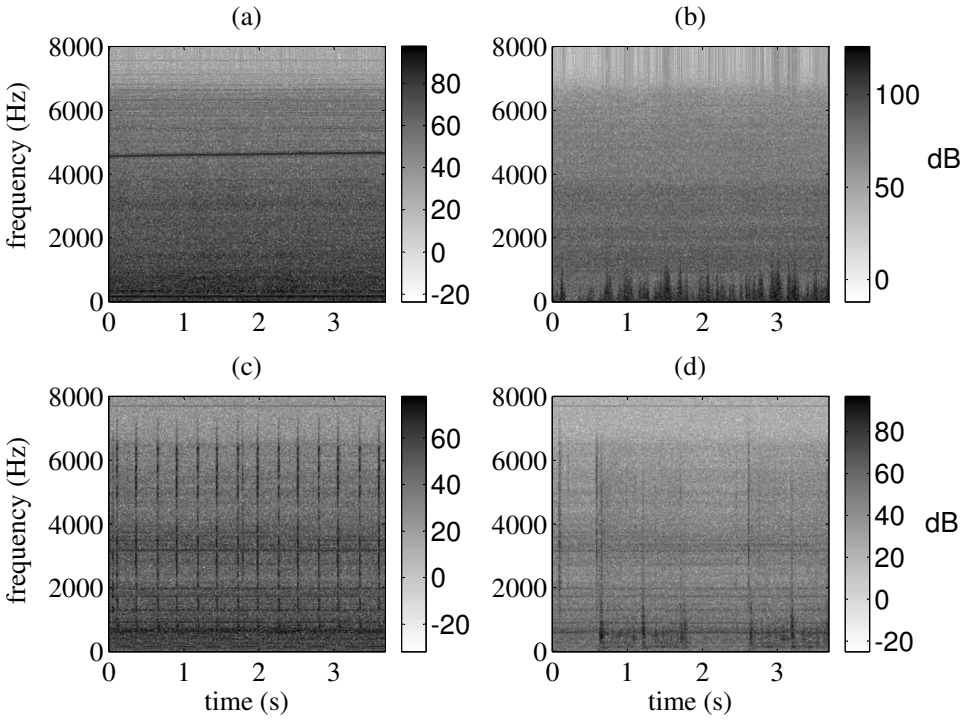


Fig. 1.3: Different noise spectrograms in a typical in-car environment. In (a) we can see the typical spectrogram of noise in a car at 80 km/h with closed windows. The spectral line at 4600 Hz is a warning signal. By contrast, figure (b) shows the noise spectrogram at 80 km/h with open windows and wind noise. In figures (c) and (d) we see the noise spectrogram at 0 km/h when the turn signal and the windscreen wiper are on, respectively. Spectrogram has been computed with 16 kHz sampling frequency and Hamming window of 32 ms shifted every 10 ms.

tion in the speech signal. A frequently used technique in this category is spectral subtraction [LB91, Bol79], which basically estimates the spectrum of the clean speech by subtracting the noise spectrum from the spectrum of the noisy speech. Other techniques use statistical filtering techniques such as Wiener or Kalman filtering [LOB78, KGG89]. Speech enhancement techniques are usually implemented in the 'pre-processing' block of Fig. 2.1.

- **Robust feature extraction**. The techniques grouped under this category use a feature extraction algorithm (cf. Sec. 2.3) that is relatively 'robust' to noise or channel distortions, in the sense that the feature vectors obtained from distorted speech are similar to those obtained from undistorted speech. Many techniques exist in this category, and range from the Perceptual Linear Predictive (PLP) features (cf. Sec. B.1) based on human perception of speech, to the Cepstral Mean Subtraction (CMS) technique, which basically subtracts the mean of the cepstral vector to cancel any channel distortion [RLS94].

- **Model Compensation**. These kind of techniques use speech models trained on clean speech, which are then adapted to the noisy speech during recognition. Among the techniques in this category we can mention parallel model combination (PMC) or HMM decomposition [GY93, VM90]. Another frequently used technique in this

category is to train the speech models with noisy speech, which has been shown to achieve good performance in noise [DRM83, LMP87].

All the techniques in the categories above, however, have their drawbacks. For the techniques in the speech enhancement category, it is sometimes the case that the speech signal is distorted during enhancement, which is at least as detrimental as the noise itself to ASR performance. Robust feature extraction techniques often achieve good performance in noisy conditions, but comparatively poor in clean speech. The model compensation techniques usually make strong assumptions on the noise modeled, which leads to poor ASR performance when the environmental noises do not match these assumptions.

Since most of the techniques in these three categories were not robust in the wide sense, some researches in the late nineties started to apply new approaches to noise robustness that do not distort speech or make assumptions on the environmental noises. Among these approaches we can mention missing data theory [CMG96], multi-band speech recognition [BD96] and multi-stream speech recognition [JEM99]. All these approaches have in common the use of *redundancy* to improve the match to the speech models. This redundancy can be inherent to the speech signal itself, as in the missing data and multi-band, or rather added to the speech representation as in the multi-stream approach. More about these new approaches can be found in Chapter 3, where the performance of some techniques based on the previous approaches will be evaluated.

1.3 Thesis Outline

The objectives of this thesis work are the following:

- assess the performance in environmental noise of a variety of robust approaches based on the use of redundancy, and compare their performance with that of our baseline system on a common speech database. The purpose of this evaluation is to find an approach likely to improve the performance of our current ASR system in noise, but which also could be implemented into a real-time ASR system. As will be seen in Chapter 3, the best results in this evaluation are obtained by the approaches that combine a better discrimination (using for that purpose a neural net) with the use of multiple streams of features (multi-stream).

- study the use of neural networks for discriminative feature reduction, and the relation of those neural approaches (also termed *connectionist* approaches) to the usual *linear discriminant analysis* (LDA).

- study the *hybrid radial basis functions/hidden Markov model* approach (hybrid RBF/HMM) with full-covariance normal basis functions as a practical alternative to the more computational intensive *hybrid multi-layer perceptron/hidden Markov model* approach (hybrid MLP/HMM).

- study the performance of multi-stream combinations of feature extraction algorithms in noisy conditions and on tasks with different complexities.

The diagram in Fig. 1.4 shows schematically the integration of the objectives above into the development of this thesis. We start in Chapter 2 with an overview of a generic ASR system and its different components. The objective of this chapter is to give a solid

basis of understanding before Chapter 5, Chapter 4 and Chapter 6, in which we explain the different approaches tested in this thesis. We follow in Chapter 3 with the start point of this thesis, namely the evaluation of 'new' ASR techniques to improve ASR performance in noisy environments. We first discuss the difference between user-centered and technology evaluation. Next we describe the elements of technology evaluation, with special emphasis on the measures for technology evaluation on ASR tasks, e.g. hypothesis/reference alignment, measures, confidence intervals and statistical tests. Since the techniques to be evaluated were implemented in different systems, and we wanted to keep the results of the different approaches comparable, we have designed two different interface levels, at the feature and state-probability levels, to use the same decoding or acoustic modeling in all the assessments. After that, we introduce the different approaches evaluated and their configurations. Finally, we present and discuss the evaluation results obtained on the AURORA 2000 task. This database is a digit recognition task with artificially-added background noises. As will be seen, the best performing systems in this evaluation are those that combine multi-stream with some kind of discriminative approach based on neural networks.

The first of these neural approaches is introduced in Chapter 4. This is based on the use of an MLP to reduce the dimensionality of the feature vector. We will see in this chapter that the problem of finding the optimum mapping for feature reduction is similar to the problem of finding the optimum classifier. In general, the optimum mapping is non-linear, but if the classes are normally-distributed with equal covariance matrices, then the optimum mapping is linear and corresponds to the solution found by LDA. However, classes for feature reduction are usually non-normal and have different covariance matrices. An alternative way to solve this problem is to use neural nets, because they have demonstrated to be powerful classifiers.

The second neural approach, known as hybrid artificial neural network/ hidden Markov model (hybrid ANN/HMM), is discussed in Chapter 5. In particular, we discuss the hybrid RBF/HMM approach, since it can be easily integrated into our current ASR system and is not as computationally expensive as the usual hybrid MLP/HMM approach.

We devote Chapter 6 to the multi-stream approach. Especially, we will study the combination of two streams of features using *feature concatenation* and *probability combination*. For the former case, we present an approach that concatenates the LDA vectors of two different feature streams, and reduces the dimensionality of the concatenated vector using also LDA.

In Chapter 7 we present the experimental results using the approaches discussed in the previous three chapters. These experiments have been carried out on two different tasks: the AURORA 2000 and the UKKCP in-car tasks. The UKKCP task is a typical tasks for in-car applications (commands, city names and spelling) that has been recorded in real environments. Therefore, and in contrast to the AURORA task, the noises have not been artificially added.

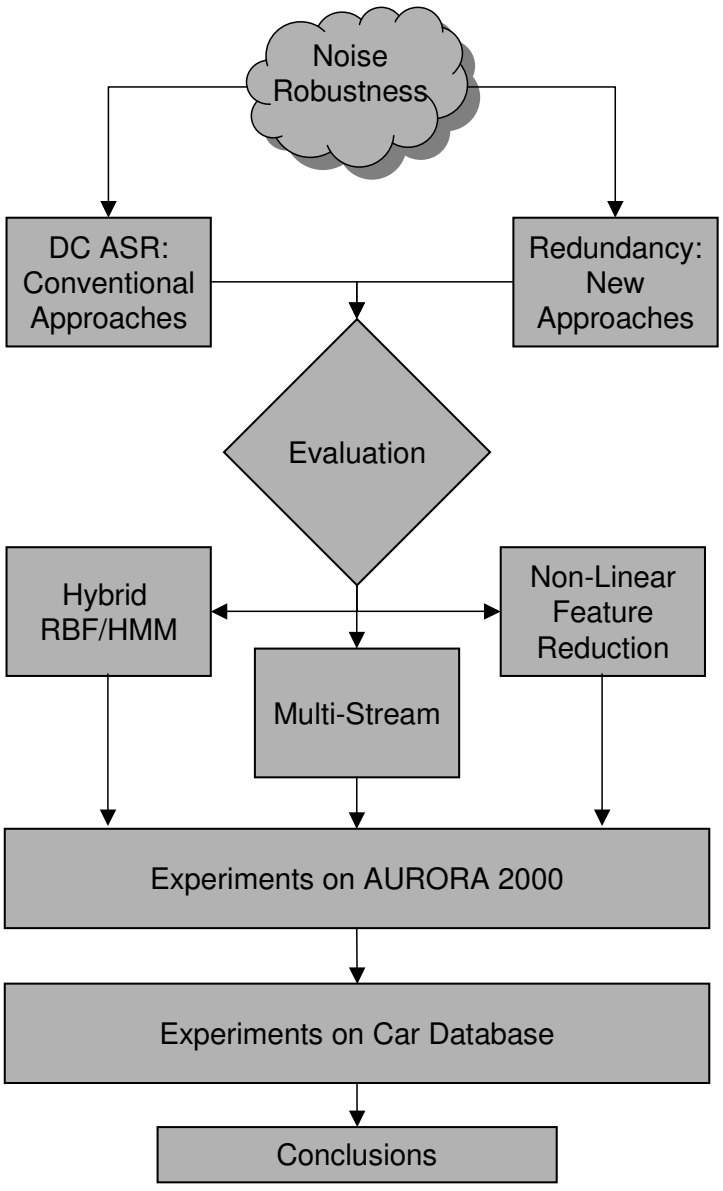
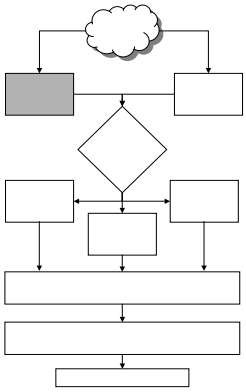


Fig. 1.4: A diagram showing the stages in the development of this thesis.



2. Automatic Speech Recognition System Architecture

In this chapter we introduce a prototypical ASR system from which virtually all current state-of-the-art ASR systems are derived. Since this kind of ASR systems uses pattern recognition to recognize the linguistic content, this approach to ASR is often called the *pattern recognition approach* [RJ93]. Almost all the current systems following this approach use hidden Markov models (HMM) to model the acoustic variability of the speech signal not only in time but also across sentences or speakers. In the first section of this chapter we present a block diagram of a generic state-of-the-art ASR system, and succinctly describe how the blocks coordinate to perform the ASR task. The following sections of the chapter are devoted to the single blocks. As this thesis focuses on the use of pattern recognition methods for ASR, we will put special emphasis on this topic in Sec. 2.4 where HMMs and their use in ASR will be described.

2.1 Introduction

The main blocks of the ASR process are schematically depicted in Fig. 2.1. The first of these blocks is the pre-processing block which is basically an A/D conversion, but may contain other processing sub-blocks as well to reduce the amount of noise or distortion present in the speech signal.

In essence, this kind of ASR system performs the following two fundamental operations:

1. discard the information in the speech signal which is not relevant to speech recognition.

2. using this relevant information find which is the most probable word sequence or sentence uttered by the speaker.

The first operation is totally or partially based on evidence from the human perception system, whereas the second operation relies on a combination of statistical pattern recognition theory and Linguistics (phonology [1] and syntaxis).

[1] In anglo-saxon literature this science is sometimes termed *phonemics*

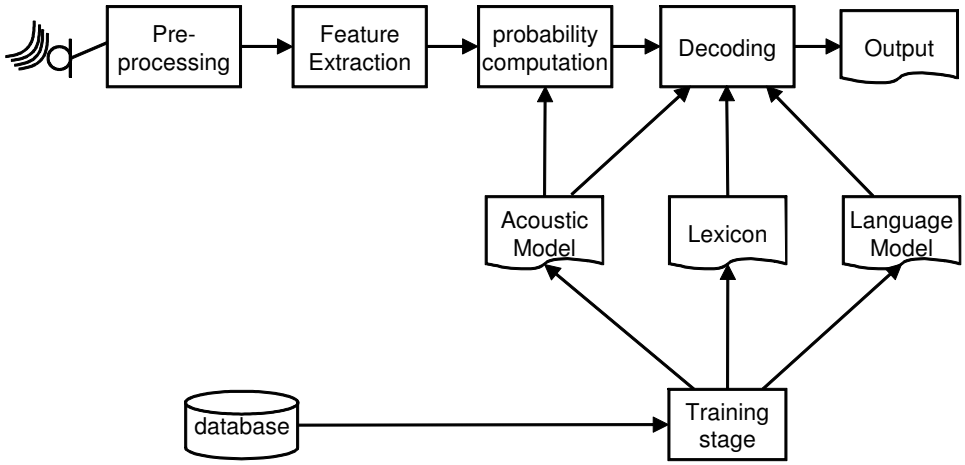


Fig. 2.1: ASR system based on statistical pattern recognition.

The pre-processing and the feature extraction blocks carry out the first operation, in order to obtain features with a high content of linguistic information and as low as possible content of non-relevant or superfluous information to speech recognition. At the output of the feature extraction block we have a sequence of vectors or *patterns* $X = (\mathbf{x}^1 \dots \mathbf{x}^T)$ which is ideally a compact representation of the sequence of the linguistically relevant sounds uttered by the speaker. This compactness of the representation is, from an engineering point of view, the main benefit of the feature extraction block, because it reduces the flow of information going into the probability computation block. As the processing time in this block is strongly dependent on the size of the feature vector at its input, it is advisable to reduce the size of the input feature vector to just the necessary.

On the other hand, the joint task of the acoustic modeling and decoding blocks is to search, among the set of all possible sentences (the so-called search space), for the uttered sentence or word sequence that best matches the pattern X. As we will see in Sec. 2.5, this is done by computing the probability of the most probable word sequence given the input pattern X in a very efficient way.

To perform this computation the ASR system in Fig. 2.1 uses the three following components, which are previously calculated during the *training* stage:

- **acoustic models** of words or sub-word acoustic-phonetic units, e.g. phonemes, which model the acoustic variability of the speech sounds due to the factors already discussed in Sec. 1.1.

- **recognition lexicon** which is a mapping between each of the words in our ASR task and the acoustic model associated with each of them. As it will be explained in Sec. 2.4.2 this word model can be monolithic and specific to the word it models, or can consist of a sequence of concatenated sub-word units.

- **language model** that controls the possible word sequences allowed by the syntax and semantic of the language.

The parameters of the acoustic models are normally jointly trained to account for the co-articulation effects between words or between phonemes. This is achieved by concatenating the acoustic models of the words in a sentence to form a large acoustic model of the

sentence. This large model is then trained using the pattern associated with the sentence, and applying the equations detailed in Sec. 2.6.

In a completely independent procedure the language model is trained or deduced from the set of possible sentences in our ASR task. The language model conveys, albeit restricted to a particular domain, the syntactic and semantic rules of the language, which impose certain constraints on the possible word sequences in that language (cf. Sec. 2.4.3).

2.2 Speech Pre-Processing

As already mentioned, the basic task of this block is to perform an A/D conversion. In a first step the speech signal is filtered using a low-pass filter to avoid aliasing and to suppress the high frequency noise. The cut-off frequency of the filter is usually less than the Nyquist rate to have low signal content around that frequency. Next the signal is usually sampled and quantized at 8 kHz and 8 or 16 bits/sample for telephone applications or at 16 kHz and 16 bits/sample for applications where the frequency content of the speech signal is not limited as for telephone applications.

To further reduce the noise content of the digitized speech signal some kind of noise reduction algorithm such as spectral subtraction [Bol79, LB91], Wiener or Kalman filtering [LOB78, KGG89], can be applied to the speech signal.

2.3 Feature Extraction

In this step the time sequence of speech samples is converted into a sequence of feature vectors each bearing the relevant linguistic information in the speech signal at a particular point in time. To perform this operation it is assumed that the speech signal is a piece-wise or short-time stationary process. This assumption is reasonable since, although over long periods of time (on the order of 1/5 sec or more) the speech signal characteristics change to reflect the different speech sounds being spoken, when examined over a sufficiently short period of time (between 5 and 100 ms), the speech signal characteristics are approximately constant.

A feature vector is obtained by first windowing the sequence of speech samples with a Hamming window of 20 to 30 ms length centered at a particular point of time. The sequence of feature vectors is generated by shifting the window every 10 ms so that a sequence of window frames is gained. Each of those window frames is then further processed to obtain the desired sequence of feature vectors.

This processing is intended to extract the relevant linguistic information encoded in the speech signal and at the same time remove as much non-relevant information as possible, which would otherwise introduce unnecessary variation in the sequence of feature vectors. As already mentioned in Chapter 1 this unnecessary variation comes from a variety of sources.

There are basically three kinds of feature extraction methods:

- **auditory-based methods**, e.g. mel-filter cepstral coefficients (MFCC), which are partially or totally based on concepts of speech perception.
- **production-based methods**, e.g. LPC-cepstrum, which are based on speech production models.

Since it has been empirically found that the LPC-cepstrum is more sensitive to noise-induced variation in the speech signal [JH96], most of the current state-of-the-art systems use the so-called mel-cepstral coefficients (MFCC) [DM80], which is based on a filter-bank, and is consequently an auditory-based technique. As depicted in Fig. 2.2, this technique basically consists of transforming each of the windowed frames into a spectral representation using a Fast Fourier Transform (FFT). Next the power spectrum of each frame is computed by taking the square of the modulus of the FFT, which is then processed in the spectral domain using a filter-bank. This usually consist of a bank of 16-20 triangular or auditory-based filters of equal length and equally spaced in the mel-frequency scale [RJ93]. In the linear frequency domain this results in a bank of filters that is unequally distributed over the frequency axis. Also, the bandwidth of the filters in the bank is much larger in the high- than in the low-frequency region. This filter-bank simulates the critical-band behavior of the human auditory system: a sound whose frequency is within a certain critical-band can influence the perception of the sound in the same band but not outside [JH96].

From a signal processing point of view, the main benefits of the filter-bank are to remove the *pitch* [2] frequency, and to reduce the signal rate. The former operation is equivalent to a non-linear smoothing in the frequency domain whereas the latter is in essence a down-sampling of the smoothed spectrum [DPH93].

Although the MFCCs as described above attain a good performance level (up to 99% connected digit recognition rate in clean conditions) when the environmental conditions during recognition match those of the training data, their performance drops dramatically as the mismatch between training and recognition conditions increases. The inherent lack of robustness of these features has stimulated the research towards improving their robustness against a variety of distortions, and towards finding other kinds of features which are inherently more robust against mismatches. In Appendix B we describe some feature extraction algorithms such as PLP that are more robust in distorted environments.

2.4 Statistical Pattern Recognition Approach to ASR

In this section we discuss the mathematical foundations of the pattern recognition approach to ASR. The fundamental problem of ASR can be stated in the following way. Let X and W_i be respectively a sequence of vectors $(\mathbf{x}^1, \ldots, \mathbf{x}^T)$ and a sequence of words $(w^1, \ldots, w^{L_i})$. The optimum word sequence W_{opt} in the minimum probability of error sense is given by:

$$W_{opt} = \arg\max_{W_i} P(W_i|X) \tag{2.1}$$

where $P(W_i|X)$ is the posterior probability of sequence W_i given the observation sequence X. This is the so-called Bayes classifier, which ensures a minimum classification error [BM94, Bis96]. Our purpose is to use HMMs, discussed in Section 2.4.1, to solve the previous problem. In fact, it is shown in Eq. 2.6 that it is rather simple to compute joint probability of an observation sequence and the state sequence Q given the model W_i. The previous problem, however, requires the computation of the probability of a word sequence *given* the observation sequence. Using the Bayes formula [Roh76], we can condition

[2] The pitch frequency is the vibration frequency of the vocal cords as the air from the lungs flows through them. The resulting signal does not contribute to the discrimination of the speech sounds, and merely carries prosodic information [RJ93]

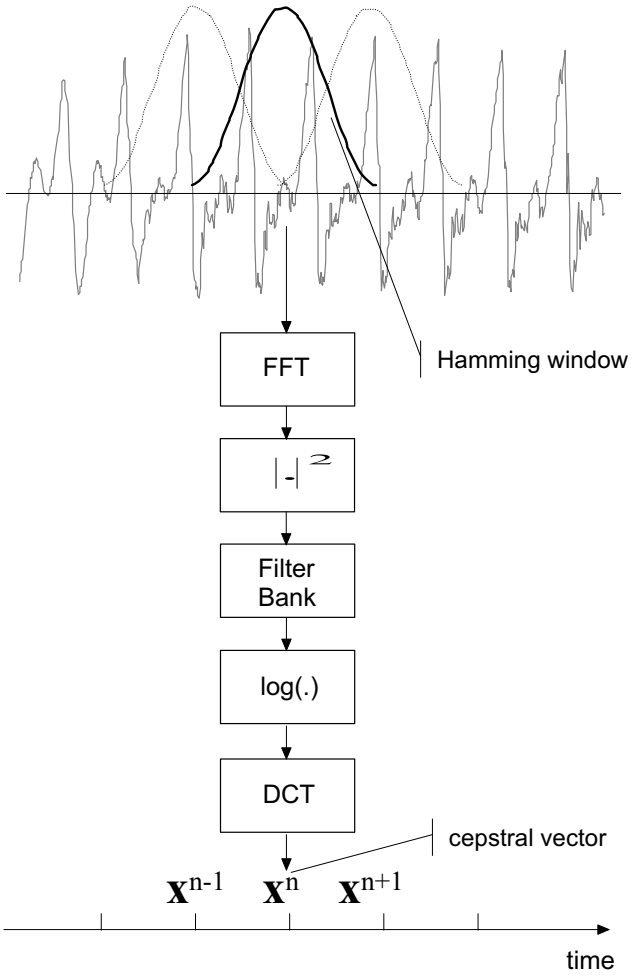


Fig. 2.2: Feature extraction process of the mel-cepstral coefficients (MFCC). First the speech signal is windowed using a Hamming window to obtain a speech frame. Next spectrum of the speech frame is computed using a FFT, and afterwards processed using a filter-bank to smooth out the pitch and other undesired variation. Finally the log filter-bank coefficients are converted using a DCT into the desired cepstral coefficients.

X to the hypothesized word sequence W_i in the probability term, thus obtaining:

$$W_{opt} = \arg\max_{W_i} \frac{p(X|W_i)P(W_i)}{p(X)} \tag{2.2}$$

The term $p(X|W_i)$ is a *likelihood* [3] , and the term in the denominator is independent during recognition of the word sequence W_i, so that we can put Eq. 2.1 in the following interesting form without any loss of performance. Let X and W_i be as in Eq. 2.1. The optimum word sequence W_{opt} in the minimum probability of error sense is given by:

$$W_{opt} = \arg\max_{W_i} p(X|W_i)P(W_i) \tag{2.3}$$

[3] In this thesis we distinguish between *likelihoods* and *probabilities*. The former are actually probability densities and are denoted with a lowercase $p(.)$. The probabilities, by contrast, are denoted with a uppercase $P(.)$

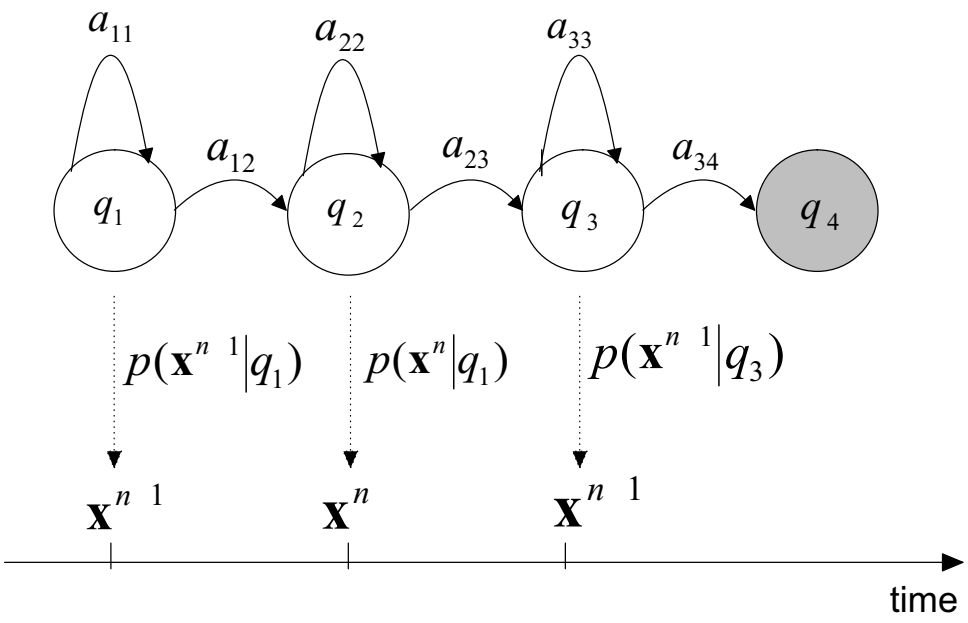


Fig. 2.3: A left-to-right HMM represented as a generator of observation vectors. The dotted line means that state q_i has emitted vector $\mathbf{x}^n$ with likelihoods (probability) $p(\mathbf{x}^n|q_i)$. Note that the state q_4 is a non-emitting state, which means that this state does not generate any observation $\mathbf{x}$. This state is used to connect an HMM to other HMMs to form word or sentence models.

This last equation reveals that the probability of a given sequence of words is indeed the product of two probabilities:

- the first probability, $p(X|W_i)$, is computed using the acoustic models of the words which have been built, for example, by assembling phoneme hidden Markov models (HMMs) as will be described in Sec. 2.4.1.
- the probability $P(W_i)$, in contrast, conveys the contribution of the language model, that is, of the syntactical and semantical rules of the language, and is computed using a language model as will be described in Section 2.4.3.

Consequently, we devote the following section to discuss HMMs and language modeling.

2.4.1 Background on Hidden Markov Models (HMMs)

The topology of a left-to-right or Bakis HMM is shown in Fig. 2.3 where the model is depicted as a stochastic generator of the observation sequence X. As we can readily see, an HMM μ consists of a chain of connected states (more precisely a 1st order stationary Markov chain [BM94]) each with a well-defined probability density function, which determines the nature of the observations $\mathbf{x}$ generated by the state q. In most of the state-of-the-art ASR systems the state densities are continuous, and accordingly we can say that a state q generates an observation $\mathbf{x}$ with a certain likelihood $p(\mathbf{x}|q)$. In Sec. 2.5.2 we will expand on this topic, to show some different ways of modeling the probability density functions of the states.

The connections a_{ij} between the states are called transition probabilities, and express the probability of a transition from state q_i to state q_j. The transitions probabilities from a given state q_i must satisfy the following conditions:

$$a_{ij} \geq 0 \quad \text{and} \quad \sum_{j=1}^{N} a_{ij} = 1 \ , \ 1 \leq i \leq N \tag{2.4}$$

A third set of probabilities which rule the behavior are the prior probabilities π_j of the states q_j. The π_j is the probability that the HMM generation process starts at state q_j. For the usual Bakis HMM (left-to-right) the π_j hold:

$$\pi_j = \begin{cases} 1 & \text{if } j = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

since for this kind of HMM we have always $q^1 = q_1$, i.e. the first state in time is always the first state (from the left) of the HMM.

Interpreted as a generator of observations the HMM is rather straightforward to understand: at a certain instant of time n the HMM stays in a certain state q_i which generates the observation vector $\mathbf{x}^n$ with a certain probability determined by the density function of the state q_i. Next the model makes a transition to state q_{i+1} or stays in state q_n in a random way. In the first case, the next observation is generated by state q_{i+1} whereas in the second case the next one is generated once again by state q_i. During this process two kinds of sequences are therefore generated: the observed sequence of observations $X = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$, and the sequence of HMM states $Q = (q^1, \ldots, q^T)$ which cannot be observed and is therefore 'hidden'. Note that this sequence of states is not unique to the given observation sequence. In fact, and since HMMs are stochastic generators, there is an infinite number of possible state sequences that may have generated the given observation sequence. Actually, it is simple to compute the likelihood of the observation sequence X given the state sequence $Q = (q^1, \ldots, q^T)$. Assuming a continuous density function of the states, the likelihood of the observation sequence X given the state sequence Q is:

$$p(X, Q|W_i) = \pi_{q^1} \prod_{n=1}^{T} a_{q^n q^{n+1}} p(\mathbf{x}^{n+1}|q^{n+1}) \tag{2.6}$$

Each of the state sequences generates the observation sequence with a different likelihood, and consequently there is a state sequence which generates the observation sequence with maximum likelihood. However, in a practical situation we do not have access to the hidden state sequence, so that we cannot compute the likelihood of an observation sequence X using the last formula. In Sec. 2.5.1 we will introduce an efficient algorithm to compute the likelihood of an observation sequence that circumvents this problem.

The suitability of HMMs for speech modeling is readily seen if we think of the non-stationary character of the speech signal. As already mentioned in Chapter 1, the statistical properties of the speech signal change to reflect the different speech sounds being uttered by the speaker to generate a distinct word. If we now think of the single states in an HMM as models for the short-time stationary properties of speech sounds, then it can be readily understood that the transition mechanism of the HMMs provides an excellent way to model non-stationarity: a leap from one state to another causes a change in the short-time characteristics of the signal. More details about HMMs and their use in ASR can be found in the books [RJ93, BM94].

2.4.2 Acoustic-Phonetic Modeling Using HMMs

The simplest acoustic modeling approach, known as whole word modeling, is to associate with each word in the lexicon an HMM which is only trained on the occurrences of that word. In that case the lexicon is just a simple one to one mapping between word and model identifiers, and there are therefore as many HMMs as words in the lexicon. The use of this kind of modeling is limited by the size of the lexicon and the amount of training data available. For small vocabulary size tasks (number of words of the order of 10) whole word modeling is usually preferred, but for larger tasks it becomes impractical partly because of the difficulty to robustly train the HMMs of the rare words [Kuh94]. In addition, to accurately model the co-articulation effects between words, several realizations of a word in different contexts are needed. Since the amount of possible contexts increases quadratically with the number of words, this further restricts the use of word level models. Moreover, word level modeling is not modular because each time a new word is to be recognized a new HMM must be trained and added to the set of models.

On the other hand, the set of possible phonemes in a language is relatively small and finite (between 13 and 75 for most of the languages, with a mean number of 30 [Mul69]) and they consequently lend themselves to statistical modeling since it is easy to find enough training patterns for each phoneme HMM, i.e. they can be robustly trained. To build a lexicon using phoneme HMMs, we usually use the concepts of phonology to transcribe each word into a sequence of phonemes, or equivalently a sequence of HMMs, which is acoustically consistent with the transcriptions of the other words in the lexicon. In Fig. 2.4, the German city names *Ehingen* and *Solingen* are respectively phonetically transcribed (using SAMPA notation) as `eIN@n` and `zolIN@n`, and have therefore the last four phonemes in common. This implies again that the transcriptions of both words in the lexicon must have the four last HMMs in common as well, if each HMM is associated with a phoneme. Otherwise two different acoustic models would be trained on a similar sound, which would result in further undesired variability in our acoustic models. Since many of the words in the lexicon have their first phonemes in common, it is possible to arrange the lexicon in a tree form instead of the list form used for the word models. The nodes in this tree are the phonemes, and each branch is a word in the lexicon. This structure is called lexicon a tree, and greatly simplifies and speeds up the decoding step [Kuh94].

The problem with the phoneme-based acoustic modeling is that the phoneme training patterns typically exhibit a higher degree of variability as compared to the word training patterns due to the strong co-articulation effects between contiguous phonemes in a word (allophones). This results in a poor sensitivity to the acoustic context [Kuh94] which leads to a more inaccurate modeling of the acoustic variability as compared to whole word modeling.

These two kind of modeling reveal the compromise between robust training of the acoustic models and the sensitivity of the models to their acoustic context. As a compromise, it is usual to use a kind of mixed modeling in which the very frequent words in the lexicon are modeled using a whole-word HMM, and the rest using phoneme HMMs. Another possibility is to use syllables instead of phonemes as sub-word units, since syllables contain most of the variability due to the co-articulation between phonemes. A problem with the syllables is the need for large speech databases to obtain reliable estimates of the syllable model parameters, because the number of syllables in a language is quite large. A solution to this previous problem is to use half-syllables instead of syllables, since their

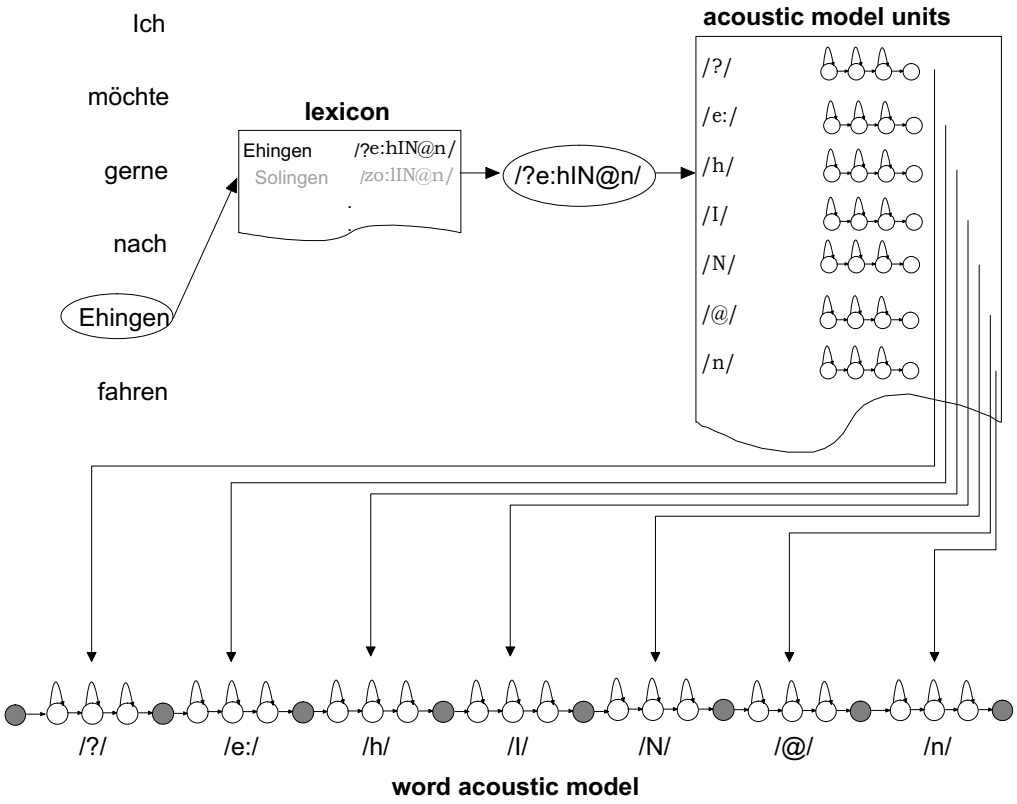


Fig. 2.4: An illustration of the building blocks principle of acoustic modeling. The phonetic transcriptions of the words are stored in a lexicon. Each phoneme in a lexicon is modeled by an HMM (phoneme HMMs). These HMMs are used to construct the composite model of the word.

number is much smaller [Kuh94]. Another very common technique is to introduce context-dependent sub-word models [Kuh94, Lee90], e.g. biphones or triphones, so as to increase the sensitivity of the HMMs to the acoustic context without reducing their robustness.

This building block principle is illustrated in Fig. 2.4, and is based on the fact that human languages are articulated (as opposed to unarticulated animal sounds [Mul69]). This fact allows us to model a very large number of sentences with a comparatively small number of statistical models.

2.4.3 Language Modeling

As already mentioned in the introductory section of this chapter, the language model conveys a restricted set of the syntactic and semantic rules valid for the language in question. Without this set of rules, the words in the lexicon could be freely combined. This would result in an explosion of the number of possible sentences, from which just a few would be syntactically and semantically correct. Therefore, the rules in the language model are extremely useful to restrict the search space in the search process carried out in the decoding block, which greatly reduces the computation time for large vocabulary applications. Additionally, the language model may partially correct the errors due to a bad acoustic model, by improving the score of the correct sentence and reducing the scores

of the incorrect ones.

This language model can be a simple grammar, in which the possible sequences of words or sentences are strictly fixed, or a statistical language model which assigns a probability $P(W_i)$ to each possible word sequence W_i [Ros00]. In fact, the first kind of language model can be interpreted as a special form of statistical language model in which the probability of a sentence is one or zero. Also, the use of grammars is restricted to very simple tasks, e.g. command-and-control, where the structure of the uttered sentences is simple. For tasks with more complex sentences the grammars are too rigid or become too complicated to correctly model the variety of the sentences. In addition it is very difficult, if not impossible, to model with a grammar non-grammatical phenomena typical of spontaneous spoken sentences, such as repetitions or grammatical errors, e.g. '`Ich hätte gern eine -äh- ein Pfund Bohnen`'. For these reasons, a statistical language model is usually preferred for complex ASR tasks. The basic idea behind statistical language modeling is to factor the probability of a given sentence W_i in the following way:

$$P(W_i) = P(w^1, \ldots, w^{L_i}) = \prod_{j=1}^{L_i} P(w_j|H_j) \tag{2.7}$$

where w_j is the j-th word in the sentence, and $H_j = (w^1, \ldots, w^{j-1})$ is called the *history* of word w_j.

A very common practice in statistical language modeling is to use an n-gram to model the history of a given word by taking just the $n-1$ previous words, that is:

$$P(w_j|H_j) \approx P(w_j|w_{j-n+1}, \ldots, w_{j-1}) \tag{2.8}$$

The value of n controls the trade-off between the accuracy of the approximation above and the accuracy of the estimated n-gram probabilities. To understand this, we just have to figure out how difficult it is to find samples of a history H_j for a certain word w_j. If the history is too long, the number of samples found in a text is usually too low, and accordingly the estimate of the probability $P(w_j|w_{j-n+1}, \ldots, w_{j-1})$ is not reliable. A common choice is to use a trigram (n=3) when the available training corpus is large (millions of words), whereas a bigram (n=2) is preferred when the corpus is small. In the next section we will see how to integrate the language model probabilities in the search process.

2.5 Finding the Optimum Sequence of Words

In the previous section we have seen the two basic elements used in the statistical modeling of the speech: acoustic models, which are built from a set of HMMs, and a language model to model the possible word sequences. In this section, we explain the algorithms used to find the optimum sequence of words W_{opt} in the sense of Eq. 2.3.

2.5.1 Decoding

Isolated Word Recognition

To start with the decoding problem, let us reduce the scope of our problem to isolated word ASR. In this case there is no sequence of words, and accordingly the language model

term is superfluous. Let us as well assume that the words in our lexicon have equal *a priori* probabilities, and that the probability densities of the states are continuous, so that it is possible to compute the state likelihoods of an observation $p(x|q,w)$. Taking all these assumptions into account, the Eq. 2.3 can be simplified to:

$$w_{opt} = \arg\max_{w_i} p(X|w_i) \tag{2.9}$$

that is, the problem is to find the word with maximum probability $p(X|w_i)$. This last probability can be further expressed as a sum of the probabilities in Eq. 2.6 over all possible state paths in the HMM of word w_i:

$$\begin{aligned} p(X|w_i) &= \sum_{\forall q^1,\ldots,q^T} p(X, q^1, \ldots, q^T|w_i) \\ &= \sum_{\forall q^1,\ldots,q^T} \prod_{n=1}^{T} a_{q^{n-1}q^n} p(\mathbf{x}^n|q^n, w_i) \end{aligned} \tag{2.10}$$

However, this method of computing the probability of a given word is very inefficient, because a great amount of possible state sequences exist. The approximate number of operations required would be $N^T - 1$ additions and $(T-1)N^T$ multiplications, which is of the order of TN^T operations. Even for relatively small number of input frames (for a second of speech $T = 100$) and a typical number of states ($N = 10$) per HMM, the amount of operations to perform is prohibitive.

A solution to this problem is to use the Viterbi algorithm which is a very efficient procedure to find the optimum word w_{opt}. The key idea behind this algorithm is to approximate the sum operation in Eq. 2.10 by a maximum operation, so that just the probability of the state path with maximum probability has to be computed. Expressed in mathematical terms we thus have:

$$p(X|w_i) \approx \max_{q^1 \ldots\, q^T} p(X, q^1 \ldots\, q^T|w_i) \tag{2.11}$$

Substituting into Eq. 2.9 results in the following optimization criterion, which is the one actually optimized using the Viterbi algorithm:

$$\tilde{w}_{opt} = \arg\max_{w_i} \max_{q^1 \ldots\, q^T} p(X, q^1 \ldots\, q^T|w_i) \tag{2.12}$$

If we now define the variable $\delta^l(j,i)$ as:

$$\delta^l(j,i) = \max_{q^1 \ldots\, q^{l-1}} p(\mathbf{x}^1 \ldots\, \mathbf{x}^l, q^1 \ldots\, q^{l-1}, q^l = q_j|w_i) \tag{2.13}$$

the Viterbi algorithm can be expressed in the following elegant form [RJ93]:

$$\begin{aligned} &\text{Initialization} && \delta^1(j) = \pi_j\, p(\mathbf{x}^1|q_j, w_i) \\ &\text{Recursion} && \delta^{l+1}(j) = \max_k \left[\delta^l(k)\, a_{kj}\right] p(\mathbf{x}^{l+1}|q_j, w_i) \\ & && l = 1, \ldots, T-1 \\ &\text{Termination} && p(X|w_i) \approx \max_k \delta^T(k) \\ &\text{Decision} && w_{opt} = \arg\max_i p(X|w_i) \end{aligned} \tag{2.14}$$

This algorithm is based on the Bellman's Principle of Optimality of Dynamic Programming [Bel67], which states that every partial path of an optimum path must be itself optimum. This can be seen in the recursion step of Eq. 2.14, which guarantees the optimality of the partial path passing through state q_j at time instant $l+1$. It can be seen as well that the final decision is made in the termination step. If at termination we find that state q_k has maximum $\delta^T(k)$, then we know that q_k has an optimum predecessor state resulting from the recursion at time instant T. Continuing backwards it is indeed possible to trace-back the optimum state sequence, *if* for each instant of time and state a trace-back pointer to the previous optimum state has been stored during the algorithm.

Although the approximation in Eq. 2.11 may seem too strong (just one of the several paths contributes to the probability) to accurately approximate the probability $p(X|w_i)$, our purpose is not this latter, but rather to find the correct word w_{opt}. Experience has in fact demonstrated that the differences between the sequence $\tilde{w}_{opt}$ in Eq. 2.12 and the exact sequence w_{opt} of Eq. 2.9 using the forward algorithm [RJ93, BM94] are minimal. The advantage of the Viterbi algorithm is that it can be implemented without multiplications, which further reduces the computational cost. This is simply achieved by taking logarithms on both sides of the recursion step in Eq. 2.14.

Connected Word Recognition

From the previous discussions we know how to use a set of HMMs (acoustic models) to recognize an isolated word. But what happens when the uttered word is a part of an uttered sentence, and it is consequently no longer clear where the words start or end?. Moreover, how should the language model term $P(W_i)$ be integrated into the decoding process, so as to take account of syntactic and semantic information in the decoding process?. The problem in this case is mathematically formulated in Eq. 2.3. As can be seen, this equation introduces a further unknown variable which was not present in the previous case: the sequence of words W_i. A possible solution would be to build HMMs for all the possible sentences by concatenating the HMMs of the single words (in the same way as word HMMs were built from sub-word HMMs in Sec. 2.4.2), apply the Viterbi algorithm to compute the probability of each of those sentences, and finally choose the sentence with maximum probability. However, this procedure is computationally very inefficient, given the large number of possible sentences, especially if the number of words in the lexicon is large.

To efficiently solve this problem a number of different algorithms have been proposed, but the most frequent algorithm in ASR is the one-pass (one-stage) dynamic programming search algorithm [NO99, NO00]. Like the Viterbi algorithm this kind of search is time-synchronous [4], and it makes use of the maximum approximation to compute the probability of the acoustic models. The minimization criterion of the one-stage algorithm can be thus stated as:

$$\tilde{W}_{opt} = \arg\max_{W_i} \left[P(W_i) \max_{q^1 \ldots q^T} p(X, q^1 \ldots q^T | W_i) \right] \tag{2.15}$$

[4] A search strategy is time-synchronous if the search hypotheses are built synchronously with the sequence of feature vectors. This is a very desirable property for ASR, because we do not have to wait until the end of the sentence to start the search.

As in the previous case, we define $\delta^l(j,i)$ as:

$$\delta^l(j,i) = \max_{q^1 \ldots q^{l-1}} p(\mathbf{x}^1 \ldots \mathbf{x}^l, q^1 \ldots q^{l-1}, q^l = q_j | w_i) \tag{2.16}$$

that is, as the score of the best path up to time l that ends in state q_j of word w_i. Additionally, we define the variable $\tau^l(j,i)$ as the start time of the best state path in word w_i up to time l that ends in state q_j of word w_i. The recursion step of the one-stage algorithm is divided into two sequential levels, which are carried out, as in the Viterbi case, for each new input frame $\mathbf{x}$. The first is the acoustic level in which all the word/state pairs are processed using the following recursions:

$$\delta^{l+1}(j,i) = \max_k \left\{ \delta^l(k,i)\ a_{kj} \right\} p(\mathbf{x}^{l+1} | q_j, w_i) \tag{2.17}$$

$$\tau^{l+1}(j,i) = \tau^l(k_{max}, i) \tag{2.18}$$

where k_{max} is the index of the best predecessor state of the maximization in the 1st equation. In a second level, the following recursion is performed over all words in the lexicon:

$$\upsilon^{l+1}(i) = \max_j \left\{ P(w_i|w_j) \delta^{l+1}(k_{last}, j) \right\} \tag{2.19}$$

where $P(w_i|w_j)$ is the probability of word w_i given the predecessor word w_j, and k_{last} is the index of the last state of word w_j. Note that in the recursion of Eq. 2.19 the time index l is not incremented, since this score is actually used to initialize the recursions in Eq. 2.17 for successor words. For that purpose, we introduce a non-emitting state between q_0 in each word (analogous to the shaded states in Fig. 2.3), to pass both the score ω and the time index at which the new word started:

$$\begin{aligned} \delta^l(0,i) &= \upsilon^l(i) \\ \tau^l(0,i) &= l \end{aligned} \tag{2.20}$$

In this fashion the score $\delta^{l+1}(j,i)$ can capture both the acoustic and the language model probabilities. To be able to quickly trace-back the optimum sequence of words W_{opt} at termination, the index of the best predecessor of word w_i at time $l+1$ is stored as well:

$$\iota(i, l+1) = j_{max} \tag{2.21}$$

At termination the scores $\delta^T(k,i)$ are used to determine the last word of the optimum sequence W_{opt} in the following fashion:

$$w^{L_{opt}} = \arg\max_{w_i} \left[\max_k \delta^T(k,i) \right] \tag{2.22}$$

To trace-back the optimum sequence we use the word $w^{L_{opt}}$, the table of stored predecessors υ and the table of time boundaries τ. We first look into table υ to know which is the best predecessor of $w^{L_{opt}}$ at time T. Next we look into table ι to get the start time of word $w^{L_{opt}}$, and use this time and the predecessor of $w^{L_{opt}}$ to repeat the process for this word and all the precedent words until the start of the sentence is reached.

2.5.2 State Probability Computation

In this section we address the question of how to model the probability density functions of the HMM states. We have assumed in the preceding section that these densities are continuous, and it is therefore possible to compute the state likelihood $p(\mathbf{x}|q, w)$ of an observation vector. When these densities are discontinuous, however, it is no longer valid, in the strict sense, to use the term likelihood. But which kind of HMMs have continuous densities and which have discrete ones? In speech recognition there have basically been— and still are— three different kinds of HMMs categorized according to the type of state densities:

- **Discrete HMMs (DDHMM)**. This kind of HMM applies first a vector-quantization algorithm to transform the input continuous sequence X into a sequence of discrete symbols $S = (s^1, \ldots, s^T)$. These symbols are extracted from a set of code-vectors $\mathcal{S} = \{s_1, \ldots, s_M\}$ stored in a code-book. The density function of a state is therefore discrete, and assigns to each symbol s_i in the code-book a probability of being emitted by the state, i.e.:
$$b_{ik} = P(s_k|q_i) \ , \ \forall i, k \tag{2.23}$$
with the constraint:
$$\sum_{k=1}^{K} b_{ik} = 1 \ , \ \forall i \tag{2.24}$$
The state emission probabilities are therefore:
$$p(\mathbf{x}^n|q_i) \underset{s_k=[\mathbf{x}^n]}{=} P(s^n = s_k) = b_{ik} \ , \ \forall i, k \tag{2.25}$$
where the square brackets [.] symbolize the vector-quantization operation.

- **Continuous density HMMs (CDHMM)**. In this kind of HMMs the density function of each state is usually modeled using a mixture of normal densities:
$$p(\mathbf{x}|q_i) = \sum_{k=1}^{K} b_{ik} \mathcal{N}(\mathbf{x}; \mathbf{m}_{ik}, \mathbf{K}_{ik}) \tag{2.26}$$
where the b_{ik} must satisfy:
$$\sum_{k=1}^{K} b_{ik} = 1, \forall i \tag{2.27}$$
if $p(\mathbf{x}|q_i)$ is to be a density function.

- **Semi-continuous HMMs (SCHMM)**. The third kind of HMM is a sort of compromise between the discrete and continuous density HMMs. As a discrete model it uses a vector-quantization algorithm to transform the sequence of observations X into a sequence of symbols S. However, this algorithm does not output just a code-book symbol s for each input frame $\mathbf{x}$, but rather the probabilities of the symbols in the code-book given the input frame $\mathbf{x}$. This implies that the symbols in the code-book are no longer a representation of a single code-vector, but rather a representation of a probability density function. This can be interpreted as a kind

of fuzzy vector-quantization which avoids the hard decisions of the discrete density case. The probability density function associated with each of the symbols in the code-book is usually normal, so that we can compute the probability of each symbol given the observation vector $\mathbf{x}$ as:

$$g_k(\mathbf{x}) = p(\mathbf{x}|\, s_k) = \mathcal{N}(\mathbf{x}; \mathbf{m}_k, \mathbf{K}_k) \tag{2.28}$$

At the output of the vector quantization we have for each input frame $\mathbf{x}^n$ a vector $\mathbf{g}^n$ which contains all the g_k, and accordingly we have a sequence $G = (\mathbf{g}^1, \ldots, \mathbf{g}^T)$ for each input utterance.

To compute the state emission probability $p(\mathbf{x}|q_i)$ we proceed as in the continuous density case, although in this case the Gaussian mixtures b_{ik} of the previous case are interpreted as symbol emission probabilities of the state, i.e. the probabilities that a given symbol s_k has been emitted by the state q_i. Therefore the state emission probabilities are:

$$p(\mathbf{x}|\, q_i) = \sum_{k=1}^{K} b_{ik}\, g_k(\mathbf{x}) \ , \ \forall i \tag{2.29}$$

where as before:

$$\sum_{k=1}^{K} b_{ik} = 1, \forall i \tag{2.30}$$

Thus semi-continuous HMMs can also be interpreted as a kind of continuous HMM where all the normal densities are shared between the states.

There are, however, other methods to model the probability densities of the observation vectors $\mathbf{x}$ in each state. In fact, we will see in Chapter 5 that it is possible to use artificial neural networks for that purpose, and still use the decoding algorithms discussed in Sec. 2.5.1 to find the optimum word sequence.

2.6 Training the Parameters of HMMs

Training is the process of computing the parameters of the HMMs, such as the state transition probabilities a_{ij}, so that the trained HMMs can be used to recognize an uttered sentence using the algorithms described in the previous points.

As seen in Fig. 2.1, an essential part of the process is a speech database which is an orthographically transcribed set of speech recordings. Ideally we would like to have as large a speech database as possible, so that most of the possible word or phoneme contexts in our recognition tasks are covered in the training phase. If additionally the recognition task is speaker independent, then the database must contain utterances from many speakers to cover the inter-speaker variation [JH96]. In general we can say that a speech database should cover, as far as possible, the sources of variation that could arise during recognition. Thus, as a general rule, the more sources of variation, the larger must be the speech database to obtain reliable HMMs. Another important factor that conditions the size of the speech database (and which is hardly mentioned in the ASR literature) is the number of parameters to be trained. This is a consequence of the so-called *curse of dimensionality* [Bis96] which somehow limits the number of parameters

trainable with a given amount of training data. In fact, it has often been observed that an increase in the complexity of the models does not lead to an increase in performance. To understand this phenomenon, let us assume that a non-linear mapping $T : \mathbb{R}^d \longrightarrow \mathbb{R}$ is to be computed using a set of input/output pairs $\mathcal{C} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_n)\}$. To perform this computation, we divide the value space of each of the d input variables into M cells, so that in the $\mathbb{R}^d$ we have a total of M^d cells. If we are given a new point $\mathbf{x}$ that falls into a certain cell, we can find an output value for that point by averaging the output values of the training points in that cell. To increase the precision of this approximation one would simply increase M. However, this also increases the number of cells, which increases exponentially to M^d. Since each cell must at least contain one training point, the number of training points must also exponentially increase, *if* the variance of our mean estimator is to be approximately the same as before. In general we can say that the number of training samples needed to train a certain number of parameters does not increase linearly with the number of parameters, but it rather grows in an exponential fashion. In Chapter 4 we will discuss some techniques to reduce the dimensionality of the feature vector (and consequently the number of HMM parameters), which are useful in mitigating the effects of this phenomenon.

A second important aspect of the training process is the *training criterion* used to train the parameters. A very common and important criterion is the *maximum likelihood* (ML) criterion so that the training criterion of the HMM parameters can be stated in the following fashion: Let X be as before, W the sequence of words corresponding to X and Θ the set of parameters of the HMM associated with W. The optimum parameter values in the ML are found using the following criterion:

$$\Theta_{opt} = \arg\max_{\Theta} p(X|W, \Theta) \quad , \quad \forall X \tag{2.31}$$

The term $p(X|W, \Theta)$ is the same as in Eq. 2.3 but in this case the sequence of words W is fixed and the set of parameters Θ is allowed to vary. Since the data vectors in X used to train the HMMs in ASR are not 'labelled', i.e. no class label is associated with each of the vectors in X, the training data set is said to be *incomplete*. Our ML estimation problem, therefore, is a case of *training with incomplete data*, and has consequently no closed-form solution [Bis96]. As in the recognition step, we have basically two options to iteratively solve the optimization criterion above:

- use the maximum approximation in Eq. 2.11, and optimize the parameters for just the sequence of states with maximum likelihood. This is the basic idea behind the *Viterbi training* [BJM83] procedure, and its generalization the *segmental k-means algorithm* [RJ93].

- optimize the set of parameters without approximating the value of $p(X|W, \Theta)$ by using the *Baum-Welch training* [Bau72] procedure.

As a consequence of the maximum approximation the number of computations per iteration in Viterbi training is lower than that of Baum-Welch training. At the same time, however, the former algorithm needs more iterations and training data than the latter to obtain HMMs with a similar statistical robustness [ME91], which counterbalances the lower computational cost per iteration of Viterbi training.

On the other hand, we have already mentioned that we are primarily interested in semi-continuous HMMs, because most of our experiments have been performed using this kind of system. We have seen in Sec. 2.5.2 that an SCHMM system has the following components: a code-book of symbols, a set of emission probabilities of those symbols for each HMM state, and the transition probabilities between the different states. As seen in the following sections, the first set of parameters is trained independently from the other two. The Viterbi training procedure is used to train the parameters of the code-book, whereas the Baum-Welch is used to compute the symbol and the transition probabilities.

2.6.1 Code-book Parameter Training

Let X, W and $(q^1, \ldots, q^T)$ be as before. The Viterbi training criterion can be expressed in the following way:

$$\tilde{\Theta}_{opt} = \arg\max_{\Theta} \max_{q^1,\ldots,q^T} p(X, q^1, \ldots, q^T | W, \Theta) \tag{2.32}$$

As already mentioned above, an iterative solution to the previous optimization is given by the Viterbi training algorithm. The algorithm is the following:

1. Generate an initial segmentation of the training data into HMM-states, which assigns to each training vector $\mathbf{x}^l$ a label $\mathbf{t}^l$ associated with a particular HMM-state. We can start for example with a flat segmentation, which assigns the same number of frames to each HMM-state present in an utterance. A better solution if we use phoneme HMMs is to use the average phoneme durations [BM94]. The results is a set $\mathcal{S} = \{(\mathbf{x}^l, q^l)\}$ of frame/state pairs which it is called henceforth *segmentation*.

2. Actualize the parameters of the SCHMMs, i.e. the $\{a_{ij}\}$, the $\{b_{ij}\}$, the $\{\mathbf{m}_j\}$ and the $\{\mathbf{K}_j\}$, over all sentences and independently of each other using the last segmentation into states.

3. Use the re-computed SCHMMs to generate a new segmentation into states for all the sentences in the training set. In this step, the segmentation is generated using a forced alignment, and a mean score over all sentences, usually the average of $\max_{q^1 \ldots q^T} p(X, q^1 \ldots q^T | w_i)$, is also computed.

4. If the score computed in the previous step is better than that of the last iteration, then go to step 2 and repeat the process, otherwise terminate the algorithm.

Our training procedure of the code-book parameters is analogous to the previous algorithm, but with some particularities. The first particularity is that just one training iteration has been performed, so that steps 3 and 4 are actually not carried out in our algorithm. A second particularity is that the initial segmentation into states of step 1 has been generated using a previously trained set of HMMs, which have been trained as described in [CKRB93]. These HMMs are semi-continuous as well, but their code-book has been trained using the Linde-Buzo-Gray (LBG) clustering algorithm [LBG80], which is an *unsupervised learning* technique, and accordingly does not need any initial segmentation to compute the parameters of the code-book [Bis96]. This is opposed to a *supervised learning* technique, which uses a segmentation $\mathcal{S}$ to train the parameters of a classifier. Our implementation of the algorithm above is a supervised learning technique, because

we assume that the segmentation $\mathcal{S}$ is *a priori* given. The third one is that we have just used the Viterbi training to compute the parameters of the code-book, i.e. the means $\{\mathbf{m}_j\}$ and the covariances $\{\mathbf{K}_j\}$, but not the emission and transition probabilities, which have been computed using the Baum-Welch algorithm of the next section. This has been done to find a compromise between training speed and statistical robustness of the trained SCHMMs. A last particularity of our method is the way we build the classes of the code-book when the number of states is very large: we first gather the states into a smaller set of state clusters, using the Lee clustering algorithm [Lee90] (cf. Appendix D), and finally we associate with each cluster a symbol in the code-book [CKRB93]. In contrast, if the number of states is small, we simply associate each state with a symbol, and we compute the ML-estimate [5] of the mean and covariance over all feature vectors assigned to that symbol by the segmentation $\mathcal{S}$.

2.6.2 The Baum-Welch Algorithm

This algorithm is also an iterative solution of the criterion in Eq. 2.31, which was first published by Baum [Bau72]. In fact, this algorithm is an special case of the more general *expectation maximization algorithm* [DLR77] (EM) used to solve the above mentioned problem of training with incomplete data. With the Baum-Welch algorithm we want to train the transition probabilities $\mathbf{A}$ and the emission probabilities of the symbols $\mathbf{B}$, using as input data the output sequence of the vector quantization $G = (\mathbf{g}^1, \ldots, \mathbf{g}^T)$, which has been computed using the code-book of the previous step.

We begin by defining the forward variable $\alpha^l(j)$ and the backward variable $\beta^l(j)$ as:

$$\begin{aligned} \alpha^l(j) &= \quad p(\mathbf{g}^1 \ldots \mathbf{g}^l, q^l = q = j|W) \\ \beta^l(j) &= \quad p(\mathbf{g}^{l+1} \ldots \mathbf{g}^T | q^l = q_j, W) \end{aligned} \tag{2.33}$$

where both variables hold:

$$p(G|W) = \sum_{\forall j} \alpha^l(j)\beta^l(j), \forall l \tag{2.34}$$

and using these two variables we can further define:

$$\begin{aligned} \gamma^l(i) &= \frac{\alpha^l(j)\beta^l(j)}{p(G|W)} \\ \eta^l(i,j) &= \frac{\alpha^l(i)a_{ij}p(g^l|q_j,W)\beta^l(j)}{p(G|W)} \\ \zeta^l(i,j) &= \gamma^l(i)\frac{b_{ij}\mathcal{N}(\mathbf{g}^l;\mathbf{m}_j,\mathbf{K}_j)}{p(\mathbf{g}^l|q_i,W)} \end{aligned} \tag{2.35}$$

Using these variables the re-estimation (recursion) formulas of the Baum-Welch algorithm

[5] This estimate is simply the sample mean, $\hat{\mathbf{m}} = (1/N)\sum_i^N \mathbf{x}^i$ and covariance, $\hat{\mathbf{K}} = (1/N)\sum_i^N (\mathbf{x}^i - \hat{\mathbf{m}})'(\mathbf{x}^i - \hat{\mathbf{m}})$, over all the frame/state pairs in the segmentation $\mathcal{S}$ having the same q_k

are:

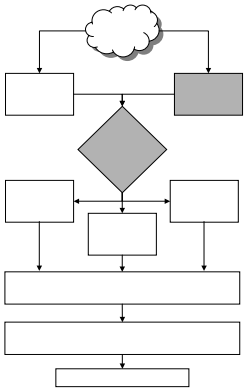
$$a_{ij} = \frac{\text{mean \# of transitions from } q_i \text{ to } q_j}{\text{mean \# of stays in state } q_i} = \frac{\sum_{l=1}^{T-1} \eta^l(i,j)}{\sum_{l=1}^{T-1} \gamma^l(i)} \tag{2.36}$$

$$b_{ij} = \frac{\text{mean \# of obs. of symbol } s_j \text{ in state } q_i}{\text{mean \# of stays in state } q_i} = \frac{\sum_{l=1}^{T} \zeta^l(i,j)}{\sum_{l=1}^{T} \gamma^l(i)} \tag{2.37}$$

The values of both sets of parameters are actualized once for training iteration.

2.7 Summary

In this chapter we have presented a generic 'state-of-the-art' ASR system, and explained its different building blocks. The pre-processing and the feature extraction blocks are responsible for discarding the variability in the speech signal not relevant to speech recognition, e.g. noises or speaker-dependent effect, and reducing the information flowing into the probability computation block. In contrast, the probability computation and the decoding blocks carry out the decoding of the linguistic content in the speech signal. To carry out this sub-task, two different kinds of statistical models are used: HMMs and language models. The first are used to model the acoustic variability, whereas the latter are used to model the syntactic and semantic variability.



3. Interfacing ASR Systems for Evaluation

This chapter is devoted to the evaluation of ASR algorithms. The evaluated algorithms have been developed during the course of the RESPITE and SPHEAR projects by our project partners. Our task was in both projects to find out whether any of those algorithms was a better alternative to our current ASR system. Consequently, we devote the first section of the chapter to the fundamental question of what kind of ASR evaluation is useful to our purpose. After that we detail the fundamental elements of technology evaluation, with especial emphasis on the metrics used to evaluate ASR systems in ASR tasks. Two previous large-scale evaluation exercises will be reviewed and taken as examples to design our own evaluation. After that a brief description of each of the algorithms evaluated is given, followed by the results of our evaluation and their discussion.

3.1 Introduction

As the number of applications of ASR technologies is increasing steadily, the need for ASR system evaluation is growing rapidly. This need is specially urgent since the number of different ASR technologies is overwhelming. However, it is not at all clear how ASR systems should be evaluated, or rather from which standpoint ASR systems should be evaluated. In the following sections we try to clarify this concept.

As defined by Crouch, Gaizauskas and Netter [CGN95] *evaluation* is in the strict sense a comparison between the *assessment* results of different *systems*. Assessments can be of two different kinds depending on the desired perspective:

- **user-centered** or **extrinsic assessments** are useful for the end-users of the system. These assessments try to find how well does the system allows an end-user to complete his intended goal in a given *environment*.

- **technology** or **intrinsic assessments** are useful from the point of view of the technologist [Gai98]. These assessments, by contrast, try to measure how well a system meets some pre-defined functional specification of the *task* it is intended to carry out on some specific test data set.

Both kinds of assessments are obviously not completely unrelated since any improvement measured in a technology assessment should lead to an observable benefit from the end-user point of view. On the other hand, the job of a technologist in the spoken language field is only possible if technology assessment and measures are used. This is easily understood if we realize that user-centered assessments measure the quality of the man-machine interaction, whereas technology assessment measure the quality of the machine or system itself using some predefined functional criterion. The former have therefore a rather subjective nature, since they partially depend on the end-user, which is too imprecise to be useful in drawing useful conclusions for a technologist, i.e. to be useful in establishing cause-effect relations in the experiments carried out by a technologist.

Thus we have two complementary assessments, each intended to fulfil the necessities of two different groups: end-users and speech technologists. Of course, as technologists we are mainly concerned with technology assessment, but to understand the limits of our predefined abstract measures we have to gain some insight into the user-centered perspective as well.

The approaches researched in the SPHEAR and RESPITE projects, however, are far from being implemented into a commercial ASR system for the moment. Consequently, any user-centered evaluation of them is pointless and even impossible. Nevertheless some user-centered considerations should always be kept in mind in any technology evaluation, since the improvements measured during technology evaluation must be large enough to be noticeable for an hypothetical end-user. This is even more true if, as stated at the beginning of the chapter, the final objective of the process is to implement the best approach in our ASR system for applications in cars.

3.2 The Elements of Technology Evaluation

The fundamental elements of technology evaluation, which should be determined before evaluation begins, are discussed in the following paragraphs.

System

This is the object of technology evaluation itself and is defined as a set of software components that carry out a certain spoken language processing task. An ASR system is always a fundamental sub-system of a spoken language processing system, and its performance is therefore decisive for the correct operation of the complete system.

Task

This is the pre-defined functional specification of the spoken language processing task to be carried out. This can be viewed as an abstract mapping between a set of input objects and a set of output objects. For instance, ASR can be interpreted as a mapping between an uttered speech signal and a written representation of it. This representation can be the optimal—in some predefined sense— string of words, a list of N-best strings, or even a word graph. Therefore we speak henceforth of the *automatic speech recognition task*. As a consequence we can talk of a system as implementing a certain task. Systems must fulfil the requirements of the task if they are to perform it successfully. Obviously systems can be decomposed into sub-systems and likewise a task can be broken down into sub-tasks.

Input/Output attributes	• language of the input or output, e.g. German, English, etc... • topic, e.g. weather reports, broadcast news, etc... • degree of spontaneity of input speech, e.g. read speech or spontaneous speech • channel distortions, e.g. telephone channel, microphones, etc... • environmental noise type and level, e.g. car-motor noise at different speeds • accent of speakers, e.g. native or non-native, dialectal, etc... • speaker independent or speaker dependent task • degree of fluency of input speech, e.g. isolated words or connected words.
Object attributes	Internal objects like lexicon, grammars or language models used to perform the task, which are of course partially determined by some of the I/O attributes.
Mode attributes	Attributes related to how the task is performed, such as depth, accuracy, robustness or efficiency of the task.
Structural attributes	Decomposition of the task into sub-tasks, since there may be more than one way to decompose the spoken language processing task

Tab. 3.1: Categorization of the task attributes of a spoken language processing task.

On the other hand, tasks can be of two kinds:

- **user-significant** tasks, are tasks where both the input and the output objects have some direct significance to a system end-user.
- **user-transparent** tasks, by contrast, are tasks where input or output objects have no direct meaning to the end-user.

Typically a user-transparent task is a sub-task of a wider user-significant task. For example, for the end-user of a car navigation system the output of the ASR task is of no relevance, and in this sense the ASR task is a user-transparent sub-task of the larger user-significant car navigation task. Obviously, technology evaluation is mainly concerned with user-transparent tasks, whereas user-centered evaluation only considers user-significant tasks.

A task is described by means of attributes which can fall into one of the categories shown in Table 3.1. System and sub-systems parameters should be adapted to the re-

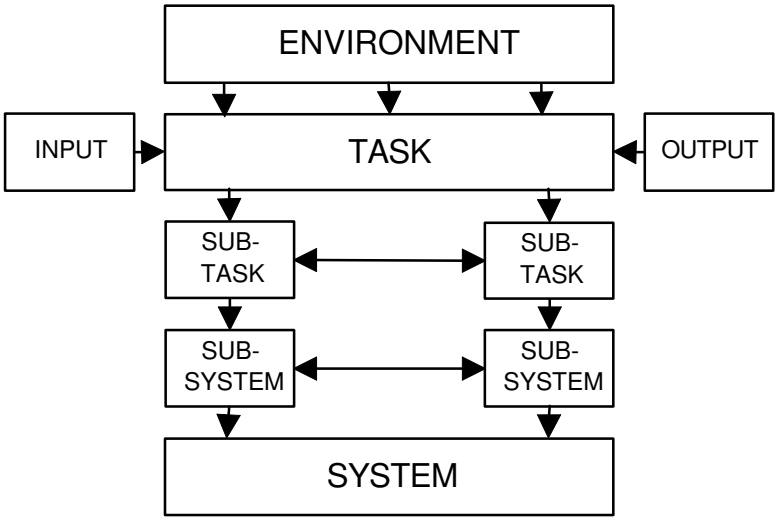


Fig. 3.1: Dependence and interaction between environment/task/system.

quirements imposed by task and sub-task attributes, if the system must perform the task accurately.

Environment

This environment is composed of the factors external to a task or sub-task that determine or influence its attributes. For user-transparent sub-tasks within a wider user-significant task, the environment is determined by the surrounding sub-tasks. For user-significant tasks, by contrast, the environment is determined not only by other user-significant and user-transparent tasks surrounding it, but also by the users themselves. From this discussion it seems clear that environments possess a rather layered nature—also termed *onion-like* [CGN95]— where a certain task at a given level may be the environment of another sub-task at a lower level. In a car navigation task, for instance, the dialogue system would be the environment of the ASR task, and would thus determine the attributes of the ASR task. Similar to tasks, environments are characterized by their attributes. If a task is to match its environment, task input/output attributes must satisfy the requirements of it, i.e. task attributes must match those of the environment.

An interesting consequence of this last fact and the layered nature of environments is that environment attributes tend to percolate down into task structure. This means that environmental attributes at a higher level influence sub-task attributes at a lower level, which implies that user-centered issues cannot be completely ignored at a user-transparent sub-task level. This conclusion further justifies our discussion in the introduction.

It may be helpful to look in Fig. 3.1 to understand the relation between the three elements explained above. As can be seen, environment attributes determine principally global task input/output attributes. The global task and the sub-tasks surrounding a given sub-task, constitute the environments of this sub-task, and therefore determine its attributes as well. After that the parameters of a given sub-system must be customized to match as accurately as possible the attributes of the corresponding sub-task.

Evaluation Data

As pointed out in the introduction, technology evaluation is carried out on a specific data set, on which a certain quality or qualities are measured. This data set or evaluation data contains three different kinds of data:

- **test data**, which is processed by the systems to obtain an output hypothesis.
- **reference data**, or the 'correct' output of the task, which is compared to the output hypothesis of the systems in order to assess them.
- **train data**, or the data used to train the parameters of the spoken language processing system.

For evaluation to be meaningful, evaluation data must have the fundamental characteristics [CGN95] listed below:

- **realistic**, which implies that data must be the kind of input data the system receives in real operation conditions.
- **representative**, which means that evaluation data should include a significant sample of all the possible inputs a system could receive during operation. For example, the test part of our evaluation should include a variety of different noise conditions present in car applications if we want to measure environmental noise robustness. But even reference data must also include at least a transcription of test data into words, if *word error rate* (cf. Sec. 3.3.2) is to be measured.
- **statistically independent train and test data**, which ensures a realistic estimation of the performance, since the real operation conditions will be quite different from those of training. In general we can say that the smaller the training data set, the more important it is to have an independent test data set. Another interesting point is whether training data should include the noises present in the test data or not. This is not fair from a theoretical point of view, but from a practical point of view it has been empirically demonstrated [LMP87, HP00] that ASR system performance can benefit from a noisy training, *even if* the noises are different from those of the test data.
- **large test data set**, since it must be large enough to have statistically significant results. As we will see in section 3.3, if the test data set is too small, it is not possible to measure statistically significant differences between the systems.
- **large train data set**, to guarantee the correct estimation of the model parameters. As argued in [PL95] it could well be that a given ASR system using a certain complex approach needs, to obtain the maximum performance out of the approach, more training data than another ASR system using a different simpler approach. The performance of this complex ASR system would be well above the performance of the simple ASR system, if enough training data was available. But if training data is scarce, it could even be that the performance of the simple ASR system is better than that of the complex ASR system, because model parameters are not well estimated. This problem is related to the so-called *curse of dimensionality* problem, which has

already been discussed in Sec. 2.6. Unfortunately, it is extremely difficult to predict how large a training data set should be to obtain the maximum performance out of a certain approach. The only possibility remains to train on different training set sizes, and afterwards test the performance of the different ASR systems on the same test data set, in the hope that a certain performance limit for the approach is found.

Metrics

This includes all the questions regarding performance measurement. These questions fall into one of the following categories:

- **qualitative**, i.e. what is actually to be measured?
- **quantitative**, i.e. which quantity should be used to observe changes in the quality?
- **methodology**, i.e. how should measurements be carried to be consistent, reliable and significant?

These three question are further discussed in the next section.

Implementation

This element groups system-dependent issues like hardware platform, operating system, programming language, etc. ...

3.3 Metrics for Technology Evaluation on ASR Tasks

In this section we deal with the metrics for technology evaluation already mentioned in the previous section. The questions regarding metrics may fall in one of the categories explained into the following subsections.

3.3.1 Qualitative

As already mentioned in Sec. 3.2, the nature of the measurements is partially determined by the quality to be measured, which may fall for an ASR task into one of the following categories:

- **accuracy qualities** measure the similarity between actual output of the ASR system and expected output for the ASR task, such as the number of correctly or falsely recognized words, sentences, etc. ...
- **robustness qualities** measure the mean accuracy over a given range of environment/task attributes, such as robustness against different environmental noises, channel distortions, etc. ...
- **efficiency qualities** measure how efficiently does the ASR system uses the hardware/software resources available to perform the ASR task.

Robustness is measured on test sets in which a certain environment/task attribute varies within a range of possibilities. This range of possibilities should ideally be determined from the actual variation range of the environment/task attribute in the intended application. Computing the accuracy of a given ASR system on this kind of test set we can evaluate its robustness to a certain kind or kinds of environmental/task attribute variation. On the other hand, efficiency is usually quantified using a measure of the amount of a resource being consumed or the amount of time needed by an ASR system to perform an ASR experiment.

3.3.2 Quantitative

We have seen in the previous point that to measure the three qualities above we need basically two kind of measures, namely *similarity* and *efficiency* measures. Another point to consider is that ASR systems are seldom used in isolation except for a few applications like automatic speech transcription or dictation. More often, an ASR system is the most important part of a complex spoken language processing system which carries out a language related *task* in a specific *environment*. Since ASR systems are thus embedded in larger spoken language processing systems, it would be reasonable to evaluate them embedded in the large spoken language processing task, because the performance measurements would then be on the end-result of the global task. However this is not always feasible for a number of reasons such as:

- assessments can be too complex,
- a complete spoken language processing system may not be available for testing,
- for a dialog application, experts don't agree about how to assess its output,
- no evaluation data readily available,
- it is not always clear what will be the end-application,
- there is usually no agreement on which measure should be used to assess the outputs of spoken language Systems.

Therefore we must assess directly the output of ASR systems since:

- it is much easier and quicker,
- there are plenty of readily available evaluation databases,
- there is general agreement on how to evaluate outputs of ASR systems.

But this kind of assessment further poses the following problem: how large should be the improvement observed in an ASR sub-task, to be measurable in the global spoken language processing task?. This is not a trivial question and it has just been faced by a few authors such as [BEG+96, GVSJ97]. Nevertheless it is shown in both cases that the correlation between WER and the corresponding high-level measure is quite high. But even if the correlation would have been low, it seems intuitive that having a good WER at the output of an ASR system is always a good property. Consequently, the performance of ASR systems is usually evaluated by assessing the output of the ASR systems.

Levenshtein Algorithm

Measures of *similarity* depend clearly on the output format of an ASR system. If this output is the best hypothesized transcription into words of the input speech, then the problem reduces to finding the 'distance' between the reference string and the hypothesized string. Many algorithms exist to compute this string distance [1], but among ASR researchers it is common practice, for its simplicity, to perform a string alignment that minimizes the Levenshtein distance metric between the output of the ASR system (hypothesis) and the correspondent reference data file. This alignment is performed for each file in the test set, and the total number of errors over all files is used to compute a final performance score of the ASR system. This Levenshtein distance [Lev66] is a kind of string edit distance [WF74]. This kind of distance computes the minimum number of edit operations needed to convert one string to the other. Those edit operations can be of three types:

- **insertions**, if a word present in the hypothesis is not in the reference,
- **deletions**, if a word present in the reference is not in the hypothesis,
- **substitutions**, if a word in the reference is substituted by another word in the hypothesis.

Each of those operations has a predefined cost which can be fixed for all the words— as in the Levenshtein distance case — or changed from word to word. The Levenshtein algorithm is an efficient algorithm to find the set of edit operations with minimum Levenshtein distance, and it is illustrated in Fig. 3.2

This algorithm is based on dynamic programming (DP) similar to the Viterbi algorithm explained in Sec. 2.14. In Fig. 3.2(a), a horizontal arrow in the matrix symbolizes a deletion, a vertical arrow an insertion, and a diagonal arrow symbolizes a substitution or a correct assignment which always has zero cost. As it is common to all dynamic programming techniques, each of the cells in the grid is assigned a distance which is computed following the Bellman optimality principle [Bel67], i.e. computed selecting as predecessor cell the one among the three possible that generates a minimum cost in the present cell. This optimum predecessor is saved and the process is repeated for all the cells in the grid, until the cell in the upper right corner is reached. After computing the optimum predecessor for this cell, the sequence of optimum predecessor cells is traced back, and thus the sequence of optimum edit operations. The result of the Levenshtein alignment between reference and hypothesis is displayed in Table 3.2(b). For a more detailed mathematical treatment of string edit distances we refer to [BC95] or to [WF74].

Word Error Rate (WER)

After performing this alignment for each of the test files we obtain a total number of insertions, deletions and substitutions on the test set, which can be viewed as the errors generated by the assessed ASR system. These numbers are then used to compute the

[1] For an excellent review of most of the current algorithms used in DNA sequencing to compare, align or search DNA strings see [Gus97]

three	8	6	5	4	3	5	**4**
four	6	4	3	2	4	**4**	7
five	4	2	1	3	**4**	6	8
one	2	**0**	**2**	**4**	6	8	10
hyp	**0**	2	4	6	8	10	12
	ref	one	oh	two	five	four	three

(a) Levenshtein distance table

hypothesis	one	DEL	DEL	five	four	three
reference	one	oh	two	five	four	three

(b) Alignment result

Fig. 3.2: Levenshtein algorithm to align the hypothesis of an ASR system with the reference. The red trace-back path in Tab. 3.2(a) shows the best alignment. The costs used to compute the Levenshtein table are $c_{sub} = 1$ and $c_{del} = c_{ins} = 2$. The 'DEL' in Tab. 3.2(b) stands for deletion.

WER score of the ASR system on the given set. This WER score can be expressed as:

$$WER = \frac{I + D + S}{N} \tag{3.1}$$

where I, D, S and N are respectively the total number of insertions, deletions, substitutions and the total number of words in the reference test set.

An alternative metric to the WER is proposed In [WN82]: the relative information loss (RIL). This is an entropy-based measure which can be computed from phonetic-pair or word-pair confusion matrices, if the elements of these matrices are interpreted as probabilities of a given confusion pair.

Statistical Comparison of Measurements

Once a WER score has been computed for each ASR system, there remains the question of how to compare the scores or hypothesis/reference alignments of two different ASR systems. The most straightforward way is to simply assume that WER scores are deterministic quantities and compare them as such. But this ignores the fact that WER scores are stochastic quantities subject to random variation. This arises from the finiteness of the test data set, which implies that our score is just an estimation of the true WER. Fortunately, there are a plethora of methods to compare stochastic quantities [Roh76], which basically fall into two categories:

- **confidence intervals** are intervals centered at the estimated WER values in which the true WER value can fall at a certain confidence level α.
- **statistical tests**, by contrast, test the hypothesis that two given Levenshtein alignments are statistically equivalent against the hypothesis that both are different at a certain confidence level α.

The diagram in Fig. 3.3 shows the evaluation process of two ASR systems based on the statistical comparison of the WER. As already mentioned, if the test set consists of data with varying environmental/task attributes, then the measured WER is a measure

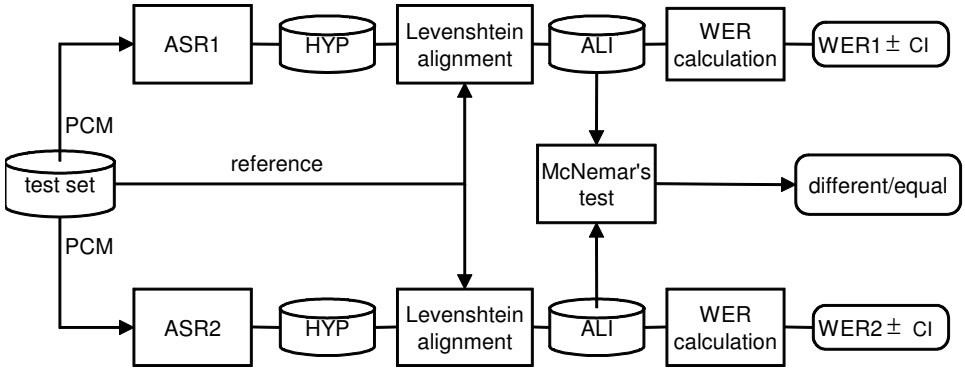


Fig. 3.3: Evaluation of the accuracy of two ASR systems on an ASR task. The HYP data are the hypothesis files of the recognizer, whereas the ALI data are the results of the Levenshtein alignments with the reference transcriptions. The McNemar's statistical test is used to determine whether the differences in the WER between both ASR systems are statistically significant. The confidence interval (CI) determines the range of values were the true value of the WER may fall.

of the robustness of a system to changes in these attributes. In the following two points we explain in more detail how the confidence intervals and statistical tests are performed.

Confidence Intervals

To compute a confidence interval for the WER, we have to find a statistical characterization of this random variable. Previous work in the field [PL95, Kuh94, GC89, Moo77] has exclusively focused on the statistical modeling of the *correctness* [2], assuming that deletions and substitutions occur according to a binomial distribution $b(N, p)$, where p is the probability of error and N is the total number of reference words. This model, however, cannot be applied when the insertions must be taken into account, because they are not considered in the correctness.

However, if we now let $N \to \infty$ with $\lambda = Np$ constant in the binomial distribution formula, we obtain the Poisson distribution $P(\lambda)$ with parameter λ [Roh76]:

$$P(K = k) = e^{-\lambda} \frac{\lambda^k}{k!} \tag{3.2}$$

where K is a random variable modeling the observed number of errors, and the parameter λ can be interpreted as the expected number of errors. We can extend the values of K and λ to include the insertions, and consequently we can use the previous distribution to find a distribution for the WER. This is easily found by denoting the associated random variable to the WER with W and recalling that $W = K/N$. Substituting in the previous equation we find:

$$P(W = w) = e^{-\lambda} \frac{\lambda^{Nw}}{(Nw)!} \quad w = 0, \frac{1}{N}, \frac{2}{N}, \ldots \tag{3.3}$$

[2] The correctness is defined as $WC = \frac{N-(D+S)}{N}$, and consequently does not include the number of insertions.

The mean and the variance of the random variable W are:

$$\begin{aligned} \mu_W &= \frac{\lambda}{N} \\ \sigma_W^2 &= \frac{\mu_W}{N} \end{aligned} \tag{3.4}$$

To estimate the mean of W — and therefore the value of the parameter λ— we can use one of the two following estimators $\hat{M}$:

- Take the value of the actual sample $w = k/N$ and assume that this is the estimate of the mean, so that the estimate has the same distribution as the random variable W, i.e.:

$$\hat{M} = W \tag{3.5}$$

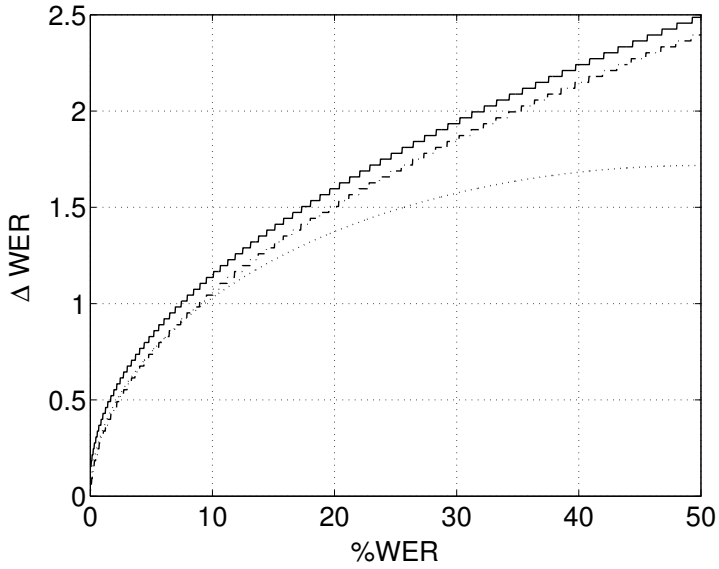
  The upper and lower limits of the confidence interval of the value of λ at a certain level α— usually 0.05% or 0.01% — can be found using the following reasoning: the upper limit is determined by the highest value of *lambda* for which the probability $P(W \leq w)$ is larger than $\alpha/2$, whereas the lower limit is determined by the smallest value of λ for which the probability $P(W \geq w)$ is larger than $\alpha/2$. As before $w = k/N$ where k is the number of errors observed including insertions, deletions and substitutions. Using the two previous probabilities and the distribution of W in Eq. 3.3, the limits λ_{inf} and λ_{sup} are found by solving for λ the following two equations:

$$\begin{aligned} \text{upper limit} \qquad & \sum_{j=0}^{k} e^{-\lambda_{sup}} \frac{\lambda_{sup}^j}{j!} = \frac{\alpha}{2} \\ \text{lower limit} \qquad & \sum_{j=k}^{\infty} e^{-\lambda_{inf}} \frac{\lambda_{inf}^j}{j!} = \frac{\alpha}{2} \end{aligned} \tag{3.6}$$

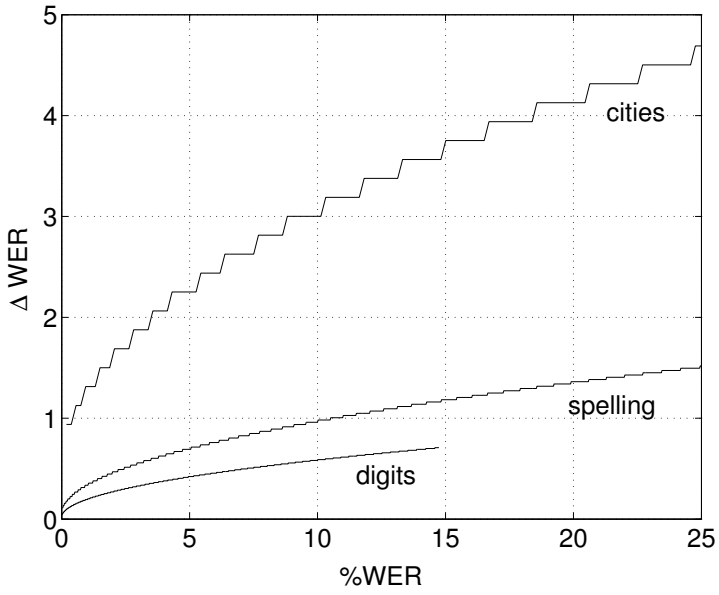
  where k is the number of observed errors, $lambda_{sup}$ and λ_{inf} are the limits of the confidence interval, and α is the desired confidence level. Both equations can be numerically solved for λ using the Newton method to numerically solve non-linear equations [HSZ96]. The solution is shown in Fig. 3.4 where the variation of the interval lengths with the WER is shown. It is interesting to observe in the previous figure that the uncertainty increases with increasing WER, which is actually logical since the Poisson distribution spreads with increasing λ. Also, the confidence intervals obtained assuming a binomial or a Poisson distribution are similar for low WERs, but differ significantly as the WER increases. In fact, for the binomial model the length of the confidence interval reaches its maximum at a WER of 50% and then decreases with increasing WER. For the Poisson model, in contrast, the length increases steadily with increasing WER.

- Another possible estimator is to take a sequence of measurements of the WERs, and compute the sample mean to estimate the true WER, i.e.:

$$\hat{M} = \frac{1}{n} \sum_{i}^{n} W_i \tag{3.7}$$



(a)



(b)

Fig. 3.4: Variation of the upper and lower confidence interval lengths with the WER. In both figures, the estimator of the true WER is the current WER value. In (a), the number of words N is 3257. The continuous line corresponds to the upper part of the interval, whereas the discontinuous corresponds to the lower part. The dotted line corresponds to half the length of the confidence interval obtained by assuming a binomial-distributed number of errors. In (b), the length of the confidence interval for the WER is shown for different sizes of the test set. The 'digits' set has 11869 words, the 'spelling' set has 4480 and the 'cities' set has 533.

Since the measurements are independent and identically distributed, the mean and variance of the estimator $\hat{M}$ are:

$$\begin{aligned} \mu_{\hat{M}} &= E\{\hat{M}\} = \frac{\lambda}{N} = \mu_W \\ \sigma^2_{\hat{M}} &= \frac{\mu_{\hat{M}}}{nN} \end{aligned} \tag{3.8}$$

Assuming that the values of $\mu_{\hat{M}}$ and $\sigma_{\hat{M}}$ are given, i.e. they are not estimated using an estimator for the mean and for the variance, the following quantity is known to be distributed according to a normal distribution of zero mean and unit variance [Roh76]:

$$\frac{\hat{M} - \hat{\mu}_{\hat{M}}}{\hat{\sigma}_{\hat{M}}} \sim \mathcal{N}(0,1) \tag{3.9}$$

where $\mathcal{N}(0,1)$ is the normal distribution of zero mean and unit variance. A confidence interval for the values of μ_W at confidence level α can be found by putting:

$$P(-d_{\alpha/2} < \frac{\hat{M} - \mu_{\hat{M}}}{\sigma_{\hat{M}}} < d_{\alpha/2}) = \alpha \tag{3.10}$$

where the $d_{\alpha/2}$ is the $\alpha/2$-quantile of the normal distribution with zero mean and unit variance. The previous equation can be put in the form of a quadratical equation and solved for $\mu_{\hat{M}}$ to give:

$$\mu_W = \hat{M} + \frac{d^2_{\alpha/2}}{2nN} \pm \sqrt{\frac{d^4_{\alpha/2}}{(nN)^2} + \hat{M}\frac{d^2_{\alpha/2}}{nN}} \tag{3.11}$$

for sufficiently large n this can be further simplified to:

$$\mu_W \simeq \hat{M} \pm d_{\alpha/2}\sqrt{\frac{\hat{M}}{nN}} \tag{3.12}$$

A representation of this last equation as a function of the estimated WER, i.e. as a function of $\hat{M}$, is shown in Fig. 3.5. As in the previous case, the length of the confidence interval increases with increasing WER, which implies that statistically significant differences between two different WER values must be larger for large WERs.

Statistical Tests

On the other hand, statistical tests on the alignments just tell us whether two given alignments are statistically different or not. All those tests define one or more random variables on the alignments, extract samples of them from each hypothesis/reference alignment, and then define a certain statistical test on some statistic hypothesized on those measurements. A variety of statistical tests have been proposed to compare ASR system outputs [Mar89, GC89], but we restrict our discussion to the description in some detail of the well-known McNemar's test. Assume we have two alignments A_1 and A_2, each

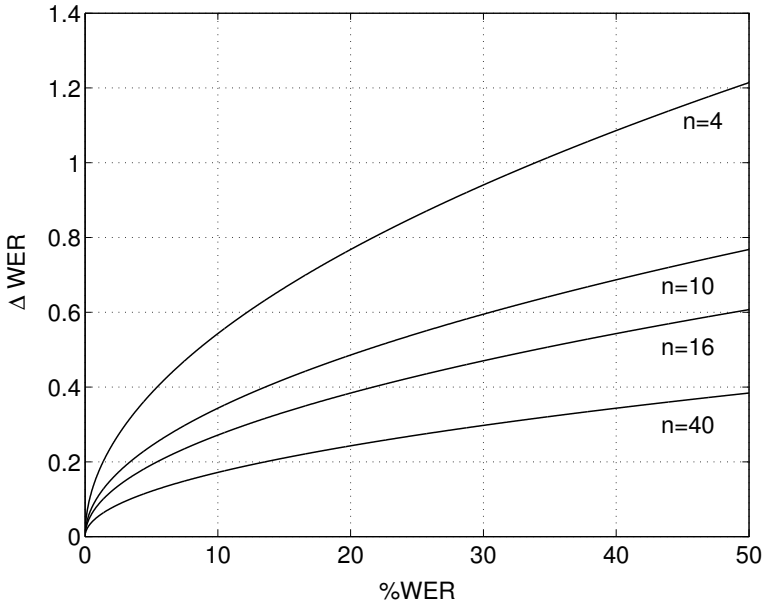


Fig. 3.5: Variation of the length of the confidence interval with the estimated WER for different values of n. The number of reference words N has been fixed to 1001, and the confidence level is 95%. Note the similarity of the curves with those in Fig. 3.4, except for the steps seen in that figure, which are not present in this case because the estimator $\hat{M}$ has a continuous density.

associated with a certain ASR system output and computed using for instance the Levenshtein distance described above. The results of both alignments can be summarized as in Tab. 3.2, where N_{00} is the number of tokens correctly classified by both systems, N_{01} is the number of tokens classified correctly by A_1 but incorrectly by A_2, N_{10} is the number of tokens classified correctly by A_2 but incorrectly by A_1 and finally N_{11} is the number of tokens incorrectly classified by both systems. A token can be a phoneme, a word or even a whole sentence. Since we are interested in the differences between both ASR systems, it seems logical to ignore the cases where both ASR are equal and to compare instead the probability $P(A_1 correct \wedge A_2 incorrect)$ with the probability $P(A_2 correct \wedge A_1 incorrect)$. If both probabilities are equal, it would be logical to think that both ASR systems are statistically undistinguishable. More formally we define K as the number of tokens in which the alignments A_1 and A_2 differ, i.e. $K = N_{01} + N_{10}$. If we now condition the previous probabilities on K we can define:

$$\begin{aligned} q &= P(A_1 correct \wedge A_2 incorrect | K = k) \\ 1 - q &= P(A_2 correct \wedge A_1 incorrect | K = k) \end{aligned} \tag{3.13}$$

The null hypothesis $\mathbf{H}_0$ is thus 'both conditional probabilities are equal' which can be expressed mathematically as:

$$\mathbf{H}_0 : q = \frac{1}{2} \tag{3.14}$$

To test the null hypothesis let's assume that N_{01} and N_{10} are distributed according to the binomial distributions [3] $B(k, q)$ and $B(k, 1 - q)$, respectively. Under the null hypothesis

[3] This assumption is in general only valid if the errors are assumed independent which is only strictly true if tokens are whole sentences, or if we are evaluating an isolated word ASR system

		A_2	
		Correct	Incorrect
A_1	Correct	N_{00}	N_{01}
	Incorrect	N_{10}	N_{11}

Tab. 3.2: Comparison of the results of two alignments.

both random variables are equally distributed and the binomial distribution is symmetric around $k/2$. Therefore we can use the following two-sided test:

$$p_{McNemar} = \begin{cases} 2\Pr(n_{01} \leq N_{01} \leq k | K = k), & \text{if } n_{01} > k/2; \\ 2\Pr(0 \leq N_{01} \leq n_{01} | K = k), & \text{if } n_{01} < k/2; \\ 1, & \text{if } n_{01} = k/2. \end{cases} \tag{3.15}$$

which can be computed directly as follows:

$$p_{McNemar} = \begin{cases} 2\sum_{m=n_{01}}^{k} \left(\begin{array}{c} k \\ m \end{array} \right) \left(\frac{1}{2}\right)^k, & \text{if } n_{01} > k/2; \\ 2\sum_{m=0}^{n_{01}} \left(\begin{array}{c} k \\ m \end{array} \right) \left(\frac{1}{2}\right)^k, & \text{if } n_{01} < k/2; \\ 1, & \text{if } n_{01} = k/2. \end{cases} \tag{3.16}$$

If $p_{McNemar}$ is under a certain predefined significance level α then the null hypothesis $\mathbf{H}_0$ must be rejected. This significance level α usually takes the values 0.05, 0.01 or 0.001, and we thus accept/reject the null hypothesis at the 95%, 99% or 99.9% confidence level, respectively. Furthermore $p_{McNemar}$ can be interpreted as a the degree of chance observed in our measurements. If for instance $p_{McNemar} = 0.38$, then we have 38% probability that the differences observed are due to chance. If, on the contrary, $p_{McNemar} = 0.01$, we have just a 1% probability that the differences are due to chance. Another interesting point to note is that if K is not large enough, then it is not possible to reject the null hypothesis. This means that the test data must have a minimum size in order to detect significant differences in the performance, as already stated in section 3.2.

Measures of Efficiency

We have already mentioned that to measure the efficiency of a given ASR system one has to measure the amount of a resource being consumed or the time a resource is in use to perform the ASR task. Any of those quantities depend on the complexity or length of the utterance being recognized, and therefore they must be normalized in some sense by the utterance complexity or length. A commonly used measure of efficiency is the Real Time Factor (RTF) which can be computed as:

$$RTF = \frac{T_{cpu}}{T_{utterance}} \tag{3.17}$$

where T_{cpu} is the total cpu time needed to recognize the utterance, and $T_{utterance}$ is the total duration of the utterance. This measure is between 1 and infinity, with values near to 1 being close to real time operation.

Another commonly used measure of efficiency is the *profile* of an ASR system, which is the amount of memory occupied by the ASR system. This figure is usually given in

kb or Mb, i.e. not normalized, but it is clearly dependent on the complexity of the ASR task. This figure is specially important for the so-called embedded applications where the amount of available memory and CPU resources are strongly limited.

3.3.3 Methodology: Previous ASR Evaluation Exercises

Using two well-known evaluations as examples, we show two very different methodologies to evaluate ASR systems. Although there have been other evaluations, usually restricted to the participants of a certain project such as SAM or Verbmobil, we have chosen the above two because they are the most well-known programs open to every institution willing to evaluate its ASR system or a particular ASR algorithm. Moreover, they are representative of two different kinds of evaluation. The one assesses and evaluates the performance of state-of-the-art ASR technology for a certain spoken language processing application, whereas the other tries to set a standard for front-end algorithms, which are a part of ASR technology, somehow independent of the end-application intended. The one aims at paving the way for the introduction of ASR technologies into real life applications, the other may be interpreted as an attempt to improve ASR technology by finding a best front-end and afterwards building upon it. And finally the one is of special interest for language engineers willing to integrate ASR technology into a spoken language processing system, whereas the other is more interesting for the ASR technologist whose aim is to improve a certain ASR system.

The ARPA/NIST Evaluation Programs

The first official ASR system evaluation program using the Resource Management (RM) corpus, which was the first created by ARPA for the sole purpose of evaluation [PFB88], was conducted in 1989, and marked the start of a series of yearly tests which continue today. These evaluation programs are coordinated by the US National Institute for the Standardization of Technology (NIST), and involve the ARPA, the US Linguistic Data Consortium (LDC), and the research institutions which participate with their ASR systems in the evaluation. Those evaluations programs are open to all research institutions willing to participate, and participating institutions vary from year to year. These programs have concentrated in the past on ASR system evaluation for ASR task, i.e. for speech-to-text transcription, but today the trend seems to be to evaluate the ASR systems in larger spoken language processing tasks. The complexity of the ASR tasks proposed for evaluation has been growing from year to year, spanning from the initial read speech RM ASR task to the spontaneous speech Call Home (CH) ASR task. In fact, in the last years there have been two ARPA/NIST ASR evaluation programs: the Broadcast News evaluation and the Recognition of Conversational Speech over Telephone evaluation. The first focused on the recognition of so-called 'found speech' in broadcast news, and was organized in a hub and spokes paradigm. This paradigm consists simply of defining a central test— the hub — which all participants must evaluate, and a set of independent test suites— the spokes — on which participants can evaluate if they are willing to test the effect of different channels, noise conditions or non-native speech on their ASR systems. The central test for this program was therefore termed Hub-4, and was extracted from the Broadcast News (BN) corpus recorded by ARPA. On the other hand, the second program concentrated on the transcription of telephone conversations, i.e of spontaneous speech. The hub of

this program was extracted from the CH corpus, and was termed Hub-5. Compared to the Hub-4 test set, the WER on the Hub-5 was much larger due to typical effects of spontaneous speech such as contracted forms, disfluencies, etc.... As the train data set, participants in both programs used the test data set of past years. To ensure the full comparability of the measurements, all the participants in both evaluation programs shared the same evaluation software package SCLITE, which performs the reference/hypothesis alignments and measurements already described in Sec. 3.3. For a good summary of ARPA/NIST evaluations until the year 1998 we refer to [YC98], and for up-to-date and detailed information about present and past ARPA/NIST evaluations, we look into the NIST web-site http://www.nist.gov/speech.

In spite of the success of those evaluations, in the sense that some of the most renowned institutions in ASR have participated in them, it is not actually clear if they are useful for comparison of ASR algorithms or approaches. This is the most interesting comparison from the standpoint of a speech technologist, since his work is mainly to implement and extend algorithms into an existent ASR system. The ARPA/NIST evaluations however compare ASR systems, *not* algorithms or approaches, i.e. they are not primarily intended to find out why a given ASR system performs well or badly. Actually, the ASR systems compared in those evaluations are so different, that sometimes it is hard to tell why a certain system is actually performing better. It could be due to a particular algorithm implemented in the best performing system, or it could as well be due to a combination of factors. Therefore it seems that to be more useful to speech technologists, those evaluations should apply some kind of restrictions or conditions on the ASR systems evaluated. This is somehow coupled to the fact that ARPA/NIST evaluations do not focus on a particular situation or phenomenon ,e.g. noise or out of vocabulary words (OOV), since ARPA speech databases usually include many different phenomena which must somehow be tackled by the competing ASR systems. What is important is the overall performance, not the performance in the presence of a particular phenomenon.

Nevertheless ARPA/NIST evaluations are certainly useful for language processing engineers willing to integrate an ASR system into their large language processing system. This seems to be the future direction favored by ARPA/NIST, since the new evaluation programs Spoken Document Retrieval (SDR) [GAVF98] and Rich Transcription (RT) evaluate ASR systems embedded in a larger language processing Task, which suggests that ARPA/NIST evaluations are becoming even more application-oriented.

The AURORA Evaluations

The first evaluation program sponsored by the EU and open to any institution willing to participate in it was started in 1999 under the aegis of the European Telecommunication Standards Institute (ETSI) and continues today. The official objective of this program was somehow more precise than that of ARPA/NIST evaluations, since it was to find an ASR front-end standard for distributed speech recognition (DSR) over mobile voice networks. The official evaluation program has been called AURORA DSR front-end evaluation, and is coordinated by the STQ DSR group of the ETSI. This group has given the guidelines and recorded different speech databases for the official evaluation. Those speech databases were made available through the European Language Resources Association (ELRA), which plays a similar role to the US LDC. A first standard was issued in the year 2000, which was based on the usual mel-filter cepstral coefficients [ETS00]. A more advanced standard

front-end, based on more robust algorithms, was released in October 2002 [ETS02].

Essentially the idea of DSR is to split the ASR process between the mobile terminal, e.g. cellular phone or PDA, and a distant end-server. The mobile terminal performs the speech signal pre-processing and feature extraction for ASR. It also encodes the feature vectors to transmit them over a mobile telephone channel to the distant end-server. This powerful end-server performs the decoding of feature vectors, acoustic modeling and decoding steps, which are more computationally expensive. This approach guarantees a minimum loss of ASR performance due to transmission effects, even over different mobile transmission channels. On the other hand, the environments of mobile applications are usually noisy or very noisy at the user's end, which implies that front-end algorithms for those applications must be robust against noise. At the same time this fact means that the evaluation data must somehow include this phenomenon, if the results of evaluation are to be significant for such applications. Evaluation was performed at the ASR task level, and the end-application initially intended was command-and-control, and therefore tasks with a small to medium vocabulary size. Since just front-end algorithms could compete in the official program, a standard back-end, i.e. acoustic modeling and decoding, software and configuration was provided to all participants. This ensured a high degree of comparability between the ASR results of different algorithms. Thus participants had to provide the results of the ASR system, built up from their front-end and the pre-defined back-end, on the evaluation database provided by ELRA. The qualities to measure were of course the overall noise robustness, the degradation due to channel encoding and the latency, i.e. the delay introduced by the front-end algorithm. The measure used to quantify the first qualities was the word accuracy, instead of the usual WER [HP00].

The first speech database issued for the program, called AURORA 1999 database, was extracted from the TI-digits database, which consists of sequences of digits spoken in American English. Different noises at different SNR levels were added artificially to the training and test speech data. Added noises were identical for both test and train sets, and no clean data for HMM training was provided. Since some participant institutions argued that their approaches could only be trained on clean data and that test noises must be different from train noises, a second speech database was provided in year the 2000 and was called AURORA-2000 [HP00]. The differences from the previous database were a separate full training set with clean speech only, and two new test conditions. The first with added noises different from those in training, and the second with a channel distortion applied to the speech data. More details about this database can be found in Sec. 7.2.1.

In spite of the extended use of the AURORA-2000 database to test noise-robustness, the database has its limitations. First of all the different noises have been added artificially to the speech signal, which implies that complex interactions between speech and noise, present in real speech recordings, are not present in the database. Second, the proposed ASR task, namely the recognition of English digits, is too little for today's typical applications of ASR, which means that results are not easy extrapolated to more real and complex tasks. A final problem of the unofficial competitions is common to the ARPA/NIST evaluations: it is difficult to compare algorithms which are implemented in different ASR systems, since it is hard to tell if the observed improvement is mainly due to a certain algorithm.

Since the parallel between our objective, namely the evaluation of acoustic modeling

algorithms for noise robustness, and that of the AURORA evaluation is evident, we present in the next section an evaluation exercise which is similar to the AURORA evaluation.

3.4 Description of our Evaluation

Task

The first step in the design of a technology evaluation is to define the task which will be used for the evaluation. Remember that in Chapter 2 we have seen that an ASR systems can be decomposed into three main, more or less independent sub-systems, namely feature extraction, probability computation and decoding. Likewise an ASR task can be decomposed into a feature extraction, a probability computation and a decoding sub-task. We have thus a situation parallel to the one depicted in Fig. 3.1, substituting the global SLP task by the ASR task, the sub-tasks in the figure by the three ASR sub-tasks just mentioned, the sub-systems by the corresponding ASR sub-systems and finally the SLP system by the ASR system. The difference with respect to the previous case is that it is no longer possible to assess the performance of a given sub-system at the sub-task level, i.e. sub-system performance must be assessed at the global ASR task level. This is because no human-generated reference for the outputs or inputs of the mentioned sub-tasks can be generated, and therefore it is neither possible to compare actual outputs with a 'best-possible' output reference, nor it is possible to provide sub-systems with a 'best-possible' human-generated input. Therefore to be able to assess the performance of a given ASR sub-system it is essential to have the other ASR sub-systems. To assess, for instance, the performance of different feature extraction sub-systems, one could use in each assessment different acoustic modeling or decoding blocks. However, if the different sub-system assessments are to be fully comparable, the configurations of the acoustic modeling and decoding sub-systems must remain fixed or as constant as possible over all assessments. Consequently, while evaluating on a given ASR sub-task, the sub-systems implementing the other ASR sub-tasks must remain as constant as possible. Unfortunately, this is not always possible, since some ASR sub-systems impose certain constraints on the preceding or following sub-systems, i.e. they are actually not fully independent from each other.

Environment

Once the task level to evaluate on has been determined, the next step is to determine which is the ASR task environment and which attributes our ASR task must have to match it. A typical application in cars is of the command-and-control type, where the functionality of some devices is controlled by voice. The number of commands is usually small or not very large, and the commands are usually spoken in isolation, e.g. '`rewind`', or at most in very short and precise sentences, like for instance in '`turn the radio off`'. Therefore phenomena typical of spontaneous speech are not an issue for those applications. Speaker independence is, on the contrary, a must for those applications, since it is certainly in advance not clear who the speaker will be. As performance of ASR systems is greatly affected by environmental noise, this is also a very important issue to consider. As was already discussed in Section 1.2, environmental noise may depend on many factors such as speed, open/closed car windows or screen wipers switched on, etc. . . .

Metrics

From Section 3.3 we know that the questions to clarify in this point are three, namely:

- **Qualities to measure**. Since the main goal of the algorithms developed in SPHEAR and RESPITE is to improve ASR performance in a variety of noise conditions, the principal quality to measure is noise robustness. Another secondary quality is the memory efficiency of the algorithms proposed, or equivalently the number of parameters of a recognizer using any of those approaches. Finally an interesting quality to observe would have been the complexity of the algorithms, which is usually measured using the RTF defined in Sec. 3.3. As it will be seen, the evaluation methodology we adopt does not let us reliably measure the RTF, so that just a qualitative judgement of complexity is possible. A second question is what attributes of the task, environment or implementation must vary to measure the selected qualities, and which must be fixed to guarantee a certain comparability. Clearly if the desire is to measure noise robustness the attributes related to environmental noise must vary, i.e. assessments must be carried out ideally over the range of noise conditions which are to be expected during real operation. Since speaker independency is mandatory for the application intended it must remain fixed, if assessments are to be comparable and meaningful for the application. So does the spontaneity of input speech, since it is not an issue for the application intended and therefore phenomena associated with it is not present in the input speech and must not be tackled by the systems. Other I/O task attributes such as language or subject area must remain fixed as well.

  We have already mentioned above that to evaluate on a given ASR sub-task the subsystems implementing the other ASR sub-task must remain as constant as possible over all assessments. This also concerns the objects used by those systems (*object attributes* of the task) to perform the corresponding sub-task. For instance, if two systems using exactly the same decoding block are to be fully comparable, then the lexicons used by the identical decoding blocks must be the same. Otherwise it could be that the results of one of the systems are worse due to a higher confusability of its lexicon or to some OOV word, and not due to differences in the acoustic modeling or feature extraction blocks. Likewise if we are to compare two different acoustic modeling approaches, then the topology, i.e. number of HMM-states and allowed transitions, of the HMMs must be fixed to guarantee comparability. As we have already mentioned, efficiency attributes are not as important in this comparison as accuracy and robustness, since the evaluated algorithms are far from being implemented efficiently. Consequently, we do not demand a uniform implementation for the evaluated sub-systems.

- **Measures**. As we want to assess the robustness of feature extraction or acoustic modeling sub-systems on an ASR task, the most straightforward measure of robustness to use is a kind of mean value of the WER over different environmental noise conditions. The WER for the single tests is computed as described in Sec. 3.3. As for the measures of efficiency, we just grade the evaluated systems according to their number of parameters and to their degree of complexity.

- **Methodology for our evaluation**. We have already mentioned at the start of the section that to ensure a high degree of comparability between ASR sub-system

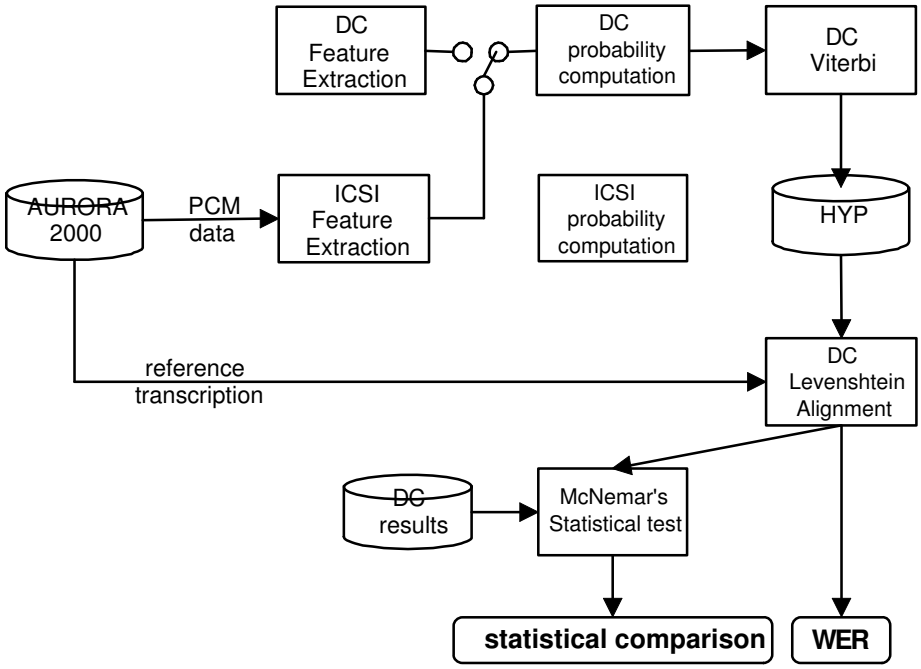


Fig. 3.6: Interfacing process through the feature interface.

assessments, the other ASR sub-systems should remain as constant as possible over all those assessments. Typically we have the situation, however, that ASR systems are not easily separated into a feature extraction, an acoustic modeling and a decoding sub-system, because sub-systems have been so closely coded, that it is no longer possible to take the code of a sub-system in ASR system A and use it in ASR system B. Consequently if the desire is to have an ASR system built up from blocks of different ASR systems, the simplest solution is to define interfaces between those blocks, so as to be able to exchange data between ASR sub-systems. Since we have three different ASR sub-systems or blocks, two different interface points naturally emerge:

- **feature interface**, between the output of the Feature Extraction block and the input of the acoustic modeling block.
- **likelihood/posterior interface**, between the output of the acoustic modeling block and the input of the decoding block.

The interfacing process is illustrated in Fig. 3.6 and Fig. 3.7 where the two different interfaces are shown. To assess the performance of a certain feature extraction algorithm implemented in system B, the feature vectors generated by this algorithm are output, converted into our system's format and input into our system through the feature interface to be further processed by the acoustic modeling and decoding blocks of our system. Of course the parameters of our system's acoustic modeling and decoding blocks must have been previously trained on these kind of features.

The situation is somehow different if the performance of a certain acoustic modeling algorithm is to be assessed. If the given acoustic modeling allows it, the feature vec-

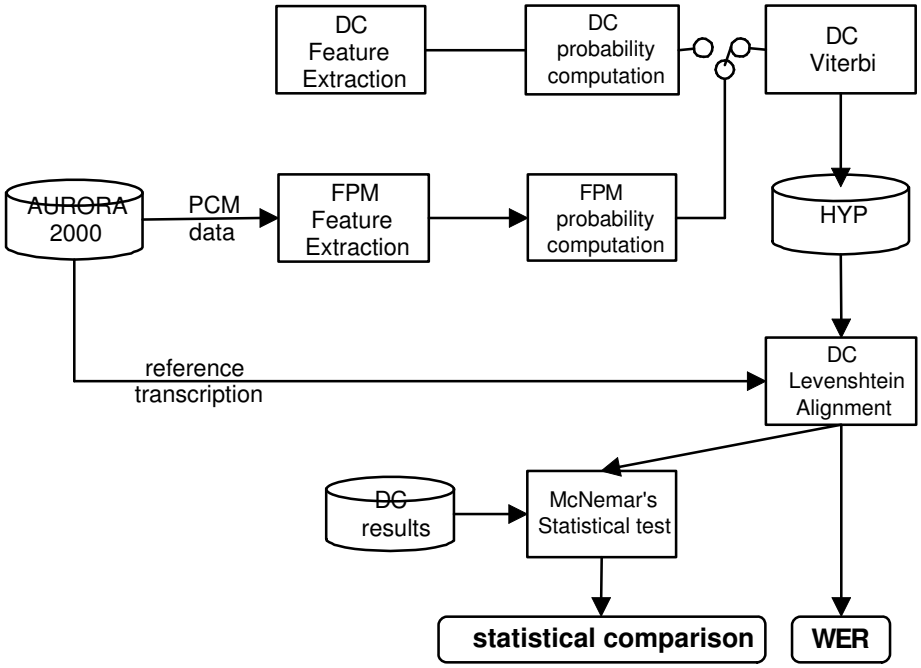


Fig. 3.7: Interfacing process through the state-likelihoods interface.

tors generated by our systems's feature extraction block are output, transformed into the format of system B and input into the acoustic modeling block of system B. This block further processes the features and generates the HMM state likelihoods which are then output, converted into our system's format and input into our system's decoding block to obtain the final hypothesis. Note that in this case the parameters of the acoustic modeling of system B have been trained using our system's features. If on the contrary the acoustic modeling algorithm in system B is not compatible with our system's features, the feature extraction block of system B must be used instead of ours. Features are then further processed by the acoustic modeling block of system B whose output is then converted into our data format and passed to the decoding block of our ASR system.

As we were not allowed to distribute our ASR system or a part of it among our project partners, the only solution left was to carry out, at our lab, the interfacing experiments just described. Since some other project partners had problems as well with a free distribution of their ASR software, we decided to keep things flexible and let them decide between sending us their ASR software or just the features or likelihoods to be input into our ASR system for recognition. If the first variant was chosen, a running version of their ASR software, including all the necessary objects such as HMMs, transform matrices, etc. . . , had to be sent to us. If on the contrary the second option was selected, feature or state likelihood vectors, output by the feature extraction or acoustic modeling block being evaluated, had to be generated for the files in the test set of the AURORA database. In addition, the feature vectors for the train set have to be generated as well, if a feature extraction block is to be evaluated. This is necessary because the parameters of our acoustic modeling block

must be trained on those feature vectors, before we can use them to recognize speech.

Evaluation Data

At this point we already have a very precise idea of the evaluation we want to carry out, and it is thus possible to think of the evaluation data we need. Ideally we would have bee able to use our own English database recorded in car environments for command-and-control applications, because it is perfectly suited to the conditions described above. Unfortunately we were not allowed to distribute this database or a part of it to the other SPHEAR and RESPITE partners, since possible competitors would then have had free access to our database. We thus had two possible options: to use another sub-optimal database or to get the systems from our project partners and evaluate them at our site using our database. This last option was however not feasible since some partners were not eager to give away their systems and not enough manpower was available at our site to fully configure, train and test the five systems for the intended task. Therefore, we decided to use another sub-optimal database that was somehow suited to our evaluation and was readily available to all project partners. The best candidate we had, at the time our evaluation started, was the AURORA-2000 database which is particularly suited to test noise robustness as mentioned in the preceding section. Of course, the database has many drawbacks such as artificially-added noises, some noises not suitable for the application intended and very small size vocabulary, but it was the best, readily available and affordable database we could have at the time.

3.5 Description of the Interfaced ASR Systems

As already mentioned in Chapter 2 and although at first sight the feature extraction and acoustic modeling blocks to be evaluated may seem quite different, all of them have in common that no assumptions on the environmental noise are made. The effect of environmental noise is mitigated by using the inherent redundancy of the speech signal or adding redundancy to it. We start with a full description of our baseline ASR system for the AURORA-2000 ASR task, since the acoustic modeling and decoding blocks of this system are used to evaluate the blocks of other ASR systems.

3.5.1 Missing Data Class-Imputation with Fuzzy Masks

Approach

This technique was first applied to noise robust ASR by researchers at the University of Sheffield [CGM94], and has undergone many refinements and improvements since then. The underlying principles of this theory are:

- **Occlusion of speech**. This principle refers to the fact that during the auditory process speech and noise do not mix up additively, like signal and noise in a normal telecommunications channel. On the contrary, experimental evidence seems to show that the strongest signal occludes or masks locally the weakest, which is then locally missing for any practical purpose.
- **Redundancy of speech**. This second principle claims that the inherent redundancy of speech may be exploited to recover, in some sense, the missing or occluded regions

of the speech signal from the present or speech-only regions. This seems to be corroborated by the fact that human listeners can cope with severely distorted speech by using the inherent high redundancy of this signal [All94, LC97, WRBB95].

Since, according to the two principles above, the speech signal can be segmented into present and missing components and the missing ones may be recovered from the present ones, an ASR system relying on MDT must solve the two following basic problems:

1. Signal-noise segmentation or masks.
2. ASR with missing and present components.

The first problem is not directly related to MDT, although it is crucial to guarantee the successful use of MDT in any ASR application. In fact, it has been shown that the potential of MDT for ASR is huge, as the striking results— virtually clean speech performance over a wide range of SNR— using *a priori* signal-noise segmentations seem to indicate [GBCJ01]. However, these *a priori* segmentations or masks are unattainable in practice because a noise-free version of the utterances is needed to obtain them. Consequently, we need a signal-noise segmentation algorithm which only relies on the noisy speech signal to obtain the mask showing the present and missing components of the signal. Unfortunately, no satisfactory solution to this problem has been found to date, and MDT must therefore be reliant on imperfect signal-noise segmentation.

In the second problem we assume that a signal-noise segmentation or mask is available, which is then used to retrieve the missing from the present components using MDT. The best performing MDT technique to date [JCGV99] is the *class imputation*, or marginalisation of the Gaussian mixture density functions of the HMMs over the missing components. These marginalised density functions are then used in the normal way. In spite of its good performance, the class imputation approach has an important inconsistency in its algorithm that may be limiting its potential performance. Since this approach requires marginalization of normal density functions for each input feature vector, diagonal covariance normal densities are normally used in the acoustic model to reduce the computational burden. This implies that the components of the feature vectors are assumed uncorrelated. At the same time, the components of the feature vector cannot be mixed, because then it is no longer possible to separate missing and present components of the feature vector. As a consequence, we are forced to use feature vectors in the frequency domain where missing and present components are easily separated. But the components of a feature vector in the frequency domain, e.g. filter-bank coefficients, are usually strongly correlated which is at odds with the use of diagonal covariance normal densities. This inconsistency between model and features may be partially mitigated by increasing the number of normal densities in the HMMs, but this increases the computational load. The interested reader can find mathematical and further details about MDT [CGJV99, GBCJ01].

Evaluated System

In order to evaluate MDT for ASR, we received, from the University of Sheffield, an ASR toolkit that implements present/missing mask generation, feature extraction and acoustic modeling for MDT speech recognition. Since MDT is chiefly an acoustic modeling approach, the logical interface point with our ASR system is the likelihood interface. Consequently, the situation in this case is as depicted in Figure 3.7.

The evaluated system was configured as described in the following points:

- **Feature extraction**. filter-bank of 32 linearly spaced filters in the ERB scale between 50 Hz and 3750 Hz [Coo91]. The instantaneous Hilbert envelope at the output of each filter was smoothed with a first order filter with an 8 ms time constant, and sampled at a frame-rate of 10 ms.. A cube root compression was applied to the frame of energy values. No further transformation, e.g. DCT, was used in this case because otherwise present and missing components would then be mixed up. Static features were supplemented with their temporal derivatives, to form a 64 dimensional feature vector.

- **Acoustic modeling**. Topology of HMMs (number of states per word and allowed state transitions)is exactly the same as in our baseline system. Continuous density HMMs with a mixture of 7 diagonal covariance normal densities per state, which results in a total of $7 \times 127 = 889$ normal densities of 64 dimensions. The total number of parameters in the acoustic modeling block is 114935 $((64+64)*889+889+2*127)$.

- **Training**. The Baum-Welch algorithm was used to compute the normal densities, the mixture weights of the densities and the state-transition probabilities. The training set used to train these parameters was the clean set of AURORA 2000.

- **Recognition**. The fuzzy-masks were generated for the files in the test sets of the AURORA-2000 database. The fuzzy-masks used in the recognition step are soft signal-noise segmentations in the spectro-temporal space. These masks are obtained by computing the instantaneous SNR value for each filter-bank coefficient and applying a softmax non-linearity to it, so as to have a value between 0 and 1 [GBCJ01]. After that, fuzzy masks and feature vectors were input into a MDT acoustic modeling that employs class imputation with bounded marginalization as in [GBCJ01] to compute the values of the state likelihoods. The final step was to input the state likelihoods through the likelihood interface into the decoding block of our baseline system. We also used the HHM-state transition probabilities, generated during HMM training with HTK, during decoding. As in our baseline system, no grammar was used.

3.5.2 Tandem Acoustic Modeling with Multiple Streams

Approach

In contrast to the previous approach, which exploits the inherent redundancy of the spectral feature vector, the multi-stream approaches incorporate into the speech recognition process different feature vectors conveying complementary information about the speech signal, or any other signal, for example lip movements, related to the speech production process. The reasons for this combination of different sources of information are manifold, the most important being the incorporation of different aspects or views of a speech sound which may help to discriminate between similar sounds. A second reason more relevant to noise robustness is the fact that different speech features may be unequally affected by certain noises, which can be used to improve performance in noise by weighting or averaging in some sense over all streams of features, so that ASR performance remains constant over a wider range of noise conditions. From a pattern recognition viewpoint

this approach is none other than the so-called mixture-of-experts approach [Bis96, Bou99] which aims at tackling situations in which an input pattern may have different statistical properties, or equivalently in which a different mapping must be used depending on the region of the input space. We will return to the topic in Chapter 6 where the multi-stream approach will be treated in some detail.

On the other hand, the tandem acoustic modeling approach [HES00] uses the output of a neural network classifier as the input features for the Gaussian mixture model, so that the system can be interpreted as having effectively two acoustic models operating in tandem. The neural net is trained on phonetic targets, and therefore the net outputs could be interpreted as phonetic posterior probabilities whose variation in time is then modeled using the GMM/HMM acoustic modeling. According to the authors in [HES00] this allows the GMM/HMM acoustic model to concentrate on the difficult phonetic transitions, since the neural net eliminates spurious variability and magnifies the phonetic transition regions. As it will be seen in Chapter 4, the neural net of this approach can also be interpreted as performing a kind of discriminative dimensionality reduction which is somehow a more natural interpretation than the one explained above.

Both the tandem and the multi-stream can be combined as in [HES00] to obtain further performance improvements. The basic idea is to take two or more neural nets and train each using different kinds of features, which should convey as much complementary information as possible. After that the outputs of the neural nets are combined during recognition to obtain a single feature vector which is then passed to the GMM/HMM acoustic model. Since the outputs of the neural nets are phonetic posterior probabilities the combination can be a simple average of their outputs. In Chapter 6 we will return to the more general question of how to combine multi-stream and tandem-like approaches.

Evaluated System

Our RESPITE partners at the International Computer Science Institute (ICSI) provided us with a package specially designed to train neural nets for classification tasks in ASR: the SPRACHcore package [4]. This is a highly modular package which includes different feature extraction algorithms, very efficient neural net training and classification programs, and a very powerful Viterbi decoder.

The main features of the evaluated system are the following:

- **Feature extraction**. The speech signal was analyzed using a window length of 25 ms and a frame shift of 10 ms. This system uses two different feature streams, namely the well-known PLP features (cf. Sec. B.1) and the Modulation Spectrogram (MSG) features developed at ICSI (cf. Sec. B.3). The PLP features in this experiment consisted of the frame energy plus a total of 12 PLP coefficients, i.e. a total of 13th coefficients. The MSG feature vector consisted of two different 14th-dimensional feature vectors which were concatenated to obtain a final 28th dimensional feature vector.

- **Acoustic modeling**. Dynamic features were appended to the PLP vector and a time window of nine enlarged PLP feature vectors was then used as input to the neural net resulting in $13 \times 3 \times 9$ input units. No dynamic features were appended

[4] http://www.icsi.berkeley.edu/~dpwe/projects/sprach/sprachcore.html

to the MSG static features but a time window of 9 MSG frames was used as well, resulting in a total of 28×9 input units. Both MLPs had just one single hidden layer of 480 sigmoidal units, and an output layer of 24 softmax units each associated with one of the phonetic classes present in the AURORA digit corpus. The activation function of the units in the hidden layer was a sigmoid whereas that of the output layer is a softmax function, as described in Section 5.5.1. The scHMMs consisted of a supervised code-book of 127 normal densities with 24 dimensions, and the same topology as in our baseline system. The total number of parameters used in the feature extraction and acoustic modeling blocks was 370011 $((351 + 24) \times 480 + (252 + 24) \times 480 + 127 \times ((24 \times 23)/2 + 24 + 24) + 127 \times 127 + 2 \times 127)$.

- **Training**. The targets of the neural nets were hard targets (output values are 1 or 0) obtained from a phonetic segmentation of the AURORA 2000 multi-condition train set. The coefficients of PLP and MSG features were normalized to have null mean and variance one, to avoid problems with the net training algorithm and to speed up the computation. The neural net training was performed using the usual error back-propagation algorithm (EBP) with cross-validation to avoid over-fitting (cf. Sec. 5.5.2), and the process was stopped when cross-validation scores did not change significantly or after a maximum of 12 EBP iterations. After MLP training, the pre-nonlinearity outputs of the MLPs were used as features in order to have features which are better adapted to the GMM/HMM acoustic model. The combination of the outputs of the PLP and MSG nets was achieved by simply adding the pre-nonlinearity outputs of both nets as reported in [HES00].

  Subsequently the combined features were used to train the scHMMs, as in our baseline training using a supervised code-book. The difference was that there was no need for an LDA transform in this case since dimensionality reduction had already been performed by the neural net. Therefore, the supervised code-book of 24-dimensional normal densities was trained using the same segmentation into states as in our baseline system (cf. Sec. 7.3.1). As in our baseline training, the quantized features were used during Baum-Welch training to compute the emission probabilities of the vq-symbols in each HMM state and the state-transition probabilities. The training set used to train neural nets and scHMMs was the multi-condition set of AURORA 2000.

- **Recognition**. We started the evaluation of the trained tandem multi-stream system by generating the pre-nonlinearity outputs of the nets for the test sets of the AURORA 2000 database, and combining them as in the training step. After that we input the combined feature vectors through the feature interface into the GMM/HMM acoustic modeling of our ASR system, configured as described in the training step, to obtain the final sentence hypothesis. As in the previous experiments, the decoding block of our ASR system does not employ any grammar at all.

3.5.3 Multi-Stream Hybrid MLP/HMM System

Approach

Neural nets are known to be powerful tools for pattern recognition tasks [Lip87, Bis96] but they fail to grasp the time variation of the speech signal when they are used in isolation.

To overcome this problem, a series of approaches using neural nets were developed but it was not until the early 90s, as the first hybrid ANN/HMM systems were being developed [BW90], that a first successful neural net approach for ASR arrived. The rationale behind hybrid ANN/HMM acoustic modeling is to combine the outstanding discrimination capabilities of neural nets with the good modeling of time variation offered by HMMs. Essentially the idea is to substitute the GMM used in conventional HMMs by an MLP (cf. Sec. 5.4) with one hidden layer, and use it to compute the state posterior probabilities as shown in Figure 5.3. Actually, what makes both approaches so different is the training algorithm, since the ANN is trained to approximate conditional state posterior probabilities instead of conditional state likelihoods, and is therefore a discriminative training by nature. This means that during training not only the likelihood of the right state is maximized but also the likelihood of the incorrect states is minimized. For a detailed review of the hybrid approach we refer to [MB95] and to Chapter 5.

This approach can be combined with the multi-stream paradigm in a way similar to the tandem approach. As in this approach, two or more neural nets are trained independently on two or more complementary feature sets. Since the neural nets have been trained using the same output targets, the outputs of both nets can be combined during recognition to obtain the final state posteriors, which are then passed to the decoding block to generate a final recognition hypothesis.

Evaluated System

The most important characteristics of the evaluated system using the previous approach are detailed in the following points:

- **Feature extraction**. The evaluated hybrid ANN/HMM system used exactly the same feature streams as the tandem multi-stream system, namely the PLP and the MSG feature sets.

- **Acoustic modeling**. The topology of the neural nets used is different from the one used in the tandem multi-stream system, since the targets of the neural nets in this case are the 127 states of the whole word HMMs of our baseline system. This makes a total of 127 output units for each neural network (cf. Sec. 7.3.1). The number of units in the hidden layer for both neural nets is 480, and the number of input units for the PLP neural net is $13 \times 3 \times 9$ whereas the MSG neural net has 28×9 input units. The activation function of the hidden and output units are the same as for the tandem system. The total number of parameters used in the acoustic modeling for this approach were 411360 $((351 + 127) \times 480 + (252 + 127) \times 480)$.

- **Training**. As in the tandem case, the coefficients of PLP and MSG features were normalized to have null mean and variance one. Both neural nets are trained independently on each of the feature sets, using exactly the same segmentation into HMM-states we used to obtain the code-book for LDA in our baseline system (cf. Sec. 7.3.1). Neural nets were also trained using the same algorithm (EBP) and methodology as for the tandem system, i.e. hard targets, cross validation, etc.... The training set used to train both neural nets was the multi-condition set of AURORA 2000.

- **Recognition**. After training both neural nets, they were used to generate the HMM-state posteriors for each speech frame in the test sets. The HMM-state posteriors obtained from each neural net were subsequently combined to obtain a single posteriors vector per frame. The combination technique was similar to the one used in the tandem multi-stream approach, but instead of adding the pre-nonlinearity outputs we added the logarithms of the outputs, i.e. we multiplied the posteriors. The resulting HMM-state posteriors vectors were then input into our ASR system through the likelihood interface, and were decoded using the Viterbi decoding block of our baseline system. As in the previous assessments, no grammar was used during the decoding.

3.5.4 Multi-Band Noise-Contaminated Training System

Approach

The multi-band paradigm for ASR is a kind of multi-stream approach in which the streams are associated with frequency sub-bands. This approach was first suggested by Bourlard [BD96, BD97] who was largely inspired by the work of Fletcher [All94], which dealt with the effects of telephone channel distortion on human speech intelligibility. In this approach the spectral sub-bands are independently processed and afterwards recombined at some later stage of the ASR process. Theoretically the benefit of this technique for noise robustness is the possibility to recognize noisy speech by discarding the noise-corrupted sub-bands and using just those that have not been affected by noise. Another potential advantage of the multi-band approach would be the possibility to model asynchrony between sub-bands, since it could well be that phoneme transitions do not occur synchronously between sub-bands. But to allow for asynchronism between sub-bands a significant adaptation of the recognizer must be performed [Bou99, WKM98, VM90], since the recombination is no longer at the HMM-state level.

In spite of some successful initial tests using a very simple multi-band approach to deal with artificial narrow-band noises and the continual refinement of this approach using more elaborated sub-band combination methods [MHB99, GB00, MHGB99, HM00], it was not possible to demonstrate any significant improvement with respect to the full-band baseline systems when the interfering noises were wide-band or real-world noises [Hag01, pp.132,170]. Nor has any of the tested multi-band asynchronous combination algorithms shown a significant and clear improvement over the synchronous or state-level combination approach [MM99].

Recently, however, a variant called multi-band noise-contaminated approach based also on the multi-band paradigm has been successfully used to deal with wide-band real-world noises [DR01] was developed by researches at the Faculté Polytechnique de Mons (FPM). The idea is based on the simple but clever observation that, if sufficiently narrow frequency bands are considered, noises inside these bands differ only in their energy level, not in the shape of their band-limited power spectrum. Consequently, if we train our acoustic models using speech data artificially corrupted by any kind of wide-band noise added at different SNR levels, it can be expected that these acoustic models are robust to other kind of noises different from that used to corrupt the training data.

Evaluated System

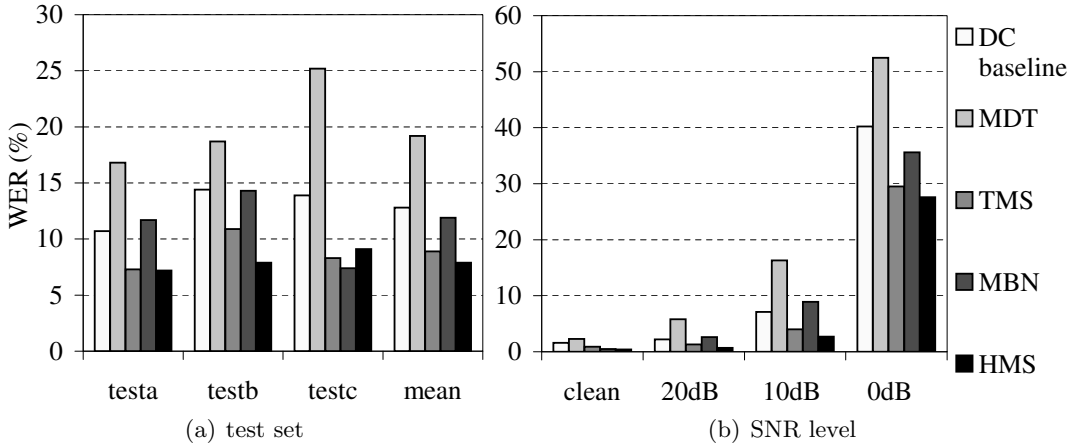
The main features of this system are explained in the following points:

- **Feature extraction**. In a first step a critical band analysis of the speech frames was carried out based on a bank of 30 trapezoidal filters equally spaced along the Bark scale. A temporal filter of the J-RASTA kind [HM94] was then applied to the temporal trajectories of the filter-bank coefficients. The 30 filtered critical band energies were then divided between 7 frequency sub-bands and therefore grouped into sub-vectors. The 7 sub-bands were a kind of compromise between the assumption of sub-band narrowness and keeping enough speech specific information in each sub-band.

- **Acoustic modeling**. Each sub-band vector was normalized to make them independent of the absolute energy of the speech frame. A temporal window of 15 frames was afterwards constructed for each sub-band and then transformed using a non-linear transform implemented via an MLP with 2 hidden layers, in a way similar to the tandem system described in Sec. 3.5.2, since phonetic targets are used as well in this case. However, the output vector space used during recognition is not the output of the output layer but the output of the 2nd hidden layer. It will be shown in Chapter 4 that this is actually theoretically related to linear discriminant analysis (LDA). These neural nets have 1000 nodes in the 1st hidden layer and 30 in the 2nd hidden layer. The non-linearly transformed features were then concatenated to obtain an acoustic feature vector that can be used for ASR in the conventional sense. The training data used to train these neural nets was a contaminated (henceforth 'contaminated training') version of the clean train set of the AURORA 2000 database. To contaminate the clean data, white noise at different SNR levels was artificially added. The acoustic modeling used was a rather large hybrid MLP/HMM acoustic model [MB95] with just one single hidden layer. The inputs of this model were 3 concatenated frames of the above described feature vector which gave $3 \times 7 \times 30$ input nodes. The number of nodes in the hidden and output layers were 1000 and 127, respectively. These 127 output probabilities modeled the conditional posterior probabilities of the states in our baseline HMM topology given the current input vector x_t. The total number of parameters in the feature extraction and acoustic modeling blocks was 1531185 parameters. A more detailed description of this system can be found in [DR01].

- **Training**. All neural nets in the acoustic modeling were trained using the usual EBP algorithm.

- **Recognition**. Unlike the previously described techniques, we were provided in this case with just the state likelihoods of the data in the test sets of the AURORA 2000 database. Therefore the state-priors for the files in the test sets of the AURORA 2000 database were generated at FPM using the feature extraction and the acoustic model described above, and recorded on CDROMs which were then sent to us for evaluation. We subsequently input the data in the CDROMs into the Viterbi decoding block of our ASR system through the likelihood interface to obtain the final sentence hypothesis. As in the previous cases no grammar at all was used during decoding.



(a) test set (b) SNR level

Fig. 3.8: WER results over the three test sets of the AURORA 2000 database. The WERs in (a) are the averages over all noise types and SNRs at 0 dB, 10 dB, 20 dB and clean, whereas those in (b) are the averages over the all the test sets and noises.

3.6 Experimental Results and Discussion

To validate the interfaces described in Sec. 3.4, we performed a series of experiments on the AURORA database. In a first set of experiments, we tested the operativeness of the feature interface with the tandem multi-stream features. Results were similar to the results obtained by ICSI with different acoustic modeling and decoding blocks, although for low SNRs (0 dB) the results of our system were slightly worse.

For the likelihood interface we used the likelihoods from the MDT system in Sec. 3.5.1 for validation. The results obtained with our decoding system were similar to those obtained at Sheffield University, but Sheffield's results were slightly better than ours because their system used a kind of grammar during decoding. This grammar forced noise or silence models at both ends of the sentences, which improved the results in the AURORA case, specially in low or very low SNRs, due to the long pauses at both ends of the sentences in the database.

3.6.1 Evaluation Results

The evaluation results of the approaches presented in the previous sections are shown in Fig. 3.8. The acronym MDT corresponds to missing data class-imputation with fuzzy masks, TMS to tandem acoustic modeling with multiple streams, MBN to multi-band noise-contaminated training, and finally HMS to multi-stream hybrid MLP/HMM. The details about the DC baseline can be found in Sec. 7.3.1.

In Fig. 3.8 we can observe:

- The hybrid MLP/HMM system is on average the best performing system. The average difference with respect to the next best system (TMS) is about 13% relative, whereas the difference with respect to our baseline is 62% relative. The differences between hybrid MLP/HMM and the tandem system, however, are not uniform across the different test sets. In matched test conditions (test a) both systems have a similar performance, whereas in unmatched test conditions (test b) the performance of the

	DC	MDT	TMS	MBN	HMS
DC	-	←	↑	↑	↑
MDT	↑	-	↑	↑	↑
TMS	←	←	-	←	↑
MBN	←	←	↑	-	↑
HMS	←	←	←	←	-

Tab. 3.3: Pairwise statistical comparison of the overall results on AURORA 2000 of the evaluated systems using the McNemar's statistical test. The direction of the arrow indicates which of the two systems is better according to this test.

hybrid system is clearly better. In contrast, the performance of the tandem system is better in the test c set, where the input speech has been additionally distorted by a channel response.

- The second best performing system— the tandem multi-stream system — is also much better than our baseline with a 45% average improvement. Moreover, this improvement is consistent across all test sets and SNR levels.

- The multi-band noise-contaminated training approach is also on average better than our baseline system (7.5% relative). However, the differences are not consistent across the different test sets. For the test a set, our baseline is clearly better than the previous system, whereas on the test b both systems have a similar performance. For the test c, in contrast, the performance of the multi-band system is much better. Moreover, this system is the best performing system on this test set, which agrees with the theoretical expectations discussed in Sec. 3.5.4.

- The worst performing system is the missing data class-imputation with fuzzy masks system. Its average performance is about 50% worse than that of our baseline, and the differences are consistent across SNRs and test sets.

To confirm that the differences observed in the previous table are statistically significant, we applied the McNemar's test on each pair of results. The results of these tests are shown in Tab. 3.3, where the arrows in each cell point to the best system according to the test. As we can see, all the differences observed in Fig. 3.8 are statistically significant.

To evaluate the complexity of the previous approaches we have qualitatively estimated the training and recognition complexity of each system. A comparison of the 'complexity of the training data' set and of the training algorithm are shown in Tab. 3.4. By 'complexity of the training data' we mean the degree of difficulty of preparing the training data. Thus, the missing data approach must be trained on clean speech for the approach to make sense, and the data must therefore be carefully recorded. For the multi-band noise contaminated approach we need clean training data as well, but we additionally have to add white noise at different SNR levels to the clean data. As we can see in the previous tables, the complexity of both the tandem and hybrid MLP/HMM multi-stream approaches is larger in the training step due to the EBP training (cf Sec. 3.5.2 and Sec. 3.5.3) needed to train the weights of the neural nets used in both approaches. During recognition, the use of the neural net instead of the LDA matrix in the tandem approach increases the computational burden. However, the most complicated system by far is the multi-band

System	training complexity data	algorithm
DaimlerChrysler baseline	+	+
Missing Data	++	+
tandem multi-stream	+	+++
Multi-Band Noise Contaminated	+++	+++
Hybrid MLP/HMM multi-stream	+	++

System	recognition complexity
DaimlerChrysler baseline	+
Missing Data	+
tandem multi-stream	++
Multi-Band Noise Contaminated	+++
Hybrid MLP/HMM multi-stream	+

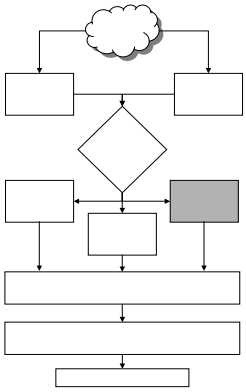
Tab. 3.4: Qualitative comparison of the complexity of the participant systems. Complexity of the training step is shown in (a), whereas the complexity of the recognition step is shown in (b).

noise contaminated system since the number of weights of the two neural nets used in this approach is very large (cf. Sec. 3.5.4).

3.7 Summary

In this chapter we have first explained the difference between user-centered and technology evaluation. Next we have discussed the elements of this last kind of evaluation, and how such an evaluation must be designed to have a meaningful evaluation. After that we have discussed the kind of measures used in technology evaluation, with special emphasis on accuracy measures. We have seen that to measure the accuracy of an ASR system we basically need an alignment between reference and hypothesized sentence, a measure on this alignment, and finally an statistical procedure to test the significance of the differences between scores.

In the second part of this chapter, we have explained our chosen evaluation problem, namely the evaluation of feature extraction or acoustic modeling blocks. The proposed solution to this problem has been to define an interface at the feature level and another at the state-likelihood level, to be able to use the same acoustic modeling or decoding block across all assessments. This guarantees a high degree of comparability between assessments, and consequently a meaningful evaluation of feature extraction and acoustic modeling blocks. We have also briefly described the different approaches we have evaluated, and finally we have shown the results of our evaluation. These results clearly show the advantage of using connectionist approaches (tandem and hybrid MLP/HMM) combined with multi-stream speech recognition to improve recognition performance across a range of noises and SNRs. For this reason, we devote the following three chapters of this thesis to the study of hybrid ANN/HMM, of tandem in the general framework of feature reduction for classification, and finally of multi-stream ASR.



4. Feature Reduction Methods for Classification

In this chapter we study the tandem approach by interpreting it as a kind of feature reduction method for classification, which is a family of mappings from a high-dimensional space into a low-dimensional one in which the separation of the classes is similar to that in the original high-dimensional space. These kind of mappings are used in ASR to be able to add context to the feature frame without significantly increasing the processing time, and the amount of parameters to train. After reviewing some literature in the field, we proceed with the fundamentals of feature reduction for classification to explain which is an optimum transform for feature reduction. Next the usual linear discriminant analysis (LDA) approach is discussed and related to the optimum mapping in the previous point. Finally a family of connectionist approaches is discussed, which use multi-layer perceptrons (MLP) to approximate the optimum mapping.

4.1 Introduction

It is well known that ASR systems can take advantage of the context of a feature frame [Fur86]. Appending dynamic features or using a large context window of 5-10 consecutive feature frames is common practice among speech researchers. Since by doing so we are increasing the dimensionality of our feature space the two following problems arise:

- Larger processing time in the acoustic modeling stage.
- The so-called *curse of dimensionality* which may cause poorly trained acoustic modeling parameters (cf. Sec. 2.6).

To solve both problems feature dimensionality reduction methods can be used. The idea is to compute a mapping from a high-dimensional into a low-dimensional space that preserves some intrinsic or extrinsic characteristic of the speech signal. As depicted in Fig. 4.1 the mapping takes a window of consecutive feature frames and maps it into a low-dimensional vector, which is then passed to the acoustic modeling stage. Depending on the nature of the characteristic preserved, the methods to compute that mapping can fall into the two following categories:

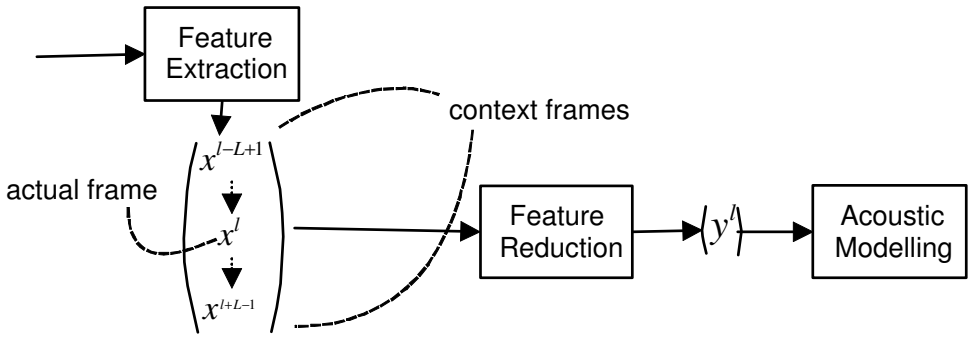


Fig. 4.1: A system representation of the feature reduction process embedded in an ASR system.

- **feature reduction for signal representation**, e.g. principal component analysis (PCA), which compute the mapping to preserve some intrinsic characteristic of the signal [Fuk90].
- **feature reduction for classification**, e.g. linear discriminant analysis (LDA), which compute a mapping to reduce the dimensionality of the input feature vector without reducing the classification capability in the original high-dimensional space [Fuk90].

Since the task of the acoustic modeling stage is actually a classification one the methods belonging to the 2nd group are theoretically more suited to our problem.

A common approach in ASR is linear discriminant analysis (LDA) which uses a non-square matrix to reduce the dimensionality. In ASR the classes used to compute the LDA matrix are usually associated with the HMM-states. To find those classes a segmentation into HMM-states of the training database is used in order to assign an HMM-state to each feature frame in the training set. As in Sec. 2.6.1, the mean vectors and covariance matrices of the classes in the LDA are then found by simply computing the mean and covariance over the feature vectors belonging to the same HMM-state (supervised learning). As seen in Sec. 4.3, the LDA approach is only optimum in the Bayes sense if the classes are normally distributed and have equal covariance matrices. A group of classes with these properties are called homoscedastic classes as opposed to heteroscedastic classes, which have different covariance matrices. These assumptions contradict the results of the statistical tests shown in Table C.1 of Appendix C where the statistical test reject the null hypothesis (normality) for all the states of the HMMs.

But even if the classes were normally distributed, the assumption of equal covariance matrices would also be violated. In Fig. 4.2 we have plotted, for instance, the distribution of the classes in the code-book (class distributions are assumed normal) on the c_1 - c_3 plane, where c_1 and c_3 are the 1st and the 3rd cepstral coefficients (other choices for the coefficients give a similar result). The code-book has been computed on the same features as for experiment in Tab. C.1. In the plot the classes have been assumed to be normally distributed, and the ellipses represent the contour lines of the densities.

To relax the strong assumption of heteroscedasticity some authors have proposed the use of an heteroscedastic discriminant analysis (HDA) linear transform [SPGC00] . However, no closed form solution exists for this linear transform, and it must therefore be found using a numerical algorithm. In addition, the HDA transform is still based on the

normality assumption which we have seen to be false in practice.

Therefore it could be interesting to try discriminative dimensionality reduction approaches that do not make such strong assumptions on the input data, and that are therefore optimum in the Bayes sense. As it is further seen in this chapter, this can in theory be achieved by using Neural Networks (NN).

4.2 Literature Overview

One of the first uses of LDA in ASR was reported in [HUN92], where a significant improvements using LDA feature reduction for large vocabulary speech recognition was reported.

Since LDA makes strong assumptions on the input data, some authors tried to find feature reduction mapping which were still linear, but that do not make such strong assumptions on the data. In [KA98], for example, a generalization of LDA to the heteroscedastic case using the ML framework is proposed. The idea is basically to embed the computation of the linear transform into the HMM training. Experiments were performed on the isolated digit part of the TI-Digits database, and it was found that the heteroscedastic transform is better than LDA, specially when the context of the actual frame is used.

In another study [SPGC00], a generalization to the heteroscedastic case of Fisher's discriminant is proposed in order to find a linear feature reduction mapping, which is termed

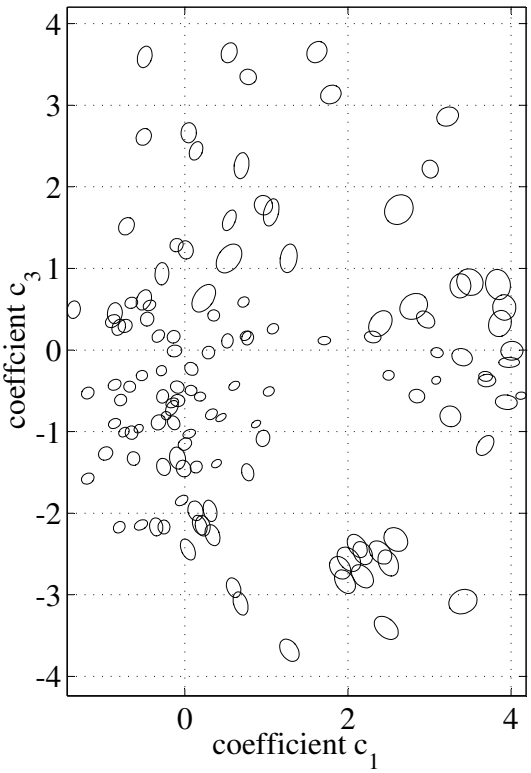


Fig. 4.2: A scatter plot of the classes in the normal code-book of an SCHMM computed on the same features as in Table C.1. Classes are assumed to be normally distributed in the plot. Note that the covariances of the classes are different.

heteroscedastic discriminant analysis (HDA). Since no closed form solution to this problem exists a quasi-Newton gradient descent procedure is used to iteratively approximate it. To improve the adaptation of the reduced feature vector to the diagonal covariances used in the acoustic modeling stage two different approaches are used:

- **HDA with MLLT**. A HDA transform followed by a maximum-likelihood linear transform which adapts the reduced feature vector to the diagonal covariances.
- **Diagonal HDA (DHDA)**. This transform includes in the criterion the adaptation of the output vector to the diagonal covariances.

Experiments were performed on the databases Voicemail and Switchboard, and showed the superiority of the combination of HDA and the MLLT transform with respect to LDA and DHDA.

In [DDC99] the maximum mutual information (MMI) criterion between input and output of the mapping is used to compute a linear discriminant transform. The advantage with respect to LDA and HDA is that the classes can be mixtures of normal densities. As in the HDA case, the optimum solution must be found iteratively. The target classes were associated with the HMM states. The features were decorrelated after the MMI transform, to improve the match to the acoustic model. Experiments were performed on the Resource Management task, and showed a significant improvement with respect to LDA.

In [TRP00] the authors used the minimum classification error (MCE) criterion and Gradient Probabilistic Descent (GPD) to find a linear transform. The approach is termed extended linear discriminant analysis with model transformation (ELDA-MT), and the essential assumption is that there is a dominant Gaussian mixture for each HMM state, which is then associated with one of the classes in the mapping computation. Experiments were performed on the Verbmobil corpus using context-independent HMMs, and showed a significant improvement of the approach with respect to classical LDA.

All the previous approaches use a linear transform which is just optimum in the Bayes sense when the classes are normally distributed and have equal covariance matrices. Therefore some authors have studied the case where a non-linear transform is used and implemented with ANNs. For example, in [RW98] the basic idea is to first train the HMMs without frame context, and then use the trained HMMs to find a mapping (linear or non-linear) between an extended feature vector with context and the original vector space where the HMMs were trained. The training of the mapping is performed using the maximum mutual information (MMI) criterion, and the state density functions of the HMMs. During the training of the mapping the parameters of the HMMs remain fixed. If the original input space of the HMMs had N dimensions and the number of context frames used is K, then the mapping transforms a $(N \times K)$-th dimensional vector into a N-th dimensional one. Experiments on the Resource Management task were conducted using a linear transform, a Multi-Layer Perceptron (MLP) and an Recurrent Neural Net (RNN). Experiments with more than one stream were also performed (cf. Chapter 6). For monophone HMMs the performance was similar or just slightly better than classical LDA. For triphone HMMs and 4 streams, in contrast, the recognition accuracy of the new system was shown to be better than LDA.

Fontaine *et al.* [FRB97] use multi-layer perceptrons (MLP) of one and two hidden layers to implement a linear and a non-linear discriminative transform, respectively. Both kinds of MLPs are trained to approximate the class posterior probabilities, in the sense

discussed later in this chapter. The classes are associated with the phonemes, and the reduced features used for recognition are the outputs of the last hidden layer. Experiments were performed on the PhoneBook database using context-independent GMM/HMM and hybrid MLP/HMM acoustic modeling. For the first kind of models a relative improvement of 25% was observed, whereas for the second kind the improvement was not so significant. Although the motivation given by the authors to use non-linear mappings (LDA makes strong assumptions on the data) is very interesting, there seems to be no result with classical LDA in the study. Also, it would be interesting to study how the approach can be extended to more complex context-dependent systems.

In [HES00] the tandem acoustic modeling approach is proposed, in which an MLP is used to approximate the phoneme posterior probabilities. During recognition the pre-nonlinearity outputs of the MLP are decorrelated using PCA and passed to the acoustic modeling block. These two steps improve the match between features and CDHMM acoustic modeling based on Gaussian mixture Models (GMM). Although in this work the MLP indeed implements a mapping from a high dimensional space (frame context is also used) into a low dimensional one (phonemes), the authors do not interpret it as a feature reduction transform for classification. They rather prefer to interpret it as two acoustic modeling blocks acting in tandem. According to the authors, the first (MLP) magnifies and sensitively maps the boundaries between phones, but coarsely reflects the mid-regions. This should be a desirable property of feature vectors, and can consequently be easily modeled by the GMM/HMM acoustic model. However, this interpretation has a weak mathematical background which may support further developments of the technique. Therefore we prefer to interpret this technique as a kind of NLDA with another kind of MLP topology. Experiments were performed on the AURORA 2000 database, and showed a large improvement over the baseline system(35% relative). Results of the new approach were also compared to those of a phonetic-based hybrid ANN/HMM system, and showed also an improvement in accuracy with respect to those systems. However, we think that this last difference is rather due to the whole-word acoustic modeling used by the tandem system. In fact, the evaluation results presented in Chapter 3 seem to support this fact, since the results of our whole-word hybrid ANN/HMM system were better than those of a similar tandem system. In another set of experiments, the robustness of the new approach to cross-corpus experiments was tested. It was found that the new approach is very sensitive to a change in the task, although we think that this is also the case for classical LDA, specially if the training corpus is rather small.

In [RHWR96] a similar approach to the tandem acoustic modeling is proposed. As in the tandem system, an MLP with one hidden layer is used to implement the feature reduction mapping. In contrast to the tandem system, however, the target classes for the mapping are the states of phonetic HMMs (a total of 169 states). Since the resulting number of outputs is too large to be used as a feature vector, a PCA transform is used to further reduce the dimensionality. The topology of the MLPs used in the experiments is also smaller than those in the tandem experiments. The input vector of the MLP is built from a context of only 5 static frames or 3 static frames with appended dynamic features. The hidden layer of the MLP has also only 100 or 50 units, which may be too few to model the high-dimensional output space. As opposed to the tandem case, which uses softmax units in the output layer, the units in that layer are sigmoidal. The acoustic model used semi-continuous HMMs and the features were the scaled outputs of a Bark-

scaled filter-bank of 20 filters. The energy of the frame was also added to the feature vector resulting in a 21-dimensional vector. A particularity of the feature extraction is that the filter-bank outputs are scaled to sum up to 1. The MLP is trained using the usual error back propagation (EBP) algorithm and after MLP training the PCA matrix is trained on the output vector of the MLP. Phone recognition experiments are performed on the Diphone database in German. The results show an improvement in phone classification for the non-linear mapping with respect to LDA, although the improvement is not very large, and of course it does not imply that this improvement is also observed in a real ASR task.

In the following section we discuss a common framework for feature reduction for classification, of which classical LDA, NLDA and tandem are special cases.

4.3 Feature Reduction for Classification

The optimum feature reduction mapping for classification is one that is optimum in the Bayes error sense. The Bayes error is the error probability of the best possible classifier, i.e. the Bayes classifier. This Bayes classifier [Fuk90] is a central concept of statistical pattern classification, and it simply states mathematically the intuitive notion that the class with maximum probability should be chosen. Let $\mathbf{x}$ be the input vector and let $\{\mathcal{C}_1, \ldots, \mathcal{C}_L\}$ a set of L labels each assigned to one of the L classes. The optimum class $\mathcal{C}_{opt}$ in the Bayes sense is then:

$$\mathcal{C}_{opt} = \arg\max_{i=1,\ldots,L} P(\mathcal{C}_i|x) \tag{4.1}$$

In Eq. 2.1 we have already used this rule to state the fundamental ASR problem of finding the most probable word sequence W_{opt}. Having defined the Bayes decision rule, the conditional Bayes error [Fuk90] is given by:

$$\begin{aligned} r(\mathbf{x}) &= \sum_{\mathcal{C}_i \neq \mathcal{C}_{opt}} P(\lfloor_i|\mathbf{x}) \\ &= 1 - \max_{\mathcal{C}_i} P(\mathcal{C}_i|\mathbf{x}) \end{aligned} \tag{4.2}$$

and the Bayes error is then the expected value of this last quantity:

$$\begin{aligned} \varepsilon_b &= \mathrm{E}\{r(\mathbf{x})\} \\ &= 1 - \sum_{i=1}^{L} \int_{\mathbf{x}\in\mathcal{C}_i} P(\mathcal{C}_i|\mathbf{x}) p(x) d\mathbf{x} \\ &= 1 - \sum_{i=1}^{L} P(\mathcal{C}_i) \int_{\mathbf{x}\in\mathcal{C}_i} p(\mathbf{x}|\mathcal{C}_i) d\mathbf{x} \end{aligned} \tag{4.3}$$

A good feature reduction mapping for classification is one that leaves the Bayes error ε_b as unchanged as possible after transformation. However, this definition is too loose to be useful in practice, and what we need is some kind of criterion that a mapping should satisfy to leave the Bayes error unchanged. As a first step toward this criterion, we define what is an admissible transform or mapping:

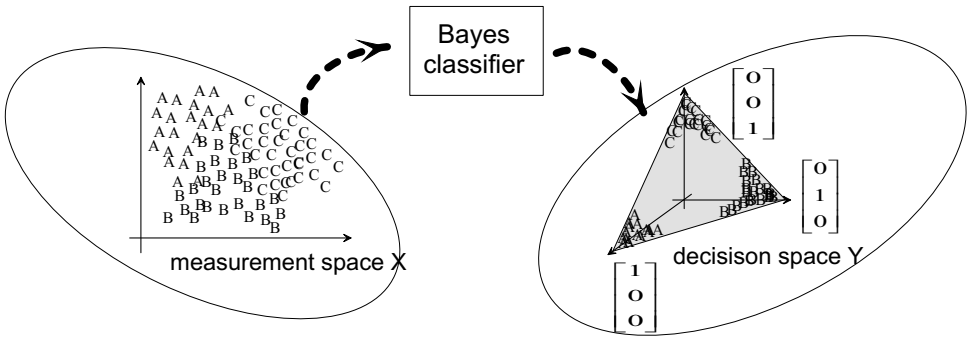


Fig. 4.3: A figure showing the distribution of the feature vectors for a three-dimensional problem after transformation using the Bayes classifier.

Definition 4.1 (Admissible mappings)
Let $C : \mathbb{R}^n \to \mathbb{R}^m$ be a mapping. If $\mathbf{y} = C(\mathbf{x})$, $\varepsilon_b = \mathrm{E}\{r(\mathbf{x})\}$ and $\varepsilon'_b = \mathrm{E}\{r(\mathbf{y})\}$ then the mapping C is said to be admissible if:

$$C: \qquad \varepsilon'_b = \varepsilon_b$$

that is, the Bayes error in the output space is the same as in the input space.

From a geometrical viewpoint an admissible mapping C preserves the relative class separability or distances between classes in both spaces. Any invertible mapping C is an admissible mapping. Another non-trivial example of admissible mapping is one that maps the high dimensional input vector $\mathbf{x}$ into the *a posteriori* probabilities $P(\mathcal{C}_i|x)$ of the relevant classes for our classification or pattern recognition problem. This is easily understood if we consider the maximum term in Eq. 4.2 for the transformed feature vector $\mathbf{y}$, that is:

$$r(\mathbf{y}) = 1 - \max_{\mathcal{C}_i} P(\mathcal{C}_i|\mathbf{y})$$

since we know that the value of $\mathbf{y}$ already contains the class posteriors of the input frame $\mathbf{x}$, it is clear that $P(\mathcal{C}_i|\mathbf{y}) = P(\mathcal{C}_i|\mathbf{x})$, and consequently $r(\mathbf{y}) = r(\mathbf{x})$. This in turn implies that the Bayes error is equal in both spaces.

The distribution of the feature vectors in the output space is illustrated in Fig. 4.3. We can see that the vectors are distributed above the positive region of the hyperplane $\sum_i^m y_i = 1$, since posteriors are positive and must sum up to 1. Note also that if the Bayes error is low, the reduced features tend to concentrate around the points $(0, \cdots, 0, 1, 0, \cdots, 0)$.

In fact, it intuitively makes sense that the posterior probabilities of the classes are the ideal features for classification since these features can be classified with the simplest classifier (bisector classifier) as shown in Fig. 4.4.

From the discussion above, it can readily be seen that a first key problem of feature reduction for classification is to define or find the classes relevant for the ASR problem. From the viewpoint of feature reduction, it would be desirable to have as few classes as possible in order to have a low-dimensional feature space. As already mentioned in the introduction, these classes are usually associated with the HMM-states of our ASR system. However, for ASR tasks with a large number of states the use of such a large feature vector is prohibitive for the reasons already given in the introduction. It could well even be that there are more states than dimensions in the input feature vector $\mathbf{x}$, so that the admissible

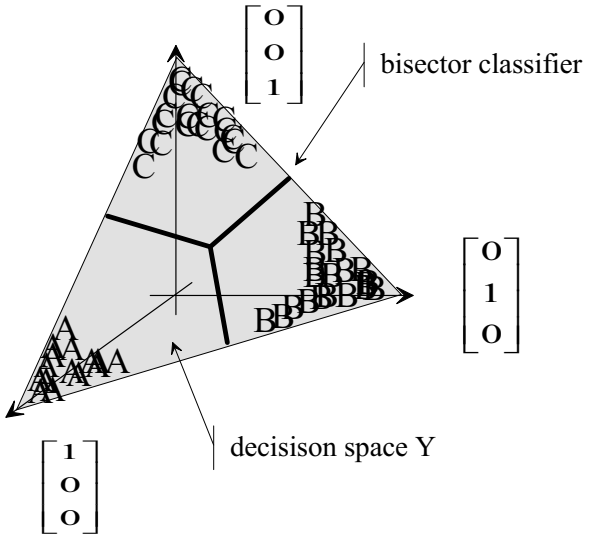


Fig. 4.4: A figure showing the bisector classifier in the transformed space.

mapping C is actually no feature reduction mapping at all, and therefore of no interest for our purposes.

To overcome this problem, we have to re-interpret the admissible mappings C as mappings into a feature space where we are still able to classify the features as good as in the original feature space $\mathbf{x}$. This interpretation is mathematically stated in the following theorem, which imposes a general condition on a mapping C to be admissible in the sense of Definition 4.1 :

Theorem 4.1 (Generalized Admissibility [DGL96])
Let $C : \mathbb{R}^n \to \mathbb{R}^m$ and $D : \mathbb{R}^m \to \mathbb{R}^k$ be two multi-dimensional mappings. Furthermore let $\mathbf{x}$ and $\mathbf{z}$ be the input vector and output vector of the mapping C respectively, and $\mathbf{y}$ the output vector of the mapping $C \circ D$. The mapping C is admissible if and only if there is another mapping D such that

$$(C \circ D)(\mathbf{x}) = (P(\mathcal{C}_1|\mathbf{x}), \cdots, P(\mathcal{C}_k|\mathbf{x}))$$

with probability one, where the $P(\mathcal{C}_i|\mathbf{x})$ are the posterior probabilities of the classes.

Note that this last theorem combines a transform C and a classifier D that computes the posterior probabilities of the classes $P(\mathcal{C}_i|\mathbf{x})$ from the transformed feature vectors $\mathbf{z} = C(\mathbf{x})$. A trivial solution is of course $C(\mathbf{x}) = (P(\mathcal{C}_1|\mathbf{x}), \cdots, P(\mathcal{C}_k|\mathbf{x}))$ and $D = I$, which is consistent with our previous statement about the optimality of the posterior probabilities of the classes. Note also that the mappings C and D are not unique, and further constraints on the reduced feature vector must be imposed to find a unique solution [KAO94]. Whether a dimensionality reduction mapping C $(m < n)$ exists that is admissible in the sense of Definition 4.1 is not stated in the previous theorem. Nevertheless, we can impose this constraint on the mapping C and try to estimate the parameters of both mappings so as to approximate the posterior probabilities of the classes. Although this does not in general lead to a mapping C admissible in the sense of Definition 4.1, the estimated mapping C

transforms the high-dimensional input space into a low-dimensional space where the Bayes error is not much higher than in the original high-dimensional space $\mathbf{x}$.

The previous theorem, however, just imposes a condition that mapping C must satisfy in order to be admissible, but does not explain how to find it. Consequently, a second problem of feature reduction for classification is to find a practical method to compute the joint mapping $F = D \circ C$. In many of the practical situations we normally have a number of input-output vector data pairs, e.g. a segmentation into states $\mathcal{S} = \{(\mathbf{x}^l, \mathbf{q}^l)\}$, and we want to find the underlying mappings C and D using the previous data pairs. This kind of problem is formally equivalent to the problem of fitting a curve to a series of observations, which can be solved using statistical regression theory. A well-known result of statistical regression states that the absolute minimum of the approximation in the minimum square error (MSE) sense is reached when the input-output mapping is the conditional expectation of the output targets $\mathbf{t}$ given the input vectors $\mathbf{x}$. More formally we can write:

Theorem 4.2 (Optimum Regression Curve [Bis96])
Given the conditional distribution of the targets and the input vectors $p(\mathbf{t}|\mathbf{x})$, the optimum regression curve $F_{opt} : \mathbb{R}^n \to \mathbb{R}^k$ in the minimum square error (MSE) sense is given by:

$$F_{opt}(\mathbf{x}) = \mathrm{E}\{\mathbf{t}|\mathbf{x}\}$$

that is, when the optimum mapping F_{opt} is the expectation of the output patterns conditioned to the input patterns.

Using this last result and a particular kind of coding for the target vectors $\mathbf{t}$ associated with the classes $\mathcal{C}$, we can estimate the class posteriors at the output of the mapping. If we let the target vectors $\mathbf{t}$ adopt the 1-of-c coding scheme, i.e. the k-th component is 1 and the others are zero, then the conditional density $p(t|x)$ can be written in the following form:

$$p(\mathbf{t}|\mathbf{x}) = \sum_{l=1}^{N} \delta(\mathbf{t} - \mathbf{e}_l) P(\mathcal{C}_l|\mathbf{x}) \tag{4.4}$$

where $\mathbf{e}_l$ is the unit vector in the l-dimension. Substituting into Eq. 4.2 this results in:

$$F(\mathbf{x}) = (P(\mathcal{C}_1|\mathbf{x}) \dots P(\mathcal{C}_N|\mathbf{x}))' \tag{4.5}$$

which is the desired output of the composed mapping $(C \circ D)$ in Theorem 4.1.

Although this equation shows that it is possible to estimate class posteriors using the Theorem. 4.2, it is of little practical utility since the conditional density $p(\mathbf{t}|\mathbf{x})$ is normally unknown. As already mentioned above, it is often the case in practice that only a segmentation $\mathcal{S}$ of the training data is available to estimate the parameters of the mapping. Moreover, since the number of available training samples in $\mathcal{S}$ is finite, we must also substitute the expectation operator E. by the average over all training samples. However, we know that Theorem. 4.2 is the solution to the approximation of the targets in the MSE sense. Consequently, an equivalent formulation of estimating the optimum mapping C in Theorem 4.1 is stated in the following definition:

Definition 4.1 (Minimum Square Error Optimization)
Let F be a mapping that must approximate the posterior probabilities $P(\mathcal{C}|\mathbf{x})$ at its output as in Theorem 4.1. If the targets t_k associated with the classes $\mathcal{C}_k$ are coded in a 1-of-c fashion, the optimum mapping F_{opt} can be found as:

$$F_{opt} = \arg\min_{\Theta} \sum_{\forall l} \| \mathbf{t}^{\mathbf{l}} - F(\mathbf{x}^{\mathbf{l}}, \Theta) \|^2$$

that is, F_{opt} is the mapping that minimizes the global MSE over all training data points between the 1-of-c coded targets and its outputs.

To be able to solve the previous optimization in practice, we usually make the following assumptions:

1. Since the number of possible mapping types is theoretically infinite, we normally restrict our solution to a family of mappings that can be represented using a certain set of parameters Θ. A typical example of this restriction is LDA, since we assume that the optimum mapping F_{opt} is linear, i.e. we assume that the relation between input vector and posteriors is linear. However, this is only true when the classes have normal distributions with equal covariance matrices [Fuk90]. If these conditions are not met, then the mapping F found using LDA is not optimum in the Bayes sense.

2. The previous minimization problem, however, has rarely a closed form solution (with the exception of F being linear as in LDA), so that we must use some kind of iterative algorithm to approximate the solution. These multidimensional minimization problems are usually very complex and have a large number of local minima, and as a consequence it is difficult to find an optimal solution using an iterative algorithm.

Using the considerations above one can devise the following two strategies to find a feature reduction mapping C:

- **Class separability criteria**. Define a class separability criterion in the output vector space of C which is somehow related to the criterion in Definition 4.1, and find the optimum C according to this criterion. This is the strategy used in LDA or HDA to find a linear feature reduction mapping. The problem is the usual strong assumptions made on the form of the mapping function. A second problem is that the separability criterion is usually not directly related to the criterion in Definition 4.1, and as a consequence the solution is sub-optimal in the Bayes sense.

- **Direct approximation of the class posteriors**. Estimate C and D simultaneously in order to directly approximate the posterior probabilities of the classes at the output of $(C \circ D)$ using the criterion in Definition 4.1. This is the strategy used in the tandem and NLDA approaches, which use multi-layer perceptrons (MLPs) and error back-propagation (EBP) to find a non-linear feature reduction mapping C. The problem with these algorithms is the high complexity of the cost functions in Definition 4.1, specially when the number of classes is large. These cost functions have many local minima, and it is therefore difficult to find the optimum solution using an iterative algorithm such as error back-propagation [RHW86, Bis96].

We see, therefore, that in the selection of the strategy to compute the mapping, we have a basic compromise between the assumption made on the mapping and the complexity of the cost function.

4.4 Target Classes Selection for Feature Reduction

In the previous point we have already mentioned that the selection of the classes is the first issue in feature reduction for classification. For some problems the selection of the classes is obvious (as in medical applications where the decision is between normal and ill), but unfortunately this is not so in ASR. A usual choice for the targets are the HMM states in our acoustic model, since these are the classes whose probability is to be computed in the probability computation step of an ASR system (cf. Sec. 2.5.2). However, the HMM states are certainly not the only possible choice. In a recent paper [SKKW03] the authors selected the Gaussian mixtures in the states as the classes for LDA, and obtained a significant improvement over the state-based LDA.

The choices for the classes we have investigated in this thesis are:

- **states** of the HMMs,
- **phonemes** occurring in the given ASR task,
- **clusters** of HMM states. To cluster the states we use the clustering algorithm proposed in [Lee90] and described in Appendix D. This clustering is based on a loss of information measure when merging two states.

4.5 Class Separability Criteria

A class separability criterion J is a function of some parameters of the pdf of the classes—normally just the means and covariances — so that it can be written as:

$$\mathrm{J} = f(\mathbf{m}_1, \ldots, \mathbf{m}_N, \mathbf{K}_1, \ldots, \mathbf{K}_N) \tag{4.6}$$

For these kind of methods the mean vector and covariance matrix of each class are computed using the input-output data pairs $\mathcal{S} = (\mathbf{x}^l, \mathbf{t}^l)$ in the training set. The problem is to find a mapping C (linear or non-linear) between a high dimensional space and a low dimensional one that maximizes the separability criterion J in the low dimensional output space. More formally we can define:

Definition 4.2 (Optimum Mapping for Class Separability Criteria)
Let J *be a class separability criterion defined in* $\mathbb{R}^m$*. An optimum feature reduction mapping* $C_{opt} : \mathbb{R}^n \to \mathbb{R}^m$ *in the sense of the criterion* J *is one that satisfies:*

$$C_{opt} = \arg\max_{C} f(\mathbf{m}_1^m, \ldots, \mathbf{m}_N^m, \mathbf{K}_1^{(m\times m)}, \ldots, \mathbf{K}_N^{(m\times m)})$$

where the $\mathbf{m}_i^m$ *and the* $\mathbf{K}_i^{(m\times m)}$ *are the means and covariances of the classes in the m-dimensional transformed space.*

This criterion has in general no closed form solution and must be solved using numerical methods such as gradient descent. Moreover, as in the case of Definition 4.1 the mapping C is assumed to have a certain form, e.g. linear, to keep the optimization mathematically tractable. If the classes in the previous criterion are assumed to be normally distributed and with equal covariance matrices, this criterion can be greatly simplified. In fact, it can be demonstrated that the optimum transform in that case is linear, and therefore this kind of feature reduction is known as linear discriminant analysis (LDA). More about this approach can be found in Appendix E.

4.6 Neural Networks for Dimensionality Reduction

In this section we discuss the second kind of strategy which can be used to compute a mapping for feature reduction. This strategy tries to simultaneously compute the mappings C and D, so as to have an estimate of the class posteriors at the output of $F = (C \circ D)$. The problem is that this computation leads to complex non-linear optimizations which cannot be solved using common iterative algorithms. During the last decade, however, artificial neural nets (ANN) have been successfully applied in hybrid MLP/HMM ASR systems to estimate the posterior probabilities of the HMM states [BM94, BW90]. Since for the problem at hand we must approximate posteriors as well, it seems logical to think that MLPs could also be successfully applied. The idea is actually to use the MLP framework to solve the optimization in Definition 4.1, and to use the MLP or a part of it as the feature reduction mapping C.

The advantages of using an MLP for feature reduction are:

- No assumption is made about the pdf of the input data in the classes.
- Non-linear mappings can be easily computed, which are more suited to the statistical distribution of the data vectors $\mathbf{x}$ in the classes $\mathcal{C}_i$.
- Posterior probabilities can be approximated using more suitable criteria than MSE.

The problem of using the MSE criterion in Definition 4.1 is that it is just optimum in the maximum likelihood sense when the class targets $\mathbf{t}$ are normally distributed [Bis96]. The class targets $\mathbf{t}$ coded using the 1-of-c coding scheme, however, are far from being normally distributed since the components take just the values 1 or 0. A more suitable density for each component would be in this case the Bernoulli discrete density. It can be shown, that by applying the maximum likelihood principle to this last density the minimum cross-entropy (MCE) criterion is obtained [Bis96]. The cross-entropy function between two vectors $\mathbf{t}$ and $\mathbf{y}$ is defined as:

$$E^n = -\sum_{\forall k} t_k \ln y_k(\mathbf{x^n}) \tag{4.7}$$

where t_k and y_k are the target variables and the corresponding actual outputs of the MLP, respectively [Bis96]. If we have a set of N data points, the total cross-entropy can be expressed as:

$$E = \sum_{\forall n} E^n = -\sum_{\forall n} \sum_{\forall k} t_k \ln y_k(\mathbf{x^n}) \tag{4.8}$$

Using the previous formulas, we can define the MCE optimization in the following way:

Definition 4.3 (Minimum Cross-Entropy Optimization)
Let Θ be the set of parameters of the mapping F, and $\mathcal{S} = \{(\mathbf{x}^l, q^l)\}$ a segmentation into states of the training data. If the states q are coded in a 1-of-c fashion, the optimum values of Θ to approximate the class-posteriors at the output of F are found by minimizing:

$$\Theta_{opt} = \arg\min_{\Theta} \sum_{\forall l} -\ln(f_c(\mathbf{x}^l, \Theta))$$

where c is the index of the active component (the index of the associated state) for the current frame $\mathbf{x}^l$, and f_c is the c-th output of the mapping F.

Note that the expression of the cross-entropy between targets $\mathbf{t}$ and outputs $\mathbf{y}$ has been simplified in the problem above, since for 1-of-c coded targets the following equality holds:

$$\sum_{i=1}^{M} t_k \ \ln(\frac{y_k}{t_k}) = \ln(y_h)$$

The MCE criterion minimizes the relative error instead of the absolute error and is therefore more suitable than the MSE criterion for target functions such as the posterior probability functions, which have regions of low values. It can be as well demonstrated that the optimum solution to the criterion in Definition 4.3 is also given by the regression curve in Theorem 4.2. Therefore, by minimizing the cross-entropy between output and target vectors we can also compute a mapping that estimates the posterior probabilities of the classes. The weights of the MLP can be trained using the error back-propagation (EBP) algorithm, which is a kind of gradient descent technique (cf. Sec. 5.5.1). More about MLPs and EBP can be found in [RHW86, Bis96].

Depending on the number of classes considered in our feature reduction problem, we propose two different MLP-based approaches each using a different kind of MLP: an MLP with one hidden layer for low number of classes, and an MLP with two hidden layers for large number of classes.

4.6.1 NN Topology for Low Number of Classes (Tandem)

If the number of classes is not very large, the output of the MLP can be directly used as a feature vector, since we know from Sec. 4.3 that the posterior probabilities of the classes are the optimum features. On the other hand, it has been shown that a one hidden layer MLP can approximate arbitrarily well any continuous functional form, provided the number of hidden units M is sufficiently large [HSW89]. Consequently, it can be interesting to train an MLP as depicted in Fig. 4.5 to estimate the posteriors of the classes. During recognition the units at the output layer are removed, and the activation of the output layer are used as a reduced feature vector. This last step is needed in order to have features more suited to the Gaussian mixture models in the states of the HMMs. Typically this kind of topology is used when the classes are the phonemes in the ASR task, as in the tandem original approach [HES00] or clusters of states. However, this topology has also been used in an approach that first estimates the posterior probabilities of the states using this MLP, and next applies a PCA to the high-dimensional output vector to further reduce the dimensionality [RHWR96].

4.6.2 NN Topology for Large Number of Classes (NLDA)

When the number of classes in is large, we cannot directly use the class posteriors as the reduced feature vector for the reasons already given in Sec. 4.3. An alternative is to split, as in Theorem 4.1, the global mapping of the neural net F into two partial mappings C and D ($F = C \circ D$) which are trained simultaneously using the input-output data pairs in the training set $\mathcal{S} = \{(\mathbf{x}^l, \mathbf{t}^l)\}$. To perform the simultaneous training of both mappings the MLP shown in Fig 4.6 can be used. In this topology, the feature reduction mapping C is implemented by the first two layers of weights (outputs of C are the activation of the second hidden layer), whereas the mapping D is implemented by the last layer of weights.

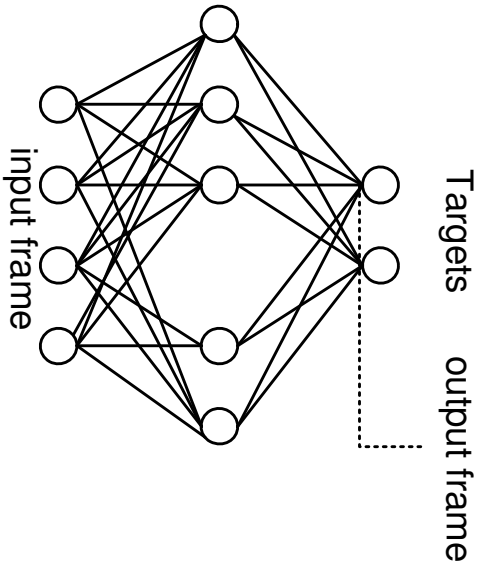


Fig. 4.5: A one hidden layer MLP used for feature reduction in the tandem system. The reduced feature vector is obtained from the pre-nonlinearity outputs.

As can be seen in the figure, the output $\mathbf{z}$ of the mapping C has less components than the input vector $\mathbf{x}$, and since we are able to correctly classify the classes using just the vector $\mathbf{z}$ then we can say that the vector $\mathbf{z}$ has similar discrimination capabilities as the original vector $\mathbf{x}$. During recognition the mapping D and the units in the second hidden layer are discarded, and the activations of the second hidden layer are then passed to the acoustic modeling.

In this case the vectors handed to the acoustic modeling do not have a straightforward interpretation. In [Bis96] it has been shown that each of the hidden layer outputs can be interpreted as the probabilities that a certain characteristic is present in the current feature vector. In fact, if the classes used to compute the NLDA mapping are phonemes, it is tentative to interpret these outputs as the probabilities of the so-called binary distinctive features [JH56] of a language. These features include categories such as 'voiced', or 'nasal'. From the point of view of phonetics these are theoretically the only features needed to distinguish between two different phonemes.

This MLP with two hidden layers can be further generalized to an MLP with 3 hidden layers, where the activations of the layer in the middle are used as feature reduced vectors. The advantage of using such a topology is that the mapping between reduced feature space and posterior probabilities is not necessarily linear. This imposes less restrictions on the mapping $(C \circ D)$, and accordingly we may find a mapping more suited to the optimum mapping F_{opt} in Theorem 4.1.

4.7 Summary

In this chapter we have discussed the principles of feature reduction for classification. We have seen that the optimum feature reduction mapping in the Bayes sense is the optimum Bayes classifier or a part of it. We have also seen that this optimum mapping can be

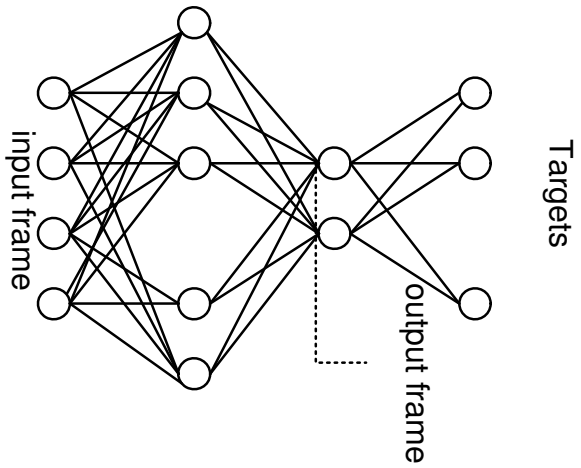
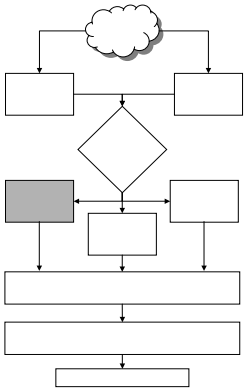


Fig. 4.6: A two hidden layer NN used for feature reduction using non-linear discriminative analysis (NLDA). The reduced feature vector is obtained from the pre-nonlinearity outputs of the 2nd hidden layer.

found by approximating, in the MSE or MCE sense, the 1-of-c coded target classes. A first fundamental question is to decide which are the ideal classes for the problem in question, namely ASR in an HMM background, and a second question is the way the coded targets should be approximated. This can be performed by defining a separability criterion in the output space of the transform, e.g. LDA, or by directly approximating the posterior probabilities of the classes as in the tandem or NLDA approaches. In Sec. 7.3.3 we show the results obtained for different choices of the classes and estimation approaches.



5. Radial Basis Functions for Hybrid ANN/HMM

Motivated by the excellent results of the hybrid ANN/HMM approach on the AURORA 2000 evaluation we present in this chapter a variant of the hybrid ANN/ HMM approach based on radial basis functions (RBF) instead of Multi-layer Perceptrons (MLP). The rationale behind this is to find an approach which still performs reasonably well but needs less training time, i.e. with less NN weights to estimate using Back-Propagation. Another good reason is that our ASR system would need almost no modification to implement the hybrid RBF/HMM approach. In fact, we will see in this chapter that our SCHMM ASR system already uses an RBF to compute the state likelihoods. The difference with respect to our intended hybrid RBF/HMM approach is the way the weights of the RBF have been computed: in the SCHMM approach the weights are computed using the *maximum likelihood* (ML) principle whereas in the hybrid RBF/HMM approach weights are computed using the *maximum-a-posteriori* (MAP) principle, i.e. using an inherently *discriminative* principle. Furthermore, if certain constraints are met, the weights trained using the MAP criterion can be interpreted as symbol emission probabilities, and accordingly they can directly be used in semi-continuous HMM systems.

5.1 Introduction

In Chapter 2 we have used the maximum likelihood criterion to find estimation formulas for the HMM parameters. However, this criterion is not well adapted, at least theoretically, to the definition of the fundamental optimum word sequence (cf. Eq. 2.1 in Chapter 2). The right criterion to find the optimum set of parameters Θ_{opt} for that problem is the *maximum a posteriori* (MAP) criterion, which maximizes the posterior probability $P(X|W,\Theta_{opt})$. Using this criterion the optimization can be expressed in the following fashion. Let $X = (\mathbf{x}^1,\ldots,\mathbf{x}^T)$ and $W = (w^1,\ldots,w^{L_i})$ be an observation sequence in the training set and the associated word sequence, respectively. The maximum a posteriori estimate of the set of parameters Θ is given by:

$$\Theta_{opt} = \arg\max_{\Theta} P(W|X,\Theta) \tag{5.1}$$

In fact, it is possible to relate the previous criterion to the maximum likelihood criterion of Eq. 2.31 in Chapter 2. Using the Bayes rule the posterior of the right sequence of models $P(W|X,\Theta)$ can be written as:

$$P(W|X,\Theta) = \frac{p(X|W,\Theta)P(W,\Theta)}{p(X,\Theta)} \tag{5.2}$$

To obtain the ML criterion, the denominator term is assumed constant during training, and it is accordingly discarded. This is clearly not true because the term $p(X,\Theta)$ changes as the parameter values in Θ change.

To gain further insight into the differences between both criteria, we can express the denominator term as:

$$p(X,\Theta) = \sum_j p(X|W_j,\Theta)P(W_j,\Theta) \tag{5.3}$$

where the sum is over all possible model sequences. If we additionally assume that the acoustic model parameters and language model parameters can be trained separately, and since the $p(X|W_j,\Theta)$ just depend on the parameters of the sequence W_j, we can write:

$$P(W_i|X,\Theta) = \frac{p(X|W_i,\Theta_i)P(W_i)}{\sum_j p(X|W_j,\Theta_j)P(W_j)} \tag{5.4}$$

Now substituting the previous result into Eq. 5.1 we obtain:

$$\arg\max_{\Theta} P(X|W_i,\Theta) = \arg\max_{\Theta} \frac{p(X|W_i,\Theta_i)P(W_i)}{\sum_j p(X|W_j,\Theta_j)P(W_j)} \tag{5.5}$$

which is equivalent to the maximization of [BM94]

$$\arg\max_{\Theta} \frac{p(X|W_i,\Theta_i)P(W_i)}{\sum_{j\neq i} p(X|W_j,\Theta_j)P(W_j)} \tag{5.6}$$

where we clearly see that the MAP criterion imposes a joint optimization of the parameters of the correct sequence of models against the parameters of all the incorrect sequences of models. Thus MAP criterion imposes a concurrent optimization of the likelihood of the correct sequence of models and the likelihoods of the incorrect sequences of models. The first one is maximized whereas the other ones are minimized. This is clearly very different from the ML criterion, since this latter criterion just maximizes the likelihood of the correct sequence of models, but pays no attention to the likelihood of the incorrect model sequences.

The estimation formula in Eq. 5.6, however, has no closed form solution, and an iterative solution is only computationally feasible for isolated word recognition. For Connected Word Recognition or Continuous Speech, the formula is impractical because the amount of possible incorrect model sequences in the denominator is extremely large. A solution to this problem is to take into account in the denominator of Eq. 5.6 just the sequences that have a similar but smaller likelihood than the correct sequence. These sentences can be found by running an ASR system in N-best hypothesis modus over the training database. Another estimation criterion closely related to Eq. 5.6 is the maximum mutual information criterion [BBdSM86, Val95], which iteratively maximizes the mutual information between the sequence of words W and the observation sequence X.

On the other hand, hybrid ANN/HMM tries to circumvent the difficulties of iteratively optimizing Eq. 5.6 by decomposing the global posteriors $P(W_i|X,\Theta)$ in the criterion of Eq. 5.1 into the local HMM-state posterior probabilities. As seen in Sec. 5.4, this decomposition allows us to reduce the problem of global discrimination, i.e. sentence level, to the simpler problem of local discrimination, i.e. state level. The state posterior probabilities are then estimated using a Multi-Layer Perceptron (MLP). This is a kind of ANN which has proved to be very useful in many classification problems [Lip87]. However, one of the drawbacks of MLPs is the large amount of weights they need to approximate posterior probabilities at the output, specially if the number of HMM-states is large. The number of operations per input frame needed by conventional error back propagation (EBP) [RHW86] to estimate the weights increases linearly with the number of weights [Bis96], which may be prohibitive for large amounts of weights and training data.

On the other hand, radial basis functions (RBF) combine at their output units more complex functions than the hyperplanes of the MLPs. This combination is for RBF usually linear, but can be made non-linear as in our case. In theory, if these complex functions match the data space of the input vector $\mathbf{x}$, the same classification performance as with MLPs may be expected but with less hidden units [BM94] and therefore less parameters.

Finally note that *discrimination* is rather a concept of parameter estimation, i.e. of training. One can for instance train an NN to estimate the state posteriors and convert those posteriors to normalized likelihoods using the prior probabilities. These state likelihoods would have been thus discriminatively trained and of course would be very different from the state likelihoods trained using the ML principle. Consequently, the discriminative properties of the hybrid ANN/HMM approach do not stem from the use of posterior probabilities but rather from the training method employed [BM94].

As explained in the next points, the use of posterior probabilities instead of likelihoods provides a relatively simply way to train the HMM parameters in a discriminative way.

5.2 Previous Work on Hybrid RBF/HMM

An study closely related to the one presented in this chapter has been carried out by Renals *et al.* [RMB91], who have used a similar approach to compute the posterior probabilities using an RBF. However, the normal kernel functions (cf. Sec. 5.3) of the RBFs were assumed diagonal, which may partially explain why the recognition results were clearly poorer than similar experiments with MLPs.

Another loosely related approach— without RBFs— was used in [BWK99] to discriminatively train the mixture densities (or equivalently emission probabilities, cf. Sec. 2.5.2) of context-dependent HMMs with tied mixtures.

In [SL92] a hybrid RBF/HMM is developed and its performance compared to a tied-mixture recognizer. A separate set of Gaussian mixture models is used for the static, 1st derivative and 2nd derivative features. The posteriors or likelihoods of the three mixture models are merged at the state level to obtain a single probability score per state, i.e. the systems are multi-stream systems (cf. Chapter 6). For the hybrid system, the normal densities are trained using the k-means algorithm whereas the mixture weights are trained using gradient descent (cf. Sec. 5.5.1). The normal densities are diagonal with the variances of the coefficients equal to the grand variance averaged over all words and states. Results show a significant improvement of the hybrid system with respect to a

tied-mixture system with a similar number of parameters. Once again, these experiments use a very poor Gaussian mixture model for both systems, which may be the reason for the good results of the discriminative approach.

The authors in [RR95] use a hybrid RBF/HMM to recognize the phonemes in the PhonDat 'Diphon' database in German. As in the previous reference, the topology of the RBF has separate sub-nets for the static, dynamic and energy features each with 256, 128 and 16 basis functions, respectively. The basis functions are multivariate Gaussian functions without the normalization factor $1/\sqrt{2\pi\,|K|}$ and with diagonal covariance matrices. The outputs of the basis functions are normalized to sum up to 1 over each sub-net. To obtain the state posterior probabilities, the three sub-nets are combined using a sigmoidal non-linearity, which forces the outputs of the RBF to be between 1 and 0. The targets of the RBF are the states in the phonetic acoustic model which are a total of 169. The particularities of this topology are that the sub-nets are combined using a sigmoidal non-linearity, and since this sigmoidal already guarantees that the weights are between 1 and 0, the weights of the RBF are not constrained to be between 1 and 0. The training procedure is carried out in two stages. A first stage to estimate means, variances and weights of the RBF is performed using the usual algorithms of hybrid ANN/HMM (cf. Sec. 5.5), but using the minimum square error criterion to approximate the state posteriors. A second stage based on the minimum classification error criterion is used to improve the discrimination of the weight estimates, especially between highly confusable phonemes. The phoneme recognition results show that this second training stage significantly improves the performance, and that the addition of more context frames in the input layer also improves the performance.

In our experiments with the hybrid RBF/HMM approach in Sec. 7.3.4 we will use a more sophisticated Gaussian mixture model with non-diagonal covariance matrices to see if we can also make improvements with a discriminative approach.

5.3 Radial Basis Functions

Radial basis functions (RBF) were originally developed to exactly interpolate functions [Pow85, Bis96] at a set of given data points. However, this is not our intended use of RBFs, because we do not want an estimator that passes through all data points, but rather one that *generalizes* the estimate to points not present in the training set (cf. Sec. 5.5.2). To this purpose we use the following RBF network:

$$f_k(\mathbf{x}) = \sum_j w_{kj}\,\phi_j(\mathbf{x}) \tag{5.7}$$

The functions ϕ_j are termed the *kernels* or basis functions of the RBF. The most usual kernels found in the applications are the normal kernels:

$$\phi_j(\mathbf{x}) = c\,\exp(-\frac{(\mathbf{x}-\mathbf{m}_j)'(\mathbf{x}-\mathbf{m}_j)}{2\sigma^2}) \tag{5.8}$$

which can be generalized to:

$$\phi_j(\mathbf{x}) = c\,\exp(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_j)'\mathbf{K}_j^{-1}(\mathbf{x}-\mathbf{m}_j)) \tag{5.9}$$

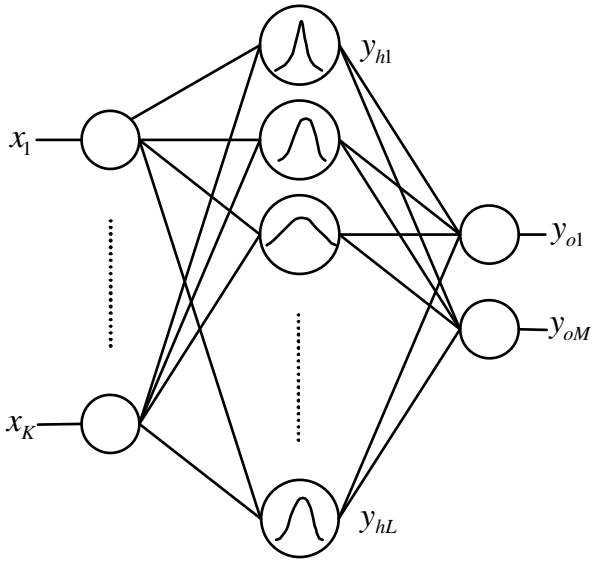


Fig. 5.1: A radial basis function network.

where the c is a normalization constant which may depend on the input vector $\mathbf{x}$. An RBF is depicted in Fig. 5.1 where we can see that an RBF consists of one hidden layer of non-linear units, each implementing one of the kernels ϕ_j, and an output layer of linear units. In Sec. 5.4.2 we will see that it is possible to use non-linear units in the output layer as well.

As explained in Sec. 5.5, the training of RBFs is usually carried out in two stages:

- A first stage to train the means $\mathbf{m}_j$ and covariances $\mathbf{K}_j$ of the kernels ϕ_j using supervised or unsupervised learning techniques (cf. Sec. 2.6.1).

- In a second stage the parameters of the kernels are kept fixed and the weights in the 2nd layer w_{kj} are then estimated using some kind of supervised learning technique, e.g. gradient descent.

If the minimum square error (MSE) criterion is used to approximate the targets and the output units are linear, then the second stage can be solved using algebraic methods. If another criterion is used, e.g. minimum cross-entropy (MCE), or the output units are non-linear, then an iterative method such as gradient descent must be used to solve the problem. The usual approach, and the one used in our study, is to use gradient descent to find the iterative equations.

Links to SCHMMs and Hybrid ANN/HMMs

As already seen in Chapter 2, the state likelihoods in an SCHMM system are computed using:

$$p(\mathbf{x}|q_k) = \sum_j b_j(k)\,\mathcal{N}(\mathbf{x};\mathbf{m}_j,\mathbf{K}_j) \tag{5.10}$$

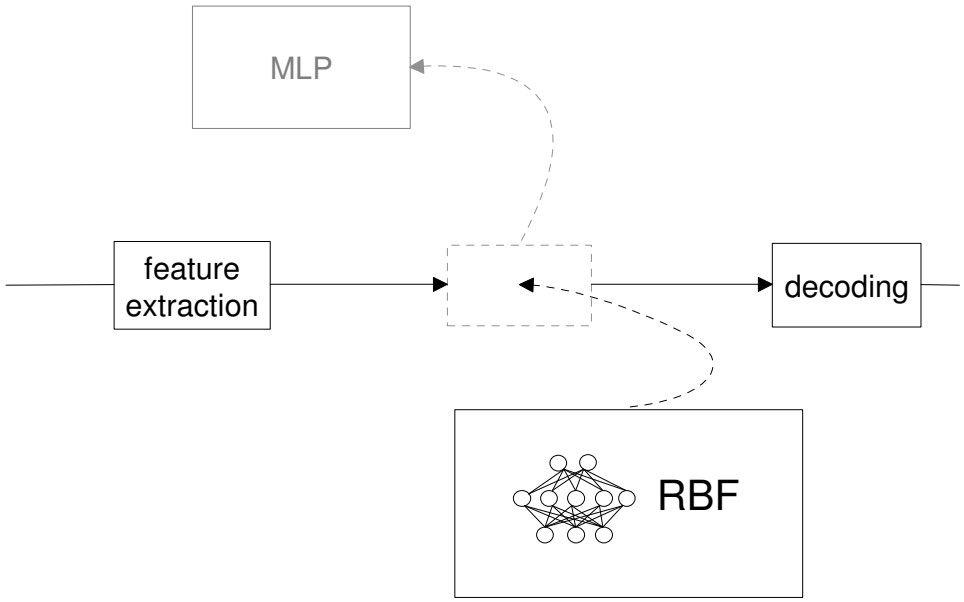


Fig. 5.2: Substitution of the usual MLP by an RBF network to compute the state posterior probabilities in hybrid ANN/HMM.

where $b_j(k)$ is the probability of observing symbol j in state $q = k$ and $\mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j)$ is a Gaussian modeling the distribution of the data $\mathbf{x}$ for the symbol j. If we now take $w_{kj} = b_j(k)$ and $\phi_j(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j)$, then we can readily see that the state likelihoods of an SCHMM are actually computed using an RBF, whose weights w_{kj} have been computed using the Baum-Welch algorithm, and whose kernel parameters have been estimated using supervised or unsupervised learning (cf. Sec. 2.6.1). A problem with the Baum-Welch algorithm is that it is based on the maximum likelihood criterion, and is therefore inherently non-discriminative.

As seen in the following sections, a solution to this problem is to use the framework of hybrid ANN/HMM to train the weights of the RBF discriminatively. The idea is simply to substitute the usual MLP by an RBF to approximate the state posteriors $P(q_i|\mathbf{x})$ as illustrated in Fig. 5.2. The main differences between RBFs and MLPs are [Bis96]:

- MLPs are said to be a form of *distributed representation* in the space of activation values for the hidden units since, for a given input vector, many hidden units typically contribute to the determination of the output value. The interference and cross-coupling between hidden units results in a network training process which is highly non-linear with problems of local minima or nearly flat regions in the error function, which can lead to very slow convergence rates. By contrast, the kernels of the RBF are a *local representation*, and accordingly for a given input vector just a few hidden units are typically active, which may reduce the problem of slow convergence.

- if the form of the kernels is well-matched to the distribution of the data, and since only a few hidden units are typically active for a given input vector, the number of hidden units of an RBF is typically smaller than that of an MLP.

- additionally, all the parameters of an MLP are usually determined at the same time

using some kind of supervised learning technique, whereas the parameters of an RBF are estimated in two steps.

In the following sections we explain how the hybrid ANN/HMM approach works, and how an RBF can be trained and used for this approach.

5.4 Hybrid ANN/HMM Acoustic Modeling Approach

In Sec. 2.4 of Chapter 2 we have seen that the statistical pattern recognition approach to ASR can be mathematically expressed with the following equation:

$$W_{opt} = \arg\max_{W_i} P(W_i|X) \tag{5.11}$$

where $W_i = (w^1, \ldots, w^S)$ is one of the possible HMM model sequences and $X = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$ is the observed sequence of features. We have mentioned in the introduction to this chapter that the suitable training criterion for Eq. 5.11 is the maximum a posteriori (MAP) criterion, i.e.:

$$\Theta_{opt} = \arg\max_{\Theta} P(W_i|X, \Theta) \tag{5.12}$$

where it is implicitly assumed that the following constraint holds:

$$\sum_i P(W_i|X, \Theta) = 1 \tag{5.13}$$

In fact, this *global constraint* is what makes any approach trained using the above criterion discriminative. This is clearly so, because when a certain word sequence W_i has a probability near to 1, then the others must be zero or near to zero to meet the constraint. A problem is that the number of possible word sequences is very large, and it is consequently difficult to use the previous constraint in practice. The difficulty is therefore to find an expression for $P(W|X, \Theta)$ which meets the previous constraint, but without having to consider all the possible word sequences.

Proceeding in a similar way as in Chapter 2 with the likelihood $p(X|W)$, we can decompose $P(W|X)$ as:

$$\begin{aligned} P(W|X) &= \sum_{\forall Q_i} P(W, Q_i|X) \\ &= \sum_{\forall Q_i} P(W|X, Q_i)P(Q_i|X) \end{aligned} \tag{5.14}$$

where $Q_i^T = (q^1, \ldots, q^T)$ is one of the possible HMM-state sequences of length T given the sequence of HMMs W. Since given the sequence of states $q^1, \ldots, q^T$ the choice of the word sequence W is independent of the observation sequence X, and since given the sequence of states $q^1, \ldots, q^T$ it is possible to recover the sequence of models W that generated it, the first term in Eq. 5.14 can be neglected.

The second term in the previous equation can be factored into a product of local posteriors:

$$\begin{aligned} P(Q_i^T|X) &= P(q^1|X)P(q^2|X, q^1)\ldots P(q^T|X, q^{T-1}, \ldots, q^1) \\ &= \prod_l P(q^l|X, Q_i^{l-1}) \end{aligned} \tag{5.15}$$

Using the usual 1st order Markov assumption (current state depends only on the preceding state), and setting the dependency of the context to just the current frame, we can approximate:

$$P(q^l|X, Q_i^{l-1}) \approx P(q^l|\mathbf{x}^l, q^{l-1}) \tag{5.16}$$

The posterior probability $P(W|X)$ is thus:

$$P(W|X) = \sum_{\forall Q_i^T} \prod_l P(q^l|\mathbf{x}^l, q^{l-1}) \tag{5.17}$$

If we now set the following local constraint on the posteriors:

$$\sum_{\forall k} P(q^l = k|\mathbf{x}^l, q^{l-1}) = 1 \tag{5.18}$$

then it can be demonstrated [BM94] that the global constraint in Eq. 5.13 is met. Therefore under certain assumptions *local discrimination*, i.e. constraint on the state posteriors, ensures global discrimination. The advantage of using local discrimination is that the number of HMM states is usually much smaller than the number of possible sequences, which greatly simplifies the constraint.

But before we deal with the training of a hybrid ANN/HMM system using the constraint in Eq. 5.18, let us explain how the optimum word sequence in Eq. 2.1 can be found in the case of a hybrid ANN/HMM system.

5.4.1 Decoding Algorithm

An hybrid ANN/HMM system directly uses the Bayes classifier in Eq. 5.11 to find the optimum word sequence W_{opt}. The main difference from the ASR systems described in Chapter 2 is that posterior probabilities, e.g. $P(W|X)$ are computed instead of likelihoods $p(X|W)$. However, we will see in this point that the Viterbi algorithm can be used in this case as well.

Assuming that the problem is to recognize isolated words, and that the usual Viterbi approximation is also valid for the posteriors:

$$P(w_i|X) = \sum_{\forall Q_i(w_i)} P(Q_i, w_i|X) \simeq \max_{\forall Q_i(w_i)} P(Q_i, w_i|X) \tag{5.19}$$

we can write Eq. 5.11 in the following form:

$$w_{opt} = \arg\max_{w_i} P(w_i|X) = \arg\max_{w_i} \max_{Q_i^T(w_i)} P(Q_i^T, w_i|X) \tag{5.20}$$

Note the similarity with the criterion of Eq. 2.12 in Chapter 2. Furthermore, if analogously to the derivation of the Viterbi algorithm in Chapter 2 we define:

$$\delta^{l+1}(k) = \max_{q^1,\ldots,q^{l+1}} P(q^1, \ldots, q^{l+1}, w_i|X) \tag{5.21}$$

then the Viterbi algorithm for hybrid ANN/HMM systems is:

$$\begin{aligned}
&\text{Initialization} && \delta^0(k) = P(q^1 = k) && (5.22)\\
&\text{Recursion} && \delta^{l+1}(k) = \max_j [\delta^l(j)\ P(q^{l+1} = k|\mathbf{x}^{l+1}, q^l = j)]\\
&\text{Termination} && P(w_i|X) \simeq \max_k \delta^T(k)\\
&\text{Decision} && w_{opt} = \arg\max_{w_i} P(w_i|X) && (5.23)
\end{aligned}$$

Note that the recursion term in the previous algorithm contains the term $P(q^{l+1} = k|\mathbf{x}^{l+1}, q^l = j)$, which depends on the previous state. To be able to use feed-forward neural networks, e.g. MLPs and RBFs, the dependency on the previous state is usually dropped,that is:

$$P(q^{l+1} = k|\mathbf{x}^{l+1}, q^l = j) \simeq P(q^{l+1} = k|\mathbf{x}^{l+1}) \tag{5.24}$$

Usually the context of the current frame x^l (a window of 5 to 9 frames) is added to the input feature vector of the NN to compensate for the last approximation. Note as well that no transition probabilities are used in the previous algorithm. The dependency of the previous state and the dependency of the current frame is jointly expressed in the term $P(q^{l+1} = k|\mathbf{x}^{l+1}, q^l = j)$.

The optimum word sequence can be computed using an algorithm analogous to the one-stage algorithm in Sec. 2.5.1, but using state posterior probabilities instead of state likelihoods as in the derivation of the previous algorithm.

5.4.2 State Posteriors Computation Using RBFs

The next question we deal with is which topology should our RBF have to effectively compute the posterior probabilities $P(q_l|\mathbf{x}_t)$ in Eq. 5.24. The use of an ANN to compute the state posteriors is depicted in Fig. 5.3, where we see that each of the outputs is the posterior probability of a certain state. Typically a single RBF computes the posteriors of *all* the states, to ensure the local discrimination discussed in Sec. 5.4.

The posterior class probabilities can be expressed as a function of the class densities using the Bayes rule:

$$P(q_k|\mathbf{x}) = \frac{P(q_k)p(\mathbf{x}|q_k)}{\sum_{\forall i} P(q_i)p(\mathbf{x}|q_i)} \tag{5.25}$$

In Sec. 5.3 we have shown that RBFs can be used to approximate the state densities $p(\mathbf{x}|q = k)$ as for instance in SCHMM systems:

$$p(\mathbf{x}|q_k) = \sum_{\forall j} b_j(k)\mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j), \tag{5.26}$$

where $b_j(k) = P(s_j|q_k)$ is the probability of observing symbol s_j in state q_k, and $\mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j)$ is a normal density associated with symbol s_j. In hybrid ANN/HMM systems, however, we want to approximate the posterior probabilities instead of the density functions.

If we substitute the previous equation in Eq. 5.25 we obtain:

$$P(q_k|\mathbf{x}) = \frac{\sum_{\forall j} P(q_k)b_j(k)\mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j)}{\sum_{\forall i}\sum_{\forall j} P(q_i)b_j(i)\mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j)} \tag{5.27}$$

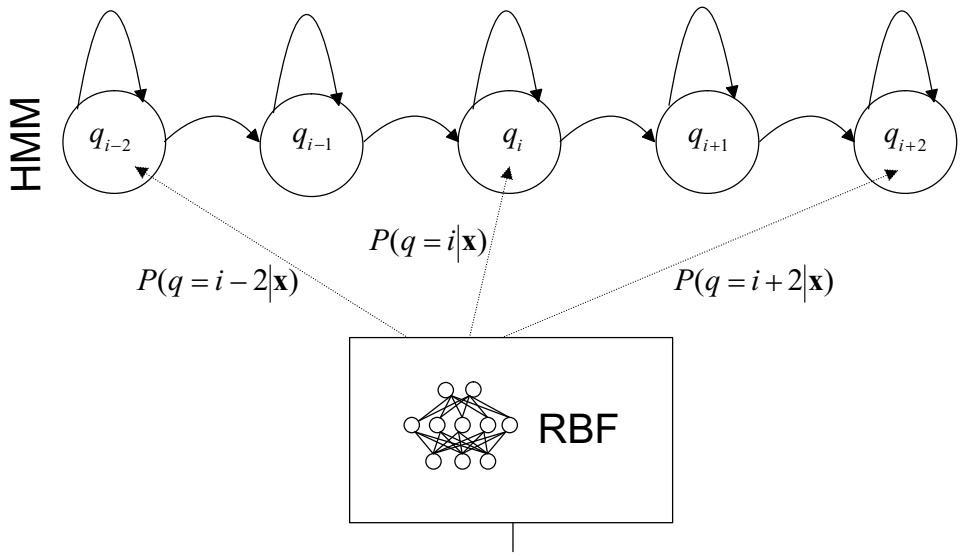


Fig. 5.3: An RBF computes the state posterior probabilities in the hybrid RBF/HMM approach.

If we now take

$$\begin{aligned} w_{kj} &= b_j(k)P(q_k) \\ \phi_j(\mathbf{x}) &= \mathcal{N}(\mathbf{x}; \mathbf{m}_j, \mathbf{K}_j) \end{aligned} \tag{5.28}$$

then we can write the last equation as:

$$P(q_k|\mathbf{x}) = \frac{\sum_{\forall j} w_{kj}\phi_j(\mathbf{x})}{\sum_{\forall i}\sum_{\forall j} w_{ij}\phi_j(\mathbf{x})} \tag{5.29}$$

Furthermore, if we define the Bayes non-linearity to be:

$$f(a_{ok}) = \frac{a_{ok}}{\sum_{\forall i} a_{oi}} \quad \text{with} \quad a_{oi} = \sum_{\forall j} w_{ij}\phi_j(\mathbf{x}) \tag{5.30}$$

(the a_{oi} are the activations at the output layer), then we obtain the following final expression for the posterior probabilities:

$$P(q = k|\mathbf{x}) = f\left(\sum_{\forall j} w_{kj}\phi_j(\mathbf{x})\right) \tag{5.31}$$

This expression is similar to the original expression of an RBF in Eq. 5.7, but with a non-linearity in the output units instead. Actually, this non-linearity is a way to ensure that state posterior probabilities meet the local constraint in Eq. 5.18. This is easily understood if we note that the outputs of the Bayes non-linearity satisfy:

$$\sum_{\forall i} f(a_{oi}) = 1 \tag{5.32}$$

5.5 Estimating Hybrid RBF/HMM Model Parameters

A training criterion analogous to the Viterbi training in Eq. 2.32 can be used to train the parameters of a hybrid RBF/HMM system, assuming that the maximum approximation in Eq. 5.19 is true. Let X, W and $(q^1 \dots q^T)$ be as in the previous equations. The optimum values for the parameters of the hybrid RBF/HMM system are found using the following criterion:

$$\tilde{\Theta}_{opt} = \arg \max_{\Theta} \max_{q^1 \dots q^T} P(q^1, \dots, q^T, W|X) \tag{5.33}$$

Note, however, that in contrast to Eq. 2.32 we intend to find an approximation $\tilde{\Theta}_{opt}$ to the optimum MAP parameters, i.e. in the sense of Eq. 5.1, instead of an approximation to the optimum ML parameters.

It is also possible to devise an algorithm, analogous to the segmental k-means or Viterbi training in Sec. 2.6.1, to train the parameters of an hybrid RBF/HMM system in the sense of the criterion of the equation above.

1. Generate an initial segmentation of the training data into HMM-states, which assigns to each training vector $\mathbf{x}^l$ an HMM state q^l. We can start for example with a flat segmentation, which assigns the same number of frames to each HMM-state present in an utterance. A better solution, if the HMM states model the phonemes, is to use the average phoneme durations [BM94]. Alternatively, if a set of HMMs, with the same topology as the ones to train, is available, we can use them to generate a segmentation by performing a forced alignment.

2. Estimate the RBF parameters (weights $\{w_{kj}\}$, means $\{\mathbf{m}_j\}$ and covariances $\{\mathbf{K}_j\}$) as explained in Sec. 5.5.1 using the last segmentation.

3. Use the new hybrid RBF/HMM system to perform another forced alignment on the training data to generate a new segmentation into states. During the forced alignment compute some kind of score, usually an average of $\max_{q^1 \dots q^T} P(q^1, \dots, q^T, W|X)$, to be able to make a decision in the next step.

4. If the score computed in the previous step is better than that of the last iteration, then go to step 2 and repeat the process. Otherwise stop the iteration.

As in the case of SCHMM systems we have employed this algorithm with some particularities. First we have used a previously trained set of HMMs to compute the initial segmentation into states. The second particularity is that we have just performed one iteration of the algorithm, so that there was no need for steps 3 and 4.

5.5.1 Training of the RBF Parameters

In this section we discuss the problem in the second step of the previous algorithm, namely the estimation of the RBF parameters. Let $F : \mathbb{R}^N \rightarrow \mathbb{R}^M$ be a mapping implemented with an RBF with components $F(\mathbf{x}, \Theta) = (f_1(\mathbf{x}, \Theta), \dots, f_M(\mathbf{x}, \Theta))$. We want to estimate the parameters Θ of the RBF, to approximate the posterior probabilities of the states $P(q_i|\mathbf{x})$ at the output of F.

Some authors [RMB91] have attempted to train all the parameters Θ of the RBF (means, covariances and weights) at the same time, using the gradient descent technique.

However, the approach is prone to finding local minima, and it is at least as computationally intensive as MLP training. For these reasons, and as already mentioned in Sec. 5.3, the training of the RBF is usually performed, as the training of the SCHMM parameters, in two sequential steps:

1. training of means and covariances (normal kernel parameters), which is equivalent to the code-book training in SCHMM.

2. training of weights, which is equivalent to the estimation of the symbol emission probabilities in SCHMM.

Normal Kernel Parameter Training

Since the normal kernels can be interpreted as the normal distributions of the symbols in the code-book of an SCHMM system, we can use the training procedure described in Sec. 2.6.1 to estimate the parameters of the Gaussian kernels. As already explained, this training procedure consists of a first step where a segmentation into states $\mathcal{S}$ is generated using a set of available HMMs with the same topology as the ones to train. Finally this segmentation is used to compute the ML-estimate of the means and covariances of the kernels in the RBF. At the end of this process we obtain the same code-book as used in our SCHMM system. The outputs of the hidden layer are also obtained as in the VQ-step of our SCHMM (cf. Sec. 2.5.2 and Sec. 7.3.1), i.e. they are obtained by first computing the values of the normal densities in the code-book for the input vector, and then dividing each of those values by the sum of all the values of the densities. This ensures that the outputs of the hidden layer sum up to one and that the variance of each output component is less than one.

Emission Probabilities/RBF Weights Training

To find an estimation formula for the RBF weights we proceed in a similar way as for SCHMMs and assume that the hidden layer outputs $\mathbf{y}_h$ are given by the functions found in the previous step, so that the means $\mathbf{m}_j$ and covariances $\mathbf{K}_j$ remain fixed during this step. The outputs of the hidden layer $\mathbf{y}_h$ are then used in the estimation algorithm of the emission probabilities or RBF weights. As already shown in Sec. 4.6, the approximation of the class-posteriors at the output of the mapping F can be solved by minimizing the cross entropy between the outputs of F and the 1-of-c coded targets of the mapping [1], i.e. the HMM states. In contrast to the problem in Sec. 4.6, however, we just have one layer of weights to estimate and consequently we do not need error back propagation.

Instead, and since the output units of the mapping F are non-linear, a simpler gradient descent technique than EBP can be used to numerically solve the problem. The idea behind gradient descent is to update the weights w_{ij} in the direction of the steepest descent.

$$w_{ij}^{l+1} = w_{ij}^{l} + \Delta w_{ij}^{l} \quad , \quad \forall i, j \tag{5.34}$$

[1] This means that the target outputs of the mapping F are vectors of the form $\mathbf{t} = (0, \ldots, 0, 1, 0, \ldots, 0)$ with just one component set to 1 and the rest to zero. Therefore each state q_k has a unique label vector $\mathbf{t}_k$ with a 1 in the k-th component

The criterion functions E of the minimization criterion are given in Eq. 4.7 and Eq. 4.8. Depending on the way weights are updated, there are two basic kinds of gradient descent training.

- For the *batch* training:

$$\Delta w_{ij}^l = -\alpha \frac{\partial E}{\partial w_{ij}} = -\alpha \sum_{\forall n} \frac{\partial E^n}{\partial w_{ij}}\bigg|_{w_{ij}^l, \mathbf{x}^n} \tag{5.35}$$

- For the *sequential* training:

$$\Delta w_{ij}^l = -\alpha \frac{\partial E^t}{\partial w_{ij}}\bigg|_{w_{ij}^l, \mathbf{x}^n} \tag{5.36}$$

In Eq. 5.35 weights are updated after the gradients at all training points $\mathbf{x}^t$ have been computed, i.e. after each training *epoch*, whereas in Eq. 5.36 weights are updated after each training pattern $\mathbf{x}^t$ has been presented to the algorithm. A compromise between both kinds of weight updating is the 'bunch' training, where the weights are updated after a bunch of n frames has been processed.

To estimate the weights we have explored the following two RBF topologies in our experiments in Chapter 7.

- **RBF with Bayes output units and weight constraint**. In the first topology the Bayes non-linearity is used at the output units as in Sec. 5.4.2, i.e.:

$$f(a_{ok}) = \frac{a_{ok}}{\sum_i a_{oi}} \tag{5.37}$$

To obtain a simpler constraint for the weights [2] we factor them in two terms:

$$w_{kj} = b_{kj} P(q = k) \tag{5.38}$$

and assume that the prior state probabilities $P(q = k)$ are already given. The weights b_{kj} must satisfy the constraint imposed by Eq. 5.18, i.e. they must:

$$\sum_{\forall j} b_{kj} = 1 \quad \text{and} \quad 0 \leq b_{kj} \leq 1 \tag{5.39}$$

Using these constraints the outputs of the RBF are ensured to be positive and to add up to 1.

This constraint can be imposed if the weights b_{kj} are a function of the form:

$$b_{kj} = \frac{\exp(c_{kj})}{\sum_{j'} \exp(c_{kj'})} \tag{5.40}$$

As demonstrated in [RMB91] the gradient of the error function in the first approach is:

$$\frac{\partial E_n}{\partial c_{kj}} = \frac{1}{a_{ok}} (y_{ok} - \delta_{kc}) w_{kj} (y_{hj} - a_{ok}) \tag{5.41}$$

[2] Actually the weights w_{kj} can be interpreted to be $w_{kj} = P(q = k, s = j)$ which is a joint probability and must therefore meet the constraint $\sum_k \sum_j w_{kj} = 1$. See Chapter 8

Note that since the weights b_{kj} add up to 1, it is possible to interpret them as symbol emission probabilities, and consequently it is possible to use them in our SCHMM ASR system which decodes the likelihoods of the states.

- **RBF with softmax output units and no weight constraint**. In the second method we use a softmax non-linearity at the output units:

$$f(a_k) = \frac{\exp(a_k)}{\sum_i \exp(a_i)} \tag{5.42}$$

As before we factor the weights in:

$$w_{kj} = b_{kj} P(q = k) \tag{5.43}$$

In this case we assume that the activation functions of the output units approximate:

$$a_k = \log(P(q = k) \sum_j b_{kj} y_{hj}) = \log P(q = k) + \log \sum_j b_{kj} y_{hj} \tag{5.44}$$

where the term $\log P(q = k)$ can be interpreted as the offset of the units in the hidden layer. Using this assumption there is no need to constrain the weights since the softmax ensures that the outputs of the RBF are positive and that they add up to 1.

The gradients can be demonstrated to be [RMB91]:

$$\frac{\partial E}{\partial w_{kj}} = (y_{ok} - \delta_{kc}) y_{hj} \tag{5.45}$$

5.5.2 Optimization in Practice

In this section we address two important practical questions to consider before training a neural network. The first is the initial values to give to the weights w_{kj}, since gradient descent is a numerical approach and consequently in need of initialization. Most of the current algorithms choose a set of random values for the weights, to avoid problems with the symmetries in the neural network. Weight values are also chosen to be small to avoid large values of the activation functions of the units, which may lead to saturations and therefore to slow convergence. In fact, we would ideally like to have activation functions with unit variance. Since the initial weights are random quantities, we have to make a choice for the probability distribution of the initial weights, which is normally chosen to be Gaussian. The components of the input vector have normally different variances, which would force choosing different variances for the distribution of the weights if we are to have activations with unit variance. A usual solution with MLPs is to normalize the input components to have zero mean and unit variance, so that all weight distributions can be chosen to have equal variances. In that case, the variance of the weight distributions is only dependent on the number of components as $\sigma \propto d^{-1/2}$ with d being the number of components in the input vector. In the case of our RBF, all the input components (outputs of the hidden layer) are between zero and one, and consequently their means and variances are not normalized to zero and unity, respectively. To initialize the weights of the RBFs (emission probabilities) we have modeled the non-normalized weights c_{kj} in

the Bayes non-linearity case and the weights b_{kj} using a uniform distribution distributed between 0 and 1.

The second issue is the stop criterion for RBF training. Since gradient descent is an iterative procedure we must use some criterion to know when to stop it. One may think that the more iterations the better would be the approximation of the posterior probabilities. In fact, if we measure the classification error on the training data we can observe that the classification rate improves after each iteration. However, if we also measure the classification rate after each iteration on an independent test set, we also observe that after a certain number of iterations the classification rate on this test set decreases. This phenomenon is called *over-fitting* and is a consequence of the so-called bias-variance trade-off when approximating any function using a *finite* set of data points [Bis96]. In fact, we are not interested in finding a classifier that perfectly classifies the data in the training set, but rather in inferring a classifier from the training data that generalizes to unseen data points. We know from the discussion in Sec. 4.3 that it is only possible to find the optimum mapping F_{opt} when the conditional density of the targets $\mathbf{t}$ given the data vectors $\mathbf{x}$ is known— or equivalently when an infinite number of data points is available—, and that the mapping is given by $F_{opt} = \langle \mathbf{t}|\mathbf{x} \rangle$. Since the number of data points available is finite, the output of the mapping F is not equal to the optimum, so that the error due to the finiteness of the training set is:

$$\epsilon(\mathbf{x}) = \mathrm{E}_S\{(F(\mathbf{x}) - \langle \mathbf{t}|\mathbf{x} \rangle)^2\} \tag{5.46}$$

where $\mathrm{E}_S\{.\}$ is the expectation over all possible training sets. Note that F depends on the training data set S. This error can be further decomposed in the following two terms [Bis96]:

$$\begin{aligned} \epsilon^2_{bias}(\mathbf{x}) &= (\mathrm{E}_S\{F(\mathbf{x})\} - \langle \mathbf{t}|\mathbf{x} \rangle)^2 \\ \epsilon_{var}(\mathbf{x}) &= \mathrm{E}_S\{(F(\mathbf{x}) - \mathrm{E}_S\{F(\mathbf{x}\})^2\} \end{aligned} \tag{5.47}$$

If we choose a mapping F to fit all the data points perfectly, the result would be a zero bias error but a rather large error due to variance. Conversely, if we choose a fixed mapping F independently of the training set S, the variance error is zero, but the bias error is large because we have paid no attention to the training data. We see, therefore, that there is a trade-off between both components of the error.

What we want to obtain is a good compromise between both components of error in Eq. 5.46, so that the total error is minimized. This can be achieved for example with *early stopping* techniques. An example of these techniques is hold-one-out validation or *cross-validation* [Bis96, BM94] [3], which divides the training set into two sub-sets. The large set with $L - N$ training vectors is used to estimate the parameters and the smaller set with N vectors is used after each training epoch to estimate the classification error of the trained RBF. This evaluation is repeated after each epoch until the classification error on the independent set of N vectors does not decrease.

To improve the efficiency of the EBP training, a technique can be used to avoid oscillations around the minimum point, which may slow down the convergence rate of EBP [4].

[3] Among speech researchers the hold-one-out method is also known as cross-validation, although strictly speaking cross-validation splits the training set into N equal sub-sets, and trains the net N-times, each time leaving one of the sub-sets from training and using it to compute the classification (generalization) error of the net after each training epoch.

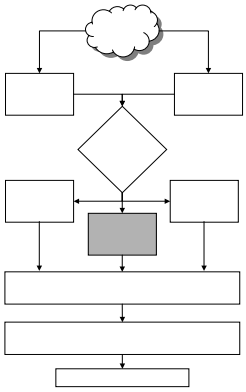
[4] http://www.icsi.berkeley.edu/speech/faq/nn-train.html

This technique works as follows: if after a certain epoch the cross-validation error is less than 0.5% absolute better than in the previous epoch, then the learn-rate is decreased by a factor of two after each epoch, until the difference in the cross-validation error between consecutive epochs is once again less than 0.5% or the maximum number of training epochs is reached (12 epochs).

Obviously, the phenomenon of the curse of dimensionality discussed in Sec. 2.6) may also have an influence on the classification performance of the neural net, if the number of training patterns is not large enough to reliably train the weights of the neural net.

5.6 Summary

In this chapter we have discussed the hybrid ANN/HMM approach. The advantage of this approach is that it uses the maximum a posteriori (MAP) criterion to train its parameters, and it is consequently a discriminative approach. The advantage of this approach as compared to other discriminative approaches is that it attains global discrimination, i.e. at the sentence level, by discriminating at the local level, i.e. at the HMM state level. Normally MLPs are the ANN used in this approach, but we have explained in this chapter the case of using RBFs instead. These ANNs have an structure similar to a GMM, and are therefore easily implementable in the current acoustic modeling of our system based on SCHMMs. We have also seen that the posterior probabilities at the output of a hybrid ANN/HMM can also be decoded using the Viterbi algorithm, and that a combination of the Viterbi training (segmental k-means) and gradient descent algorithms may be used to train the parameters of the hybrid RBF/HMM system.



6. Combining Multiple Streams of Features

In this chapter we present the multi-stream approach to ASR. Broadly formulated, this approach aims at improving the performance of classifiers by incorporating into the recognition process different sources of information about the speech signal. In particular, we devote this chapter to the study of combining different streams of feature vectors, each obtained from a different feature extraction algorithm. As already seen in Chapter 3, this approach obtained an excellent result on our AURORA 2000 evaluation, combined with the hybrid MLP/HMM or the tandem approaches. A first objective of this chapter is to understand how the multi-stream approach contributes to the excellent results. Another objective is to study whether this approach can also improve the performance of our LDA-based SCHMM system. We start the chapter with an introduction to the multi-stream approach, and follow with an overview of past research on the topic. Next we explain the multi-stream approaches tested in this thesis, which are the synchronous concatenation of feature vectors, the concatenation of LDA-transformed feature vectors and probability combination.

6.1 Introduction

The multi-stream recognizers discussed in this chapter combine different streams of feature vectors, extracted using different feature extraction algorithms, to obtain a better performance than any of the one-stream recognizers. The approach is based on the observation that different representations of speech often lead to different kinds of errors. This suggests that the errors of two classifiers using different feature extraction blocks are often complementary, and it is therefore possible, at least theoretically, that one of the features compensates for the errors of the other. Also, it has been shown [Rog94] that to achieve complementarity of errors it is usually better to use different feature vectors than different classifier architectures for each stream.

This complementarity of errors is mathematically formulated using the *correlation* of the errors. In fact, it has been theoretically and experimentally found [TG96, KHDM98, KB00, EB00] that the lower the correlation of the errors of two classifiers, the better is the performance of the combined classifier. If two classifiers based using different feature

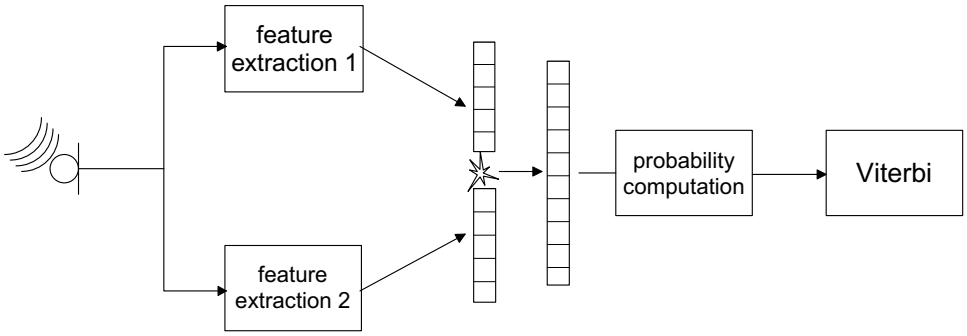


Fig. 6.1: Feature concatenation of two streams of feature vectors.

extraction algorithms have highly correlated error patterns, it is very unlikely that their combination improves the results of any of the features alone.

This last fact introduces the first aspect in multi-stream with different feature vectors, namely the selection of the feature vectors to be combined. This is a very difficult point because the correlation of the error patterns just give us an idea of the potential gain to be expected, but not of the actual gain, since this also depends on other factors such as combination method or higher classification stages. Normally the actual gain can only be determined after performing a speech recognition test with the combined features.

The second important aspect is the way the streams of feature vectors are combined. Depending on the nature of the feature vectors to be combined, the combination technique can be:

- **Asynchronous**, if the streams are processed and decoded independently to a certain level where they are merged to generate the final hypothesis.
- **Synchronous**, if the streams are combined at the frame level, i.e. every 10 ms.

In our experiments we have only considered synchronous combination of streams, because the available feature vectors were not inherently asynchronous. The two basic types of synchronous combination strategies are:

- **Feature concatenation**, where the feature vectors are simply concatenated to form an extended feature vector that is then further processed as a normal feature vector as shown in Fig. 6.1.
- **Probability combination**, where the outputs of the probability computation of both feature vectors blocks are combined somehow to obtain a single probability for each HMM-state as shown in Fig. 6.2.

The motivations for multi-stream recognition are manyfold, but they can be basically grouped into two classes [Hag01]: psycho-acoustic and engineering motivations. The former group includes all the motivation derived from human speech perception, whereas the latter includes those derived from pattern recognition theory and praxis. Among the psycho-acoustic motivations we can mention the following arguments:

- **Integration of different time scales**. As can be derived from a series of studies on human speech perception [AG98, Ghi94], the robustness of speech to interferences

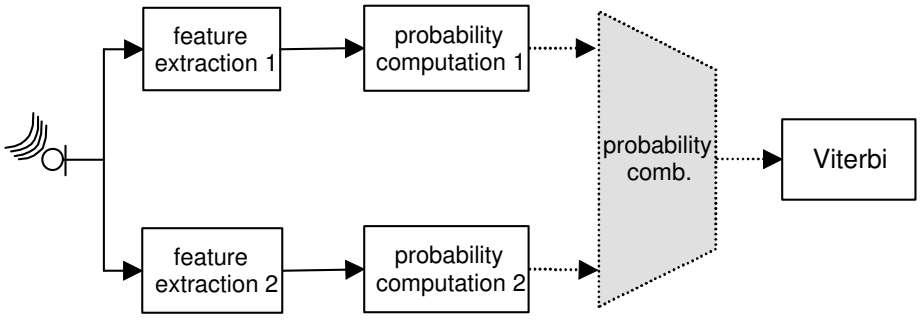


Fig. 6.2: Probability combination of two feature streams.

seems to be a result of effective integration or combination of several different time scales in the speech perception process. As a consequence, and since short-term information is already exploited by current ASR systems, it would be interesting to integrate long-term information into the ASR chain.

- **Redundancy of speech perception**. In another set of perceptual experiments [Gre97], it was found that the human speech perception system is, as the speech signal itself, highly redundant. This redundancy guarantees a high degree of robustness when some region of the speech signal is corrupted by noise.

On the other hand, some of the arguments from a pattern recognition point of view are:

- **Averaging of different classifiers**. In [Bis96] it was theoretically demonstrated that the averaging of the outputs of different classifiers can reduce the expected squared error by a factor equal to the number of classifiers being used, if the errors of the classifiers have zero mean and are uncorrelated. If these conditions are not met, then the squared error is reduced as well but the reduction is not as large.

- **Training on various environmental conditions**. Using a set of classifiers each trained on different conditions, e.g. trained on different noise conditions [Shi00], can often improve the performance of a single recognizer trained on just one condition. This is a way to increase robustness when the operation conditions of the recognizer are not exactly known.

- **Combination of various feature extractions**. Another reason to use multi-stream systems is the large number of different feature extraction, acoustic modeling and training algorithms available in the speech recognition literature. It is often the case, for example, that one of those algorithms works particularly well for certain environmental conditions. By combining several of those algorithms in one single system, a better robustness to environmental conditions can theoretically be achieved.

6.2 Prior Work on Multi-Stream ASR

In [KFS00] a comparison of different combination techniques using two different kinds of feature vectors (Articulatory Features and MFCCs) is presented. The acoustic modeling used in the experiments was based on context-dependent SCHMMs. The size of the

code-books used was 256 for the MFCCs and 324 for the articulatory features, and both are therefore larger than the code-books used in most of our experiments (cf. Chapter 7. The features are combined at three different levels: feature level, state level and word level. Experiments were conducted on the Verbmobil corpus, and recognition tests were performed on the official 1996 Verbmobil evaluation set. In a first set of experiments, streams were combined at the state-level (probability combination) using product, minimum, maximum and sum rules. Results show that product and minimum rule improve the baseline results, whereas maximum or sum rule are worse than baseline. To combine the streams at the word level, the ROVER algorithm was used [Fis97]. In feature level combination, the dimensionality of the concatenated feature vector (65 components) was reduced to 39 components by applying a heuristic dimensionality reduction algorithm. The results of the different combination levels show that state-level and word-level combination improve the performance of the baseline and have a similar performance, whereas feature-level combination does not significantly improve the baseline performance.

In another study [KB00], the authors argue that the added ensemble-incurred error increases with the degree of classifier error correlation. A method to reduce this correlation is to jointly train classifiers for multi-stream ASR. Since usual 'hard' rules, such as sum or product rules, cannot be used to jointly train classifiers (not continuous and differentiable), a family of 'soft' combination rules is defined and used to jointly train the classifiers. These 'soft' rules are a generalization of the 'hard' rules. Experiments are conducted on the OGI Numbers95 corpus using hybrid MLP/HMM acoustic modeling. In a first set of experiments, the two MLPs trained on J-RASTA-PLP and MFCC feature are combined using the conventional 'hard' techniques, and as expected the 'AND' rules are clearly superior to the 'OR' rules. Varying the parameter of the parameter-dependent 'soft' rules did not significantly change the performance. In a last set of experiments, it appears that joint embedded training of the two MLPs using the 'AND' soft-combination together with the 'hard' product rule during test significantly outperforms the product rule combination of MLPs that have been independently trained. However, this difference, albeit significant, is rather small and further experiments are needed to see if larger gains can be obtained.

As a justification of the results of the product rule, we can cite the result in [Kir99] where it was found that the product rule increases discriminability for correct classifications and decreases discriminability for incorrect classifications at the HMM-state level. This fact has a positive effect at higher decoding levels (Viterbi).

In [CLA00] the authors explore the use of using different features for each sub-band in multi-band ASR [BD96, BDR96, BDR97], and also the use of different full-band features in multi-stream ASR. The combination rules used are the unweighed sum and product rules. The combined feature vectors are the MFCC, PLP and J-RASTA-PLP features. Experiments are performed on the NOISEX database with a vocabulary size of 32 words. Results for multi-band ASR with different features per sub-band show a significant improvement with respect to the multi-band with the same features per sub-band. In the multi-stream experiments, the performance of the combined multi-stream systems (heterogeneous) was compared to the performance of one-stream systems with a similar size to the multi-stream systems (homogeneous). Results show that the product rule outperforms the sum rule, and that the heterogeneous systems are better than homogeneous systems in clean conditions. In noisy conditions, by contrast, the authors were not able to observe a significant advantage of heterogeneous systems over to homogeneous systems.

The authors of [SEK+00] analyze the combination of short-term features (PLP) with long-term ($\sim$ 1s) features (LDA-RASTA, MSG and TRAPS). Experiments are carried out on the AURORA 1999 noisy digit recognition task. They use tandem feature extraction for each feature vector to map it into a 24-dimensional space which is common to all reduced features, since all tandem MLPs have been trained on the same phoneme segmentation. Consequently, to combine the different 24-dimensional tandem feature vectors, they simply take the average over all the vectors to be combined. The number of feature vectors combined in that way are 2, 3 and 4 feature sets. The acoustic modeling used in these experiments has been conventional whole-word GMM/HMMs with 16 states per model and 3 mixtures per state. Results show that the non-linear transform (tandem) greatly improves the results. In addition, multi-stream combination as described before improves further the results of tandem. As in our experiments with PLP and MSG features and a similar system, the improvement of adding the MSG features is about 1% absolute in these experiments.

In a set of Large Vocabulary ASR experiments with the Broadcast News task, the authors of [JEM99] compare the results of multi-stream systems with mono-stream systems of similar size, i.e. of similar learning capacity. The acoustic modeling used is phoneme-based hybrid MLP/HMMs, and the study is carried out using a combination of MSG and PLP feature vectors. The sizes of the MLPs used are 344k and 758k parameters. Results show that multi-stream obtains similar performance to mono-stream with a similar number of MLP parameters. However, multi-stream can alleviate the memory requirements and the computational cost, since both MLPs can be trained on different machines. An interesting result on the 1998 Broadcast News evaluation is also presented, which shows that multi-stream improves performance in unmatched test conditions but does not apparently improve performance in matched test conditions.

In another study combining long-term and short-term features [HS99], a new kind of long-term temporal pattern (TRAPS) is presented. The idea is simply to take slices across the time axis of the filter-bank time-frequency representation of the signal. Therefore, if we have L filter in the filter-bank then we obtain L sequences of TRAPS. Each of these slices is mapped into 29 phonetic classes using an MLP to obtain the neural TRAPS. To obtain a single stream of features from these 15 neural TRAPS, an MLP is used to combine them. Experiments with these new kind of features are performed on the OGI Stories and the OGI Numbers corpus. The acoustic modeling used hybrid MLP/HMMs. The results clearly show that TRAPS alone do not improve baseline results, but combined with baseline features improve baseline performance.

In an experimental analysis by [EB00] on the AURORA 2000 noisy digit recognition task, the usefulness of *conditional mutual information* (CMI) to design feature combinations is studied. The rationale behind this measure is that feature pairs with a high CMI are highly correlated, and may therefore be more easily modeled using feature combination, since this allows the modeling of the correlations. Conversely, classifiers trained using different features whose outputs have a low CMI, are good candidates for probability combination. The acoustic modeling used in the experiments is hybrid MLP/HMM. The results reported in this work do not support the assumption that feature combination should be favored when the CMI of the features is large. However, they clearly support the assumption that low CMI between pairs of features is a good indicator of the benefit of combining them. This agrees with the previous results in [KB00] where it was shown

that good-performing combinations have indeed low-correlated classifier outputs.

In [WKMG98] the combination of syllable-based and phoneme-based recognizers was studied. As in previous attempts, the idea was to integrate short-term (phoneme) and long-term (syllable) information in the ASR process. The syllable-based recognizer used MSG features and hybrid MLP/HMM acoustic models trained to classify into 124 semi-syllabic categories. The phoneme-based recognizer, in contrast, used log-RASTA-PLP features and also hybrid MLP/HMM trained to classify into 32 phonetic classes. In contrast to previous approaches, the combination of both recognizers was performed at the *whole-utterance* level. This is performed by generating one N-best list for each recognizer and merging both. For each hypothesis sentence in the merged list, two acoustic scores are generated by performing a forced alignment on the sentence using both recognizers. The final score for each sentence is obtained by a weighted sum of both scores and the language model score. Results were obtained on the OGI Numbers corpus with a 32-word vocabulary, and show that the combination of both recognizers is significantly better than any of the recognizers alone. To demonstrate the result was not due to the larger number of parameters of the combined system, tests using larger MLPs and one feature were also performed, but none of the results was better than the combined system.

Finally, in a study by [Hag01] a new kind of combination rule for multi-stream systems is presented and evaluated: the full combination approach. Hybrid MLP/ HMM acoustic modeling was used in all the experiments. In a first set of experiments, the new rule is compared to the mono-stream systems and standard sum and product rules (cf. Sec. 6.3.2). The streams used are formed from different feature vectors (PLP, MFCC and J-RASTA-PLP) to achieve a good degree of complementarity. The results of this first set of experiments show that neither the 'standard' nor the full combination multi-stream approaches were significantly better than the 'best' mono-stream recognizer in any of the tested noise conditions. In another set of experiments, diversity of streams was achieved by computing the delta and delta-delta of the feature vectors. Experiments were carried out using PLP and J-RASTA-PLP features, and show that for the PLP features results can be greatly improved by using the full combination rule. For the J-RASTA-PLP features, by contrast, no significant improvement was observed when using any of the variants of the full combination rule.

6.3 Synchronous Combination of Streams

As already mentioned in the introduction, one of the two fundamental aspects in multi-stream ASR is the way to recombine or merge the streams of features. In synchronous combination of streams, the streams are combined at every frame, and must consequently be generated with the same frame rate.

The objective of this section is to introduce the combination rules we have tested in our experiments, with special emphasis on rules that may be used with LDA and tandem-like feature vectors.

6.3.1 Feature Concatenation

This is the simplest way of combining two synchronous streams of features. As shown in Fig. 6.1, feature vectors are concatenated into one single large vector, which is then processed by a single classifier.

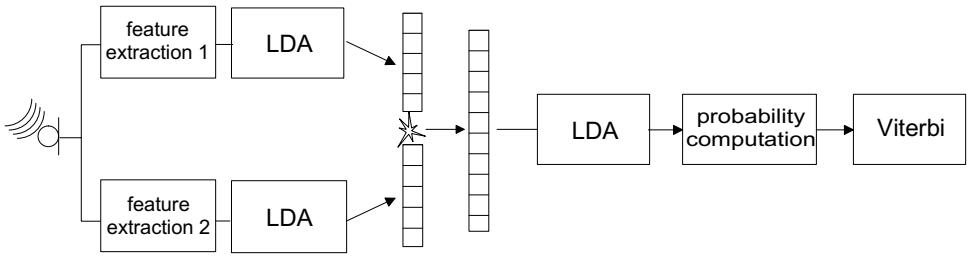


Fig. 6.3: Concatenation of LDA feature vectors.

One of the disadvantages of this technique is that it leads to classifiers with a larger number of parameters, and consequently longer training and processing times. Also, if the amount of memory resources is limited, it could even be impossible to train such a system [JEM99]. Moreover, if the amount of training data is small, the reliable estimation of the large number of parameters can be very difficult due to the curse of dimensionality (cf. Sec. 2.6).

A possible solution to the previous problem is to use some kind of dimensionality reduction technique as explained in Chapter 4. If LDA is used to reduce dimensionality of the single streams we can apply LDA to both streams independently, concatenate both output vectors, and further reduce the dimensionality of the concatenated vector using a third LDA transform as illustrated in Fig. 6.3.

6.3.2 Probability Combination

In this case, each of the input streams is independently processed by a specific classifier to obtain a probability vector for each stream. These probability vectors are afterwards recombined to have a single probability vector that is passed to the decoding stage (see Fig. 6.2) or to another classifier, as in the tandem original approach (cf. Sec. 4.6.1).

In fact, from a pattern recognition point of view the probability combination approach can be formalized and generalized using the concept of mixture-of-experts [Bou99, Bis96]. This approach was originally developed by [JJNH91] to model feature vectors that have different distributions in different regions of the space. As shown in Fig. 6.4, the outputs of the two classifiers are first weighted and then combined. The weights are controlled by a gating network to force the use of the right classifier for the current input vector. A training procedure exists to jointly optimize the parameters of both classifiers and the gating network. As can be deduced, the main aspects in this mixture-of-experts approach are the combination rule of the probabilities, the weighting method, and the optimization of the classifier and gating network parameters.

Although the previous approach is applied to one single vector, it can obviously be applied to our multi-stream problem as well. However, in our case we do not use any weighting, nor jointly optimize the parameters of the different classifiers. The reason for the first restriction is that in the case of multi-band ASR [Hag01] no improvement was observed when using different weighting techniques for the sub-bands. Moreover, in the same work, the author used relative frequency weighting in multi-stream experiments, but no improvement was observed with respect to the equal weights combination. A reason for the second restriction is also that just a small improvement was observed in [KB00] when classifiers were jointly optimized. Consequently, we simply assume equal weights for all

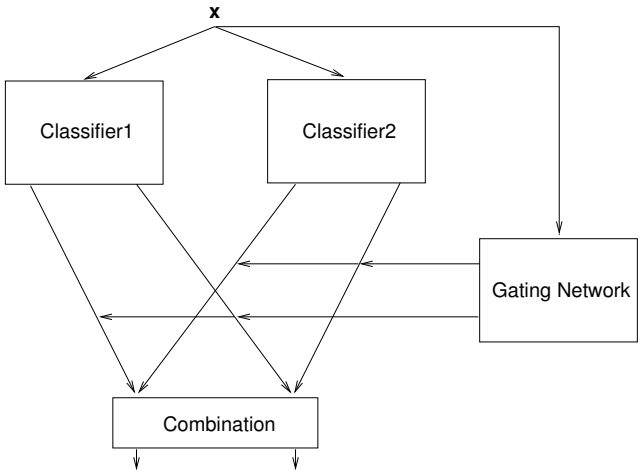


Fig. 6.4: The mixture-of-experts approach to model a feature vector that has different distributions in different regions of the space.

streams, and we optimize each classifier independently, so that our multi-stream system reduces to that already shown in Fig. 6.2.

The remaining question is, therefore, the kind of recombination rule to use. Many different possibilities exist to combine the output probabilities of two classifiers— see for example [Hag01]—, but all of them are derived from two basic types: the sum rule and the product rule.

Sum Rule

As the name suggest, the basic idea of this rule is to add the probabilities of the different classifiers. Depending on the kind of acoustic modeling, we have two versions of this rule:

- **Posterior based systems (hybrid MLP/HMM)** Let q_k be the k-th HMM state and $\mathbf{x}_l$ the feature vector corresponding to stream l. Further, let us introduce a set of mutually exhaustive random events $\{b_l : \ l = 1, \ldots, L\}$, each denoting the event 'stream l is the most reliable stream AND the other streams are unreliable'. Using those random events we can decompose the posterior probability $P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L)$ in the following way:

$$\begin{aligned} P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) &= \sum_{l=1}^{L} P(q_k, b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \\ &= \sum_{l=1}^{L} P(q_k \mid b_l, \mathbf{x}_1, \ldots, \mathbf{x}_L) P(b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \\ &= \sum_{l=1}^{L} P(q_k \mid \mathbf{x}_l) P(b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \end{aligned} \tag{6.1}$$

  The term $P(b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L)$ denotes the reliability of the event b_l. This reliability can be interpreted as the outputs of the gating network in Fig. 6.4, whereas the

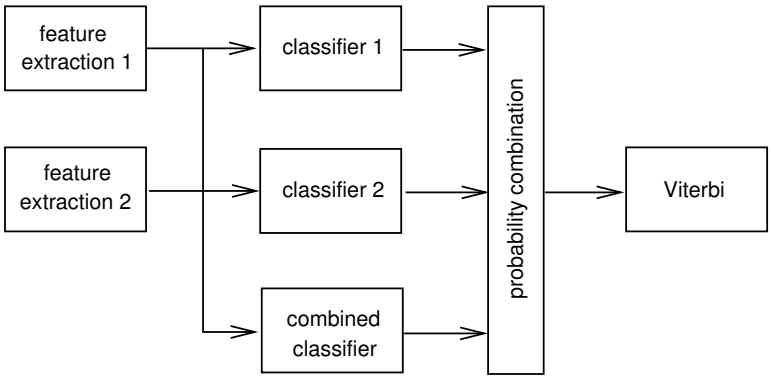


Fig. 6.5: The full combination approach to solving the non-exhaustiveness problem. The input to the combined classifier is the concatenation of both feature vectors.

posteriors $P(q_k \mid \mathbf{x}_l)$ would be the outputs of the classifiers in the same figure. If we further assume that this reliability term is independent of the observation vectors $\mathbf{x}_i$ then we can write:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \sum_{l=1}^{L} P(q_k \mid \mathbf{x}_l) P(b_l) \tag{6.2}$$

If we further assume that the b_l are equiprobable then we can write:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \frac{1}{L} \sum_{l=1}^{L} P(q_k \mid \mathbf{x}_l) \tag{6.3}$$

The assumption of exhaustiveness of the b_l is however not true in general because it can be that two or more streams are reliable at the same time. Therefore the previous derivation is a kind of heuristic justification of the sum rule.

A possibility to circumvent the problem of non-exhaustiveness is to use the so called full combination approach for the sum rule [Hag01, MHB99]. In this approach, a set of random events $\{b_i : \ 1 \leq i \leq 2^L - 1\}$ is also constructed, but each of the events is associated with a group of streams instead of being associated with one single stream. The events associated with the variables b_i are of the form 'streams in the group i are reliable and the others are unreliable'. Since we have a set of L streams, there are 2^L possible sub-sets, and excluding the empty set we have $2^L - 1$ events. These events are exhaustive, and it is therefore absolutely correct to write:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \sum_{l=1}^{2^L - 1} P(q_k \mid \hat{\mathbf{x}}_i) P(b_i \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \tag{6.4}$$

As seen if Fig. 6.5, the first particularities with respect to Eq. 6.1 is the number of classifiers, which in this case is $2^L - 1$ because each classifier is, as before, associated with one of the events b_i. The second particularity is the input vector $\hat{\mathbf{x}}_i$ to these classifiers, since this is formed by concatenating all the streams in the group i.

Making the same assumptions as before, we arrive at a similar equation to Eq. 6.3:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \frac{1}{2^L - 1} \sum_{i=1}^{2^L-1} P(q_k \mid \hat{\mathbf{x}}_i) \tag{6.5}$$

A problem with this approach is the larger training and processing time, since the number of classifiers needed grows exponentially with the number of streams.

- **Likelihood based systems (SCHMM)** Using the Bayes rule we can derive the sum rule for likelihood based systems from the formulae derived in the previous point

$$\begin{aligned} \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_L)} &= \frac{P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L)}{P(q_k)} \\ &\approx \sum_{l=1}^{L} \frac{P(q_k \mid \mathbf{x}_l)}{P(q_k)} P(b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \\ &= \sum_{l=1}^{L} \frac{p(\mathbf{x}_l \mid q_k)}{p(\mathbf{x}_l)} P(b_l \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) \end{aligned} \tag{6.6}$$

Applying once again the assumptions of the previous point we obtain:

$$\frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_L)} = \sum_{l=1}^{L} \frac{p(\mathbf{x}_l \mid q_k)}{p(\mathbf{x}_l)} P(b_l) \tag{6.7}$$

and finally:

$$\frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_L)} = \frac{1}{L} \sum_{l=1}^{L} \frac{p(\mathbf{x}_l \mid q_k)}{p(\mathbf{x}_l)} \tag{6.8}$$

As before, the full combination approach can also be applied using a similar formula to Eq. 6.8 but with supplementary terms:

$$\frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_L)} = \frac{1}{2^L - 1} \sum_{i=1}^{2^L-1} \frac{p(\hat{\mathbf{x}}_i \mid q_k)}{p(\hat{\mathbf{x}}_i)} \tag{6.9}$$

Product Rule

As in the previous type of rule, we have different forms of the rule for likelihood-based and for posterior-based ASR systems:

- **Posterior-based systems (hybrid MLP/HMM)**. Using the same independence assumption of the previous point, we can decompose the joint state-posterior probability of the streams into the product of state-posteriors of the single streams as

follows:

$$\begin{aligned} P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) &= \frac{P(q_k)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_L)} p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k) \\ &= \frac{P(q_k)}{p(\mathbf{x}_1) \cdot \ldots \cdot (\mathbf{x}_L)} \prod_{l=1}^{L} p(\mathbf{x}_l \mid q_k) \\ &= \frac{P(q_k)}{p(\mathbf{x}_1) \cdot \ldots \cdot (\mathbf{x}_L)} \prod_{l=1}^{L} \frac{P(q_k \mid \mathbf{x}_l) p(\mathbf{x}_l)}{P(q_k)} \\ &= \frac{\prod_{l=1}^{L} P(q_k \mid \mathbf{x}_l)}{P(q_k)^{L-1}} \end{aligned} \tag{6.10}$$

If we now assume that all states have equal prior probabilities, the term in the denominator can be discarded and the following product rule analogous to the one for likelihoods is obtained:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \prod_{l=1}^{L} P(q_k \mid \mathbf{x}_l) \tag{6.11}$$

The full combination variant of previous rule is:

$$P(q_k \mid \mathbf{x}_1, \ldots, \mathbf{x}_L) = \prod_{i=1}^{2^L-1} P(q_k \mid \hat{\mathbf{x}}_i) \tag{6.12}$$

where we have assumed, as in the previous equation, that the states are equiprobable.

- **Likelihood-based systems (SCHMM)**. The likelihood of the stream $\mathbf{x}_l$ in the state q_k is $p(\mathbf{x}_l \mid q_k)$. If we now assume that the L streams are independent, the joint state likelihood of the streams can be expressed as:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k) = \prod_{l=1}^{L} p(\mathbf{x}_l \mid q_k) \tag{6.13}$$

This last equation is the product combination rule of streams for likelihood based systems.

As can be deduced from the product rule, a single expert may prevent the recognition of a certain state q_k if the likelihood for the expert is sufficiently low. This kind of combination rule, therefore, emphasizes the classes on which all the classifiers agree but suppresses those on which the classifiers disagree, especially if some of them give a very low likelihood to the class.

A problem with this rule is the independence assumption used in Eq. 6.13. When the features are strongly correlated, we lose all the correlation information if we use the previous equation to combine both streams. A possible solution is to use the full combination approach, since this approach allows us to model the concatenated vectors, and consequently the correlations between them. In this case, the form of the full combination approach is

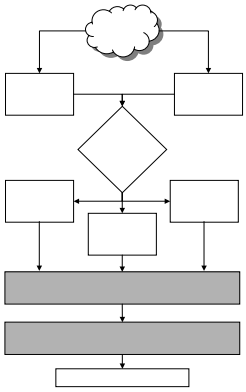
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_L \mid q_k) = \prod_{i=1}^{2^L-1} p(\hat{\mathbf{x}}_i \mid q_k) \tag{6.14}$$

As already seen in Sec. 6.2, several studies have empirically demonstrated the superiority of the product rule compared to the sum rule for many ASR tasks [KFS00, KB00, CLA00]. This is the reason why we have decided to use this rule in our experiments with multi-stream in Chapter 7.

6.4 Summary

In this chapter we have introduced multi-stream ASR using different feature vectors. We have seen in the literature overview that it is hard to tell from the correlation structure of two feature vectors whether feature concatenation or probability combination is better. Nevertheless, it is also clear that feature combination normally needs more hardware and data resources than probability combination, so that probability combination is often preferable. For this last kind of combination, if the correlation of the outputs of two classifiers trained on different feature sets is low, then both classifiers are good candidates for probability combination. In addition, several studies seem to show that the product rule is better than the sum rule for probability combination.

A way to alleviate the resource problem of feature concatenation is to use an LDA transform to reduce the dimensionality of the concatenated feature vector. We have seen, as well, that probability combination can be formalized and generalized using the concept of mixture-of-experts. For probability combination, we have discussed two sets of rules namely the sum rules and the product rules. The problem with the first rule is the exhaustiveness assumption, whereas the problem with the second is the independence assumption. Theoretically, a solution to both problems can be to use the full combination approach in both cases.



7. Experiments and Results

In this chapter we present the results obtained using the approaches discussed in the previous three chapters. The ASR experiments have been performed on two different speech databases: the AURORA 2000 digit recognition task and the UKKCP task. The first database has a relatively small training data set, and has therefore been used as a quick test bed. However, the noises in the AURORA task have been artificially added, and the size of the vocabulary is too small. These facts limit the significance of the results on this database for practical applications. For this reason we have also tested the most promising approaches on our in-car ASR task UKKCP, which has been recorded in a real in-car environment, and includes spelling and city names in the test set as well. The chapter starts with an overview over the experiments. Next we describe the databases used in our experiments: the AURORA 2000 database and the UKKCP car database. This is followed by the experiments on the AURORA database, on which all the approaches treated in the previous chapters have been tested. The second part is devoted to the UKKCP experiments.

7.1 Introduction

Most of the experiments in this chapter have been carried out on the AURORA 2000 database, since this database has an small training set and the approaches to test can consequently be trained very quickly. As already mentioned above, we have only tested the best-performing approaches on the UKKCP database.

An overview of the groups of experiments carried out can be seen in Tab. 7.1. The groups of experiments not performed on the UKKCP database are marked with a '-' in the second column. In the optimum feature set experiments we have compared different feature reduction algorithms using various acoustic modelling approaches. In the experiments with discriminative feature reduction we have compared the results of the different neural net topologies and classes for feature reduction proposed in Chapter 4. In the hybrid RBF/HMM experiments we have compared this system with two systems based respectively on SCHMMs and hybrid MLP/HMMs. Finally, in the experiments with multiple streams of features we have compared different feature combination strategies.

	AURORA	UKKCP
Optimum Feature Set	Sec. 7.3.2	-
Discriminative Feature Reduction	Sec. 7.3.3	Sec. 7.4.2
Hybrid RBF/HMM Systems	Sec. 7.3.4	-
Multiple Streams of Features	Sec. 7.3.5	Sec. 7.4.3

Tab. 7.1: Overview of the experiments with both databases.

7.2 Description of the Speech Databases

7.2.1 The AURORA 2000 Task

This database is a digit recognition task in American English derived from the TI-DIGITS database [Leo84]. Since this database was originally designed to evaluate speech recognition methods for mobile telephone applications, the original sampling rate of the TI-DIGITS was reduced to 8 kHz, which is standard in those applications.

As shown in Table 7.2 this database has two training sets, one with noisy speech data. The advantage of using noisy speech data (also known as multi-style training) is that the parameters of the HMMs can be adjusted to model the effect of interfering noise. This has been proven to be in practice usually better than clean training, even if the interfering noise during recognition was not present in the training set [LMP87]. In addition clean databases must be recorded in controlled environments, which is usually very expensive, especially if the amount of speech data to record is large. In contrast, there is already a lot of noisy recorded speech, which could be used with comparatively little cost and effort. In our AURORA 2000 experiments we have trained the HMMs using the multi-condition training set, because this kind of training data is more similar to that of our in-car UKKCP database (cf. Sec. 7.2.2).

The confidence intervals for the experiments with this database can be extracted from Fig. 3.4(a) and Fig. 3.5. In the first figure we can measure the confidence intervals for the single recognition experiment results shown in Appendix A, whereas in the second figure we can measure those of the mean results presented in this chapter. Thus for this figure the curve with $n = 10$ corresponds to the results in the figures labelled with 'SNR levels', the curve with $n = 16$ corresponds to the figures labelled with 'test sets', and the curve with $n = 40$ corresponds to the mean results shown in the figures labelled with 'test sets'.

7.2.2 The DaimlerChrysler In-Car Database: UKKCP

This large database was recorded to provide enough speech data for reliable training of the speech models. Just a fraction of this huge database has been used in our experiments. The main features of this database are summarised in Table 7.3.

Database description	It consists of digit sequences in American English of up to seven digits, which have been corrupted with artificially added real noises at different SNR levels.
Recordings	Speech was sampled at 8 kHz rate. Sampled speech was passed through a band-pass filter with frequency characteristics corresponding to those of the G.712 ITU recommendation [HP00]. This filter simulates the frequency response of a typical PCM transmission channel. Noises were first filtered with the G.712 filter, attenuated to achieve the desired SNR levels, and finally added to the speech signal to obtain the noisy speech files. A total of 8 different noises were added to the speech signal. These noises were recorded in typical application scenarios for telecommunications terminals: suburban train, crowd of people, car, exhibition hall, restaurant, street, airport, train station. All the noises had a fairly low pass characteristic, and the recordings in the street and at the airport contained non-stationary segments.
Speaker	Male and female adult speakers.
Database structure	Two training sets: multi-condition or multi-style training data set, and clean training data set. The speech data in both sets is exactly the same and contains a total of 8440 files with approximately 1.5 hours of speech. The speech data in the multi-condition training set has been contaminated with four different noises (suburban train, crowd of people, car and exhibition hall). The test set is composed of three test sets: the matched test set (test a), the unmatched test set (test b) and the distorted test set (test c). The test a contains speech contaminated with the same noises as those in the multi-condition training set, whereas the speech in the test b has been contaminated with the noises not present in the training set (restaurant, street, airport and train station). In the test c the data contaminated with suburban train noise and street noise have been further filtered using the MIRS filter which simulates the frequency response of a typical telecommunication terminal for GSM [HP00]. Only 4 of the 7 possible SNR levels (0 dB, 10 dB, 20 dB and clean) were used in our experiments. This resulted in a total of 16016 test files.

Tab. 7.2: Main features of the AURORA 2000 database.

The confidence intervals for the experiments with this database can be extracted from Fig. 3.4(b), where the curves for the three tests are depicted.

7.3 Experiments on the AURORA 2000 Database

7.3.1 Baseline System Configuration

Feature Extraction

The first step in our baseline system is to employ spectral subtraction [Bol79, LB91] to reduce the distortion due to additive noises. Our ASR system uses a feature extraction algorithm based on a bank of filters distributed uniformly over the mel-frequency scale described in Sec. 2.3, and in more detail in [RJ93]. However, those filters differ from

Database description	British English database that consists of digit sequences, alphabet letters, short command sentences, e.g. "radio off", and city names.
Recordings	Recorded in real car environments at a sample rate of 16 kHz. The noises present in this database come from a great variety of sources such as rain or wipers. However, and as already explained in Sec. 1.2, the most important noise components are due to motor, tires and aerodynamical phenomena.
Speaker	Male and female native Britons. The sentences in the test set were spoken by 16 women and 19 men.
Database structure	A total of 32107 speech files were used for training which resulted in nearly 10,000,000 training vectors after feature extraction. The test set consisted of 3894 utterances spoken by 16 women and 19 men. The test lexicon contained a total of 2827 words, which were arranged into the following sub-lexicons: • **digits** sub-lexicon (denoted as 'digmb' in the experiments) that uses whole-word HMMs to model the digits and the words 'hash', 'square' and 'star'. • **spelling** sub-lexicon that also uses whole-word HMMs to model the English alphabet letters. • **cities** sub-lexicon that uses a mixture of context-dependent, context-independent (phonetic) and whole-word models. For each file in the test set only one of the sub-lexicons was active during the test to simulate the operating characteristics in a spoken dialog system.

Tab. 7.3: Main features of the UKKCP car database.

the usual triangular response in that their response has been designed using perceptual considerations. Since the sampling frequency of the AURORA-2000 database is 8 kHz, the number of filters, or equivalently mel-filter coefficients used in our feature extraction is 16. A discrete cosinus transform (DCT) is next applied to these coefficients to further reduce the number of coefficients to 13. The feature extraction process of our baseline system is depicted in Fig. 2.2 of Chapter 2.

To compensate for possible channel distortions, e.g. convoluted frequency response of a microphone, we apply cepstral mean subtraction (CMS) on the cepstral vector. Since CMS also eliminates the spectral tilt, which is highly speaker dependent, the mean cepstral vector is just adapted during the speech segments to better adapt it to the characteristics of the current speaker.

Acoustic Modeling

The topology of the HMMs are whole-word models of the English digits. These models have a different number of states, between 8 and 15 states, to account for the different length of the digits. A one state pause model and a noise model with 9 states are also used to model the inter-word pauses and the noisy silences at both ends of the speech files, repectively. The total number of HMM-states is 127.

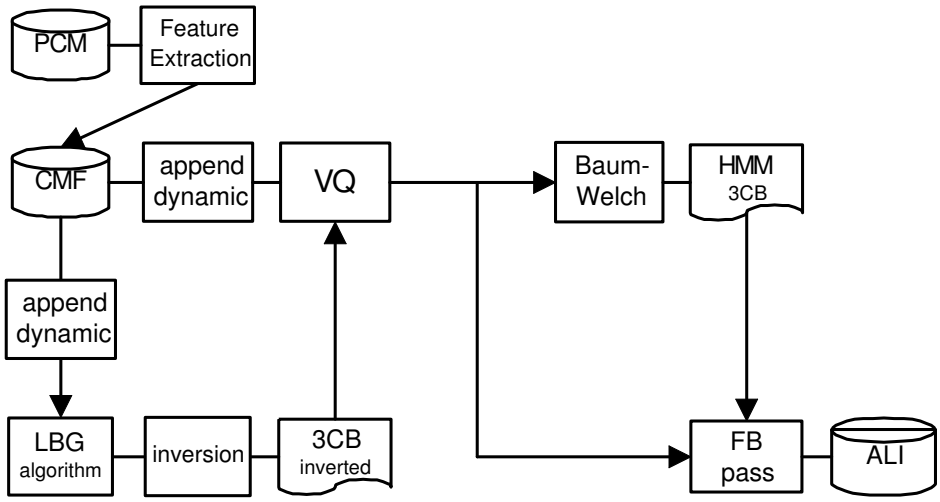


Fig. 7.1: Generation of the segmentation into states (ALI) by performing a forced alignment on the speech files. The SCHMMs used in this forced alignment have been trained using unsupervised clustering (LBG) and the Baum-Welch algorithm. The vector quantization (VQ) computes the probabilities of the symbols in the unsupervised code-books. Cylinders represent a set of files (PCM, feature, etc...), rectangles a process and the clipped rectangles are the results of the processes.

Training

Our training process basically consists of two steps: a first to generate a segmentation of the training set into HMM states, and a second stage that used this segmentation to train our final set of SCHMMs. The first stage is shown in Fig. 7.1. As can be seen in the figure, the first step in this stage is to append the 1st and 2nd derivative vectors to the static feature vector, resulting in a 39 component feature vector. Since our ASR system is based on the semi-continuous HMM acoustic modeling approach described in Sec. 2.5.2, a second step is to estimate a code-book to quantize the feature vectors. The estimation process uses the Linde-Buzo-Gray (LBG) algorithm [LBG80], and the result of this process is a set of mean vectors and covariances each associated with a symbol or class in the code-book. In our case three separate code-books for the static, 1st derivative and 2nd derivative feature vectors are estimated. The static code-book has a total of 512 symbols, and the 1st and 2nd derivative code-books 256 symbols, which results in a total of 1024 symbols. The code-books obtained using the LBG algorithm are termed *unsupervised* code-books, because there is no *a priori* association between feature vectors and symbols, as already discussed in Sec. 2.6.1. After estimating the code-books, they are used to quantize the feature vectors in the training set as described in Sec. 2.5.2. The particularities of our vector quantization (VQ) are:

1. symbol likelihoods cannot fall under a certain minimum threshold below the maximum likelihood value.

2. a maximum of 10 symbols per code-book is allowed.

3. the likelihood of the remaining symbols is normalized to sum up to one.

Since we have three different code-books, we obtain three different quantized feature vectors per feature frame. Next the quantized feature vectors are used as input to the Baum-Welch algorithm in Sec. 2.6.2, to estimate the emission probabilities of the symbols for each HMM-state and code-book, and the transition probabilities between HMM-states allowed by the HMM-topology. There are three different sets of emission probabilities for each HMM state, one with 512 symbols for the static features, and two with 256 symbols for the 1st and 2nd derivative features. To obtain a single likelihood per state, we first compute the likelihood of each code-book/set of emission probabilities pair, and then we multiply the partial likelihoods of each set, just as in multi-stream probability combination discussed in Sec. 6.3.2, i.e.:

$$p(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \mid q_i) = \prod_{l=1}^{3} \sum_{k=1}^{K_l} b_{ik}\, g_k(\mathbf{x}) \tag{7.1}$$

where the q_i, b_{ik} and $g_k(\mathbf{x})$ are as before the state i, the emission probability of the symbol k in state i and the normalized likelihood of symbol k for the input frame $\mathbf{x}$. The last step of this stage is to apply a Baum-Welch pass or forced alignment on the quantized feature vectors, in order to obtain the desired segmentation into HMM states of the training data. This step generates an alignment file (ALI) for each file in the training set, which contains the state/frame associations.

The second and final stage of our training process is to estimate the final SCHMMs as shown in Fig. 7.2. The first step is to compute a supervised code-book using the ALI files computed in the previous stage. This code-book is calculated by first constructing a context window of 9 frames around the current frame—four frames before and four frames after— to form a single 117-dimensional feature vector. Since each of the code-book classes is associated with an HMM state, the code-book means and covariances are computed by averaging the high dimensional vector over all the frames assigned to the same state in the ALI files. This results in a code-book with a total of 127 code-book symbols. For a more detailed description of the computation of the supervised code-book see [CKRB93]. Afterwards linear discriminant analysis (LDA) is applied to reduce the dimensionality of this code-book (cf. Chapter 4). Our LDA matrix projects the 117-dimensional space into a 32-dimensional space. At the end of this process we have a 32-dimensional supervised code-book with 127 symbols. This code-book is used together with the 117×32 dimensional LDA-matrix to quantize the high-dimensional cepstral feature vector of 117 components in the same way as in the previous unsupervised step. Finally the VQ feature vectors are passed to the Baum-Welch algorithm to estimate the **A**-matrix (transitions) and the **B**-matrix (emissions) of the SCHMMs. At the end of the training process, therefore, we have a 117×32 dimensional LDA-matrix, a supervised code-book with 127 symbols and the trained HMMs with a 127×1024 dimensional B-matrix of symbol emission probabilities and 127×127 dimensional A-matrix of state transition probabilities which are later on passed to the ASR system.

Recognition

Our ASR system is depicted in Fig. 7.3, where we can see the HMMs, code-book and LDA matrix computed during the supervised training stage. As in the supervised stage

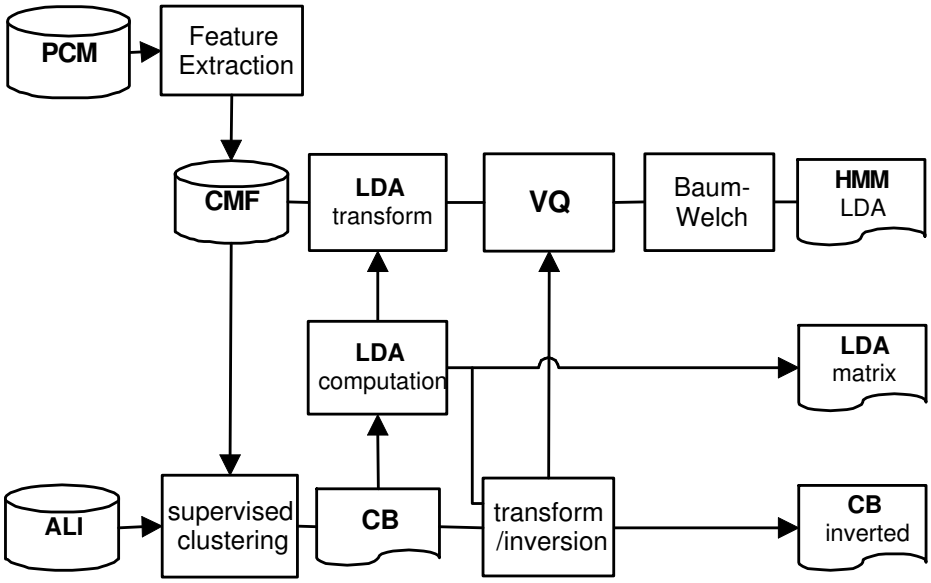


Fig. 7.2: The supervised stage of the DC training system. The segmentation into states (ALI) has been found in the unsupervised stage depicted in Fig. 7.1. The vector quantization (VQ) computes the probabilities of the symbols in the supervised code-book. As before, cylinders represent a set of files (PCM, feature, etc...), rectangles a process and the clipped rectangles are a result of a certain process.

of training, a high-dimensional vector of 117 components is constructed and transformed into a 32-dimensional vector using the LDA-matrix computed during training. This last vector is then quantized employing the supervised code-book of 127 classes, and a VQ algorithm as the one used during training. Next the VQ vector is passed to the acoustic modeling block, which computes the HMM-state likelihoods by multiplying the VQ-vector by the B-matrix, which contains the emission probabilities of the code-book symbols b_{ik} (cf. Sec. 2.5.2).

The decoding algorithm used is the one-pass (one-stage) algorithm explained in Sec. 2.5.1. No statistical language model or grammar has been used for this task, and consequently any HMM can be followed by any of the HMMs. In addition, no pause or noise models are forced at the beginning and end of the sentences, even though it would be beneficial in the experiments with the AURORA database, since there are long non-speech segments at the ends of the files in this database.

7.3.2 Optimum Input Feature Set

As a first step towards finding ASR techniques better than those used in our baseline system, we have compared the results of our current feature extraction with those of other state-of-the-art feature extraction algorithms. The algorithms we have compared are the following:

- **P*erceptual* L*inear* P*redictive* (PLP)** features [Her90], which estimate the LP all-pole model of the speech, but taking into account perceptual considerations such

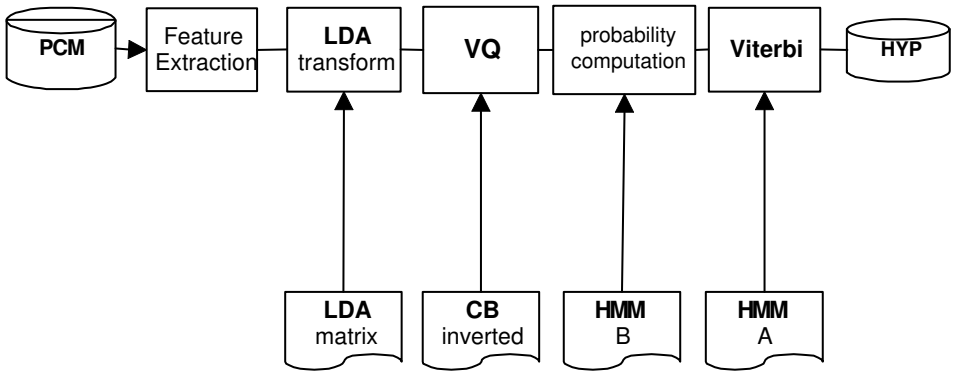


Fig. 7.3: A diagram of our baseline ASR system.

as critical-band spectral selectivity, equal-loudness curve and the intensity-loudness power law. The input speech was first pre-emphasized using a high pass filter. As in our baseline system, a Hamming window of 25 ms and a frame shift of 10 ms was used to analyze the input speech. The power spectrum was computed using an FFT of 256 samples. The critical band analysis is performed with a bank of 17 trapezoidal filters equally spaced in the Bark scale. The number of extracted PLP cepstral coefficients per frame was 12 plus frame energy.

- **The R*el*A*tive* S*pec*T*r*A (J-RASTA) PLP** features [HM94] combine the perceptual considerations of the PLP features, plus the fact that human perception of speech-like sounds depend on the spectral difference between the current speech sound and the preceding sound [SSN87]. From an engineering point of view, the effect of J-RASTA processing is to eliminate the spectral components of the input signal that vary more slowly or quickly than the speech itself. The advantage of the J-RASTA processing is that it can remove both additive (noise) and convolutive (channel) distortions. As in the PLP case, speech is pre-emphasized, and subsequently analyzed using a Hamming window of 25 ms and a frame shift of 10 ms. As in the previous algorithm, the power spectrum was computed using an FFT of 256 samples, and the filter-bank is exactly the same as before. The number of J-RASTA PLP coefficients is 10 plus frame energy, and the value of the J constant in the non-linearity is 10^{-6}, which remains constant, i.e. it is not adapted to the noise level in the actual speech file.

- **M*odulation* S*pectro*G*ram* (MSG)** features [Kin98] are also inspired by similar perceptual considerations as the PLP and J-RASTA features, but other perceptual phenomena, based on human perception of reverberant speech, are also taken into account to obtain a representation of the speech which is stable across a range of acoustic distortions. The MSG features are also spectral features which are further processed in the modulation frequency domain [1]. As before, the speech signal is

[1] In the sense of this text, the modulation frequency domain is the Fourier transformed domain of the time axis in a time-frequency representation of the speech signal. This implies that the modulation spectrum is the Fourier transform of the time variation of a certain spectral, cepstral or PLP coefficient. A global modulation spectrum can be obtained by averaging the modulation spectra of the coefficients.

analyzed using a Hamming window of 25 ms and a frame step of 10 ms. The power spectrum is computed using an FFT of 256 points. The filter-bank analysis of these features used a bank of 14 triangular filters uniformly distributed in the Bark frequency scale. In our experiments, we use 14 MSG coefficients to model the lower modulation frequencies (0-8 Hz), and 14 coefficients to model the higher modulation frequencies (8-16 Hz).

For more details about the previous algorithms see Appendix B. Two different kinds of acoustic modeling approaches have been used to compare the performance of the feature extraction algorithms:

- SCHMMs with LDA, as in our baseline system.
- hybrid MLP/HMMs.

The rationale behind these experiments across different acoustic modeling approaches was to pinpoint whether the differences in performance of the features were partly due to problems of matching with the acoustic modeling.

Experiments with SCHMMs

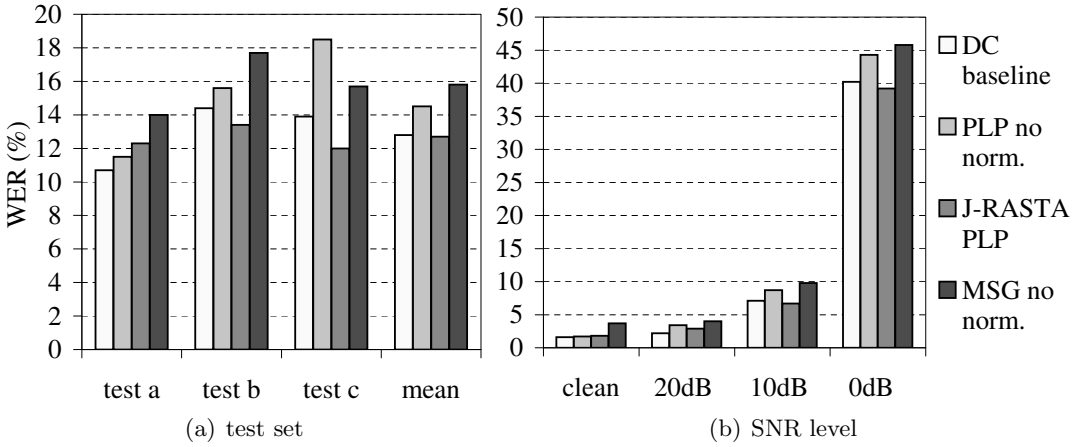
The acoustic modeling used in this set of experiments is the same as in our baseline LDA-based ASR system (cf. Sec. 7.3.1). However, some minor changes in the LDA-matrix computation have been carried out to adapt it to the different sizes of the feature vectors. For the PLP and J-RASTA features the size of the context window of the LDA matrix was 9 frames as in our baseline, but for the MSG features we reduced this size to 5 frames due to the large size of the MSG vector. If we had used a 9 frame window, the size of the covariance matrices of the supervised code-book would have been 252×252, which is difficult to train reliably given the small amount of training data in the AURORA database (cf. Sec. 2.6).

In a first set of experiments, we compared the features in their original non-normalized form.

In Fig. 7.4 we can observe:

- The only competitive approach to our current baseline feature extraction is the JRASTA approach. The other approaches (PLP and MSG) fall far behind our baseline. This could be due to the fact that both our baseline and JRASTA perform some kind of normalization or filtering on the feature vector, which makes them less sensitive to the noises and distortions.
- JRASTA performance is significantly better for low or very low SNR (c.i. $\pm 0.3\%$ and $\pm 0.7\%$, respectively), and in mismatched conditions (test b and test c with c.i. $\pm 0.3\%$ for both sets), but worse in matched conditions (test a with c.i. $\pm 0.3\%$ as well) and high SNR levels (c.i. $\pm 0.1\%$ and $\pm 0.2\%$ for clean and SNR 20 dB, respectively). The reason for that is that the spectral subtraction and CMS approaches of our baseline use silence detection, that partly relies on the assumption that the energy of the speech signal is larger than that of the noise signal. This silence detection is used to compute the mean noise spectrum and mean cepstral vector , respectively. If the

Further insight into the topic can be found in [Kin98, MN98, HM94]



(a) test set (b) SNR level

Fig. 7.4: Results of different feature extraction algorithms using SCHMMs. The full results can be found in Appendix A in tables Tab. A.1, Tab. A.2, Tab. A.3 and Tab. A.4.

noise energy is comparable to that of the speech (very low SNR), then the spectral subtraction may adapt the estimate of the noise spectrum in speech segments and the CMS probably computes the mean of the cepstrum in non-speech segments, which reduces the effectiveness of both approaches against noise and convolutional distortions, respectively. On the other hand, the JRASTA algorithm just filters out the spectral components of the input signal which vary slowly in time, e.g. channel response, irrespective of the relation between noise and signal energy. This filtering cancels out any signal components in the low modulation frequency region, without making a distinction between noise and speech components. Consequently, the performance of the approach is relatively bad in matched test conditions. In unmatched conditions, however, this filtering is beneficial, since the noises in the training and test sets are different. Moreover, since the approach does not rely on the energy of the input signal, its performance in very low SNRs is clearly better than that of our baseline.

- The MSG features perform rather poorly, which is somehow misleading, in view of the excellent results of these features reported in [Kin98] on a similar digit task. However, the acoustic modeling approach used in that work was completely different (hybrid MLP/HMM) which may be crucial to obtaining good results with these features. Later in this section we further investigate this fact by training hybrid MLP/HMM system on the MSG features.

As derived from the previous experiments, the normalization or filtering of the feature vectors can be of some advantage in noise. In fact, it has been shown in the past that the effect of noise on the feature vectors is a shift in the mean value and a change in the variance of the coefficients [MRGS98]. Therefore, it seems logical to devise some kind of mechanism to compensate for both phenomena. This can be achieved by normalizing the components of the feature vectors to approximately have null mean and unit variance. The values of the mean and the variance can be basically computed in two ways: in off-line modus or in on-line modus. If mean and variance are computed off-line, then the contents of each file must in advance be read to compute the mean over all the frames in the file.

This adds, at least, a factor of 1 to the Real Time Factor (RTF) defined in Sec. 3.3, and consequently the efficiency of our algorithm decreases. Conversely, if the mean and variance are computed on-line, the mean and variance are adapted each time a new frame is read, which does not significantly increase the RTF. The price we pay is that the mean and variance of the normalized features are no longer exactly 0 and 1, although this fact is not very important as long as the values are near 0 and 1.

The normalization we have used is simply described with the following set of equations:

$$\begin{aligned} \mu_i^{n+1} &= \mu_i^n + a\,(x_i^n - \mu_i^n) \\ \sigma_i^{n+1} &= \sigma_i^n + b\,((x_i^n - \mu_i^n)^2 - \sigma_i^n) \\ \bar{y}_i^n &= \frac{x_i^n - \mu_i^n}{\sqrt{\sigma_i^n}} \quad , \quad \forall i, n \end{aligned} \tag{7.2}$$

In our experiments we used $a = b = 0.005$ [TH97, HES00], so that the mean and variance were adapted quite slowly. The initial values for the means μ_i and variances σ_i were computed by averaging over all the files in the multi-condition training set. These initialization values are read for each file in the test set, and are not updated after each file has been processed, that is, the initial values of mean and variance were equal for all the files in the test set. This is clearly different from the adaptation of the cepstral mean used in our baseline system, since in our ASR system the initialization values are different for each file, because they depend on the previously read files. This last adaptation allows for a better adaptation to the particular environmental conditions of a test file, if the files previous to it have similar environmental conditions. However, if this condition is not met, then a mean and variance adapted on different environmental conditions would be used to normalize the feature vectors, which could distort them, especially if the adaptation constants a and b are small. This situation occurs, for example, when we want to train an MLP for a hybrid MLP/HMM ASR system with a set of speech files. The files are usually listed in random order to avoid local minima and adaptation to a particular speaker or noise condition, which may lead to slow convergence of the net weights. If these speech files include a number of different noises and distortions, as in the AURORA database case, then we have exactly the problem previously described.

A possibility to circumvent this problem is to use the normalization used in this point, i.e. to use the same initial mean and variance vectors for all the files in the training set. Because of its simplicity, we have used this normalization in all our experiments with neural nets, and in the experiments in this point, although in this last case we could have used a normalization analogous to our baseline system. A clear disadvantage of this simple normalization is that the adaptation to the test conditions is rather poor, especially if the adaptation constants a and b are small. Further discussion about this question can be found in Chapter 8.

The normalization in Eq. 7.2 can also be interpreted as a filtering. This filtering is non-linear, since the denominator term is changing with the signal as well. Nonetheless, if we ignore this fact and assume that the standard deviation is approximately constant, we have a linear IIR filter, with the following z-transform:

$$\bar{X}_i(z^{-1}) = \frac{1}{\sigma_i}\,\frac{1 - z^{-1}}{1 - (1-a)z^{-1}}\,X_i(z^{-1}) \tag{7.3}$$

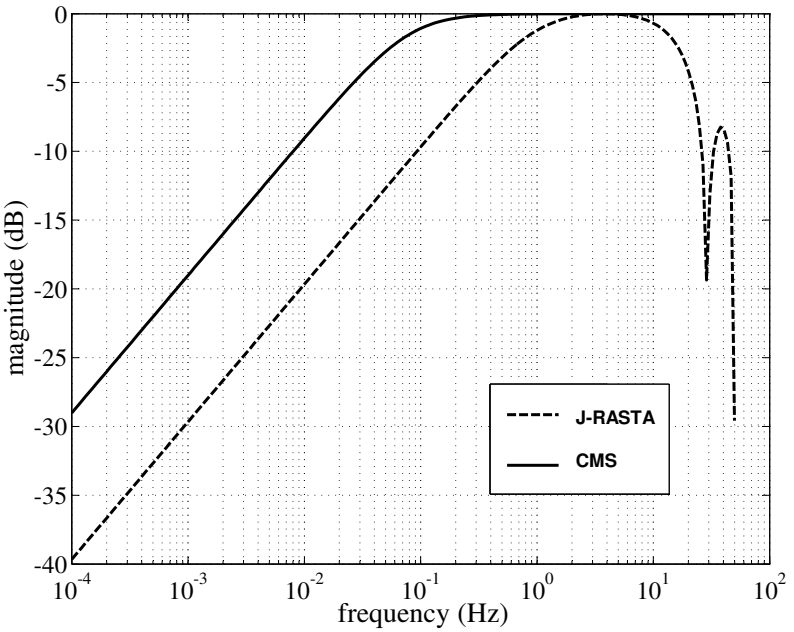
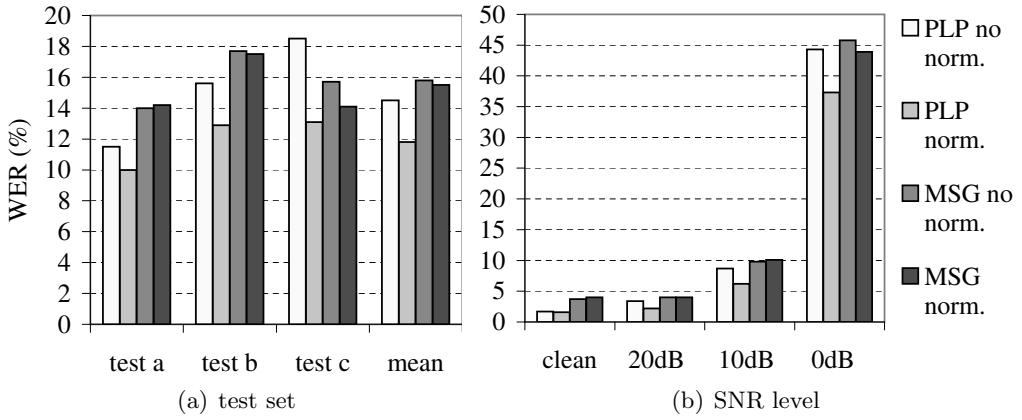


Fig. 7.5: Responses in the modulation frequency domain of the JRASTA (dashed line) and the mean subtraction (solid line) filters. Note the higher attenuation (approx. 10 dB) of the JRASTA filter in the low frequency region, and also of the frequencies above 20 Hz.

If we compare in Fig. 7.5 the response of this filter in the modulation frequency domain with that of the JRASTA filter, we can see that the attenuation of the JRASTA filter in the low-frequency region is about 10 dB better than that of our normalization. In addition, the JRASTA filter attenuates the frequencies above 20 Hz, which may further reduce any interference in this frequency region, that is, any time-varying components of the interfering noise that vary with a period shorter than 0.05s.

In Fig. 7.6 we compare the effects of normalization on the performance of the different features. There we can observe:

- After normalization the performance of the PLP features increases notably (on average 2.8% absolute). The improvement is noticeable for each test set, but it is particularly large in the test b and test c sets, i.e. in the unmatched test conditions. In clean conditions the performance of non-normalized and normalized PLPs is identical, but in noisy conditions the normalized PLPs perform clearly better. This confirms our assumption that normalization contributes greatly to the performance in noise of feature extraction algorithms, especially in low SNR environments. Interestingly, the normalization does not reduce the performance in clean speech, and it even increases the performance in matched test conditions. The performance increase is consistently better across all noises in the test sets (cf. Tab. A.2 and Tab. A.5 in Appendix A), which suggests that the normalization is beneficial for all kind of noises. These results entirely agree with those presented in [TH97], where the effect of mean and variance normalization on the PLP features was analyzed. The authors also compare the performance of mean normalization with mean and variance normalization, and show that the latter is especially beneficial in very low



(a) test set (b) SNR level

Fig. 7.6: Comparison of the results of normalized vs. non-normalized feature vectors using SCHMMs. The detailed results can be found in Appendix A in Tab. A.2, Tab. A.4, Tab. A.5 and Tab. A.6.

SNR conditions.

- Compared to JRASTA, the normalized PLP features are superior in matched test conditions (test a) but are just slightly better in unmatched (test b). In contrast, their performance is worse than JRASTA when the input speech has been distorted by a channel response (test c). This result suggests that the temporal filter used in the JRASTA algorithm is more effective in eliminating the undesired distortion than the normalization in mean and variance, which is a logical conclusion in view of Fig. 7.5. Therefore, the added attenuation (approx. 10 dB) of the JRASTA features in the low-frequency region is clearly beneficial in channel-distorted speech. Moreover, street and airport noises in the test b set (noises N2 and N3 in the tables of Appendix A) are fairly non-stationary [HP00], which may explain why the results of JRASTA are better than those of the normalized PLP features for both noise conditions (cf. Tab. A.3 and Tab. A.5 in Appendix A), since the JRASTA filter eliminates any component varying with a period shorter than 0.05s.

- Normalization of the MSG features has no great effect on their performance, except for the test c where it clearly improves. These results are logical in view of Fig. B.2 in Appendix B, where we see that the MSG features have been already normalized in the feature extraction process.

Experiments with Hybrid MLP/HMM

As discussed in Chapter 5, the hybrid MLP/HMM approach uses an MLP to compute the posterior probabilities of the HMM states. In the past, this approach has consistently shown its superiority or at least its competitiveness with the usual CDHMM approach based on Gaussian mixture models.

The PLP and MSG features used in these experiments are exactly the same as in the experiments with SCHMM in the previous section. The MFCCs used in this section, on the contrary, differ from those used in the previous, since no CMS has been applied to

them. In other words, the MFCCs computation is identical to our baseline, but without subtraction of the cepstral mean.

All input feature sets have been normalized as in the previous section, i.e. to have zero mean and variance. However, the main reason in this case is to simplify the initialization of the weights between the input and the hidden layers (cf. Sec. 5.5.2). As we have seen in the previous section, this normalization has a great impact on the performance of a given feature extraction in noise. As a consequence, any improvement observed in the hybrid MLP/HMM experiments cannot be solely attributed to the modeling capabilities of the MLP. Note also that this normalization is actually performing a kind of CMS on the MFCC features (plus a variance normalization, of course), and they are therefore not so different to the features used in our baseline system.

The MLPs used in the experiments have the following topology:

- One large hidden layer, and two layers of weights. This topology is depicted in Fig. 4.5 of Chapter 4, and is known to be able to approximate any decision boundary if the number of units in the hidden layer is sufficiently large [HSW89].

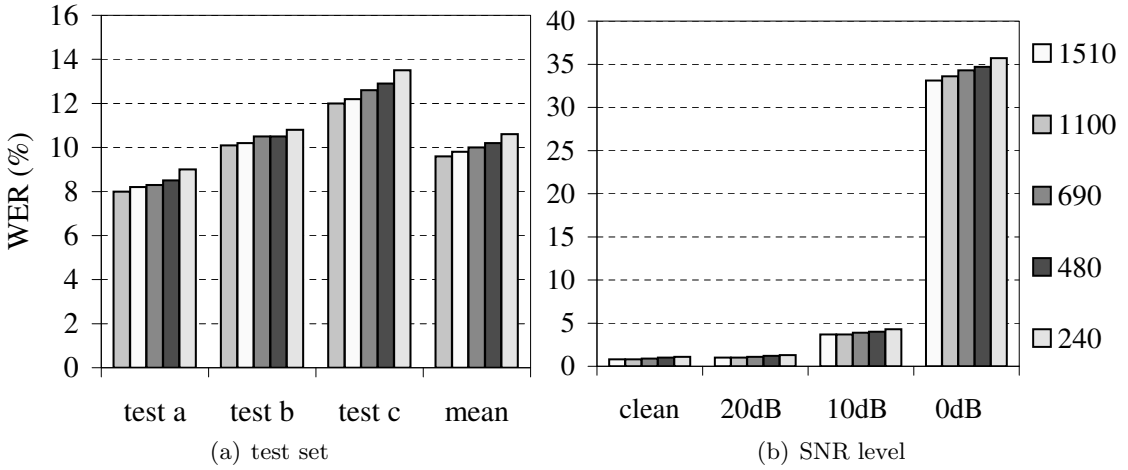
- The 1st and 2nd derivatives are appended to each PLP and MFCC feature frame, and a context window of 9 extended frames is appended together to form the input vector of the MLP. This resulted in a vector of 351 components for the MFCCs and PLPs. In the MSG case the frame already contains context information, so that it is no longer necessary to append 1st and 2nd derivatives. The extended vector of 9 context frames has 252 components in the MSG case.

- The units in the hidden layer use sigmoidal non-linearities, and the units in the output layer use softmax non-linearities.

- The targets of the MLP are the 127 states of the whole-word HMMs. To define the target output vector for each input data vector, we used the segmentation into HMM states of the training data generated to compute the supervised code-book (cf. Sec. 7.3.1). The states were then coded in a 1-of-c fashion (cf. Sec. 5.5.1) to generate the final target vectors.

- The training criterion to approximate the 1-of-c coded targets is the minimum cross-entropy (MCE) criterion introduced in Sec. 5.5.1.

- The MLPs are trained using the QuickNet toolkit of the International Computer Science Institute (ICSI) at Berkeley. The package implements very efficiently the error back-propagation algorithm commonly used to train MLPs. To avoid over-fitting of the net weights to the training data, the package uses early stopping. This technique is implemented using cross-validation, which splits the training set in two sub-sets, the larger to train the weights and the smaller to estimate the classification error (validation error) after each training epoch (cf. Sec. 5.5.2). To improve the efficiency of EBP training and avoid oscillations of the algorithm around a minimum point, the QuickNet package uses the technique already explained in Sec. 5.5.2. The start learn-rate used in our experiments was 0.008. The net weights are adapted neither in a sequential nor in a batch way, but rather in 'bunch' way (cf. Sec. 5.5.1). This means that the global error function is actualized after a bunch of 16 frames



(a) test set (b) SNR level

Fig. 7.7: Results of a hybrid MLP/HMM system using MFCC features. Performance for different sizes of the hidden layer of the neural net is measured using the WER. More detailed results can be found in Appendix A in Tab. A.7, Tab. A.8, Tab. A.9, Tab. A.11 and Tab. A.10.

has been read, and the difference between the actual and the previous 'bunch' error is then used to actualize the net weights.

In a first set of experiments, we study the dependence between size of the MLP and performance of the hybrid MLP/HMM system. These results are also interesting for our feature reduction experiments in Sec. 7.3.3, because a similar MLP topology is used there to reduce the dimensionality of the feature vector.

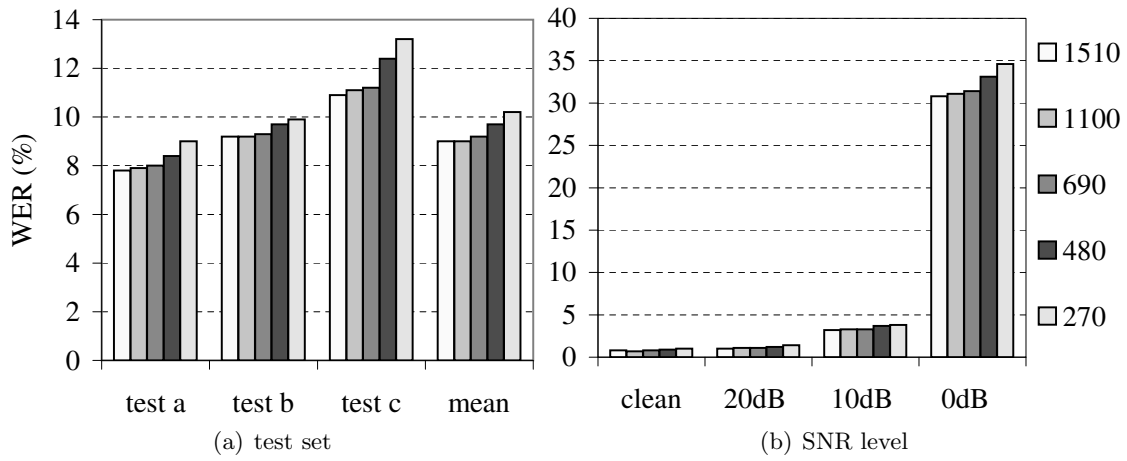
In the results shown in Fig. 7.7 and in Fig. 7.8 we can observe:

- Average WER improves only slowly with linearly growing number of hidden layer units, i.e. with the size of the MLP, for both feature sets. This agrees with the results presented in [EM99], where the dependence between MLP size, training data size and recognition accuracy is studied. In this work the authors show that the recognition accuracy of hybrid MLP/HMM systems grows slowly with the size of the hidden layer, when the amount of training data available is restricted.

- The improvement with growing size of the MLP is significant for very low SNR, but almost not noticeable for clean speech or high SNR. This observation makes sense because the multi-condition training set does not include files with SNR of 0 dB. Consequently, the complexity (size) of the MLP must grow to improve *generalization* to noise levels unseen during training.

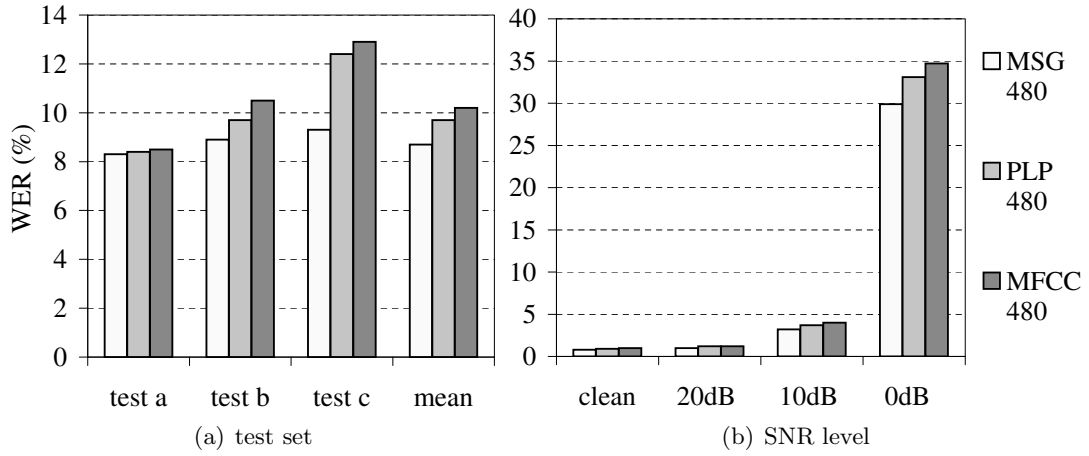
Since the training time grows linearly with the size of the net, a good compromise between training time and performance is to choose a hidden layer size of 480 units. This is the default size of the hidden layer that is used henceforth in our experiments with MLPs.

In a second set of experiments, we compared the performance of the MFCC, PLP and MSG features, using the 'optimum' size for the hidden layer found in the previous set of experiments. The results of these experiments are shown in Fig. 7.9 where we can observe:



(a) test set (b) SNR level

Fig. 7.8: Results of an hybrid MLP/HMM system with PLP input features. Performance for different sizes of the hidden layer of the neural net is measured using the WER. Detailed results can be found in Appendix A in Tab. A.12, Tab. A.13, Tab. A.14 Tab. A.16 and Tab. A.15.



(a) test set (b) SNR level

Fig. 7.9: Comparison of the MSG,MFCC and PLP features using a word-based hybrid MLP/HMM acoustic model. The size of the hidden layer of the neural net (480 units) is the same in all experiments. Detailed results can be found in Appendix A in Tab. A.17, Tab. A.16 and Tab. A.10.

- In contrast to the experiments with SCHMM, the MSG features are the best performing for any of the test sets and SNR levels. Compared to the results using SCHMMs, the improvement in performance of the MSG features is striking (almost 7% absolute average improvement) , which clearly indicates that our SCHMM acoustic modeling has difficulties to model the distribution of the MSG features. This result agrees with previous results in [Ell99] using CDHMM acoustic modeling with mixtures of diagonal covariance matrices. Since our SCHMM acoustic modeling uses full-covariance matrices, it should be in principle able to model the high correlation between the MSG coefficients. Therefore, the problem of the MSG features with our acoustic modeling must be their highly non-normal nature.

- The three features have a similar performance in clean or high SNR speech, but differ for low to very low SNRs.

- The features have similar performance in matched conditions (test a). In unmatched test conditions, however, their performance differs significantly. Note the excellent performance of the MSG features on the test c, which suggests a high degree of robustness to convolutional distortions [Kin98].

- The PLP features perform clearly better than MFCC also in this case, especially on the test b set. This agrees with the previous results obtained using SCHMMs, and is actually surprising since the MFCC features used incorporate spectral subtraction in the algorithm, which should be more effective in noisy conditions than the perceptually-inspired PLP features. However, a comprehensive comparative study of the PLP and MFCC features publicized in [PMP01] shows that the performance of normalized PLP features is superior to normalized MFCCs (without spectral subtraction) on a telephone task. This seems to confirm the validity of our conclusions, although our results with MFCCs are still misleading, and seem to cast doubt on the effectiveness of the spectral subtraction algorithm implemented in our ASR system.

- Comparing the performance of the hybrid ASR system with that of our baseline ASR system, we find that the performance of the former is clearly superior, at least on this simple digit recognition task. For instance, the performance of the PLP features is 2% absolute better using hybrid MLP/HMM acoustic modeling. This difference can only be attributed to the outstanding modeling properties of the MLP, since the PLP features used are exactly the same for both acoustic modeling types. The performance improvement is particularly large for the SNR conditions 'seen' by the MLP during training, i.e. for clean, 20 dB and 10 dB, for which the relative performance improvement is over 40%. Incidentally, the improvement for the unmatched test case (test b) is much larger than for the matched test case (test a), although the noises in the test b set were not 'seen' by the MLP during training. This observation, together with the previous one, seem to suggest that the level of the noises present during training is more important for the final performance than their actual spectral shape. Nevertheless, it can also be that this fact is exclusive to the AURORA 2000 database, since the noises in it have similar long-term spectral characteristics [HP00].

Summary

In this section we have compared the performance of different feature extraction algorithms on the AURORA 2000 database. Tests have been carried out using two different kinds of acoustic modeling approaches: SCHMMs and hybrid MLP/HMMs. For both approaches, we have found that the normalized (in mean and variance) PLP features are the best performing on the task, with on average 1% absolute better performance than the other features. Further, we have found that normalization can improve the performance of feature extraction, especially in the PLP case with a 3% absolute improvement. Compared to the performance of the same features using SCHMM acoustic modeling, the performance of hybrid MLP/HMM is roughly 2% absolute better.

7.3.3 Discriminative Feature Reduction

In this section we present and discuss the experiments and results related to Chapter 4, where the mappings for discriminative feature reduction are discussed. The tested feature reduction approaches are the original tandem, the tandem clustering-of-states, the tandem with PCA, and the non-linear discriminant analysis (NLDA) approaches.

In all the experiments of this section, we have used the PLP features as input features to the feature reduction transforms. The configuration of the PLP feature extraction is exactly the same as in the experiments of the previous section.

Tandem Experiments

The topology of the MLP used to reduce the dimensionality has one hidden-layer with two layers of weights as shown in Fig. 4.5. As in the experiments with hybrid MLP/HMM in Sec. 7.3.2, the input vector of the MLP is formed by appending delta and delta-delta features to the static PLP vector, and taking a window of 9 consecutive extended feature vectors to form an input vector of 351 components ($9 \times 3 \times 13$). The hidden layer size is chosen to be 480 units, because we have already obtained a satisfactory performance in hybrid MLP/HMM with this size. The nonlinearity in the hidden units was a sigmoid. The targets of the MLP were the English phonemes present in the digit corpus, which resulted in a total of 24 output units each associated with a phoneme. The units in the output used a softmax nonlinearity. To train the MLP, we used the same toolkit and configuration as in the hybrid MLP/HMM case. As before, the input features were normalized to have null mean and unit variance. The phonetic segmentation of the multi-condition training of AURORA 2000 set was obtained from ICSI, and is therefore the same as the one used in [SEK$^+$00, EG01]. The tandem features were obtained from the pre-nonlinearity outputs of the MLP, for the reasons already given in Sec. 4.6.1. As opposed to previous implementations of the tandem approach, no decorrelation (PCA transform) of the MLP outputs was necessary, because the normal densities in our code-books are full-covariance matrices, i.e. they can model the correlation between the tandem coefficients.

In a first set of experiments, we generated the tandem features using the MLP described above, and we used them to train two different acoustic models. The first is analogous to the unsupervised code-book in our baseline system (cf. Sec. 7.3.1 and Fig. 7.1), whereas the second is similar to the LDA-based supervised code-book of our baseline. For the first we applied the LBG algorithm to the tandem features, in order to obtain an unsupervised

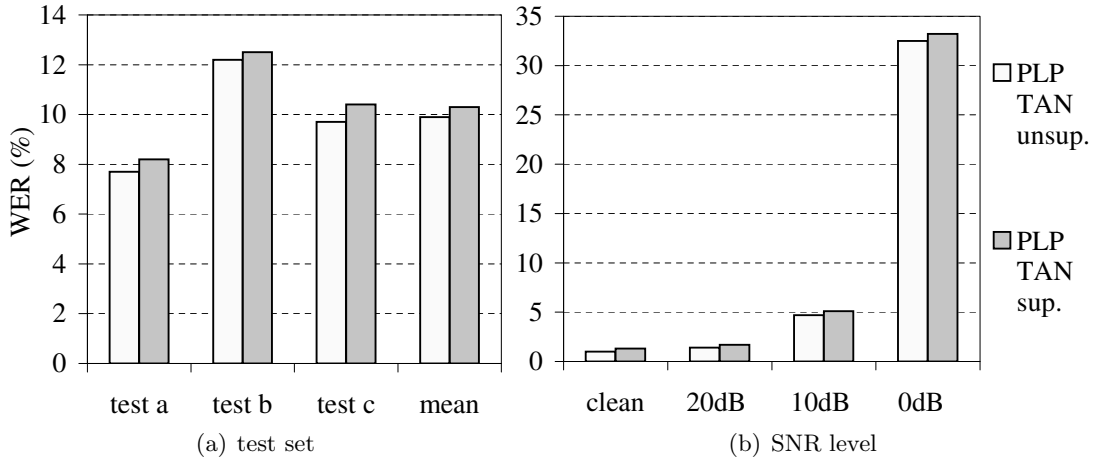


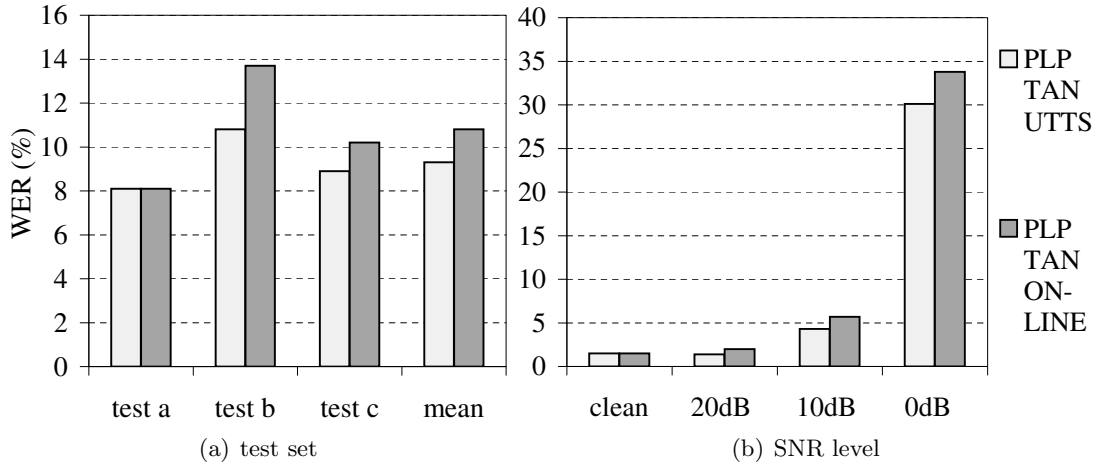
Fig. 7.10: Comparison of supervised and unsupervised code-books using tandem with PLP features. More detailed results can be found in Tab. A.18 and Tab. A.19 of Appendix A.

code-book of 512 classes. Next this was used to quantize the tandem feature vectors, just as in our baseline system, and the quantized vectors were then used to train the emission probabilities of the HMM states using the Baum-Welch algorithm. For the second acoustic modeling, we used the segmentation into states obtained in our baseline training to estimate a supervised code-book of 127 classes, each associated with an HMM state. This code-book was directly computed on the 24-dimensional tandem features, which are already dimensionality-reduced features analogous to 32-dimensional LDA vectors of our baseline. Subsequently, we used the supervised code-book to quantize the tandem features, and the quantized features were used to train, using once again the Baum-Welch algorithm, the emission probabilities of the states (see Fig. 7.2).

In Fig. 7.10 we show the results of both systems. As can be seen in the figure, the unsupervised code-book is consistently better than the supervised, across test sets and SNRs, although the difference is not large. Moreover, the unsupervised code-book has about 4 times more parameters than the supervised, which certainly contributes to the performance differences.

If we compare the previous results with those in [EG01], we see that the results on test a and test b are similar to ours, whereas on test b our results are worse. An important difference with the results in the previous article is the kind of acoustic modeling used. While the previous authors use CDHMMs, we use in our experiments SCHMMs. Another difference with respect to the mentioned authors could be the kind of normalization applied to the input features of the MLP. In our MLP experiments, we have made use of on-line normalization (cf. Sec. 7.3.2), but we believe that the previous authors have used per-utterance feature normalization. This last kind of normalization may improve the performance of the tandem system, especially in mismatched test conditions (test b). To confirm this hypothesis, we have performed an experiment using per-utterance normalization in a PLP tandem system.

The results of the same tandem system with per-utterance and with on-line normalized features are shown in Fig. 7.11. The acoustic modeling used in this comparison is the same



(a) test set (b) SNR level

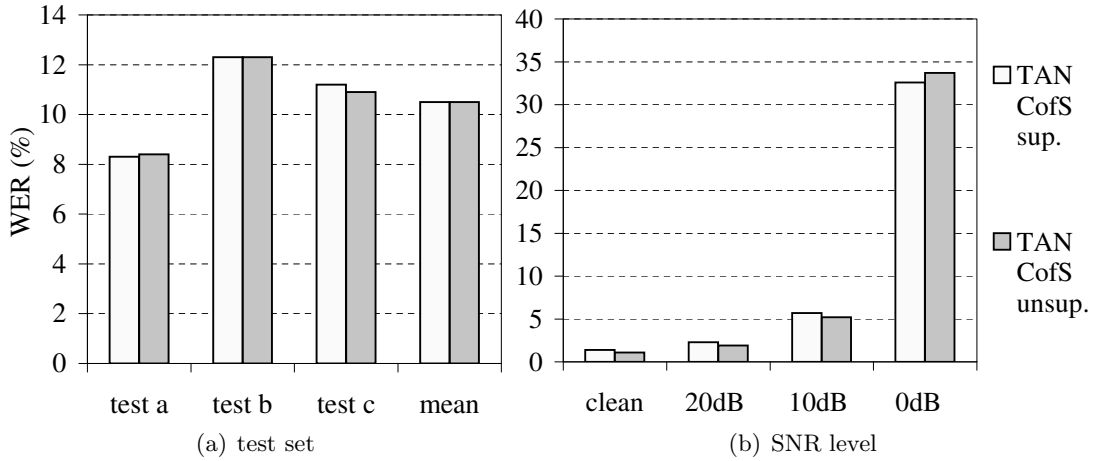
Fig. 7.11: Comparison of per utterance and on-line normalization of PLP features using tandem with unsupervised clustering acoustic modeling. More details about these results can be found in Appendix A in Tab. A.20 and Tab. A.21.

as the unsupervised code-book of 512 classes in the previous figure [2]. As can be seen in the figure, the differences in matched test conditions (test a) are not significant, but they are quite large for unmatched test conditions (test b and test c). In the test b case the difference is almost 3% absolute (21% relative). As already argued in Sec. 7.3.2, we think that the problem is rather the way the on-line normalization is applied to the feature vectors. For each test file, the means and the variances (computed over the whole training set) of the coefficients are read, and adapted according to Eq. 7.2. If the values of a and b in the equations are small (as is the case in our experiments), the means and variances are adapted very slowly to the test conditions. Since for each new test file the initial means and variances are read, which have been computed over the whole training set), the adaptation to conditions unseen during training is bad. This is not the case for per-utterance normalization, because the mean and variances are first computed over each test file, and then used to normalize the feature vectors in the file. Of course, per-utterance normalization cannot be used in real-time operation, because the whole sentence must be read before recognition begins.

Tandem Clustering of States Experiments

We present in this section the experiments with the tandem clustering-of-states, which uses clusters of states (cf. Sec. 4.4) to define the targets of the NN. As in the previous experiments, the input features are 13-dimensional PLPs, and the feature reduction transform is implemented using an MLP of one hidden-layer with 351 units ($9 \times 3 \times 13$) in the input layer and 480 units in the hidden layer. The difference with respect to the tandem approach are the targets of the MLP, which are no longer phonemes, but clusters of HMM states. The basis segmentation to train the MLP for these experiments is also the segmentation into HMM states used to train the supervised code-book of our baseline

[2] Note that results of the on-line-normalized system are different from the previous, because of a small bug in the decoding stage. However, this is not relevant for this experiment, since we only want to establish whether the per-utterance is better than on-line normalization



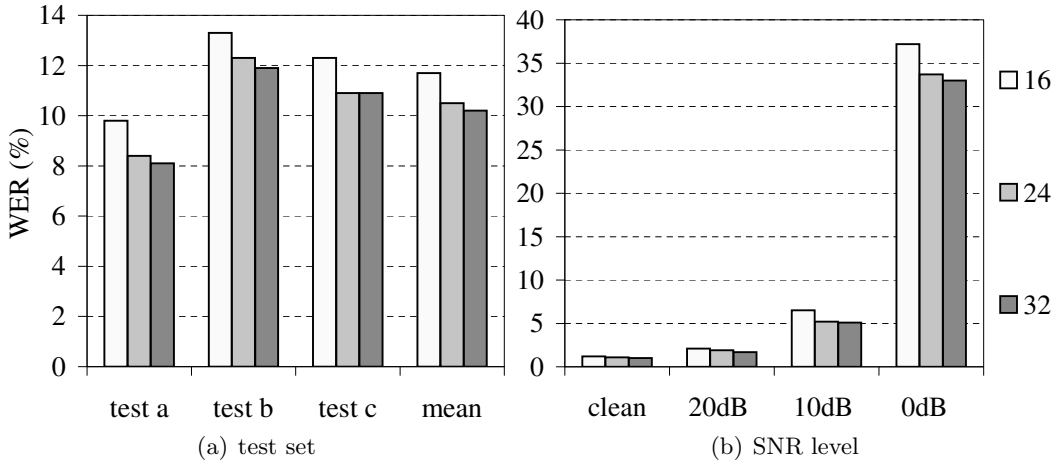
(a) test set (b) SNR level

Fig. 7.12: Comparison of supervised and unsupervised code-books using tandem-CofS with 24 clusters. More details can be found in Appendix A in Tab. A.22 and Tab. A.23.

system (cf. Sec. 7.3.1). To find the clusters of states we used the Lee clustering procedure explained in Appendix D. The segmentation into the desired clusters is found by using a cluster/state table to map the basis state segmentation into the cluster segmentation. Once this is obtained, we train the MLP using EBP as in the previous experiments. After MLP training, the tandem CofS features are obtained from the pre-nonlinearity outputs of the MLP, and are used, as in the tandem original approach, to train the SCHMMs.

As in the previous experiments, our first step was to compare the results of supervised and unsupervised SCHMMs using this feature reduction algorithm. The configuration of both kinds of acoustic models is the same as in the tandem experiments. In Fig. 7.12 we can see that the results of both kinds of SCHMMs are only marginally different across different test sets. Across different SNRs, the unsupervised SCHMM are slightly better for high SNR, whereas supervised SCHMM are better for low to very low SNRs.

In a second set of experiments with this approach, we tested the influence of the number of clusters on the performance of the approach. We clustered the HMM states in 16, 24 and 32 clusters, and modeled the resulting feature vector using an unsupervised code-book. In Fig. 7.13 we can see that performance decreases with decreasing number of clusters. However, the differences in performance between the systems using 24 and 32 clusters are not significant for any test set, and are only significantly different for very low SNRs. In contrast, the differences between 24 and 16 clusters are significant for all test sets and SNR levels, especially for low and very low SNR levels. It seems, therefore, that 24 is a kind of optimum value for the number of clusters, and that further increasing the number of clusters does not significantly improve the performance. Actually, this is logical if we think of the clustering process as an algorithm to find similar states in the HMMs of the words. These clusters of similar states should roughly correspond to the phonemes in the vocabulary, which explains why 24 clusters is a kind of optimum value, since the number of phonemes present in the digit corpus is also 24. The results shown in Fig. 7.13 entirely agree with the results of similar experiments in [SH03], where the results of different clustering approaches for the tandem CofS approach are compared.



(a) test set (b) SNR level

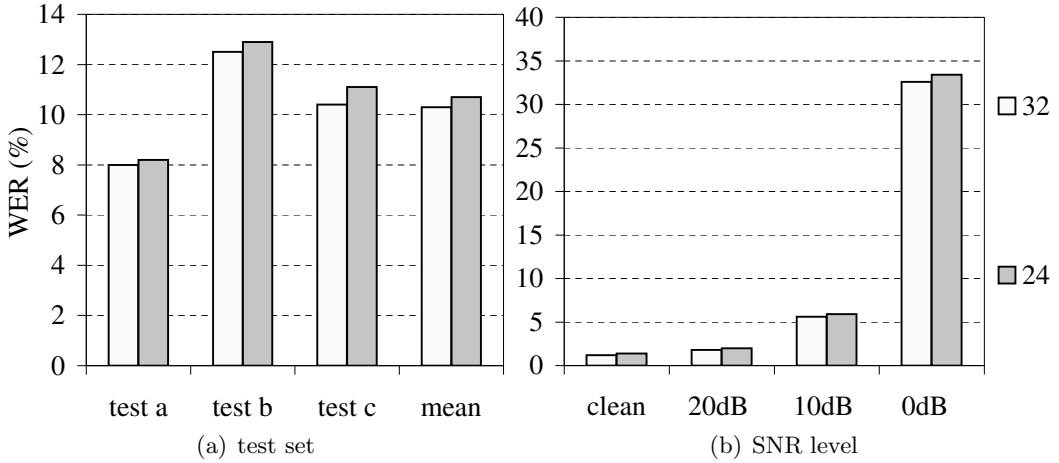
Fig. 7.13: Different number of clusters for tandem-CofS using an unsupervised code-book with 512 symbols. The detailed results can be found in Appendix A in Tab. A.24, Tab. A.23 and Tab. A.25.

Tandem with PCA Experiments

As suggested in Sec. 4.4 and by the authors in [ESS01, RHWR96], an alternative to clustering could be to train the NN using the HMM states as targets and then apply a linear feature reduction transform (PCA) to the high-dimensional output of the NN. However, this method is certainly not optimum in a theoretical sense. Nevertheless, we tried this approach on the AURORA 2000 database since in this case the training of the NN is not very time consuming. The NN topology was the same as in the previous points except for the number of output units which was set to the number of HMM states. The targets of the NN were consequently the 127 HMM states coded using 1-of-c coding. After training the NN in the same way as in the previous cases, the data in the training set was input to the MLP to generate a set of 127 dimensional vectors. This data set was then used to compute a PCA matrix to reduce the dimensionality to N components. Afterwards, this N-dimensional vector was used to train the supervised SCHMMs as in the previous tandem experiments. In Fig. 7.14 we can analyze the results of the described approach using a different number of PCA coefficients. As can be seen, the performance improvement with 32 PCA coefficients is small, and probably not statistically significant. This agrees with the previous results using clustering-of-states, and demonstrates that PCA is actually performing a kind of clustering as well, since in PCA analysis the principal axes of variation are found. These principal axes are actually determined by the components of the 127-dimensional vector (states) that are active at the same time. Each PCA coefficient is coupled to one of these directions, and quantifies how much of its principal direction is present in the actual 127-dimensional vector.

Non-Linear Discriminant Analysis Experiments

Finally, the last dimensionality reduction method used an NN with two hidden layers in a bottle-neck topology as shown in Fig. 4.6. As already mentioned in Sec. 4.6, the idea is to use an NN with two hidden layers to approximate the state posterior probabilities and to use the pre-nonlinearity outputs of the 2nd hidden layer as NLDA features. The NN



(a) test set (b) SNR level

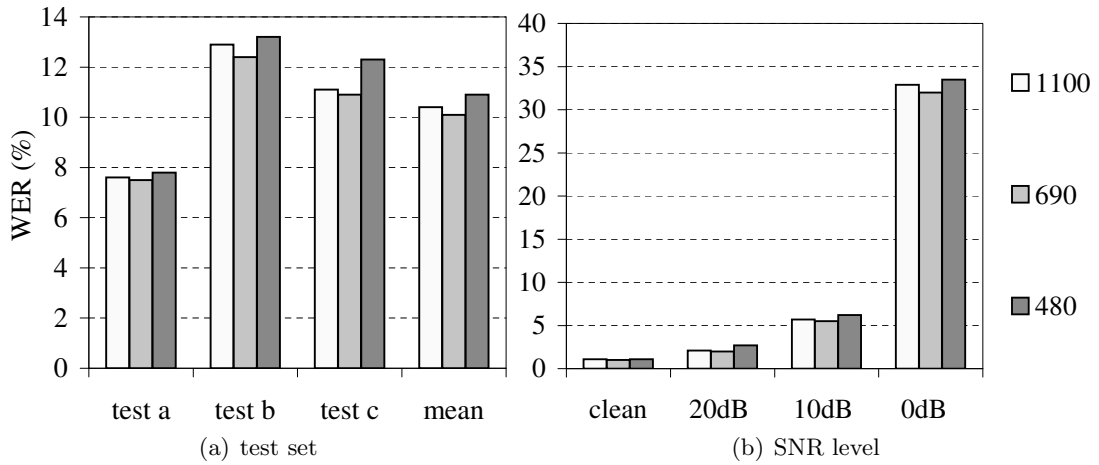
Fig. 7.14: Comparison of different size of PCA matrix (24 and 32) for tandem-PCA experiments using supervised SCHMMs. More detailed results can be found in Appendix A in Tab. A.26 and Tab. A.27.

targets of the MLP training were chosen to be the HMM states coded in the usual 1-of-c coding, and consequently the baseline segmentation into states was used as well. The units in the hidden layers were sigmoidal unit, whereas the units of the output layer used softmax non-linearity. The MLPs were trained using, as before, the EBP algorithm, with the techniques described in Sec. 5.5.2 to avoid over-fitting to the training data set. After training, the N-dimensional pre-nonlinearity outputs of the second hidden layer were used to train supervised SCHMMs as in our baseline system.

In a first set of experiments, we fixed the size of the 2nd hidden layer, and varied the size of the first to observe the changes in recognition performance. In Fig. 7.15 and Fig. 7.16 we can observe that:

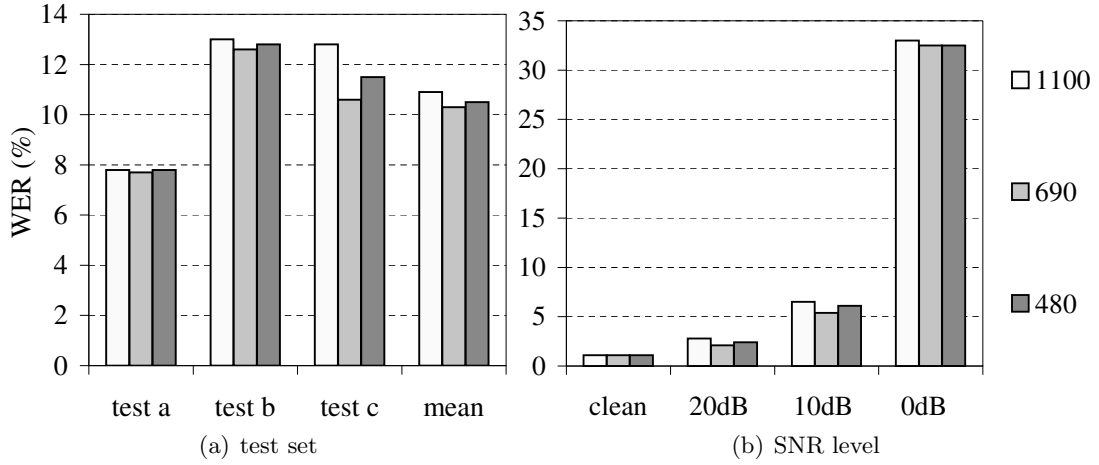
- The performance in both cases reaches its maximum at 690 units, although the average differences between different sizes of the first hidden layer is rather small. The only noticeable difference is for the test c conditions, where the difference between the best and the worst is about 2% absolute (17% relative), and for 10 dB SNR, where the previous difference is 1% (17% relative). In contrast to the experiments in Sec. 7.3.2, where the performance gradually increases with growing size of the MLP, the performance for this kind of MLP (two hidden layers) and for this kind of problem (dimensionality reduction) does not monotonically increase with the size of the MLP. However, it is hard to tell with this limited amount of experiments, if there is a clear trend or not in the evolution of the results. Since in most cases there is no significant difference between 480 and 690 units in the first hidden layer, we have decided to use 480 units in the first hidden layer in our further tests.

- Comparing the performance of both cases for equal size of the first hidden layer, we see that the performance difference is on average small and not significant. This agrees with the previous results of the other dimensionality reduction approaches, which also did not show any rapid increase or decrease in the performance as the size of the MLP changed. Moreover, similar experiments varying the size of the second hidden layer were conducted in [FRB97] using phonetic targets instead of



(a) test set (b) SNR level

Fig. 7.15: NLDA experiments with 32 units in the 2nd hidden layer, and a different number of units in the 1st hidden layer. More details about these results can be found in Appendix A in Tab. A.31, Tab. A.32 and Tab. A.33.



(a) test set (b) SNR level

Fig. 7.16: NLDA experiments with 24 units in the 2nd hidden layer and different number of units in the 1st hidden layer. More detailed results can be found in Appendix A in Tab. A.28, Tab. A.29 and Tab. A.30.

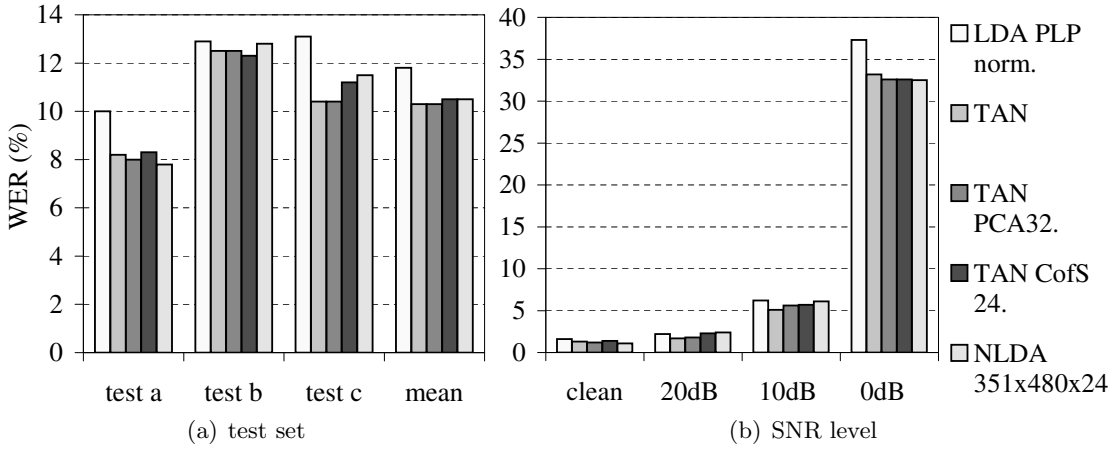


Fig. 7.17: Comparison of different feature reduction approaches using the same kind of features (PLP) and acoustic modeling (supervised training in Fig. 7.2). More detailed results in Appendix A in Tab. A.5, Tab. A.19, Tab. A.26 and Tab. A.22 and Tab. A.30.

HMM states. The relative differences in performance observed by the authors are similar to our results.

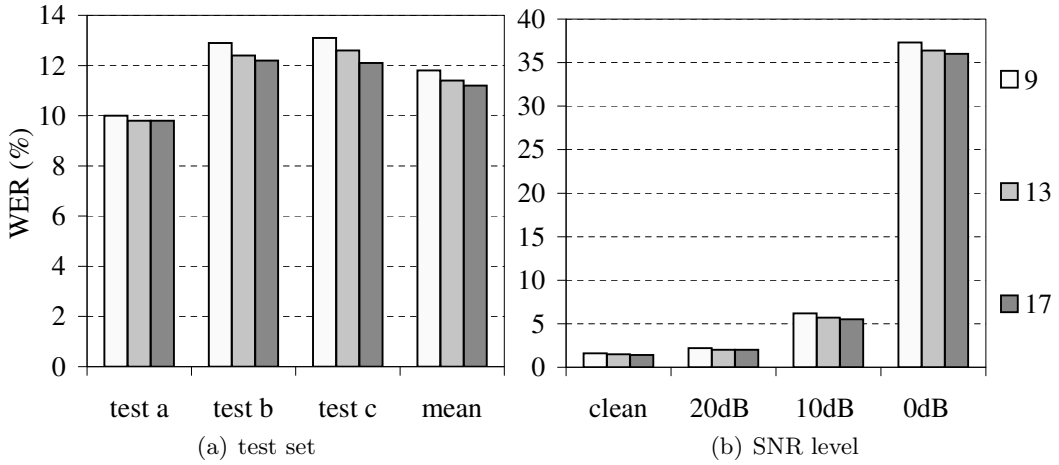
Comparison of Feature Reduction Approaches

In this section we compare the feature reduction approaches introduced in the previous section, and the LDA-based feature reduction using normalized PLP features. To have a high degree of comparability, all the compared systems use supervised SCHMMs trained as in our baseline system. In Fig. 7.17 we observe that:

- All non-linear dimensionality reduction approaches have a similar performance, and on average the differences in performance are not statistically significant. However, we see that the NLDA approach is more effective in matched test conditions than in unmatched. These results have also been reported in [MHC03], and seem to indicate that the different approaches are actually performing a similar mapping.

- Compared to the LDA-based system using normalized PLP features, the non-linear dimensionality reduction approaches consistently bring a performance improvement for all test sets and SNRs. This improvement is specially large for low to very low SNRs, and for the test sets test a and test c. These results agree with the previous results reported in [EG01, SEK$^+$00, HES00]. However, the reported improvements in those contributions were larger than ours because the tandem results were compared to the AURORA official baseline results [HP00], which used MFCC features without any kind of normalization or LDA.

Although the previous partial conclusion agrees with our theoretical expectations, it is also true that the improvement observed cannot be solely attributed to the improved discrimination capability, since the input high-dimensional spaces of the LDA and the non-linear approaches are different. In the LDA case, the context spans a time-window of 90 ms, whereas in the non-linear cases the context is much larger due to the appended delta and delta-delta vectors.



(a) test set (b) SNR level

Fig. 7.18: Results of the experiments with normalized PLP LDA-based recognizer using different sizes of the context window (9, 13 and 17 frames). More detailed results in Appendix A in Tab. A.5, Tab. A.34 and Tab. A.35.

To study the relevance of this difference, we increased the context window of LDA to 13 and 17 frames , respectively. This last number of frames is roughly equivalent to the context used in the non-linear approaches, since the length of the delta analysis window is 5 frames. As already mentioned, this is obtained by appending the delta and delta-delta coefficients to the static features to form an extended feature vector, and then combining 9 consecutive extended vectors into a high-dimensional vector. The reason why we do not use delta and delta-delta coefficients in LDA is to avoid numerical problems in the computation of the LDA matrix, since the delta and delta-delta coefficients introduce obscure linear dependencies between the coefficient of the high-dimensional vector, which may lead to non-invertible covariance matrices. In Fig. 7.18 we can observe that:

- The performance increases in average as the size of the context window increases, especially in the unmatched test cases (test b and test c). In the matched test case (test a) the improvement is not statistically significant (c.i. ± 0.2 at 95% level).
- If the results of the 17 frames case are compared with those of the non-linear transforms in Fig. 7.17, we see that the results of the non-linear approaches are clearly better in the test a and test c sets, but very similar in the test b case. In the test a case, the improvement is roughly 20% relative. In the test c set, the improvement is much larger for the noise present in the training data (N1) than for that not present (N2) (cf. Tab. A.19 and Tab. A.35), and the average improvement in this test set is about 15% relative. Differences between the non-linear approaches and LDA are only clear for very low SNR, where the non-linear approaches are clearly superior.

These observations suggests that non-linear dimensionality reduction is actually doing its job, i.e. mapping into a space with optimum discrimination between the classes, in matched test conditions, but it does not seem to work in unmatched test conditions. A reason could be the already mentioned feature normalization process (cf. Sec. 7.3.2), which does not guarantee a good adaptation to the test conditions in the unmatched test sets. It could well be that the non-parametric MLP-based approaches are more sensitive to

this problem than the parametric LDA approach. Alternatively, it could also be that the problem is the training procedure of the MLPs, since it seems to adapt excessively to the conditions of the training set. More about this point can be found in Chapter 8.

Summary

In this section we have tested different configurations of some non-linear dimensionality reduction approaches, and we have compared their performance to the usual LDA. Results show that the differences between the different non-linear approaches are on average not significant. Compared to the usual LDA approach, however, the non-linear approaches bring a significant performance improvement, although this improvement is mostly for very low SNR levels and matched test conditions. In unmatched noise conditions, the advantage of using non-linear feature reduction is not clear. This could be due to an overfitting of the MLP to the noises of the training set, which may be compounded by a poor normalization of the feature vectors. It seems, therefore, that the striking advantage of the tandem approach over our baseline found in Chapter 3, is not only due to the non-linear feature reduction, but rather due to a series of factors— PLP features, normalization in mean and variance, larger context window and non-linear transform— that add up to the excellent result obtained in Chapter 3.

7.3.4 Hybrid RBF/HMM Systems

In this section we present the results using hybrid RBF/HMM acoustic modeling as discussed in Chapter 5. The purpose of these experiments is to establish whether the discriminative training of the emission probabilities (weights between the hidden and the output layer of the RBF) improves recognition performance.

The RBF topology has been chosen to be:

- 32 input nodes, because the features at the input of the RBF are the output of a 32x117-dimensional LDA transformation matrix. As in the Feature Reduction Experiments with PLP features, a 9-frame time context window of 13- dimensional PLP feature vectors is used.

- 127 hidden nodes, because supervised clustering is used to compute the Gaussian basis functions in the hidden layer (cf. Sec. 5.5.1). This supervised clustering procedure is analogous to the one used in our LDA-based baseline system, and assigns a multidimensional full-covariance normal density of 117 components to each HMM state. As a consequence, the number of Gaussian basis functions is equal to the number of HMM states. As in our baseline, these high-dimensional densities are used to estimate the LDA matrix, and this again is used to reduce the dimensionality of the densities to 32 components. These 32-dimensional full-covariance normal densities are chosen to be the basis functions of our RBF. To compute this supervised code-book, we have used a different segmentation into states from that used in the PLP LDA-based system in Section 7.3.2, i.e. the segmentation used in our baseline LDA-based system. Instead, we have trained unsupervised SCHMMs, as in our baseline system in Section 7.3.2, using the PLP feature vectors. These unsupervised SCHMMs have been used in a forced alignment to obtain a segmentation into HMM states. This new segmentation is used to obtain the desired supervised code-book

as in the previous experiments. We divide the output of each basis function by the sum of the outputs of the other basis functions in order to have hidden layer outputs that sum up to 1. This normalization was reported to improve the results in [SL92], and is actually what the vector quantization of our baseline system is performing. Moreover, this normalization makes the initialization of the hidden layer weights easier, since it suffices to choose random values between 0 and 1.

- 127 output nodes, since this is the number of states in the HMM acoustic modeling. It has been shown in Sec. 5.5.1 that it is possible to approximate HMM state-posterior probabilities using this topology, either by using the Bayes or the softmax non-linearity in the output units. In both cases, to compute the posterior probabilities, the state-prior probabilities of the classes are needed. In all our experiments with hybrid RBF/HMM, it has been assumed that all the prior probabilities are equal.

Since the mean vectors and covariance matrices are found by a supervised clustering, the only parameters that have to be trained using the gradient descent algorithm are the mixture weights between hidden and output units, that is 127 * 127 = 16129 weights to train. The number of weights to train by gradient descent has thus been considerably reduced in comparison with the hybrid MLP/HMM system in the experiments of the previous sections. This results also in a considerable reduction of training time as compared to the hybrid MLP/HMM system. The weight estimation equations can be found in Sec. 5.5.1. The targets used in the gradient descent training were based on the segmentation into HMM states used in our LDA-based baseline system. Note that this is not the same segmentation as the one used to estimate the parameters of the basis functions, since this is based on unsupervised SCHMMs trained on the PLP features.

As already mentioned in the previous paragraphs, two variants of the previous topology have been tested. These variants use the following two different kinds of non-linearity in the output layer:

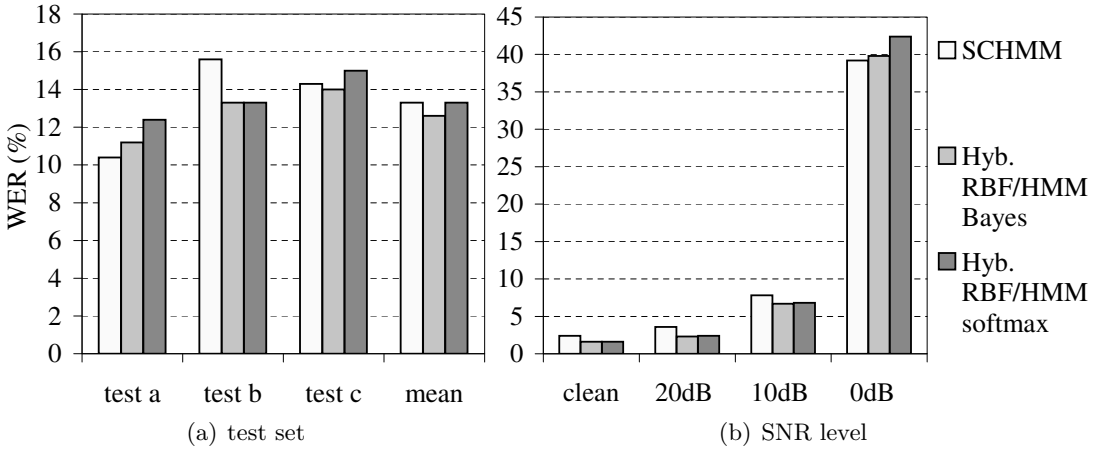
- A softmax non-linearity which converts the log-likelihoods at the input of the output layer into posterior probabilities. The advantage of assuming log-likelihoods at that point is that there is no need for constraints on the values of the weights w_{ij}, because the softmax non-linearity ensures that the outputs of the RBF are positive, and between 0 and 1. This fact greatly simplifies the derivation of the estimation equations, as can be seen in Sec. 5.5.1.

- A Bayes non-linearity with weight constraints that ensures that the RBF outputs are true posterior probabilities. To perform this experiment we used the same NN topology as in the previous experiment, but we forced the weights between hidden and output layer to be greater than zero and to sum up to one. As explained in Sec. 5.5.1, this was achieved by defining a set of 'dummy' weights that were passed through a softmax non-linearity to obtain the true constraint-compliant weights w_{ij}.

To learn more about both variants we refer to Section 5.5.1.

In Fig. 7.19 we show the results of the experiments with the hybrid RBF/HMMcan approach. Interesting points about these results are:

- Comparing both hybrid RBF/HMM variants, we observe that using a Bayes non-linearity and weight constraint is on average better than using a softmax non-linearity



(a) test set (b) SNR level

Fig. 7.19: Comparison of different results using Hybrid RBF/HMM with an SCHMM system using the same input PLP features. More details about these results can be found in Appendix A in Tab. A.36, Tab. A.37 and Tab. A.38.

in the output units. In particular, we see that the former approach is 10% relative better for the test a and 7% relative for the test c.

- Discriminative acoustic modeling is better for clean speech or mismatched noise conditions. In Tab. A.36, Tab. A.37 and Tab. A.38 of Appendix A we can find the detailed results where the previous trend can be better identified. The previous trend can also be observed in the results of test c, since the results for N2 (not present in training) are better than in the conventional approach, and conversely the results for N1 (present in training) are worse. This trend is also observed in the hybrid MLP/HMM results of Sec. 7.3.2, where the improvements for mismatched test conditions and SNR levels present in the training set were clearly larger (25% relative on average).

- For very low SNR (0 dB) and matched test conditions the discriminative approaches are worse than the conventional approach (see Tab. A.37 and Tab. A.38). For high SNR (20 dB) or low SNR (10 dB) conditions and matched test conditions, the results of both approaches are on average similar. This contrasts with the results of hybrid MLP/HMM, since the improvement using this approach, although not so large as in the unmatched case, was significant (16% relative on average).

It is difficult to tell why hybrid RBF/HMM is just a little better than a similar SCHMM system. Among the suspected causes we can mention:

- The segmentation used to estimate the supervised code-book is not the same as the one used to estimate the weights using EBP. This may have caused 'blurred' outputs because during EBP training the unit with maximum output in the hidden layer (hidden layer outputs are normalized to sum up to one) may not correspond to the unit in the output layer with the 'one'.

- In these experiments we have used the hybrid RBF/HMM approach which directly inputs the posteriors to the Viterbi decoding for posterior-based systems described

in Section 5.4.1. Alternatively, we can normalize the posteriors by the *a priori* probabilities of the states to obtain normalized likelihoods, and use these likelihoods and the state transition probabilities of the SCHMM system, to decode the sentence using the Viterbi algorithm for likelihood-based systems described in Section 2.5.1. Actually, in [RMB91] it has been argued that it is usually better in practice to normalize the posteriors by the state prior probabilities. As further discussed in Chapter 8, a first reason is that large differences in the prior probabilities of the states cause the infrequent states to be ignored during classification. Therefore, a normalization of the posteriors by the priors usually helps to reduce this effect. A second reason is simply that the state prior probabilities are usually different in the training and the test set. Since discriminative training implicitly includes the prior probabilities of the states in the training data, the use of non-normalized posteriors can lead to poor performance in certain test sets. Moreover, in [SL92] the normalization of the posteriors significantly improved the performance. However, in our case the posterior probabilities of the states in training and test are very similar (in both cases we have only sequences of digits).

- During EBP training we have assumed that the HMM states were equiprobable, i.e. equal prior probabilities, although the state in the pause model, must have a different prior to the states in the digit models. As further discussed in Chapter 8 it is possible to include the prior probabilities in the training by using a different constraint on the weights.

Summary

In the experiments of this section we have compared different topologies of the RBF for hybrid RBF/HMM. The topology using the Bayes non-linearity at the output performs best, although its performance is just slightly better than the performance of a similar SCHMM system.

7.3.5 Multiple Streams of Features

We have basically carried out two sets of experiments with the multi-stream approach presented in Chapter 6:

- Multi-stream together with feature reduction approaches and SCHMM.
- Multi-stream with hybrid MLP/HMM

The reason for applying multi-stream on different systems was to see if improvements were consistent across different kinds of acoustic modeling approaches.

Multi-stream with SCHMM Systems

In a first set of experiments, we have tested the effectiveness of the multi-stream approach when used together with LDA for dimensionality reduction. Three different kinds of combination have been tested, namely:

- concatenation of the LDA vectors, followed by feature reduction using LDA, as shown in Fig. 6.3.

- product of likelihoods as described in Sec. 6.3.2.

- full combination approach using the product of likelihoods rule, as described in Sec. 6.3.2 and shown in Fig. 6.5.

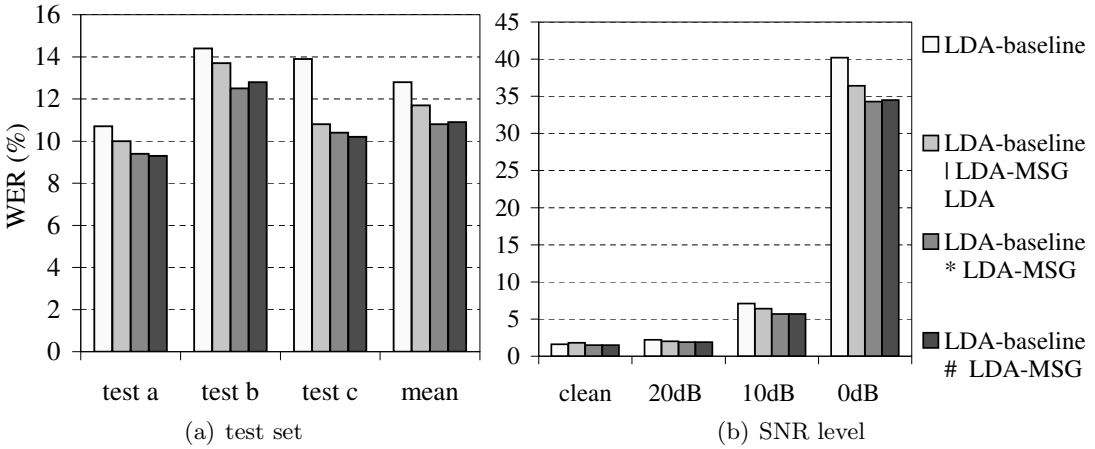
The systems we have combined are our LDA-based baseline system (cf. Fig. 7.4), and the LDA-based system using MSG features (cf. Fig. 7.6), both described in Sec. 7.3.2.

To train the SCHMMs for the concatenation of LDA vectors, we have first generated the LDA vectors for each of the files in the training set and systems to be combined. Next, for all the frames in the training set, we have concatenated the LDA vectors of both systems on a frame by frame basis to form a large 64-dimensional LDA vector. Some experiments were also carried out using a context window of concatenated LDA vectors (3 frames), but the results were almost the same as without any context. Consequently, we have used the 64-dimensional concatenated feature vector to generate a code-book using the baseline segmentation into HMM states and the same supervised clustering algorithm as in our baseline system. From this code-book, we have estimated the LDA transform that has been used to reduce the dimensionality of the concatenated vector from 64 to 32 dimensions. Next the 32-dimensional feature vectors are soft-quantized using the estimated supervised code-book, and the quantized vectors are used to estimate the emission probabilities of the HMM states as in our baseline system (cf. Sec. 7.3.1).

To train SCHMMs using the product of likelihoods rule, we have simply used the same procedure as in the 3 code-book system of our baseline (cf. Sec. 7.3.1), but with just 2 code-books. Hence, we have quantized the feature vectors using the code-books of each of the systems to be combined, and subsequently we have simply concatenated the quantized vectors of both code-books. Finally, the concatenated vector was used to estimate two sets of emission — one for each code-book — and the transition probabilities of the SCHMMs using the Baum-Welch algorithm. Since we have in this case 2 supervised code-books, the number of emission probabilities per state to estimate using Baum-Welch is 2×127. To obtain a single likelihood per state, we first compute the likelihood of each code-book/set of emission probabilities pair, and then we multiply the obtained likelihoods as shown in Eq. 7.1.

Similarly, to estimate the emission and transition probabilities of the full combination system using the product of likelihoods rule, we have used the LDA-baseline code-book, the LDA-MSG code-book and the code-book of the first multi-stream system (feature concatenation of LDA vectors followed by LDA dimensionality reduction) to obtain three sets of vector-quantized features. As before, we have concatenated the three quantized vectors, and we have used the concatenated vectors to train emission and transition probabilities using Baum-Welch. The number of emission probabilities to be trained in this case is $3 \times (127 \times 127)$. To obtain a single likelihood per state we use the same procedure as in the previous system. The results of the experiments are shown in Fig. 7.20 where we can observe:

- Feature concatenation followed by feature reduction with LDA (LDA-baseline | LDA-MSG LDA) significantly improves (cf. Fig. 3.5) the performance of the baseline (LDA-CMF) for all test sets, especially for the test c where the improvement is clearly over the significance threshold (c.i. $\pm 0.5\%$). For high SNR levels, the differences between this approach and the baseline are not significant, but for low SNR levels



(a) test set (b) SNR level

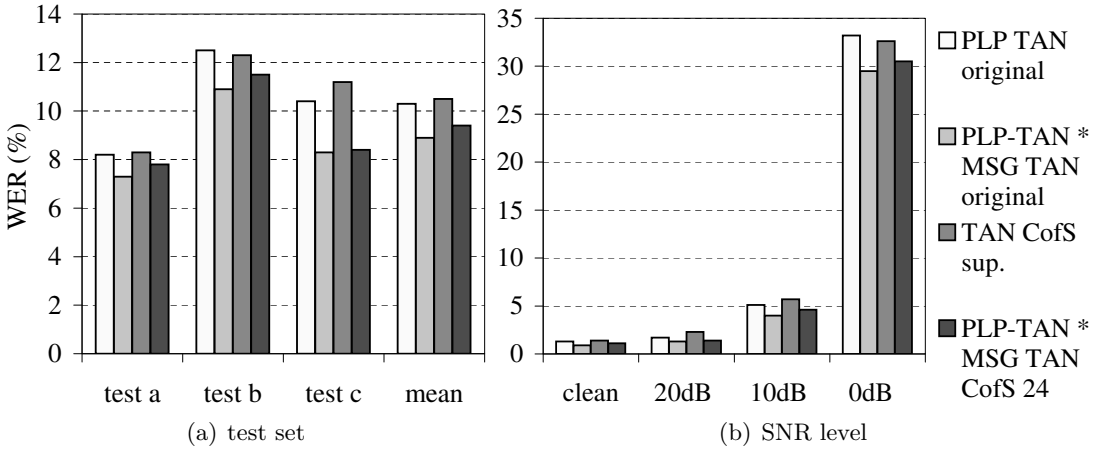
Fig. 7.20: Comparison of different multi-stream approaches using LDA dimensionality reduction and SCHMMs. The operator '$*$' stands for product of likelihoods, the '|' for feature concatenation (followed in this experiment by LDA reduction), and '#' for full combination using product of likelihoods. More detailed results can be found in Appendix A in Tab. A.1, Tab. A.41, Tab. A.40 and Tab. A.39.

the feature concatenation approach is significantly better, both for 10 dB SNR (c.i. $\pm 0.5\%$) and 0 dB SNR (c.i. $\pm 1.4\%$).

- Probability combination using the product rule (LDA-baseline $*$ LDA-MSG) is also better than our baseline for all test sets. Compared to the previous multi-stream approach, this approach is clearly better for the test b set, and just above the threshold of significance for the test a set. For the test c, in contrast, the difference is not statistically significant (c.i. $\pm 0.5\%$). As in the previous case, the differences for high SNR with the baseline are not statistically significant (c.i. $\pm 0.3\%$), but for low SNR the improvement with respect to the baseline and the previous approach is clear.

- On the other hand, the full combination approach using the product rule (LDA-baseline # LDA-MSG) does not bring any significant performance improvement over simple probability combination for any of the test sets or SNR levels. These results seem to suggest that our baseline features (mel-cepstral coefficients) and the MSG features have rather uncorrelated errors, and there is consequently no need to add a model of the concatenated feature vector.

In another set of experiments, we have tested the performance of the combination of multi-stream with non-linear feature reduction approaches. In a first experiment, we have combined two tandem systems (cf. Sec. 7.3.3) one using PLP features and the other MSG features. An MLP has been trained for each of the feature types, as explained in Sec. 7.3.3. Next the pre-nonlinearity outputs of the MLPs are added (pre-nonlinearity outputs are log-probabilities) to obtain a single dimensionality-reduced feature vector for each input frame. This feature vector is afterwards used to train a supervised code-book and the SCHMMs as already explained in Sec. 7.3.3. Two different non-linear dimensionality reduction approaches have been compared: the first is the tandem original approach, and the second is the tandem clustering-of-states approach (cf. Sec. 7.3.3). In Fig. 7.21 the



(a) test set (b) SNR level

Fig. 7.21: Comparison of different multi-stream approaches using non-linear feature reduction approaches and SCHMMs. In this case, the operator '$*$' stands for the sum of pre-nonlinearity outputs of the neural nets. The detailed results can be found in Appendix A in Tab. A.19, Tab. A.42, Tab. A.22 and Tab. A.43.

results of the two previous systems and the tandem original system using only PLP features are shown. In this figure we can observe:

- The multi-stream tandem original approach (PLP-TAN $*$ MSG TAN original) is significantly better for all tests and SNR conditions than the tandem original approach with one feature stream. The improvement is especially large in unmatched test conditions (test b and test c), and in low SNR levels.

- Similarly, the multi-stream tandem clustering-of-states approach (PLP-TAN $*$ MSG-TAN CofS) improves the performance of the same approach using only one feature stream. The improvement is significant but not very large for the sets test a and test b, but is fairly large for the test c set. A significant improvement is also observed for all SNR levels, especially in the low SNR region.

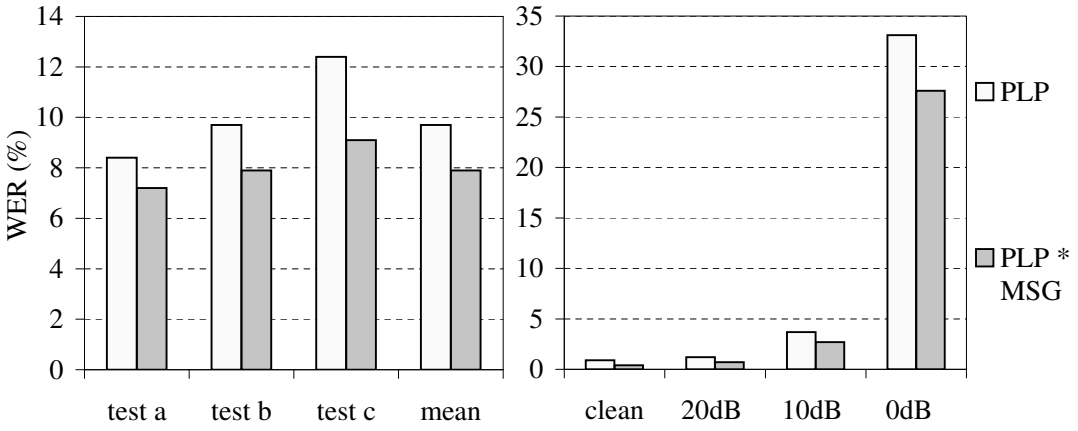
Multi-stream Experiments with Word-Based Hybrid MLP/HMM

In a further set of experiments with the multi-stream approach, we have used a word-based hybrid MLP/HMM system to test whether the improvements were consistent across different kinds of acoustic modeling techniques.

As in our previous experiments with hybrid MLP/HMM systems (cf. Sec. 7.3.2), the PLP feature vector has 13 components, and the delta and delta-delta coefficients are appended to the static features. The input layer of the MLP for the PLP features has, as before, 351 units, the hidden layer 480 units, and the output layer 127 units, one for each state in the HMMs. The MLP for the MSG features is also the same as in the previous experiments, and has 252 input units, 480 hidden units and 127 output units.

In these experiments we have simply combined the outputs of the two nets using the product rule, as explained in Sec. 6.3.2, to obtain one vector of posterior probabilities.

From Fig. 7.22 we can deduce:



(a) test set (b) SNR level

Fig. 7.22: Tests of multi-stream combination using the product rule. The operator $*$ stands for the product of posterior probabilities. The acoustic modeling approach used in this experiment is word-based hybrid MLP/HMM. More details about the results can be found in Tab. A.14 and Tab. A.44.

- Probability combination using the product rule with both MLPs significantly improves the performance for all test conditions and SNR levels (20% on average). This improvement is especially large in unmatched test conditions and low SNR levels.

- Compared to the previous results with multi-stream and SCHMMs, these results are on average significantly better than the previous. However, the results for test c are significantly worse than the best results in Fig. 7.21, but the results for test b are much better (around 30% relative). For the test a the results are not significantly different. This trend is also observed in the results with just one stream of features (cf. Fig. 7.10 and Fig. 7.9).

Although the last results seem to be consistent with the previous results with just one feature stream, it was reported in [HES00] that the 'tandem acoustic modeling' outperforms hybrid MLP/HMM acoustic modeling. In that study, the authors support the thesis that the superiority of a tandem system derives from the double acoustic modeling performed by the MLP and the CDHMM— from here the name tandem. We believe, however, that the reason for this discrepancy with our results is the different kind of acoustic model used by those authors in the tandem and hybrid MLP/HMM experiments. In the former, word-based SCHMMs were used, whereas in the latter case phone-based HMMs were used. Since it is well known that, given a sufficient amount of training material, word-based HMMs are superior to phone-based HMMs, it could well be that the 'superiority' of the tandem approach is just due to the more accurate acoustic model. This belief is confirmed by the previous experiments, since they show that a word-based hybrid MLP/HMM system is superior to the tandem system.

Summary

We have seen in this section that the use of multiple streams improves the performance in about 10-20% relative. However, and as already commented in Chapter 6, the number of parameters of the multi-stream system is larger than that of the system using just one stream. This may be the reason for the positive results obtained in the previous section, as already reported in some of the studies reviewed in Sec. 6.2. In spite of that, multi-stream is still interesting because it provides a way to save training time and hardware resources.

7.4 Experiments on UKKCP

7.4.1 Baseline System Configuration

Feature Extraction

Our feature extraction for this database uses a similar configuration to the experiments with the AURORA database (cf. Sec. 7.3.1). The main differences between the feature extractions are due to the higher sampling rate in the UKKCP database (16 kHz vs. 8 kHz in the AURORA database). As before, the speech signal is first denoised using spectral subtraction. Next the denoised speech is analyzed using a Hamming window of 22 ms, which is shifted every 10 ms. The spectrum of the windowed frame is computed using a 512 point FFT. As in the AURORA experiments, a filter-bank is applied to the computed spectrum to obtain 19 filter-bank coefficients (in the AURORA experiments we have used a bank of 16 filters). Once computed, a logarithmic compression is applied to these coefficients to reduce their dynamic range. A 19 point Discrete Cosine Transform (DCT) is applied to the compressed coefficients to obtain 13 cepstral coefficients. As in the AURORA experiments, Cepstral Mean Subtraction is used to cancel any channel distortion.

Acoustic Modelling

This set of HMMs consists of 70 whole-word models, 81 phonetic, 136 context-dependent, plus one pause model and 8 HMMs to model non-linguistic phenomena (noise, breathing, etc...). The most frequent words are modelled using whole-word models, whereas the others are modelled using a mixture of context-independent and context-dependent models.

Training

The training process of the HMM parameters has, as in the AURORA experiments, two well-differentiated parts: a first is carried out to compute, in an unsupervised fashion, a segmentation into HMM states (Fig. 7.1), and a second part that uses this segmentation to estimate, in a supervised way, a set of SCHMMs (Fig. 7.2). Although the procedure is similar to the training process in the AURORA experiments, there are some peculiarities owing to the context-dependent acoustic models used in these experiments. In the first training part and after computing the unsupervised code-books, the parameters of the whole-word and phoneme SCHMMs are estimated, and the latter models are then used to initialize the parameter values of the context-dependent SCHMMs [CKRB93]. Next, these context-dependent models are jointly trained with the phoneme and whole-word

models to obtain the final set of SCHMMs. This final set of models is used to generate a segmentation into HMM states of the training data, which is utilized in the second part to generate a supervised code-book. This code-book is generated in a similar fashion to the AURORA experiments, the only difference being the classes used in the code-book. These are no longer HMM states, but rather clusters of HMM states obtained by clustering the states using the Lee clustering algorithm (cf. Appendix D). In our case, the total number of states is 1496, which are clustered into 1024 clusters. As in the AURORA baseline, a multidimensional normal distribution is used to model each of these clusters. The modeled vector consists of 9 contiguous cepstral frames appended to form a large 117-dimensional feature vector. This code-book is used to compute an LDA transform, to reduce the vector dimensionality of the code-book to 32 dimensions. After estimating the supervised code-book, and quantizing the feature vectors using the LDA transform and the reduced code-book, the whole-word and phoneme SCHMMs are trained using the Baum-Welch algorithm. The parameters of the phoneme models are used to initialize the context-dependent models, which are in a second step jointly trained with the phoneme and word models using the Baum-Welch algorithm as well.

The obtained HMMs (a total of 296 word, context-independent and context-dependent models) are then used in our ASR system as in the AURORA experiments (cf. Sec. 7.3.1). Unlike those experiments, however, the lexicon is no longer a simple list of word models, but rather a lexicon tree whose branches are the models of the words constructed by concatenating context-independent or context-dependent HMMs— except words having their own HMM—, as was explained in Sec. 2.4.2. Although the structure of the lexicon is different, the decoding algorithm is also the one-pass (one-stage) algorithm. Since in most of the in-car applications an ASR system inputs data to a dialogue system, the ASR system knows in advance which group of words are a valid input to the actual state of the dialog system. This enormously restricts the size of the lexicon for a given sentence, and consequently speeds up the recognition time (the search space is reduced considerably). Consequently, we have grouped the words in the lexicon into three groups (sub-lexicons), namely cities, digits and spelling. The first consists of city names in the UK, the second of whole-word digit models plus models for the words 'hash' 'square' and 'star', and the third of whole-word models of the alphabet letters (spelling).

7.4.2 Discriminative Feature Reduction

Tandem Experiments

The feature vector used for the tandem system is exactly the same as in the baseline system, i.e. a mel-cepstral coefficients using spectral and cepstral mean subtraction. As in the AURORA experiments, the training process of this system consists of three parts: segmentation into phonetic classes of the training data, training of the NN using the previous segmentation, and finally training of the SCHMMs using the pre-nonlinearity outputs of the NN. The tandem system uses a set of 54 phonemes as the target classes of the feature reduction transform. The segmentation into the phonetic classes has been obtained by training unsupervised phonetic SCHMMs and using them to automatically segment the training data. Since the feature vector is the same as in the baseline, the unsupervised code-books were also the same as in that system, and consequently the quantized feature vectors were also the same as in our baseline. The main difference with

respect to the segmentation part of our baseline is the absence of whole-word HMMs, and that no segmentation step with context-dependent models is performed. We have used a set of HMMs consisting of 54 phoneme models plus one pause model and 8 HMMs to model different non-linguistic phenomena. Note that the number of phonemes is less than the number used in the baseline system. The reason is that the number of phonemes used in the baseline (81 phoneme models) is too large to be used as a feature vector. Hence, we have only selected the 54 most frequent phonemes. Since no word models were used, the digits and alphabet letters in the training lexicon have been transcribed using these 54 phonemes as well. The set of 63 HMMs (54 + 8 +1) was trained using, as in our baseline, the Baum-Welch algorithm. After that a segmentation into HMM states (ALI files in Fig. 7.1) is generated using the trained HMMs. This segmentation into states is afterwards mapped to a segmentation into 56 phonetic classes which are the 54 phonemes plus the pause and a class representing all the non-linguistic HMMs.

Once the segmentation into classes has been obtained, the next step is to estimate the weights of the NN. The topology of the NN used is as shown in Fig. 4.5, and has 351 input units, 480 hidden units, and 56 units in the output layer. Note that the topology is the same as that used in the AURORA experiments, except for the number of units in the output layer. The input vector of the NN is formed by concatenating 8 context frames to the actual frame (4 before and 4 after), where the frames are the static feature vectors with appended delta and delta-delta vectors (a 39-dimensional vector). The weights of the NN are also trained in this case using the error back-propagation algorithm.

After training the NN, the next step was to generate the reduced 56-dimensional feature vectors using the trained NN. The 56-dimensional feature vectors were then used to train supervised HMMs as was done in our baseline system. To estimate the supervised code-book, we used the segmentation into state clusters (a total of 1024) already used in our baseline. In this case, however, there was no need for an LDA transform because the feature vector had already been reduced. Each of the normal distributions in the supervised code-book was therefore 56-dimensional. As before, this code-book was used to quantize the feature vectors as explained is Sec. 7.4.1, and the quantized vectors were used subsequently to estimate the parameters of the SCHMMs using the Baum-Welch algorithm. The topology of the SCHMMs was exactly the same as in our baseline, i.e. it was a mixture of context-independent, context-dependent and whole word models with a total of 296 HMMs. The experiments with this system are shown in Fig. 7.23 where we can observe:

- As opposed to our experiments with the AURORA database, the results on the digit sub-task using the tandem original approach are significantly worse than our baseline (c.i. ± 0.4 at 95% confidence level). A similar trend is observed in the 'spelling' sub-task (c.i. ± 1.1), where the HMMs are also whole word models.

- The difference between the baseline and the tandem approach is also significant in the 'cities' sub-task (c.i. ± 3.75), where most of the words are modeled using context-dependent HMMs.

We believe that these disappointing results are due to the following factors:

- The most probable cause of this result is the mismatch between the target classes of our NN (phonemes) and the classes used in our acoustic modeling which include

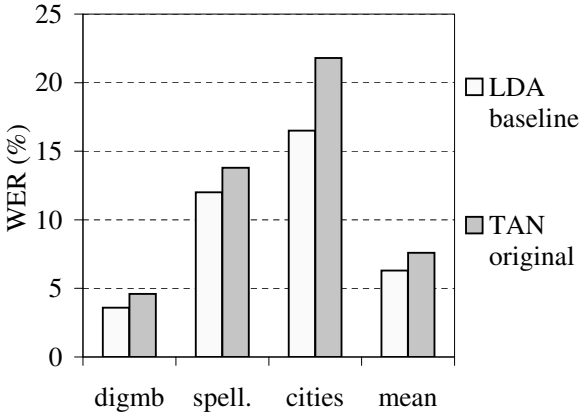


Fig. 7.23: A comparison on the UKKCP database between our baseline system and the original tandem approach for dimensionality reduction (cf. Sec. 4.6.1). The LDA feature vector has 32 dimensions whereas the tandem feature vector has 56 dimensions.

context-dependent, context-independent and whole word HMMs. As already pointed out in [ESS01], this mismatch increases the overlap between the HMM states, because each NN output (phone target) may be shared between some HMM states. The number of states sharing the same NN output grows with the increasing number of HMM states, and consequently the overlapping between them also grows. Since the number of states (127) was rather small in the AURORA experiments, the overlapping between states was not very large. Therefore, discriminating between HMM states was not difficult, and the key factor for the success of this approach on the AURORA database was rather the improved statistical modeling of the classes for feature reduction (cf. Sec. 4.6). In contrast, the number of states in the UKKCP experiments is rather large (1496), and the overlapping between them after tandem feature reduction is consequently larger than that in the LDA output space. Also, the confusability between words of the 'spelling' and 'cities' test sets is much larger than that of the digits. Moreover, in the 'cities' test set the models of the words are constructed using context-dependent and context-independent models, which are shared across different words. An increase in the overlap at the state or model level, therefore, has also more impact on the confusable sets. Theoretically, a solution to this problem is to use NLDA as in the AURORA experiments, because this feature reduction approach uses the HMM states or code-book classes as classes for feature reduction, and is therefore somehow matched to the classifier in the acoustic modeling.

- The NN for dimensionality reduction has been trained on the whole training set, in which all the possible targets of the NN (56 phonetic targets) occur. In the 'digmb' and 'spelling' test sets, by contrast, some of the targets (outputs of the NN) do not occur, and consequently some outputs of the NN should be always zero during testing. Although those outputs are zero or near to zero during training, this is probably not the case during testing, which further introduces confusability during classification in the 56-dimensional feature space. A possible solution would be to use a technique similar to missing data theory in the acoustic modeling (cf. Sec. 3.5.1),

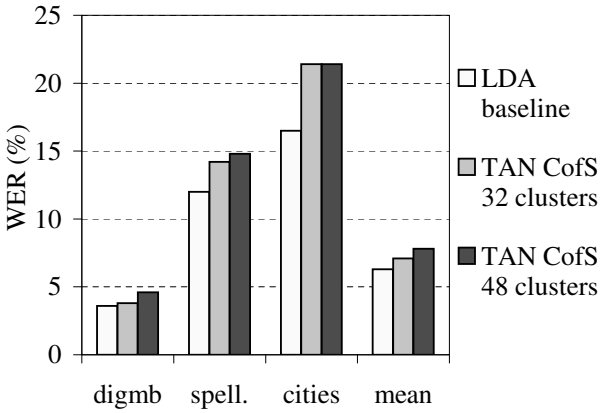


Fig. 7.24: Experiments on UKKCP using the tandem CofS approach.

since we know for each test set which are the valid outputs, and it is therefore possible to marginalize the normal densities.

- Another factor that could have influenced the performance of this approach could be the difference in the *a priori* probabilities of the phonetic classes. The most frequent class occurs 44.5% of the time, whereas the less frequent only 0.022% of the time. By contrast, the same frequencies of occurrence in the analogous AURORA experiment (cf. Sec. 7.3.3) were respectively 22% and 1.3%, which explains why in the AURORA case this effect was not so important. As is further explained in the point where the NLDA experiments are discussed, these large differences in the frequency of occurrence may cause the rare classes to be 'ignored' during classification.

Tandem Clustering of States Experiments

To see if we could improve the match between the classes in the acoustic model and those used in dimensionality reduction, we have clustered the states of the HMMs as was done in Sec. 7.3.3 in the experiments with the AURORA database.

The topology of the NN used in these experiments is very similar to that used in the previous point— a one hidden layer MLP—, the only difference being the number of units in the output layer, which is equal to the number of state clusters used. As before, the number of input and hidden units is 351 and 480, respectively.

The clusters of states are obtained using the same procedure as in the AURORA experiments, i.e. the Lee clustering algorithm (cf. Sec. D). However, the reduction in the number of classes is in this case much larger, because the number of states is 1496 and the number of clusters lies between 16 and 56. This strong reduction may cause the clusters to greatly overlap, and consequently discrimination between them becomes difficult.

The segmentation into state clusters of the training data is obtained from the segmentation into HMM states used to estimate the LDA transform in our baseline system. This is simply done by mapping a given HMM state into the corresponding cluster.

Once this segmentation has been obtained, the weights of the NN are estimated, also using the EBP algorithm and the same configuration as in the previous cases.

In Fig. 7.24 we can observe:

- For the 'digits' test set the difference between our baseline and the tandem system with 32 clusters (TAN CofS 32) is not statistically significant (c.i. $\pm 0.3\%$). In contrast, the difference with the tandem system with 48 clusters is statistically significant.

- In the 'spelling' sub-task the differences with respect to the baseline are statistically significant for both cases, although the differences between both tandem results are not statistically significant (c.i. $\pm 1.2\%$).

- Finally, for the 'cities' test set the differences between the tandem approaches and the baseline are also statistically significant (c.i. $\pm 4\%$).

Interestingly, and in contrast with the analogous experiments on the AURORA database (cf. Fig. 7.13), no improvement is obtained when using the tandem Clustering of States approach. On the contrary, it seems from this limited set of experiments that the performance even decreases as the number of clusters grows. The explanation for the bad results compared to our baseline is similar to the one given in the tandem original approach experiments above. The only difference is that the intrinsic cause of the failure is no longer the mismatch between acoustic modeling and feature reduction classes, but rather the large overlapping between the clusters (clusters are obtained in this case by clustering the 1496 states into 32 or 48 classes). By contrast, in the AURORA experiments the reduction in the number of classes was rather small (127 states to 16, 24 or 32 classes), and in addition the number of words in the vocabulary was only 11.

On the other hand, the performance decrease with increasing number of clusters seems to support the thesis explained in the previous point about the redundant number of outputs in the 'digmb' and 'spelling' test sets. The results show indeed that a reduction of the number of clusters— and consequently of the redundant outputs— is only beneficial in those test sets.

Non-Linear Discriminant Analysis Experiments

Since none of the tandem feature reduction approaches brought any improvement on UKKCP, we decided to also test the NLDA approach already used in the AURORA experiments. The advantage of using this approach is that the classes used for dimensionality reduction are the same as those used in the acoustic modeling (states). According to the theoretical discussions in Chapter 4, this should result in a better discrimination in the dimensionality-reduced space. In fact, and as already discussed in the mentioned chapter, this approach uses the same classes as LDA but makes weaker assumptions on the data— classes are not assumed to be normal — and uses a non-parametric method to find the mapping.

The used NNs had two hidden layers and the same input space as the NN in the previous tandem experiments, i.e. 351 input units. The first hidden layer had 480 units, and the second hidden layer 32 units. Since we have used two different kinds of targets, the number of units in the output layer has 1496 or 1024 units. In the first case, the targets of the NN are the states in the acoustic modeling, whereas in the second case the targets are the classes associated with the symbols in the code-book of our baseline system (1024 symbols). To train the first NN we have used exactly the same segmentation into states as in the baseline system. For the second NN we have used a segmentation into

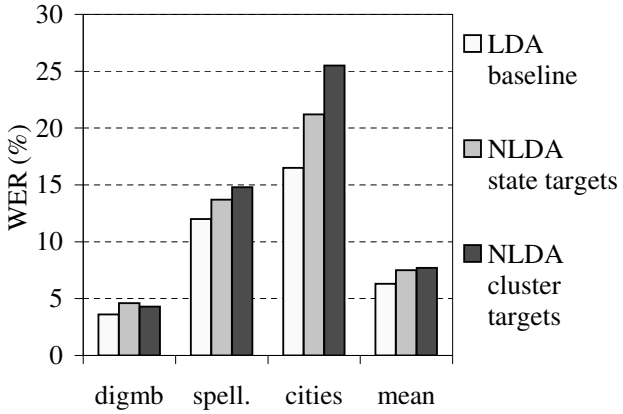


Fig. 7.25: Experiments on UKKCP using different variations of the NLDA approach.

clusters derived from the previous segmentation into states by simply mapping each state into its corresponding cluster. Both NNs have been trained using the usual EBP algorithm configured in the same way as in the previous experiments.

As before, after training an NN we generate the reduced feature vectors of the training set by inputting the high-dimensional input vector into the NN. The difference with the tandem approach is that the 32-dimensional reduced feature vector is generated at the output of the second hidden layer instead of at the output layer.

The set of feature files with the 32-dimensional vectors is then used to train the SCHMMs as in the tandem approach. Using the results shown in Fig. 7.25 we can see that:

- The NLDA approach using state targets to train the NN is significantly worse than the LDA baseline for all test sets. The difference is well above the confidence level (c.i. $\pm 0.3\%$) in the 'digmb' test set, and just outside the confidence interval in the 'spelling' and 'cities' sets (c.i. $\pm 1.2\%$ and $\pm 4\%$, respectively).

- When clusters of states are used as targets, the results are also worse than our baseline system for all test sets. Compared with the previous approach, the results seem to improve for the 'digmb' set, but at the same time the performance apparently deteriorates for the other two sets. Nevertheless the differences between both target types are just slightly above the significance limit in each test set, which casts doubt on the statistical significance of these differences.

These results with the NLDA approach are disappointing because we hoped to obtain a non-linear mapping to a reduced space with better discrimination properties, since no assumption on the statistics of the feature vectors in the clusters is made. Clearly, there are other disadvantages that counteract the assumed benefits. We believe that the poor results are caused by a mixture of the following problems:

- The prior probabilities of the states show a lot of variation. As shown experimentally in [BB93], this differences cause the number of training samples for each class to vary significantly, which leads to worse classification performance for the rarer classes. In [BCH98] it was also observed that classes with a low prior probabilities

were 'ignored' during test. The authors in [LBB+98] show that the cause of this phenomenon is the use of techniques such as early stopping or weight decay, which lead to 'smoother models' with better generalization to unseen data [Bis96]. This bias towards 'smoother' solutions, however, leads to inaccurate fitting of the optimal function, and the result is a tendency to 'ignore' the classes with low frequency. In fact, the classes in the segmentation used to train the NN for feature reduction have extremely different frequencies of occurrence. These frequencies range from the 4078130 times of the pause model to 5 times the 3rd state in the model 'aUl'. As a consequence, our NN for NLDA is strongly biased towards classifying the input frame as a pause frame, which clearly has an effect on the structure of the reduced feature space of 32 dimensions (2nd hidden layer output). Furthermore, this problem is difficult to trace if we only look at the overall target classification rate, because this rate in the training and cross-validation sets can be large, in spite of the fact that there are classes— namely the less frequent— which are not recognized at all. A number of solutions to this problem are also proposed in [LBB+98], and are further discussed in Chapter 8.

- Also related to the previous problem is the fact that the large number of classes (1496 or 1024) leads to very complex decision boundaries in the input space. Since techniques to improve the generalization of the net, such as early stopping, tend to find 'smoother' solutions, it is very difficult to find an accurate solution when the decision boundaries are very complex. A possible simple solution would be to relax the 'smoothness' constraints, but this could result in over-fitting to the training data. Another more complex alternative would be to build a hierarchical classification space, and use a hierarchical NN to classify the reduced feature vector [Sch96, FF98], instead of using just one layer of weights— and therefore a simple linear classifier— between the reduced feature vector (outputs of the 2nd hidden layer) and the outputs of the NN. The idea is to keep the topology of the NN for NLDA in its non-linear transform part (the first two layers of weights), and use a hierarchical NN between the 2nd and the output layers.

- Although to train the NN we have used the same segmentation into HMM states as in our LDA baseline, it could well be that NN training using EBP needs a more accurate segmentation to attain a high classification accuracy. In fact, by using the segmentation into HMM states we assume that it contains a kind of 'ground truth', i.e. frames are assigned to the right classes, although we know that the segmentation has been obtained using imperfect models [RMB91]. Moreover, the larger the number of classes to be discriminated the more uncertain are the assignments in the segmentation, because the distance between the classes decreases. When LDA is used, errors in the class-assignment of a given frame have little effect on the means and covariances of the code-book (and therefore on the LDA matrix), since both mean and covariance of a class are obtained by *averaging* over all the frames assigned to that class. In contrast, when EBP is used to train an NN for feature reduction the frames are individually input to the NN to obtain the output, compute the error and adapt the weights of the NN. Although, this case would correspond to the pure sequential training of the net (cf. Sec. 5.5), which has not been used in our experiments, the influence of a single misclassified frame is still strong if we, as in

our case, only adapt the weights after accumulating the errors of 16 frames (bunch training).

- Finally, a last reason for the deceiving results could be differences in the *a priori* probabilities of the classes in training and test. In fact, this could be our case since our test lexicon includes three different sub-lexicons (digits, spelling and cities), which are separately activated to account for the different stages in the dialog (cf. Sec. 7.4.1). When one of the sub-lexicons is activated, only the classes (states or clusters) present in that sub-lexicon have a prior probability of occurring. As demonstrated in [BM94, pg. 181], the mismatch between test and training priors leads to poor recognition performance when NNs are used as classifiers in hybrid ANN/HMM speech recognizers. In that case, a successful solution to the problem is to normalize the posterior probabilities, i.e. the outputs of the NN, by the *a priori* probabilities estimated on the training data set [BM94]. This normalization converts the posteriors to normalized likelihoods, which can then be used as normal likelihood in standard GMM/HMMs. In our case, however, the NN is not used as a classifier but rather as a mapping function between two feature spaces. Therefore, it is not possible to normalize the outputs of the NN without distorting the low-dimensional feature space. One possible solution is to use one of the methods suggested in [LBB+98] (used in the first point to equalize the prior probabilities of the different classes) to adapt during training the prior probabilities of the classes to those in the test set. However, this requires the retraining of both the NN and the HMMs, which is too cost-intensive. Furthermore, at training time of the HMMs it is often the case in practice that the *a priori* probabilities in the training set cannot be determined.

Summary

In this section we have investigated the performance of some alternative feature reduction approaches on the UKKCP database for in-car applications. We have found that none of the new approaches improves the performance of a similar system using conventional LDA feature reduction. For the tandem systems, the main reason seems to be the inherent mismatch between the classes for feature reduction, and the classes used in the acoustic modeling. For the NLDA system, there seem to be multiple reasons that contribute to the disappointing results: large differences in the *a priori* probabilities of the classes, errors in the segmentation into classes, and differences between the *a priori* probabilities of training and test.

7.4.3 Multiple Streams of Features

In a last set of experiments, we have tested the performance of the multi-stream speech recognition, applied together with the LDA and tandem approaches, on the UKKCP database.

A first experiment was to test the performance of a multi-stream system resulting from the probability combination of two LDA-based systems, since this approach significantly improved the performance of our AURORA baseline (cf. Sec. 7.3.5). A first step was thus to generate the MSG features for the second system. Since the sample rate of this database is larger than in AURORA (16 kHz vs. 8 kHz in the latter), the number of

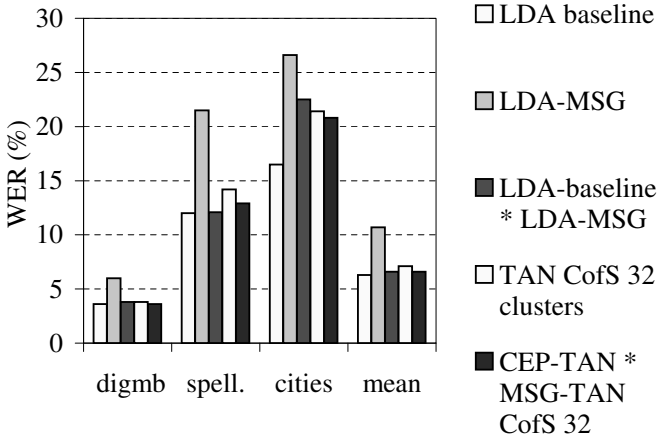


Fig. 7.26: Performance comparison on UKKCP of a multi-stream system combining two LDA-based systems (LDA-baseline $*$ LDA-MSG), and a multi-stream system combining two tandem systems (CEP-TAN $*$ MSG-TAN CofS). The results of the baseline system, and those of a similar system using MSG features are also given for comparison.

filters in the bank-of-filters used in the MSG analysis is 18 instead of 14. This increase in the number of filters results in an MSG feature vector of 36 dimensions. After generating the feature vector, the next step was to train an LDA matrix, code-book and HMMs for this system. We performed the same steps as in our baseline system (cf. Sec. 7.4.1), with the particularities that the number of context frames used as input to the LDA matrix is just three. This resulted in a 108×32 dimensional LDA matrix. Once the MSG system was trained, we combined this system with our baseline using probability combination in a similar fashion to the analogous AURORA experiment. The main aspect in this experiment is the large number of state emission probabilities needed for the combined system (2048 emissions per state) since each code-book had 1024 classes.

The second multi-stream system combined two tandem systems. One was the CofS tandem system with 32 targets discussed in Sec. 7.4.2, whereas the other is a similar system that used MSG instead of cepstral features. The input to the NN was a window of 9 MSG frames, and as a consequence the input layer of the NN had 324 units. As before, the hidden and the output layers had 480 and 32 units, respectively. After training both NN, we combined their pre-nonlinearity outputs to obtain the final reduced feature vector, just as in the AURORA experiments of Sec. 7.3.5. Subsequent steps were also analogous to those performed for the CofS tandem system with 32 clusters in Sec. 7.4.2. In Fig. 7.26 we can observe:

- As expected, the results of the MSG system lay far beyond those of our baseline since MSG features are ill-adapted to GMM modeling (cf. Sec. 7.3.2).
- In contrast to the AURORA experiments the LDA-based multi-stream system performs worse than the LDA baseline, although the differences are only significant for the 'cities' test set (c.i. $\pm 4\%$). For the other test sets, the differences lay near or inside the significance interval (c.i. $\pm 0.2\%$ and $\pm 1\%$ for 'digmb and 'spelling', respectively).
- As for the multi-stream system, the results show a significant improvement in the

	AURORA	UKKCP
Optimum Feature Set	normalized PLPs are on average 9% relative better than MFCC-based baseline.	-
Discriminative Feature Reduction	Non-linear approaches better than LDA in matched test condition but with similar performance in unmatched.	Non-linear approaches are not better than LDA.
Hybrid RBF/HMM Systems	Hybrid RBF/HMM only slightly better than SCHMM and worse than hybrid MLP/HMM.	-
Multiple Streams of Features	Adding the MSG feature stream improves performance in about 10-20% relative.	Addition of the MSG feature stream does not significantly improve performance.

Tab. 7.4: Overview of the main results with both databases.

'spelling' test set (c.i. $\pm 1.1\%$), but not significant for the other test sets.

It seems, therefore, that the clear benefit of using this approach observed in the AURORA experiments is not clear in this case. The disappointing results with the LDA-based systems could be due to the following factors:

- The large number of emission probabilities to be trained (1496×2048 parameters) in the multi-stream system, and the limited number of training data, lead to a 'curse of dimensionality effect (cf. Sec. 2.6). This results in lower performance, especially in the test set 'cities where the units with the lowest occurrence in the training set (and consequently those having poorly-trained parameters) are used. A solution would be to use more training data, to see if the results significantly improve.

- Another reason for the results could be also the bad matching between MSG features and GMM acoustic model. Although this mismatch was also present in the AURORA experiments, it may be further aggravated by the complex decision space of the UKKCP task. In fact, the results in the previous figure show a large performance degradation of 70% relative for the MSG features, whereas the degradation was only of 23% relative in the AURORA case.

Summary

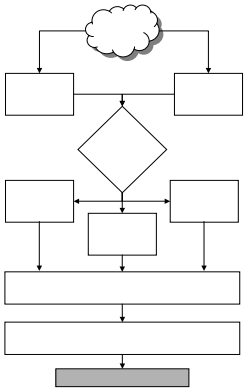
The results of using multi-stream speech recognition for the UKKCP task are not as conclusive as for the AURORA task. Although a certain improvement was observed when multi-stream was used in tandem systems, a significant degradation (especially in the 'cities' test set) was observed in LDA-based systems.

7.5 Summary

An overview of the main results obtained in this chapter is shown in Tab. 7.4. We have seen that the evaluation results obtained by the tandem system in Chapter 3, are a contribution of:

- the normalized PLP features in mean and variance,
- the non-linear feature reduction using neural nets (tandem),
- the use of multiple streams of features with the MSG as complementary features to the PLP.

On the other hand, the test RBF/HMM system obtained a slightly better performance than a similar system based on SCHMM, although its performance was clearly worse than that of a similar hybrid MLP/HMM. Finally, we were not able to observe any improvement on the UKKCP car database using non-linear feature reduction or multiple streams of features.



8. Conclusions and Future Research

As already mentioned in Chapter 1 a central problem in ASR is of handling the huge variability of the speech signal. One of the sources of this variability is the noise or distortion that distorts the speech signal in the course of the recognition process. This thesis has been devoted to the study of techniques that improve the performance of ASR in noisy environments (noise robustness). A first part of this thesis has been devoted to evaluation of ASR systems and techniques and to conduct an experimental evaluation of some ASR techniques (Chapter 3), whereas a second part has been devoted to the study of the best techniques in the previous experimental evaluation (Chapters 4,5 6, and 7).

8.1 Conclusions

In the first part, we have discussed concepts of user-centered evaluation and technology evaluation. Also, we have described the elements of technology evaluation, and put special emphasis on the measures used in evaluation of ASR systems. As explained the WER is the most common measure for ASR performance. This measure is a statistical quantity since it is measured on a finite set of speech files. Consequently, some methods are needed to establish the statistical significance of the differences between two or more WER values. Basically, we have two kinds of methods:

- Confidence intervals. We have proposed in Section 3.3 a new statistic for the WER based on the Poisson distribution, which also takes the insertion errors into account. We have seen in the same section that the length of the confidence interval for the WER increases monotonously with the WER. This is different from the usual model using the Binomial distribution, since in that case the length of the confidence interval reaches a maximum at WER of 50%. Nevertheless, the differences between both models are small for low WER, which is usually the case for any practical system.
- Statistical tests, such as the McNemar's test.

A second objective of this first part was to devise a methodology to evaluate ASR techniques that guarantee a high degree of comparability of the different assessments. To this

end we have interfaced the ASR systems of our project partners and ours at two interface points, namely:

- feature interface,
- state likelihoods/posteriors interface,

By interfacing at the feature interface we have ensured that the probability computation and decoding blocks are the same as in our baseline system, and by interfacing at the state likelihood interface we have ensured that at least the decoding block is the same as in our baseline system. The evaluation experiments in Section 3.6.1 using the previous methodology show:

- Performances of the different techniques evaluated are very different. These range from 19.2% average WER of the missing-data system in Section 3.5.1 to the 7.9% average WER of the hybrid MLP/HMM multi-stream system in Section 3.5.3. Our baseline system described in Section 7.3.1 achieved a 12.8% average WER.
- Best systems over all test sets and SNR levels are the tandem multi-stream (55% relative WER improvement over our baseline system) and the hybrid MLP/HMM systems (63% relative WER improvement).

Since the differences in performance cannot be solely attributed to a particular technique, we decided to devote the 2nd part of this thesis to the tandem, the hybrid ANN/HMM and the multi-stream approaches to understand the reasons for the good performance observed in the evaluation.

In our first experiments in Chapter 7, we have compared the performance of different feature extraction algorithms, namely our baseline based on mel-cepstral (MFCC), perceptual linear predictive (PLP), relative spectral PLP (JRASTA-PLP) and modulation-filtered spectrogram (MSG) features. The rationale behind these experiments was to understand how important was the use of PLP features in the results of the tandem and hybrid MLP/HMM system of the evaluation. Experiments with the different feature vectors showed:

- filtering or normalization of the feature components along the time axis is important to improve noise robustness. As the results on the PLP features show, normalization in mean and variance improves the average results by about 20% relative. Since the MSG features are already normalized (cf. Sec. B.3), a further normalization has almost no effect on the results.
- The normalized PLP features are significantly better than our baseline over all test conditions of the AURORA database. The improvement on average is about 8% relative. In contrast, the MSG features perform worse than our baseline features when GMM/HMMs are used, but perform clearly better than our baseline feature when the MLP/HMMs are used. This result suggests that the MSG features are ill-matched to GMM/HMM modeling.

In the first chapter of the second part, we have discussed the general framework of feature reduction for classification to understand the similarities and differences between LDA and non-linear approaches such as tandem or non-linear discriminative analysis (NLDA).

The most interesting theoretical result in Chapter 4 is that the optimum mapping for feature reduction is related to the Bayes classifier, and that a mapping for feature reduction can be theoretically found by simultaneously optimizing two mappings C and D to approximate the posteriors of the classes at the output of the composed mapping $(C \circ D)$. This is actually what LDA essentially does, and this is also what the tandem and NLDA approaches do. After the theoretical discussion two questions arise:

- which are suitable classes for feature reduction?
- how should the posteriors be approximated?

In our experiments in Section 7.3.3 and in Section 7.4.2, we have used three different kinds of classes namely states, phonemes and clusters of states. To approximate the state posteriors we have used matrices and discriminability criteria, as in LDA, and MLPs and direct approximation of posteriors using the MCE criterion, as in the tandem and NLDA approaches. The results show that:

- the use of per-utterance normalization is crucial to obtaining a high performance in mismatched test conditions. When using our on-line normalization (cf. Sec. 7.3.2), the performance in mismatched test conditions is not as good as with per-utterance normalization. The reason seems to be the weak adaptation of the on-line normalization algorithm to the environmental conditions during test.
- The class choice (states, cluster of states of phonemes) does not seem to have a great impact on the results. Differences in the results on the AURORA database are on average not significant (less than 2% relative), and on the UKKCP database the differences are on average small (less than 10% relative).
- Using the same kind of features, feature normalization and a similar size of the context window, the approaches using a non-linear feature reduction mapping are only superior to LDA in matched noise conditions. On average, however, the performance of the non-linear approaches is 10% relative better than that of LDA on the AURORA database.
- When the complexity of the task and correspondingly the complexity of the acoustic modeling grow, the MLP-based approaches seem to have problems finding a feature reduction mapping that does not increase the overlapping between the states in the acoustic modeling. Possible causes are:
  - Phonetic classes, as in the tandem approach, are not suitable as a feature reduced space because they increase the overlapping between the states in the acoustic modeling, which is especially detrimental when the number of states is large.
  - The great differences in the frequency of the targets cause the infrequent targets to be ignored during recognition.
  - High complexity of the classification boundaries in the case of NLDA, due to the large number of target states in the output layer.
  - Sensibility to errors in the 'ground truth' of the segmentation, which has been obtained using non-discriminative HMMs.

We have also presented the hybrid RBF/HMM, as a compromise between SCHMM and hybrid MLP/HMM. As in hybrid systems the emission probabilities are trained discriminatively, but the normal densities in the code-book are trained as in SCHMM. In contrast to previous authors [RMB+94, SL92, RR95], we have performed our experiments with full covariance normal densities in the RBF. The results in Chapter 7 show that:

- Imposing constraints on the weights (Bayes nonlinearity) works better than imposing no constraint at all on the weights (softmax nonlinearity.
- Improvement with respect to a similar SCHMM is small (only 5% relative on average). Moreover, in matched conditions the SCHMM performs better (8% relative), but in mismatched conditions the hybrid system outperforms the SCHMM system (17% relative).

A reason for this small improvement could be that the greatest potential for discriminative training is in the kernel functions and not in the weights of the RBF. In addition, it could be that a normalization of the state posteriors by the state priors is needed to obtain a good result, or that the posterior probabilities of the states must be considered in the training process.

In Chapter 6 we have also discussed multi-stream speech recognition, as a complementary approach to feature reduction or hybrid acoustic modeling, which may further improve the performance of the whole system. A new technique has been presented that concatenates the LDA vectors obtained from different feature vectors and further reduces the dimensionality of the concatenated vectors using LDA (cf. Sec. 6.3.1). We have also tested the probability combination at the state level (SCHMM and hybrid MLP/HMM) and at the feature level with tandem systems. The experiments with mel-cepstral and MSG features show that:

- The new multi-stream technique with LDA improves the performance of our baseline system about 9% relative.
- With the mentioned features, probability combination is superior to feature concatenation (8% relative on average on AURORA).
- Probability combination at the state-likelihood level may improve the performance of a system, if there is sufficient training data available to train the additional emission probabilities. In the UKKCP experiments multi-stream was not able to improve the result of our baseline due to the additional large number of emission probabilities to be trained.
- Probability combination at the feature level with tandem systems improves the performance, although the improvement is much larger on AURORA (on average 17% relative improvement) than on UKKCP (8% average relative improvement).

8.2 Future Research

We begin our suggestions for future research with a series of possible improvements to our current NN training method,

- Since we have only randomized the order of the files in the training file list, and randomization of the training data is important to obtain an NN that generalizes to unseen data [BM94, pg. 167] using EBP, a better approach would be to randomize the order of the input-output data pairs presented to the neural net. For this purpose we must create a data set of high dimensional input data vectors, e.g. the 351-dimensional input vector in our experiments with PLP features, and randomize it.

- Prior probabilities of target classes (states, phonemes, etc...) are usually very different in ASR, which causes infrequent classes to be 'ignored' during classification (cf. Sec. 7.4.2). Possible solutions to this problem are:

  - Prior scaling. In the minimum cross-entropy error function in Definition 4.3 we can introduce a scaling factor s_c for each class $\mathcal{C}_c$:

    $$s_c = 1 - \alpha \left(1 - \frac{1}{P(\mathcal{C}_c)N}\right) \tag{8.1}$$

    where $P(\mathcal{C}_c)$ is the prior probability of class $\mathcal{C}_c$ and N is the number of classes. Changing the α factor ($0 \leq \alpha \leq 1$) we can control the amount of prior scaling. The criterion in this problem is thus:

    $$\Theta_{opt} = \arg\min_{\Theta} \sum_{\forall l} -s_c \ln(f_c(\mathbf{x}^l, \Theta)) \tag{8.2}$$

  - Probabilistic sampling, whereby the training patterns in the training set are chosen at random in the following way: a class is chosen randomly with the probability of choosing each class C_i being $(1-\alpha)P(C_i)+\frac{\alpha}{N}$, and then a training sample is chosen at random among the training patterns associated with class C_i.
  - Post scaling of posteriors. As already mentioned in Sec. 7.3.4, the normalization of the outputs of the neural net by the prior probabilities of the classes may improve the recognition performance during recognition. However, this technique can only be applied when the neural net performs as a classifier.
  - Equalization of class memberships. The simplest method to equalize the class prior probabilities is to reduce the number of patterns associated with each class or to duplicate the samples in each class.

  More information about the previous techniques and some results on ECG (electrocardiogram) classification tasks can be found in [LBB+98].

- The segmentation $\mathcal{S}$ used to train the neural nets for hybrid system or feature reduction may contain errors, because it has been obtained using HMM models which are not discriminative in nature (cf. Sec. 5.1). This is specially important when the number of classes in the problem is large, since their overlapping is then also stronger. A solution would be to use more than one iteration of the Viterbi training algorithm in Section 5.5 to obtain a better segmentation after each iteration.

- A good feature normalization of the input features is also important to ensure that the components are between 0 and 1, and to obtain a good performance in mismatched conditions. As already mentioned in Section 7.3.2 and in Section 7.3.3,

the feature normalization used in our experiments does not sufficiently adapt to the conditions in the test environment, which reduces its effectiveness in unmatched test conditions. Our feature normalization approach reads the average mean and variance vectors, computed over the whole training set, for each new test file. This was done to simplify the normalization of files in a randomized training list. An improved method would be to first normalize before randomizing the training data as already explained in the first point. By first normalizing, we can adapt the means and variances after each training file has been read, and then use the adapted means and variances to initialize the adaptation of the next file, which should improve the adaptation to the actual environmental conditions.

For the non-linear discriminant analysis (NLDA) approach discussed in Sections 4.6.2, 7.3.3 and 7.4.2 we can suggest the following investigations:

- In our experiments with feature reduction with neural nets, we have used the percent correct classified frames in the cross-validation set as a stop criterion for the training. Although this criterion is suitable for classification tasks, it may be too coarse for NLDA, since it does not suffice that the correct class obtains the best posterior probability. Feature reduction is a mapping between two feature spaces, and it is consequently important that all the components of output feature space are approximated correctly. An alternative stop criterion would be to measure the average cross-entropy in the cross-validation set, and to stop the training when the cross-entropy increases.

- In our experiments with the NLDA approach we have used an MLP with two hidden layers, and we have used the pre-nonlinearity outputs of the 2nd hidden layer as the reduced feature vector. This assumes, however, that the mapping between the reduced features and the class posteriors is relatively simple— just one layer of weights —, which may not be sufficient in practice. As already mentioned in Section 7.4.2, when the number of states in the acoustic modeling is large, the number of classes in NLDA is also large, which leads to very complex decision boundaries. Therefore, it could be interesting to test other topologies for the MLP with more than 2 hidden layers, to allow for more complex mappings between the reduced vector space and the class posteriors.

- Since in our experiments on UKKCP we have seen that the nonlinear feature reduction approaches seem to work well on small tasks, e.g. digits or spelling, it could be interesting to use a different feature reduction mapping for each of the test sets in UKKCP. This is possible because we know at each dialog step in our UKKCP task— and in any in-car task in general— which HMM models are active for the given step. Therefore we just need to discriminate between these models, i.e. between the digits or the alphabet letters. An interesting practical solution would be to use two nonlinear transforms for the digits and spelling sub-sets and an LDA transform for the cities test set.

Further suggestions for future research on hybrid ANN/HMM systems are:

- Since the discriminative training of the emission probabilities of the code-book symbols does not bring a large performance improvement, it would be interesting to try

an approach that trains the parameters of the code-book in a discriminative way. An attempt in this direction is the tied posterior approach [SRR00], which uses an MLP to compute the symbol probabilities, and the usual Baum-Welch algorithm to train the emission probabilities of the symbols in the states. The approach is based on the observation that the state likelihoods:

$$p(\mathbf{x}|q_i) = \sum_{k=1}^{K} b_{ik}\, p(\mathbf{x}|s_k) \tag{8.3}$$

can be converted to posterior probabilities by dividing by $p(\mathbf{x})$ and using the Bayes rule:

$$P(q_i|\mathbf{x}) = \sum_{k=1}^{K} b_{ik}\, \frac{P(s_k|\mathbf{x})}{P(s_k)} \tag{8.4}$$

In this approach, the symbols s_k are the phonetic classes in the vocabulary, and the posteriors $P(s_k|\mathbf{x})$ are computed using an MLP as in hybrid MLP/HMM systems.

- To estimate the weights in our hybrid RBF/HMM approach, we have factored the weights as in Eq. 5.38 and assumed that the prior state $P(q_k)$ probabilities were already given and fixed. However, the weights in the RBF w_{ij} may be interpreted as the joint probability $w_{ij} = P(q = i, s = j)$ which leads to the general constraint for the weights:

$$\sum_i \sum_j w_{ij} = 1 \tag{8.5}$$

  The estimation formulae using this constraint are different from those derived in Section 5.5.1, and may lead to better recognition results.

- To estimate the parameters of the normal kernels we have assigned each code-book symbol to an HMM state, assumed that the statistics of each code-book symbol can be modeled using a single normal density, and computed the parameters of the densities using supervised clustering (cf. Sec. 5.5.1). However, and as seen in Appendix C, the distribution of the data in the code-book classes is not normal. Therefore, it would be interesting to use a mixture of supervised and unsupervised to model each code-book symbol with a mixture of normal densities. The idea is to group the training vectors into as many sub-sets as classes in the code-book using supervised training. Then we apply a clustering algorithm, e.g. LBG, to the data in each sub-group to find a set of normal densities for each class in the code-book. After this process we have an extended set of normal kernels in the RBF, which can be used to improve the accuracy of the modeling.

Appendix

A. Full Results on AURORA 2000

In this appendix we show the detailed or full tables of results of the experiments on AURORA 2000. As already mentioned in Chapter 7, we have not experimented with all SNR levels (clean, 20 dB,15 dB 10 dB, 5 dB, 0 dB and -5 dB) available in the database, since this would have been time consuming, and has little influence on the drawn conclusions. The selected SNR levels are clean, 20 dB, 10 dB and 0 dB. For each experiment, i.e. for each noise/SNR pair, there are 1001 test files, and consequently there is a total of 16016 files.

The tables of results are structured in the following way:

- the rows contain the results across different noises and test sets for a given SNR level. The last row contains the mean results over all SNR levels.
- the columns contain the results across different SNR levels for a given noise or test set. The last column contains the mean results over all noises and test sets.

The appendix is divided in four sections each corresponding to one of the subsections of Sec. 7.3: optimum feature set, feature dimensionality reduction, hybrid RBF/HMM and Multiple Feature Streams.

A.1 Optimum Input Feature Set

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.4	1.8	1.5	1.4	1.5	1.4	1.8	1.5	1.4	1.5	1.9	2.7	2.3	1.6
20 dB	2.2	1.8	1.7	1.7	1.8	1.7	2.3	2.2	1.7	1.9	3.4	4.2	3.8	2.2
10 dB	3.6	6.2	4.3	5.8	4.9	9.1	8.7	8	6.7	8.1	7.2	11.8	9.5	7.1
0 dB	26.3	49.3	28.4	34.2	34.5	52.1	49.5	44.1	38.7	46	28.3	52	40.1	40.2
mean	8.3	14.7	8.9	10.7	10.7	16	15.5	13.9	12.1	14.4	10.2	17.6	13.9	12.8

Tab. A.1: DC LDA-based baseline.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.1	2	1.7	1.8	1.6	1.1	2	1.7	1.8	1.6	1.8	2.3	2	1.7
20 dB	1.6	2.1	2.1	2.2	2	2.1	5.1	4.6	2.4	3.5	2.6	10.1	6.3	3.4
10 dB	4.3	6.1	5	6.3	5.4	7.5	15.6	10.8	7.3	10.3	6.5	17.7	12.1	8.7
0 dB	31.7	39.8	40.4	35.8	36.9	44.4	56.8	43.1	43.7	47	48.2	59.5	53.8	44.3
mean	9.6	12.5	12.3	11.5	11.5	13.7	19.8	15	13.8	15.6	14.7	22.4	18.5	14.5

Tab. A.2: PLP no norm. with LDA and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.8	2.4	2.1	1.5	1.9	1.8	2.4	2.1	1.5	1.9	1.7	1.6	1.6	1.8
20 dB	1.8	2.4	2	3.5	2.4	3.4	3.3	2.8	4.1	3.4	2	4.1	3	2.9
10 dB	4.7	6.9	4	7.3	5.7	10.1	7	6.5	8.7	8	4.4	7.8	6.1	6.7
0 dB	29.6	39.8	47.2	40.1	39.1	38.4	41.1	35.4	46.4	40.3	32.1	42.7	37.4	39.2
mean	9.4	12.8	13.8	13.1	12.3	13.4	13.4	11.7	15.1	13.4	10	14	12	12.7

Tab. A.3: J-RASTA PLP with LDA and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	3.8	3.6	3.2	4.1	3.6	3.8	3.6	3.2	4.1	3.6	3.9	3.8	3.8	3.7
20 dB	2.5	3.5	2.4	4	3.1	5.8	4.5	4.4	4.2	4.7	3.5	5.8	4.6	4
10 dB	6.8	9.4	5.5	9.2	7.7	16.2	10.1	10.5	11.2	12	7.7	12	9.8	9.8
0 dB	37.9	50.5	35.4	43.2	41.7	66.8	43.9	44.8	46.7	50.5	41.4	47.8	44.5	45.8
mean	12.7	16.7	11.6	15.1	14	23.1	15.5	15.7	16.5	17.7	14.1	17.3	15.7	15.8

Tab. A.4: MSG non norm. with LDA and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.4	2	1.6	1.8	1.7	1.4	2	1.6	1.8	1.7	1.4	1.8	1.6	1.6
20 dB	1.4	1.7	1.7	1.6	1.6	1.3	3.3	3.4	1.8	2.4	1.5	4.3	2.9	2.2
10 dB	3.5	5.6	4.7	5.3	4.7	6.2	9.7	8.9	5.8	7.6	3.9	8.5	6.2	6.2
0 dB	26.4	37.3	34.7	30.9	32.3	40	44	39	37.5	40.1	36.8	47.2	42	37.3
mean	8.1	11.6	10.6	9.8	10	12.2	14.7	13.2	11.7	12.9	10.8	15.4	13.1	11.8

Tab. A.5: PLP on-line normalized with LDA and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	4	4.1	3.5	4.8	4.1	4	4.1	3.5	4.8	4.1	4.1	3.9	4	4
20 dB	2.9	3.9	2.8	4.2	3.4	5.2	4.2	4.2	5.2	4.7	3.5	4.7	4.1	4
10 dB	7.5	10.6	5.2	9.1	8.1	17.5	9.8	10.7	11.4	12.3	8.4	10.8	9.6	10.1
0 dB	37.1	54.6	31	42.3	41.2	67.1	40.9	44.8	43.1	48.9	36.8	41.3	39	43.9
mean	12.8	18.3	10.6	15.1	14.2	23.4	14.7	15.8	16.1	17.5	13.2	15.1	14.1	15.5

Tab. A.6: MSG on-line normalized with LDA and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.1	0.7	0.8	0.8	0.8	1.1	0.7	0.8	0.8	0.6	1.2	0.9	0.8
20 dB	0.7	1.1	1.1	1	0.9	0.9	1.3	1	1.1	1	0.8	1.6	1.2	1
10 dB	2.5	3.2	2.5	2.9	2.7	4	4.3	4.5	4.4	4.3	3.2	5.7	4.4	3.7
0 dB	24.1	35.1	23	28.3	27.6	37	37.7	34.4	28.4	34.3	36.5	46.8	41.6	33.1
mean	7	10.1	6.8	8.2	8	10.6	11.1	10.1	8.6	10.1	10.2	13.8	12	9.6

Tab. A.7: MFCC hybrid MLP/HMM with 1510 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.7	1.2	0.7	0.9	0.8	0.7	1.2	0.7	0.9	0.8	0.6	1.2	0.9	0.8
20 dB	0.8	1.2	1	1	1	0.7	1.5	0.9	1.1	1	0.9	1.7	1.3	1
10 dB	2.5	3.6	2.2	2.9	2.8	4.5	4.3	4.6	4.5	4.4	3	5.7	4.3	3.7
0 dB	24.5	35.4	23.2	29.6	28.1	37.2	37.5	35.3	28.5	34.6	37.3	47.7	42.5	33.6
mean	7.1	10.3	6.7	8.6	8.2	10.7	11.1	10.3	8.7	10.2	10.4	14	12.2	9.8

Tab. A.8: MFCC hybrid MLP/HMM with 1100 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.2	0.8	0.8	0.9	0.8	1.2	0.8	0.8	0.9	0.6	1.3	0.9	0.9
20 dB	1	1.3	1.2	1	1.1	1	1.4	1.1	1.2	1.1	1	1.6	1.3	1.1
10 dB	2.8	3.5	2.6	3.2	3	4.8	4.3	4.7	4.7	4.6	3.4	5.5	4.4	3.9
0 dB	25.1	35.4	23.4	30.2	28.5	38.1	38	35.6	29.6	35.3	38.7	49	43.8	34.3
mean	7.4	10.3	7	8.8	8.3	11.1	11.2	10.5	9	10.5	10.9	14.3	12.6	10

Tab. A.9: MFCC hybrid MLP/HMM with 690 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.2	0.8	0.9	0.9	1	1.2	0.8	0.9	0.9	0.8	1.4	1.1	1
20 dB	0.8	1.3	1.2	1.1	1.1	1	1.5	1.2	1.5	1.3	1.1	1.9	1.5	1.2
10 dB	3.1	3.7	2.8	3.5	3.2	4.4	4.3	4.8	4.5	4.5	3.3	5.9	4.6	4
0 dB	25.5	35.6	24.5	29.9	28.8	38.3	38.5	36	29.5	35.5	39	50.3	44.6	34.7
mean	7.6	10.4	7.3	8.8	8.5	11.1	11.3	10.7	9.1	10.5	11	14.8	12.9	10.2

Tab. A.10: MFCC hybrid MLP/HMM with 480 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.1	1.3	1.1	1	1.1	1.1	1.3	1.1	1	1.1	1.1	1.5	1.3	1.1
20 dB	1.1	1.5	1.3	1.3	1.3	1.3	1.6	1.2	1.4	1.3	1.2	2	1.6	1.3
10 dB	3.3	3.8	3.1	4	3.5	5	4.6	4.9	4.8	4.8	3.7	6.3	5	4.3
0 dB	26.7	36.2	25.5	31.7	30	38.3	39.9	35.7	31.1	36.2	41.4	51.1	46.2	35.7
mean	8	10.7	7.7	9.5	9	11.4	11.8	10.7	9.5	10.8	11.8	15.2	13.5	10.6

Tab. A.11: MFCC hybrid MLP/HMM with 240 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.7	1	1	0.8	0.8	0.7	1	1	0.8	0.8	0.6	1	0.8	0.8
20 dB	0.6	1	1.1	0.9	0.9	1	1.5	1.2	1.2	1.2	0.7	1.6	1.1	1
10 dB	2.1	3	2.4	2.9	2.6	3.5	4.3	4.1	3.9	3.9	2.6	4.1	3.3	3.2
0 dB	21.4	31	28.1	26.8	26.8	31.6	33.4	29	30.2	31	33.7	43.3	38.5	30.8
mean	6.1	9	8.1	7.8	7.8	9.2	10	8.8	9	9.2	9.4	12.5	10.9	9

Tab. A.12: PLP hybrid MLP/HMM with 1510 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.6	0.9	0.8	0.7	0.7	0.6	0.9	0.8	0.7	0.7	0.6	1.1	0.8	0.7
20 dB	0.8	1	1.2	0.9	0.9	1	1.5	1.1	1.1	1.1	0.8	1.8	1.3	1.1
10 dB	2.4	3.1	2.4	2.9	2.7	3.6	4.4	4.1	4.3	4.1	2.7	3.9	3.3	3.3
0 dB	22.1	31.3	28.6	27.2	27.3	30.9	33.4	29.1	30.1	30.8	34.5	43.8	39.1	31.1
mean	6.4	9	8.2	7.9	7.9	9	10	8.7	9	9.2	9.6	12.6	11.1	9

Tab. A.13: PLP hybrid MLP/HMM with 1100 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.2	0.9	0.8	0.9	0.8	1.2	0.9	0.8	0.9	0.5	0.9	0.7	0.8
20 dB	0.9	1.1	1.1	1	1	0.9	1.5	1.1	1.4	1.2	1	1.6	1.3	1.1
10 dB	2.6	3.1	2.3	3.3	2.8	3.8	4	3.8	3.9	3.8	2.7	4.1	3.4	3.3
0 dB	22.1	31.6	28.7	27.6	27.5	31.6	33.5	29.6	30.6	31.3	34.3	45.1	39.7	31.4
mean	6.6	9.2	8.2	8.1	8	9.2	10	8.8	9.1	9.3	9.6	12.9	11.2	9.2

Tab. A.14: PLP hybrid MLP/HMM with 690 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.4	1	0.7	1	1	1.4	1	0.7	1	0.8	1.5	1.1	1
20 dB	1.1	1.5	1.6	1.2	1.3	1.3	1.9	1.3	1.2	1.4	1.3	1.9	1.6	1.4
10 dB	2.9	3.6	2.9	3.9	3.3	4	4.8	4.1	4.1	4.2	3.7	4.9	4.3	3.8
0 dB	25.6	33.3	30.9	32.1	30.4	33.1	36.9	31	31.8	33.2	41	50.5	45.7	34.6
mean	7.6	9.9	9.1	9.4	9	9.8	11.2	9.3	9.4	9.9	11.7	14.7	13.2	10.2

Tab. A.15: PLP hybrid MLP/HMM with 270 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.2	1.0	0.6	0.9	0.8	1.2	1.0	0.6	0.9	0.6	1.0	0.8	0.9
20 dB	0.9	1.2	1.3	1.0	1.1	0.9	1.7	1.4	1.2	1.3	1.0	1.8	1.4	1.2
10 dB	2.6	3.4	2.7	3.4	3.0	4.1	4.6	4.3	4.3	4.3	3.0	4.5	3.8	3.7
0 dB	23.9	32.2	29.8	28.7	28.7	32.1	35.0	30.5	31.8	32.4	38.7	48.5	43.6	33.1
Average	7.1	9.5	8.7	8.4	8.4	9.5	10.6	9.3	9.5	9.7	10.8	14.0	12.4	9.7

Tab. A.16: PLP hybrid MLP/HMM with 480 units in the hidden layer.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.7	1.1	0.8	0.7	0.8	0.7	1.1	0.8	0.7	0.8	0.8	1.1	0.9	0.8
20 dB	0.9	1.1	1	1	1	0.8	1.1	1.1	0.9	0.9	1.1	1.4	1.2	1
10 dB	2.7	2.9	2.7	3.1	2.8	3.4	4.1	3.2	3.5	3.5	3.2	3.7	3.4	3.2
0 dB	23.7	32.2	31.5	26.9	28.5	32.1	31	26.8	31.4	30.3	28.8	34.7	31.7	29.9
mean	7	9.3	9	7.9	8.3	9.2	9.3	7.9	9.1	8.9	8.4	10.2	9.3	8.7

Tab. A.17: MSG hybrid MLP/HMM with 480 units in the hidden layer.

A.2 Discriminative Feature Reduction

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.9	1.1	1.1	1.1	1	0.9	1.1	1.1	1.1	1	0.8	1	0.9	1
20 dB	0.9	1.6	1.1	1.1	1.1	1.4	2	2.1	1.6	1.7	0.8	2.3	1.5	1.4
10 dB	2.7	3.4	2.9	3.9	3.2	6.8	6.2	6.8	6.7	6.6	2.8	5.5	4.1	4.7
0 dB	19.9	31.3	25.5	25.8	25.6	41.2	41.2	37	38.8	39.5	22.5	41.9	32.2	32.5
mean	6.1	9.3	7.6	7.9	7.7	12.5	12.6	11.7	12	12.2	6.7	12.6	9.7	9.9

Tab. A.18: PLP tandem original with unsupervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.2	1.7	1.3	1.2	1.3	1.2	1.7	1.3	1.2	1.3	1	1.6	1.3	1.3
20 dB	1	1.6	1.3	1.3	1.3	2.1	2.5	2.6	2.2	2.3	1	2.4	1.7	1.7
10 dB	2.7	3.7	3.2	3.9	3.3	7.2	6.3	7.7	7.5	7.1	3.3	6	4.6	5.1
0 dB	20.5	34.1	27.3	25.2	26.7	41.7	40.6	37.7	37.9	39.4	24.2	43.7	33.9	33.2
mean	6.3	10.2	8.2	7.9	8.2	13	12.7	12.3	12.2	12.5	7.3	13.4	10.4	10.3

Tab. A.19: PLP tandem original with supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.6	1.7	1.4	1.6	1.6	1.6	1.7	1.4	1.6	1.6	1.3	1.5	1.4	1.5
20 dB	1.0	1.3	1.3	1.1	1.2	1.4	1.9	1.8	1.5	1.7	1.1	2.0	1.6	1.4
10 dB	3.1	3.3	3.7	4.2	3.6	5.0	5.3	5.4	4.3	5.0	2.9	5.3	4.1	4.3
0 dB	24.7	36.6	17.9	25.0	26.1	43.2	34.3	32.3	30.0	35.0	24.2	33.2	28.7	30.1
mean	7.6	10.7	6.1	8.0	8.1	12.8	10.8	10.2	9.4	10.8	7.4	10.5	8.9	9.3

Tab. A.20: PLP tandem original approach with per-utterance normalization.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
Clean	1.4	1.8	1.6	1.5	1.6	1.4	1.8	1.6	1.5	1.6	1.2	1.6	1.4	1.5
20 dB	1.1	1.9	1.3	1.2	1.4	2.4	2.8	2.8	2.6	2.7	0.9	3.4	2.2	2.0
10 dB	2.9	4.4	2.9	3.9	3.5	8.8	7.7	8.2	8.0	8.2	2.9	6.9	4.9	5.7
0 dB	19.7	35.0	23.4	25.7	25.9	45.4	43.5	40.1	40.5	42.4	21.1	43.2	32.2	33.8
mean	6.3	10.8	7.3	8.1	8.1	14.5	13.9	13.1	13.2	13.7	6.5	13.8	10.2	10.8

Tab. A.21: PLP TAN original approach with on-line normalization.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.2	1.9	1.2	1.3	1.4	1.2	1.9	1.2	1.3	1.4	1.1	2	1.5	1.4
20 dB	1.2	1.7	1.8	1.2	1.4	1.8	3.2	2.4	2.1	2.3	1.3	7.1	4.2	2.3
10 dB	3.1	4.1	3.6	5	3.9	8.2	6.4	6.9	7	7.1	4.1	8.9	6.5	5.7
0 dB	20	32.7	26.9	27.2	26.7	42.3	38	35.2	38.4	38.4	25.7	40	32.8	32.6
mean	6.3	10.1	8.3	8.6	8.3	13.3	12.3	11.4	12.2	12.3	8	14.5	11.2	10.5

Tab. A.22: PLP TAN clustering-of-states with supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.9	1.5	1.1	0.9	1.1	0.9	1.5	1.1	0.9	1.1	0.8	1.6	1.2	1.1
20 dB	1	1.7	1.6	1.2	1.3	1.3	2.8	1.8	1.4	1.8	1.2	5	3.1	1.9
10 dB	3.2	3.8	3.6	5.4	4	6.9	6.7	6.2	6.7	6.6	3.6	6.7	5.1	5.2
0 dB	20.6	33.1	28.9	27	27.4	42.4	38.3	37.5	41.1	39.8	27.8	41.1	34.4	33.7
mean	6.4	10	8.8	8.6	8.4	12.8	12.3	11.6	12.5	12.3	8.3	13.6	10.9	10.5

Tab. A.23: PLP TAN clustering-of-states with unsupervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.2	1.4	1	1.4	1.2	1.2	1.4	1	1.4	1.2	1	1.6	1.3	1.2
20 dB	1.3	1.8	1.7	1.5	1.5	1.8	2.9	2.1	2.2	2.2	1.5	4.4	2.9	2.1
10 dB	4.5	4.6	4.8	5.7	4.8	8.1	7.5	7.5	8.5	7.9	4.6	9.3	6.9	6.5
0 dB	25.5	36	34.1	30.9	31.6	43	43.1	39	43.6	42.1	32	44.8	38.4	37.2
mean	8.1	10.9	10.4	9.8	9.8	13.5	13.7	12.4	13.9	13.3	9.7	15	12.3	11.7

Tab. A.24: PLP TAN CofS with 16 clusters and supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.9	1.5	0.8	0.9	1	0.9	1.5	0.8	0.9	1	0.9	1.4	1.1	1
20 dB	0.8	1.6	1.1	1.3	1.2	1.2	2.3	1.8	1.3	1.6	1.1	5.1	3	1.7
10 dB	2.7	3.6	3.1	5	3.6	7.1	6.3	6.6	6.7	6.6	3.3	7.4	5.3	5.1
0 dB	20	32.2	28	27.2	26.8	41.1	37.3	36.5	39.6	38.6	26.5	41.6	34	33
mean	6.1	9.7	8.2	8.6	8.1	12.5	11.8	11.4	12.1	11.9	7.9	13.8	10.9	10.2

Tab. A.25: PLP TAN CofS with 32 clusters and supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.6	1.2	1.3	1.2	1	1.6	1.2	1.3	1.2	0.9	1.5	1.2	1.2
20 dB	0.8	1.6	1.5	1.3	1.3	1.1	2.6	2.4	1.6	1.9	0.9	4.2	2.5	1.8
10 dB	2.8	4.7	3.1	4.3	3.7	8	7.6	7.5	6.9	7.5	3.4	7.7	5.5	5.6
0 dB	18.8	38.6	22.2	23.3	25.7	45.1	39.6	38.1	35.6	39.6	25.6	39.3	32.4	32.6
mean	5.8	11.6	7	7.5	8	13.8	12.8	12.3	11.3	12.5	7.7	13.1	10.4	10.3

Tab. A.26: PLP TAN PCA with 32 coefficients and supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.4	1.7	1.2	1.4	1.4	1.4	1.7	1.2	1.4	1.4	1	1.6	1.3	1.4
20 dB	1	1.8	1.5	1.5	1.4	1.4	2.6	2.4	1.7	2	1.2	5.4	3.3	2
10 dB	2.9	4.4	3	4.8	3.7	8	7.8	8.3	7.3	7.8	3.8	9.2	6.5	5.9
0 dB	20.6	38.6	22.7	24	26.4	45.1	40.7	39.1	36.7	40.4	26.1	40.6	33.3	33.4
mean	6.4	11.6	7.1	7.9	8.2	13.9	13.2	12.7	11.7	12.9	8	14.2	11.1	10.7

Tab. A.27: PLP TAN PCA with 24 coefficients and supervised clustering.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.5	1.2	1	1.1	1	1.5	1.2	1	1.1	1	1.5	1.2	1.1
20 dB	1	1.6	1.4	1.2	1.3	1.8	3.9	2.8	2.4	2.7	1	11	6	2.8
10 dB	2.5	4	2.9	3.8	3.3	9.1	8.3	8.2	7.8	8.3	2.9	16.4	9.6	6.5
0 dB	18	33.3	26.3	24.2	25.4	44.6	39.9	36.9	38.6	40	24.7	43.9	34.3	33
mean	5.6	10.1	7.9	7.5	7.8	14.1	13.4	12.2	12.4	13	7.4	18.2	12.8	10.9

Tab. A.28: PLP NLDA $351 \times 1100 \times 24 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.4	1.3	1	1.1	1	1.4	1.3	1	1.1	0.9	1.3	1.1	1.1
20 dB	0.9	1.5	1.4	1.1	1.2	1.7	2.4	3.1	2.5	2.4	1	5.8	3.4	2.1
10 dB	2.5	3.5	2.8	4	3.2	7.6	6.3	8.2	8.4	7.6	2.9	8	5.4	5.4
0 dB	18.9	33.6	25	24.3	25.4	44.6	38.4	36.9	38.1	39.5	25.7	39.9	32.8	32.5
mean	5.8	10	7.6	7.6	7.7	13.7	12.1	12.3	12.5	12.6	7.6	13.7	10.6	10.3

Tab. A.29: PLP NLDA $351 \times 690 \times 24 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.5	1.3	1	1.1	0.8	1.5	1.3	1	1.1	0.8	1.5	1.1	1.1
20 dB	0.6	1.6	1.5	1.2	1.2	1.7	2.8	3.2	2.5	2.5	0.8	8.5	4.6	2.4
10 dB	2.4	3.9	2.8	4.1	3.3	8.6	6.9	8.5	8.3	8	3.2	12.4	7.8	6.1
0 dB	18.9	33.1	26.5	24.1	25.6	44.5	38.5	36.4	38.6	39.5	24.7	40.6	32.6	32.5
mean	5.6	10	8	7.6	7.8	13.9	12.4	12.3	12.6	12.8	7.3	15.7	11.5	10.5

Tab. A.30: PLP NLDA $351 \times 480 \times 24 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.9	1.5	1.1	1.2	1.1	0.9	1.5	1.1	1.2	1.1	0.7	1.7	1.2	1.1
20 dB	0.6	1.5	1.3	1.2	1.1	1.3	2.8	2.9	2.4	2.3	0.6	6.6	3.6	2.1
10 dB	2.2	3.8	2.7	3.7	3	8.1	7.2	8.3	8.5	8	2.7	10	6.3	5.7
0 dB	17.8	33.8	24.5	25	25.2	44.7	39.5	39.5	38	40.4	24.7	42.1	33.4	32.9
mean	5.3	10.1	7.4	7.7	7.6	13.7	12.7	12.9	12.5	12.9	7.1	15.1	11.1	10.4

Tab. A.31: PLP NLDA $351 \times 1100 \times 32 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.2	1.1	0.9	1	0.8	1.2	1.1	0.9	1	0.9	1.5	1.2	1
20 dB	0.7	1.4	1.3	1.1	1.1	1.4	2.3	2.6	1.9	2	0.9	6.9	3.9	2
10 dB	2	3.8	3.1	3.9	3.2	7.8	6.3	8	8	7.5	2.7	9.6	6.1	5.5
0 dB	17.2	33.1	24.8	24	24.7	44.5	38	37.6	37.1	39.3	24	40.7	32.3	32
mean	5.1	9.8	7.5	7.4	7.5	13.6	11.9	12.3	11.9	12.4	7.1	14.6	10.9	10.1

Tab. A.32: PLP NLDA $351 \times 690 \times 32 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1	1.5	1	1.1	1.1	1	1.5	1	1.1	1.1	1	1.4	1.2	1.1
20 dB	1	1.7	1.6	1.4	1.4	1.5	3.5	2.7	2.5	2.5	0.9	10.4	5.6	2.7
10 dB	2.6	4.4	2.9	4	3.4	8.2	8.3	7.8	8.7	8.2	2.9	12.9	7.9	6.2
0 dB	17.6	35.2	24.5	24.2	25.3	45.6	42.1	37.9	39	41.1	23.2	46	34.6	33.5
mean	5.5	10.7	7.5	7.6	7.8	14	13.8	12.3	12.8	13.2	7	17.6	12.3	10.9

Tab. A.33: PLP NLDA $351 \times 480 \times 32 \times 127$ topology and supervised clustering

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.3	1.7	1.4	1.7	1.5	1.3	1.7	1.4	1.7	1.5	1.4	1.8	1.6	1.5
20 dB	1.6	1.7	1.8	1.7	1.7	1.5	2.8	2.6	1.6	2.1	1.9	3.2	2.5	2
10 dB	3.7	5.4	3.8	5.2	4.5	6.3	8.7	7.5	5.2	6.9	4.1	7.3	5.6	5.7
0 dB	25.6	37.4	33.3	30.5	31.7	39.4	43.3	36.5	36.9	39	35.2	45.9	40.5	36.4
mean	8	11.5	10	9.7	9.8	12.1	14.1	12	11.3	12.4	10.6	14.5	12.6	11.4

Tab. A.34: PLP normalized with LDA of 13 context frames and supervised code-book.

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.1	1.9	1.6	1.3	1.4	1.1	1.9	1.6	1.3	1.4	1.2	1.7	1.4	1.4
20 dB	1.4	1.9	2	1.6	1.7	1.8	2.8	2.6	1.6	2.1	1.7	2.7	2.2	2
10 dB	3.8	5	3.6	5	4.3	6	7.9	7.7	5.7	6.8	3.9	6.5	5.2	5.5
0 dB	25.2	37.7	33.7	30.5	31.7	39.9	42.4	35.5	35.9	38.4	33.9	45.8	39.8	36
mean	7.8	11.6	10.2	9.6	9.8	12.2	13.7	11.8	11.1	12.2	10.1	14.1	12.1	11.2

Tab. A.35: PLP normalized with LDA of 17 context frames and supervised code-book.

A.3 Hybrid RBF/HMM Systems

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	2.2	2.6	2.3	2.7	2.4	2.2	2.6	2.3	2.7	2.4	2.1	2.8	2.4	2.4
20 dB	1.7	2.6	1.8	2	2	2.9	5.9	4.7	3.3	4.2	2.1	9.4	5.7	3.6
10 dB	3.9	7.5	4.2	6.2	5.4	8.9	13.5	11.4	8.3	10.5	4.8	10.2	7.5	7.8
0 dB	25	41	30.7	31.2	31.9	45.2	51	46.6	38.9	45.4	34.7	48.4	41.5	39.2
mean	8.2	13.4	9.7	10.5	10.4	14.8	18.2	16.2	13.3	15.6	10.9	17.7	14.3	13.3

Tab. A.36: PLP feature vectors with LDA feature reduction and SCHMM acoustic modeling

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.7	1.8	1.7	1.4	1.6	1.7	1.8	1.7	1.4	1.6	1.8	1.8	1.8	1.6
20 dB	2.2	2.1	2.2	2.2	2.1	2.2	2.8	3	2.1	2.5	2.2	2.8	2.5	2.3
10 dB	4.5	6	4.9	7	5.6	6.3	10	9.2	6.6	8	5.4	7.6	6.5	6.7
0 dB	28.9	37.8	40.2	35.8	35.6	36.6	47.8	41.9	38.9	41.3	39.6	51	45.3	39.8
mean	9.3	11.9	12.2	11.6	11.2	11.7	15.6	13.9	12.2	13.3	12.2	15.8	14	12.6

Tab. A.37: PLP feature vectors with LDA feature reduction and hybrid RBF/HMM acoustic modeling with Bayes non-linearity at the output units

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.6	1.9	1.9	1.3	1.6	1.6	1.9	1.9	1.3	1.6	1.4	2.1	1.7	1.6
20 dB	2.2	2.2	2.2	2.3	2.2	2.1	2.7	2.8	2	2.4	2.7	3.3	3	2.4
10 dB	5.1	6.4	5.6	7.8	6.2	6.1	9.6	7.8	6.4	7.4	6.2	7.9	7	6.8
0 dB	33	41.1	44.6	40.6	39.8	38	48.1	41.3	40.7	42	43.9	53.2	48.5	42.4
mean	10.4	12.9	13.5	13	12.4	11.9	15.5	13.4	12.6	13.3	13.5	16.6	15	13.3

Tab. A.38: PLP feature vectors with LDA feature reduction and hybrid RBF/HMM acoustic modeling with softmax non-linearity at the output units

A.4 Multiple Streams of Features

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.5	1.6	1.6	1.6	1.5	1.5	1.6	1.6	1.6	1.5	1.4	1.9	1.6	1.5
20 dB	1.8	1.8	1.8	1.6	1.7	1.9	2.2	2.2	1.6	1.9	2	2.3	2.1	1.9
10 dB	3.6	5.5	3.3	4.1	4.1	8.8	7.1	6.9	6.6	7.3	3.9	7.3	5.6	5.7
0 dB	24.7	42	23.3	30	30	51.1	39	37.2	35.2	40.6	24.9	38.4	31.6	34.5
mean	7.9	12.7	7.5	9.3	9.3	15.8	12.4	11.9	11.2	12.8	8	12.4	10.2	10.9

Tab. A.39: Multi-stream full-combination of state likelihoods (LDA-baseline # LDA-MSG)

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.5	1.7	1.5	1.5	1.5	1.5	1.7	1.5	1.5	1.5	1.4	1.8	1.6	1.5
20 dB	1.7	1.5	1.7	1.8	1.6	1.8	2.3	2.1	1.9	2	1.7	2.5	2.1	1.9
10 dB	3.4	5.5	3.1	4.4	4.1	9.2	6.8	7.1	6.4	7.3	3.8	7.3	5.5	5.7
0 dB	25.2	41.1	24.6	30.2	30.2	49.3	38.5	35.9	33.6	39.3	26.3	38.5	32.4	34.3
mean	7.9	12.4	7.7	9.4	9.4	15.4	12.3	11.6	10.8	12.5	8.3	12.5	10.4	10.8

Tab. A.40: Multi-stream combination of state likelihoods (LDA-baseline $*$ LDA-MSG)

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	1.8	2.1	1.9	1.6	1.8	1.8	2.1	1.9	1.6	1.8	1.7	2.3	2	1.8
20 dB	1.9	1.8	2.1	1.9	1.9	1.8	2.6	2.1	2	2.1	2.1	2.6	2.3	2
10 dB	4	6.4	3.8	5.1	4.8	10	7.7	7.5	7.3	8.1	4.1	8.1	6.1	6.4
0 dB	25	46.3	24	31	31.5	55.9	42	38.9	35.3	43	24.5	41.7	33.1	36.4
mean	8.1	14.1	7.9	9.9	10	17.3	13.6	12.6	11.5	13.7	8.1	13.6	10.8	11.7

Tab. A.41: Multi-stream concatenation of LDA vectors (LDA-baseline | LDA-MSG LDA)

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.8	1.2	1.1	0.9	1	0.8	1.2	1.1	0.9	1	0.7	1.2	0.9	0.9
20 dB	0.6	1.3	1.3	1	1	1.4	1.9	1.8	1.4	1.6	0.5	1.8	1.1	1.3
10 dB	2.1	3.6	2.5	3.2	2.8	6.4	4.7	5.8	5.6	5.6	2.5	4.3	3.4	4
0 dB	19	35.5	21	22.1	24.4	44.1	32.8	33.4	32.3	35.6	21.7	33.9	27.8	29.5
mean	5.6	10.4	6.4	6.8	7.3	13.1	10.1	10.5	10	10.9	6.3	10.2	8.3	8.9

Tab. A.42: Multi-stream tandem original approach (PLP-TAN $*$ MSG-TAN original).

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.9	1.4	1.1	1	1.1	0.9	1.4	1.1	1	1.1	0.9	1.4	1.1	1.1
20 dB	0.7	1.2	1.2	1.1	1	1.8	2	1.6	1.5	1.7	1	2.7	1.8	1.4
10 dB	2.7	4	2.6	3.9	3.3	8.1	5.7	5.6	6.2	6.3	2.9	5	3.9	4.6
0 dB	19.9	36.5	23	24.3	25.9	48.1	31	34.3	34.2	36.9	22.2	31.6	26.9	30.5
mean	6	10.7	6.9	7.5	7.8	14.7	10	10.6	10.7	11.5	6.7	10.1	8.4	9.4

Tab. A.43: Multi-stream tandem CofS approach with 24 clusters (PLP-TAN $*$ MSG-TAN CofS 24).

	test a					test b					test c			
SNR	N1	N2	N3	N4	mean	N1	N2	N3	N4	mean	N1	N2	mean	mean
clean	0.3	0.7	0.5	0.4	0.4	0.3	0.7	0.5	0.4	0.4	0.4	0.6	0.5	0.4
20 dB	0.5	0.8	0.9	0.6	0.7	0.5	1.1	1.0	0.6	0.8	0.5	1.1	0.8	0.7
10 dB	2.1	2.7	2.4	2.5	2.4	2.7	3.5	3.0	2.6	2.9	2.4	3.2	2.8	2.7
0 dB	20.7	29.7	27.4	24.3	25.5	27.9	28.6	24.8	28.3	27.4	28.6	36.6	32.6	27.6
mean	5.9	8.4	7.8	6.9	7.2	7.8	8.4	7.3	7.9	7.8	7.9	10.3	9.1	7.8

Tab. A.44: Multi-stream combination of hybrid MLP/HMM posterior probabilities (PLP $*$ MSG).

B. Feature Extraction Algorithms

B.1 Perceptual Linear Prediction

Perceptual linear prediction (PLP) technique is a combination of spectral analysis and linear prediction coding (LPC) analysis, that was first introduced by Hermansky in 1990 [Her90]. The main idea of this technique is to take advantage of three characteristics derived from the psycho-acoustic properties of the human ear for estimating the audible spectrum, which are spectral resolution of the critical band, equal-loudness curve, and intensity-loudness power law [Her90].

The algorithm of the PLP technique, which is illustrated in Figure B.1, is briefly described here:

1. The time-frequency analysis is identical to the feature extraction of our baseline system (cf. Sec. 7.3.1). The speech signal is framed with a Hamming window of 25 ms that is shifted every 10 ms. The spectrum of each frame is computed using an FFT of 256 points, and finally squared to obtain the power spectrum of the frame.

2. Analogous to the mel scale in MFCC, the power spectrum is convolved with a bank of filters using bark scale, where it defines the critical bandwidth as 1 bark, and

$$bark(f_c) = 6sinh^{-1}(\frac{f_c}{600}). \tag{B.1}$$

3. Next step is conversion of the power spectra to a loudness scale, using a transfer function which is an approximation to the non-equal sensitivity of human hearing at different frequencies and simulates the sensitivity of hearing at about the 40 db level.

4. This is followed by reduction of the spectral amplitude variation with intensity-loudness power-law, using the cubic root amplitude compression.

5. Then, the operation uses autoregressive (AR) modeling to apply standard LPC analysis [RJ93]. The outputs are the vector a_k containing the predictor coefficients.

6. The AR coefficients are then transformed using the cepstral recursion into cepstral coefficients c_n.

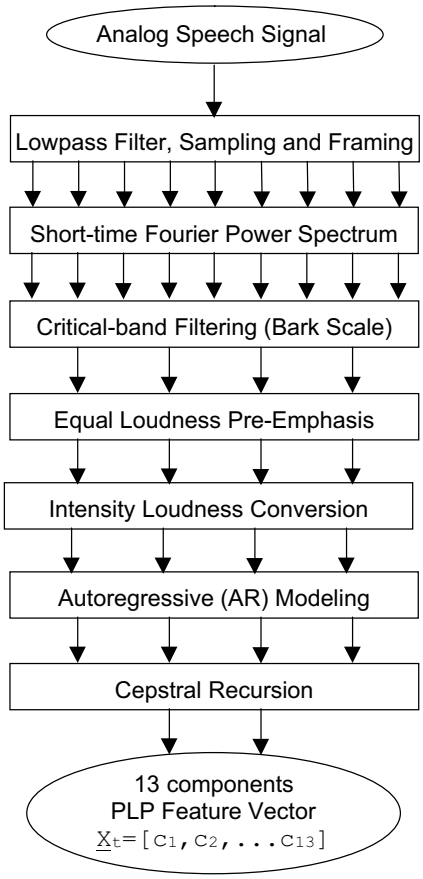


Fig. B.1: The algorithm of PLP feature extraction technique. It generates a 13-dimensional feature vector for each frame.

B.2 J-RASTA PLP

The J-RASTA PLP technique is an extension of the PLP with the aim of improving the robustness of PLPs to unknown spectral shaping (convolutive distortion) and noise (additive distortion) [HM94]. This is achieved by filtering the time-sequence of filter-bank coefficients, so as to remove the components of the input signal that vary more slowly or quickly than the speech signal. The algorithm comprises the following steps:

1. Compute the critical-band power spectrum, using the same configuration as in PLP.

2. Transform spectral amplitude through a compressing static nonlinear transformation. For the J-RASTA processing, the transform is chosen to be the Lin-Log transform:

$$y = \ln(1 + J\,x)$$

that is linear for $Jx \ll 1$ and logarithmic for $Jx \gg 1$. The rationale of using this transform is to have a characteristic with which both additive and convolutive distortions can be eliminated [HM94]. Although the value of J can be adapted during recognition to account for the different noise levels, we have used a fixed value for J ($J = 10^{-6}$) to avoid the complexity of having to train the recognizer with different values of J [HM94].

3. Filter the time trajectory of each transformed spectral component. The filter used in our experiments has the following Z-transform:

$$H(z^{-1}) = 0.1z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.94z^{-1}} \tag{B.2}$$

   The frequency response of the filter in the modulation filtering domain is shown in Fig. 7.5. This filter is a band-pass filter, with low-pass cut-off frequency of 0.956 Hz and a zero at 29.02 Hz. The modulation frequencies below and above the previous frequencies is effectively removed by the J-RASTA filter.

4. Transform the filtered speech representation through expanding static nonlinear transformation. This expanding transform is:

$$x = \frac{\exp(y)}{J} \tag{B.3}$$

   which is not exactly the inverse of the compressing transform to avoid negative coefficients.

5. As in conventional PLP, multiply by the equal loudness curve and raise to the power 0.33 to simulate the power law of hearing.

6. Compute an all-pole model of the resulting spectrum, as in conventional PLP feature extraction.

B.3 Modulation-Filtered Spectrogram

Modulation-filtered spectrogram (MSG) was originally developed by Brian E.D. Kingsbury [Kin98]. The objective is to generate visual displays of speech that are stable across a range of acoustic distortion, and to model the long term properties of speech. The MSG algorithm, which is illustrated in Figure B.2, is described as follows [Kin98]:

1. The speech signal is analyzed every 10 ms using the same time-frequency analysis as in the previous feature extraction algorithms, i.e. windowing with a 25 ms Hamming window, , FFT of 256 points and squaring to obtain the power spectrum of the windowed frame.

2. The power spectrum of each frame is convoluted with a bank of fourteen overlapping, triangular filters that are equally spaced in the Bark scale.

3. The critical-band power spectrum is converted into an amplitude spectrum by taking the square root of the filter bank output.

4. The time sequence of critical-band amplitude signals is then filtered by two different finite impulse response filters in parallel: a lowpass filter with a 0-8 Hz pass band and a bandpass filter with an 8-16 Hz pass band. It has been shown that the primary carrier of linguistic information in the speech signal are the changes in its spectral structure at rates between 1 and 16 Hz.

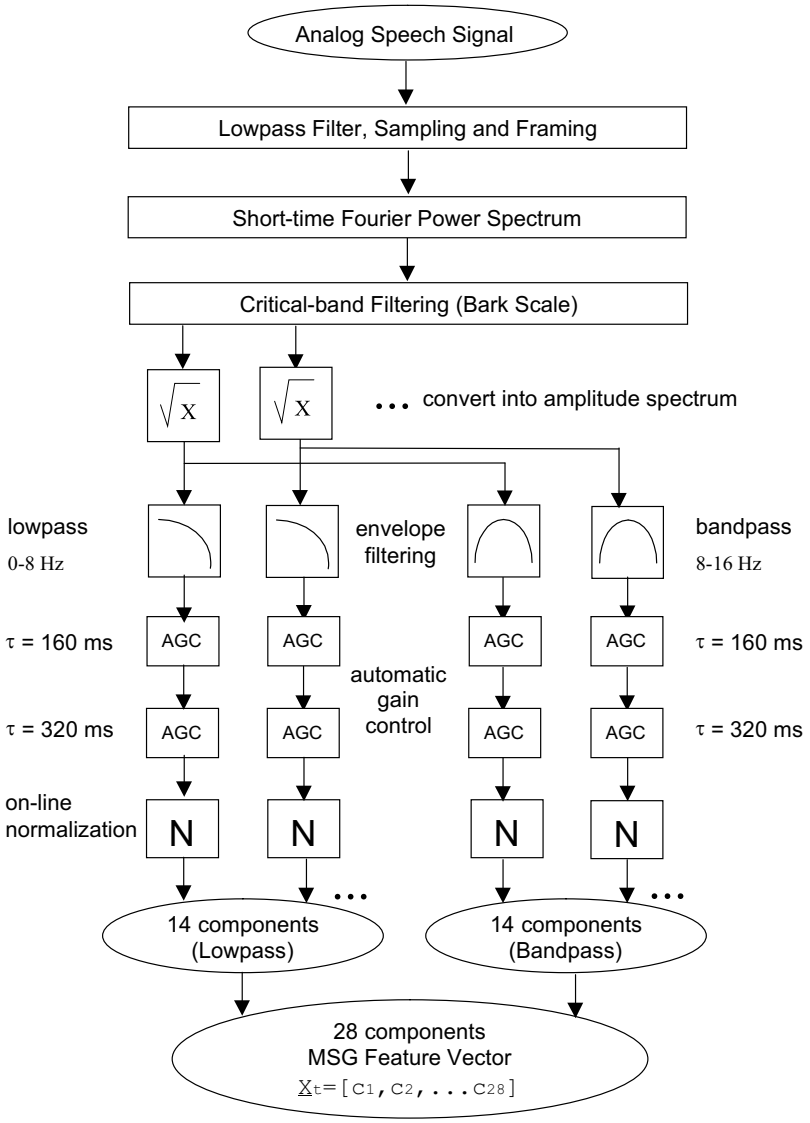


Fig. B.2: The algorithm of the MSG feature extraction technique. A 28-dimensional feature vector is generated for each input speech frame.

5. Both the low-pass and band-pass streams are processed through two feedback automatic gain control (AGC) units where the first AGC has a time constant of 160 ms and the second has a time constant of 320 ms. This AGC is essentially a square root compressor with a variable gain that depends on the dynamic of the input.

6. All features are normalized to have means of zero and variances of one using an online normalization procedure. The feature means and variances are adapted online using single pole lowpass filters with a time constant of 2 seconds.

The MSG feature vector used has thus 28 components. These 28 components, which are fourteen lowpass feature components and fourteen bandpass feature components, correspond to one point in a 28-dimensional continuous space. This vector being a function of time, it may be plotted for convenience reasons on a time-frequency plane, with bilinear smoothing used to produce the final image. The image, called the modulation spectrogram, is a visual representation of the speech signal and might be compared to the classical time-frequency Fourier transform, or wide-band spectrogram, under different signal to noise ratio (SNR) and reverberation conditions. It has been experimentally observed that the MSG is much less sensitive to noise and reverberation compared with a wide-band spectrogram [Kin98].

C. The Multivariate Omnibus Test

This statistical test for multivariate distributions is extracted from [DH94]. Let $X' = (X_1, \ldots, X_n)$ be a $p \times n$ matrix of n observations on a p-dimensional vector with sample mean and covariance $\bar{X} = n^{-1}(X_1 + \ldots + X_n)$ and $S = n^{-1}\check{X}\check{X}$ where $\check{X} = (X_1 - \bar{X}, \ldots, X_n - \bar{X})$.

Create a matrix with the reciprocals of the standard deviation on the diagonal:

$$V = \mathrm{diag}(S_{11}^{-1/2}, \ldots, S_{pp}^{-1/2}) \tag{C.1}$$

and form the correlation matrix $C = V\,S\,V$. Define the $p \times n$ matrix of transformed observations:

$$R' = H\,\Lambda^{-1/2}\,H'\,V\,\check{X}' \tag{C.2}$$

with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, the matrix with the eigenvalues of C on the diagonal. The columns of H are the corresponding eigenvectors, such that $H\,H' = I_p$ and $\Lambda = H'\,C\,H$. Using population values for C and V, a multivariate normal of p-dimensions can thus be transformed using the transform in Eq. C.2 into p independent normal random variables r_i.

Next the univariate skewness and kurtosis of the p normal random variables are computed. The univariate skewness and kurtosis are defined as:

$$\sqrt{b_{1i}} = \frac{\mathrm{E}\{r_i^3\}}{(\mathrm{E}\{r_i^2\})^{3/2}} \qquad b_{2i} = \frac{\mathrm{E}\{r_i^4\}}{(\mathrm{E}\{r_i^2\})^2} \tag{C.3}$$

where the expectation operator $\mathrm{E}\{.\}$ is substituted by the average over the samples in the matrix R.

Now defining $B_1' = (\sqrt{b_{11}}, \ldots, \sqrt{b_{1p}})$, $B_2' = (b_{21}, \ldots, b_{2p})$ and ι as a p-dimensional vector of ones, the test statistic used to test the normality of the samples is:

$$t = \frac{n\,B_1'\,B_1}{6} + \frac{n\,(B_2 - 3\iota)\,(B_2 - 3\iota)}{24} \sim \chi^2(2p) \tag{C.4}$$

which is approximately true if the number of samples n is large.

To test this statistic at a given confidence level α, we first compute the α-quantile of the $\chi^2(2p)$ distribution. If the value of the first term in Eq. C.4 is less than the α-quantile, then the hypothesis 'the samples X are normally distributed' (H=0) is accepted, otherwise the hypothesis is rejected (H=1). To have an idea of the reliability of the test, we also

compute the probability (SIG) that the values of the random variable t are larger than the observed value τ. A small value of this probability casts doubt on the null hypothesis (H=0).

This test has been applied to a sample of 10000 99-dimensional frames each obtained by concatenating 9 consecutive cepstral frames (cf. Sec. 2.3) as in our AURORA baseline system. The idea was to test whether the high-dimensional vectors assigned to each HMM state during segmentation were normally distributed or not. This question is important because we model the distribution of the high-dimensional vectors in each state with a normal distribution. This is done so as to easily compute the LDA matrix. Moreover, LDA is only optimum when the classes in the problem are normally-distributed with equal covariance matrices. If classes are not normal, then a non-linear transform, such as tandem or NLDA, may be more suitable for the problem (cf. Chapter 4).

As we can see in Tab. C.1, the null hypothesis, i.e. 'the samples X are normally distributed', is rejected for all the states in the acoustic model.

Tab. C.1: Results on the AURORA task of the omnibus test for multivariate normality applied performed for each of the HMM states of our AURORA baseline system (cf. Sec. 7.3.1).

index	H	SIG	τ	index	H	SIG	τ
1	1	0.000	18803.334	2	1	0.000	214.922
3	1	0.000	142.968	4	1	0.000	257.532
5	1	0.000	186.802	6	1	0.000	399.140
7	1	0.000	464.147	8	1	0.000	336.830
9	1	0.000	319.347	10	1	0.000	738.060
11	1	0.000	1088.001	12	1	0.000	2849.041
13	1	0.000	649.269	14	1	0.000	491.176
15	1	0.000	1647.328	16	1	0.000	2167.772
17	1	0.000	2926.284	18	1	0.000	3088.197
19	1	0.000	1207.500	20	1	0.000	1612.405
21	1	0.000	1382.024	22	1	0.000	2055.311
23	1	0.000	2340.929	24	1	0.000	961.823
25	1	0.000	290.990	26	1	0.000	5707.109
27	1	0.000	8145.369	28	1	0.000	582.506
29	1	0.000	4403.377	30	1	0.000	3947.823
31	1	0.000	5350.831	32	1	0.000	10121.061
33	1	0.000	6730.448	34	1	0.000	1205.426
35	1	0.000	684.989	36	1	0.000	709.152
37	1	0.000	3103.205	38	1	0.000	1363.626
39	1	0.000	1834.836	40	1	0.000	3223.378
42	1	0.000	1565.325	43	1	0.000	416.695
44	1	0.000	706.375	45	1	0.000	2190.368
46	1	0.000	2497.844	47	1	0.000	6321.939
48	1	0.000	434.474	49	1	0.000	402.830
50	1	0.000	1183.904	51	1	0.000	8900.903
52	1	0.000	8063.188	53	1	0.000	4423.925

index	H	SIG	τ	index	H	SIG	τ
54	1	0.000	996.764	55	1	0.000	284.626
56	1	0.000	2157.040	57	1	0.000	640.780
58	1	0.000	1589.951	59	1	0.000	5967.369
60	1	0.000	4470.749	61	1	0.000	3782.581
62	1	0.000	3882.799	63	1	0.000	3906.436
64	1	0.000	1401.568	65	1	0.000	637.178
66	1	0.000	5424.541	67	1	0.000	590.326
68	1	0.000	1470.743	69	1	0.000	3482.925
70	1	0.000	4038.993	71	1	0.000	4851.352
72	1	0.000	5340.950	73	1	0.000	3528.198
74	1	0.000	2424.902	75	1	0.000	693.144
76	1	0.000	1216.653	77	1	0.000	2580.229
78	1	0.000	1848.289	79	1	0.000	3412.022
80	1	0.000	19601.690	81	1	0.000	7759.270
82	1	0.000	3133.072	83	1	0.000	1154.494
84	1	0.000	831.406	85	1	0.000	1044.084
86	1	0.000	455.494	87	1	0.000	699.777
88	1	0.000	536.077	89	1	0.000	1849.438
90	1	0.000	1337.177	91	1	0.000	383.292
92	1	0.000	2215.499	93	1	0.000	1548.559
94	1	0.000	339.748	95	1	0.000	2186.692
96	1	0.000	1062.868	97	1	0.000	503.406
98	1	0.000	307.032	99	1	0.000	1548.478
100	1	0.000	3994.566	101	1	0.000	1853.915
102	1	0.000	1153.358	103	1	0.000	1308.439
104	1	0.000	2946.047	105	1	0.000	2648.991
106	1	0.000	1363.743	107	1	0.000	1348.794
108	1	0.000	4594.557	109	1	0.000	51272.300
110	1	0.000	8451.115	111	1	0.000	4782.682
112	1	0.000	870.844	113	1	0.000	1118.097
114	1	0.000	1897.745	115	1	0.000	8802.870
116	1	0.000	4270.577	117	1	0.000	1890.686
118	1	0.000	1758.572	119	1	0.000	1246.749
120	1	0.000	1514.934	121	1	0.000	1883.521
122	1	0.000	2218.971	123	1	0.000	1098.081
124	1	0.000	1176.425	125	1	0.000	1068.061
126	1	0.000	926.954	127	1	0.000	1149.767

D. The Lee Clustering Algorithm

This algorithm uses the entropy of the states and of the merged states to measure the information lost in the clustering process. To compute the entropy of the states we use the counts of the code-book symbols in a state q_i $N_i(k)$. These counts have been found during Baum-Welch training (cf. Sec. 2.6.2). The total number of counts in a state q_i is:

$$N_i = \sum_k N_i(k) \tag{D.1}$$

The emission probabilities of the code-book symbols in the states are obtained by normalizing the symbol counts, that is:

$$b_i(k) = \frac{N_i(k)}{N_i} \tag{D.2}$$

The entropy of state q_i can be computed using the emission probabilities:

$$H_i = \sum_k b_i(k) \ln(b_i(k)) \tag{D.3}$$

If we merge two states q_i and q_j, the counts of the code-book symbols in the merged state $q_{i'}$ are simply:

$$N_{i'}(k) = N_i(k) + N_j(k) \tag{D.4}$$

The entropy of the merged state $q_{i'}$ can also be computed as before. The information lost when merging the two states q_i and q_j can be computed as:

$$L(q_i, q_j) = N_{i'} H_{i'} - N_i H_i - N_j H_j \tag{D.5}$$

This measure is used in the clustering procedure to decide which states or clusters must be clustered. The clustering procedure is an iterative procedure that stops when the desired number of clusters is reached. The algorithm is sketched in the following steps:

1. an $M \times M$ table with the states in the rows and columns containing the loss of information $L(q_i, q_j)$ when states q_i and q_j are merged in each cell.

2. for each state q_i find the state q_j that minimizes the loss $L(q_i, q_j)$, and merge both states to form the cluster c_i.

3. compute an $M/2 \times M/2$ table containing the losses of information $L(c_i, c_j)$ when clusters c_i and c_j are merged.

4. for each cluster c_i find the cluster c_j that minimizes the loss $L(q_i, q_j)$, and merge both cluster to form the cluster c_i.

5. stop if the desired number of clusters is reached, otherwise return to step 3 and repeat the process.

E. Linear Discriminant Analysis (LDA)

If the classes used for discriminative feature reduction have equal covariance matrices, it can be shown [Fuk90] that class separability criteria can all be written in the following form:

$$\mathrm{J} = \mathrm{tr}(\mathbf{S}_1^{-1}\mathbf{S}_b) \tag{E.1}$$

where $\mathbf{S}_b$ is the generalized inter-class scatter matrix [Fuk90], and $\mathbf{S}_1$ can be the within-class scatter matrix $\mathbf{S}_w$ or the mixture scatter matrix $\mathbf{S}_m$.

In the usual linear discriminant analysis (LDA) the matrices $\mathbf{S}_1$ and $\mathbf{S}_b$ are defined as:

$$\begin{aligned} \mathbf{S}_1 &= \mathbf{S}_w = \sum_{i=1}^{N} P(\mathcal{C}_i)\ \mathbf{K}_i \\ \mathbf{S}_b &= \sum_{i=1}^{N} P(\mathcal{C}_i)\ (\mathbf{m}_i - \mathbf{m}_0)'(\mathbf{m}_i - \mathbf{m}_0) \end{aligned} \tag{E.2}$$

where the $P(\mathcal{C}_i)$ are the prior probabilities of the classes $\mathcal{C}_i$. It can be shown [Fuk90] that the optimum linear mapping C for the criterion in Equation E.1 in the sense of Definition 4.2 using the matrices in E.2 is the solution to the following eigenvalue problem:

$$(\mathbf{S}_w^{-1}\mathbf{S}_b)C = C\Lambda \tag{E.3}$$

where Λ is a diagonal matrix that contains the eigenvalues of the $S_m^{-1}S_b$ matrix, and the matrices $\mathbf{S}_m$ and $\mathbf{S}_b$ are defined in the n-dimensional input space of the mapping C. The differences between the projections of the LDA and PCA is illustrated in Fig. E.1 where two normal densities with different means and equal covariance matrices are depicted. We can clearly see that the overlapping between the densities in the LDA case is smaller than in the PCA case. In fact, the common area under both normal distributions, i.e. the overlapping, is the Bayes error, and the LDA projection leads, in the depicted case —normal densities with equal covariances—, to the optimum mapping in the Bayes sense. In [Fuk90] it has been demonstrated that the optimum mapping for the criterion J in Eq. E.1 over all possible mappings is given by the *a posteriori* probabilities of the classes, which implies that the criterion J in Eq. E.1 is consistent with the general criterion in the previous point.

If the clusters do not have equal covariance matrices but the data is still normally-distributed, then the optimal transform in the Bayes sense can be shown to be a quadratic

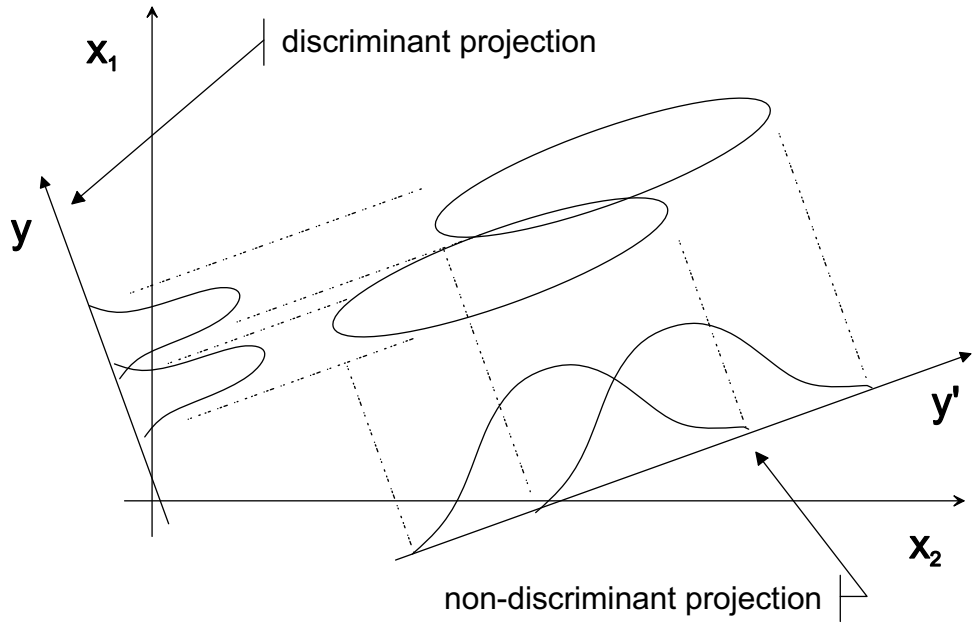


Fig. E.1: Difference between PCA and LDA. The area under both distributions is the Bayes error, and as can be readily seen this area is much larger for the PCA than for the LDA transform.

transform [Fuk90]. However, a linear transform is still preferred, e.g. HDA, in those cases due to its simplicity. In Fig. E.2 we can see a two-class problem where the normally-distributed classes have similar mean vectors but different covariance matrices. In this case the LDA criterion in Eq. E.1 does not work, since the classes are not separated by the scatter of means but rather by the scatter of covariance matrices. A possible solution is to use a different criterion which takes into account this last fact. A good criterion for the case in Fig. E.2 is the Bhattacharyya Distance criterion [Fuk90] which finds the directions along which the variances of the two classes are different, as can be seen in Fig. E.2. However, this criterion is only valid for two class problems, and is therefore not useful for feature reduction in ASR.

E.1 Relation Between LDA and Optimum Features

Actually it is possible to link the class separability criterion in Eq. E.1 to the MSE criterion in Definition 4.1. As demonstrated in [AO89] the class separability criterion in Eq. E.1 is equivalent to approximating in the MSE sense the posterior probabilities of the classes. To briefly explain this last assertion, let $C : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $D : \mathbb{R}^m \rightarrow \mathbb{R}^k$, and $F = (C \circ D)$ be three multidimensional affine (linear) mappings. If we want to compute the posterior probabilities of the classes at the output of F we can approximate the 1-of-c coded targets $\mathbf{t}$ as in the criterion of Definition 4.1, i.e.:

$$(C, D) = \arg, \min_{\mathrm{C,D}} \mathrm{E}\{\| \mathbf{t} - D(C(\mathbf{x})) \|^2\} \tag{E.4}$$

The solution to this criterion is not unique and some restrictions on the affine mappings must be imposed. This is clear since, assuming that both mappings are invertible, the

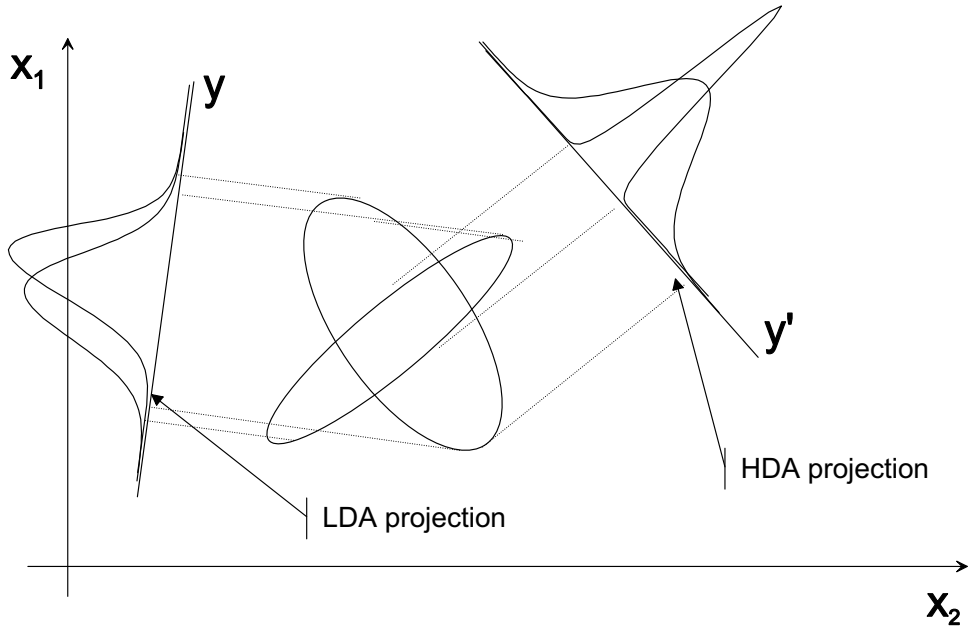


Fig. E.2: Comparison of LDA with HDA. As in Fig E.1 the area under both distributions is the Bayes error. Note that this area is smaller in the HDA case.

solutions (C, D) and (AC, DA^{-1}) are both solutions of the problem. Consequently, let us impose that the output u of the mapping C has null mean and identity covariance matrix, i.e.:

$$\begin{aligned} \mu_u &= 0 \\ \mathbf{K}_u &= I \end{aligned} \tag{E.5}$$

Our affine mappings are therefore:

$$\begin{aligned} C(x) &= \mathbf{A}'(\mathbf{x} - \mathbf{m}_x) \\ D(x) &= \mathbf{B}'u + \mu_y \end{aligned} \tag{E.6}$$

where the matrix $\mathbf{A}$ must satisfy the equation:

$$A'\mathbf{K}_x A = I \tag{E.7}$$

Using the Lagrange Multipliers method it can be found [AO89] that the matrices $\mathbf{A}$ and $\mathbf{B}$ must satisfy:

$$\begin{aligned} \mathbf{K}_{xy}\mathbf{K}_{yx}\mathbf{A} &= \mathbf{K}_x\mathbf{A}\Lambda \\ \mathbf{B} &= \mathbf{K}_{yx}\mathbf{A} \end{aligned} \tag{E.8}$$

If we now assume that the covariance matrix of the input data is non-singular we can write:

$$\mathbf{K}_x^{-1}\mathbf{K}_{xy}\mathbf{K}_{yx}\mathbf{A} = \mathbf{A}\Lambda \tag{E.9}$$

which is formally equivalent to the solution in Eq. E.3. Clearly, the total covariance of the input data $\mathbf{K}_x$ is equal to the inter-scatter matrix of the classes $\mathbf{S}_m$ in Eq. E.1. The difference lays in the matrix $\tilde{\mathbf{S}}_b$ which can be demonstrated to be equal to:

$$\tilde{\mathbf{S}}_b = \mathbf{K}_{xy}\mathbf{K}_{yx} = \frac{1}{N}\sum_{i=1}^{N} N_i^2(\mathbf{m}_i - \mathbf{m}_0)'(\mathbf{m}_i - \mathbf{m}_0) \tag{E.10}$$

As can be seen this introduces an emphasizing factor (N_i^2) which clearly emphasizes the classes appearing with higher frequency. As shown in Chapter 7 this emphasizing factor may be a problem when the prior probabilities of the classes are very different. It is in fact possible to extend this last result or interpretation to the case where the mapping D is still linear and C non-linear [Bis96, AO89].

List of Abbreviations

A

A/D Analog to Digital.

ANN Artificial Neural Network.

ARPA Advanced Research Projects Agency.

ASR Automatic Speech Recognition.

C

CDHMM Continuos Density Hidden Markov Models.

CMI Cumulative Mutual Information.

CMS Cepstral Mean Subtraction.

D

DCT Discrete Cosinus Transform.

DP Dynamic Programming.

DSR Distributed Speech Recognition.

E

EBP Errror Back Propagation.

ELRA European Language Resources Association.

EM Expectation-Maximization.

ETSI European Telecommunication Standards Institute.

F

FFT Fast Fourier Transform.

G

GMM Gaussian Mixture Model.

GPD Gradient Probabilistic Descent.

H

HDA Heteroscedastic Discriminant Analysis.

HMM Hidden Markov Models.

I

I/O Input/Output.

ITU International Telecommunications Union.

L

LBG Linde-Buzo-Gray.

LDA Linear Discriminant Analysis.

LDC Linguistic Data Consortium.

LPC Linear Predictive Coding.

M

MAP Maximum a Posteriori.

MCE Minimum Cross-Entropy.

MDT Missing Data Theory.

MFCC Mel-filter Cepstral Coefficients.

ML Maximum-Likelihood.

MLLT Maximum Likelihood Linear Transform.

MLP Multi Layer Perceptron.

MMI Maximum Mutual Information.

MSE Minimum Square Error.

MSG Modulation SpectroGram.

N

NIST National Institute for the Standardization of Technology.

NLDA Non-Linear Discriminant Analysis.

NN Neural Network.

O

OOV Out-of-Vocabulary.

P

PCA Principal Components Analysis.

PCM Pulse Codede Modulation.

PLP Predictive Linear Prediction.

PMC Parallel Model Combination.

PSD Power Spectral Density.

R

RASTA RelAtive SpecTrAl.

RBF Radial Basis Function.

RIL Relative Information Loss.

RTF Real Time Factor.

S

SCHMM Semi-Continuous Hidden Markov Models.

SLP Spoken Language Processing.

SNR Signal to Noise Ratio.

STT Speech-to-Text.

W

WER Word Error Rate.

Bibliography

[AG98] A. Arai and S. Greenberg. Speech intelligibility in the presence of cross-channel spectral asynchrony. In *Proc. ICASSP'98*, volume 2, pages 929–932, 1998.

[All94] Jont B. Allen. How do humans process and recognize speech? *IEEE Trans. Speech and Audio Processing*, 2(4):567–577, October 1994.

[AO89] H. Asoh and N. Otsu. Non-linear data analysis and multi-layer perceptrons. In *Int. Joint Conference on Neural Networks*, pages 411–415, 1989.

[Bau72] L.E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov chains. *Inequalities*, 3:1–8, 1972.

[BB93] E. Barnard and E.C. Botha. Back-propagation uses prior information efficiently. *IEEE Trans. on Neural Networks*, 4(5):794–802, September 1993.

[BBdSM86] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proc. ICASSP'86*, pages 49–52, Tokio, Japan, 1986.

[BC95] H. Bunke and J. Csirik. Parametric string edit distance and its application to pattern recognition. *IEEE Trans. Systems, Man and Cybernetics*, 25(1):202–206, January 1995.

[BCH98] E. Barnard, R.A. Cole, and L. Hou. Location and classification of plosive constants using expert knowledge and neural-net classifiers. *J. Acoust. Soc. Amer.*, 1998.

[BD96] H. Bourlard and S. Dupont. A new asr approach based on independent processing and recombination of partial frequency bands. In *Proc. ICSLP'96*, Philadelphia, October 1996.

[BD97] H. Bourlard and S. Dupont. Sub-band based speech recognition. In *Proc. ICASSP'97*, 1997.

[BDR96] Hervé Bourlard, Stéphan Dupont, and Christophe Ris. Multi-stream speech recognition. Technical report, IDIAP, December 1996.

[BDR97] Hervé Bourlard, Stéphan Dupont, and Christophe Ris. Robust speech recognition based on multi-stream features. In *Proc. ESCA'97*, 1997.

[BEG+96] M. Boros, W. Eckert, F. Gallwitz, G. Görz, G. Hanrieder, and H. Niemann. Towards understanding spontaneous speech: Word accuracy vs. concept accuracy. In *Proc. ICSLP '96*, volume 2, pages 1009–1012, Philadelphia, PA, 1996.

[Bel67] R. Bellman. *Dynamische Programmierung und selbstanpassende Regelprozesse*. Oldenburg, München-Wien, 1967.

[Bis96] Christopher M. Bishop. *Neural Networks For Pattern Recognition*. Oxford University Press, 1996.

[BJM83] L. Bahl, F. Jelinek, and R. Mercer. A maximum-likelihood approach to continuous speech recognition. *IEEE Trans. Pattern Anal. Machine Intelligence*, 5(2):179–190, 1983.

[BM94] H. Bourlard and N. Morgan. *Connectionist Speech Recognition: A Hybrid Approach*, volume 247 of *Kluwer international series in Engineering and Computer Science*. Kluwer Academic Publishers, 1994.

[Bol79] S.F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27:113–120, 1979.

[Bou99] H. Bourlard. Non-stationary multi-channel (multi-stream) processing: Towards robust and adaptive asr. In *Proc. ESCA Workshop on Robust Methods for Speech Recognition in Adverse Conditions*, pages 1–9, 1999.

[BW90] H. Bourlard and C.J. Wellekens. Links between markov models and multilayer perceptrons. *IEEE Trans. Pattern Anal. Machine Intelligence*, 12(2):1167–1178, 1990.

[BWK99] F. Beaufays, M. Weintraub, and Y. Konig. Discriminative mixture weight estimation for large gaussian mixture models. In *Proc. ICASSP'99*, 1999.

[CGJV99] M.P. Cooke, P.D. Green, L. Josifovski, and A. Vizinho. Robust speech recgnition with missing and unreliable acoustic data. Technical Report CS-99-05, Department of Computer Science, University of Sheffield, 1999.

[CGM94] M.P. Cooke, P.D. Green, and M.D.Crawford. Handling missing data in speech recognition. In *Proc. ICSLP'94*, pages 1555–1558, Yokohama, 1994.

[CGN95] R. Crouch, R. Gaizauskas, and K. Netter. Report of the study groups on assessment and evaluation. Technical Report cmp-lg/9601003, European Commission DG XIII, April 1995.

[CKRB93] F. Class, A. Kaltenmeier, and P. Regel-Brietzmann. Optimization of an hmm-based continuous speech recognizer. In *Proc. EUROSPEECH'93*, 1993.

[CLA00] H. Christensen, B. Lindberg, and O. Andersen. Employing heteregeneous information in a multi-stream framework. In *Proc. ICASSP'2000*, 2000.

[CMG96] M. Cooke, A. Morris, and P. Green. Recognising occluded speech. In *Proc. ESCA Workshop on the Auditory Basis of Speech Peception 1996*, pages 15–19, July 1996.

[Coo91] M.P. Cooke. *Modelling auditory processing and organisation.* PhD thesis, Department of Computer Science, University of Sheffield, 1991.

[DDC99] K. Demuynck, J. Duchateau, and D. Van Compernolle. Optimal feature subspace selection based on discriminant analysis. In *Proc. EUROSPEECH'99*, 1999.

[DGL96] L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Number 31 in Springer Series on Applications of Mathematics. Springer-Verlag, New-York, 1996.

[DH94] J.A. Doornik and H. Hansen. An omnibus test for univariate and multivariate normality. Economics papers W4&91, Economics Group, Nuffield College, University of Oxford, November 1994.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statis. Soc. Ser. B*, 1(39):1–22, 1977.

[DM80] S. Davis and P. Mermelstein. A comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 4(28):357–366, 1980.

[DPH93] John R. Deller, Jr., John G. Proakis, and John H. Hansen. *Discrete Time Processing of Speech Signals.* Prentice Hall PTR, 1993.

[DR01] S. Dupont and C. Ris. Multi-band with contaminated training data. In *Workshop on Consistent and Reliable Cues for sound analysis (CRAC)*, Aalborg, Denmark, September 2001.

[DRM83] B. Dautrich, L. Rabiner, and T. Martin. On the effects of varying filter bank parameters on isolated word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 680–683, 1983.

[EB00] Daniel P.W. Ellis and Jeff A. Bilmes. Using mutual information to design feature combinations. In *icslp00*, Beijing, China, 2000.

[EG01] D.P.W. Ellis and M.J. Reyes Gómez. Investigations into tandem acoustic modelling for the AURORA task. In *Proc. EUROSPEECH'01*, 2001.

[Ell99] D.P.W. Ellis. The icsi respite aurora multi-stream recognizer. http://www.icsi.berkeley.edu/ dpwe/respite /multistream/aurora1999.html, 1999. Diary of the results on the AURORA 1999 database.

[EM99] Dan Ellis and Nelson Morgan. Size matters: and empirical study of neural network training for large vocabulary continuous speech recognition. In *Proc. ICASSP'99*, 1999.

[ESS01] D.P.W. Ellis, R. Singh, and S. Sivadas. Tandem acoustic modelling in large-vocabulary recognition. In *Proc. ICASSP'2001*, Salt Lake City, 2001.

[ETS00] ETSI. Speech processing, transmission and quality aspects (stq): Distributed speech recognition; advance front-end feature extraction algorithm;compression algorithms. ETSI standard (ES) ETSI ES 201 108, European Telecommunications Standards Institute (ETSI), October 2000.

[ETS02] ETSI. Speech processing, transmission and quality aspects (stq): Distributed speech recognition; advance front-end feature extraction algorithm;compression algorithms. ETSI standard (ES) ETSI ES 202 050, European Telecommunications Standards Institute (ETSI), October 2002.

[FF98] J. Fritsch and M. Finke. Applying divide and conquer to large scale pattern recognition tasks. In G. Orr and K.-R. Müller, editors, *Neural Networks: tricks of the trade*, Lecture Notes in Computer Science, pages 315–342. Springer Verlag, 1998.

[Fis97] J.G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *IEEE Workshop on Automatic Speech Recognition and Understanding*, Santa Barbara, California, USA, 1997.

[FRB97] V. Fontaine, C. Ris, and J.M. Boite. Nonlinear discriminant analysis for improved speech recognition. In *Proc. EUROSPEECH'97*, volume 4, pages 2071–2074, Rhodes, 1997.

[Fuk90] Keinosuke Fukunaga. *Introduction To Statistical Pattern Recognition (2nd Ed.)*. Academic Press Professional, Inc., 1990.

[Fur86] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:52–59, 1986.

[Gai98] R. Gaizauskas. Evaluation in languagea and speech technology. *Computer Speech and Language*, 12(4):249–262, October 1998.

[GAVF98] J.S. Garofolo, C.G.P. Auzanne, E.M. Voorhees, and B. Fisher. The trec spoken document retrieval track: A success story. In *Proc. 8th Text Retrieval Conference TREC-8*, pages 107–130, Gaithersburg, Maryland, November 1998.

[GB00] H. Glotin and F. Berthommier. Test of several external posterior weighting funcitons for multiband full combination asr. In *Proc. ICSLP'2000 Beijing*, Beijing,China, 2000.

[GBCJ01] P. Green, J. Barker, M. Cooke, and L. Josifovski. Handling missing and unreliable information in speech recognition. In *AISTATS'01, 8th International Workshop on Artificial Intelligence and Statistics*, Key-West, Florida, January 2001.

[GC89] L. Gillick and S.J. Cox. Some statistical issues in the comparison of speech recognition algorithms. In *Proc. ICASSP'89*, pages 532–535. IEEE, 1989.

[Ghi94] O. Ghitza. Auditory models and human performance in tasks related to speech coding and speech recognition. *IEEE Trans. Speech and Audio Processing*, 1(2):115–131, 1994.

[Gre97] S. Greenberg. Auditory function. In M. Crocker, editor, *Encyclopaedia of Acoustics*, pages 1301–1323. John Wiley and Sons Inc., 1997.

[Gus97] Dan Gusfield. *Algorithms on String, Trees and Sequences*. Cambridge University Press, 1st edition, 1997.

[GVSJ97] John S. Garofolo, Ellen M. Voorhees, Vincent M. Stanford, and Karen Sparck Jones. TREC-6 1997 spoken document retrieval track overview and results. In *Text REtrieval Conference*, pages 83–91, 1997.

[GY93] M.J.F. Gales and S.J. Young. Hmm recognition in noise using parallel model combination. In *Proc. EUROSPEECH'93*, 1993.

[Hag01] A. Hagen. *Robust Speech Recognition based on multi-stream Processing*. PhD thesis, EPFL, 2001.

[Hai98] U. Haiber. *Sprecheradaption in einem Spracherkennungssystem mit stochastischer Modellierung*. Berichte aus der Mathematik. Shaker Verlag, November 1998.

[Her90] H. Hermansky. Perceptual linear predictive (plp) analysis for speech. *J. Acoust. Soc. Amer.*, pages 1738–1752, 1990.

[HES00] H. Hermansky, D.P.W. Ellis, and S. Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Proc. ICASSP'2000*, Istambul, 2000.

[HM94] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Trans. Speech and Audio Processing*, 2(4):587–589, October 1994.

[HM00] A. Hagen and A. Morris. Comparison of hmm experts with mlp experts in the full combination multi-band approach to robust asr. Technical report, IDIAP, July 2000.

[HP00] H.G. Hirsch and D. Pearce. The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ASR: challenges for the next millenium*, Paris, September 2000. ISCA ITRW ASR 2000.

[HS99] H. Hermansky and S. Sharma. Temporal patterns (traps) in asr of noisy speech. In *Proc. ICASSP'99*, Phoenix, Arizona, March 1999.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feed-forward networks are universal approximators. *Neural Networks*, 2(5):59–366, 1989.

[HSZ96] W. Hackbush, H.R. Schwarz, and E. Zeidler. *Teubner-Taschenbuch der Mathematik*. B.G. Teubner, 1996.

[HUN92] R. Haeb-Umbach and H. Ney. Linear discriminant analysis for improved large vocabulary continuous speech recognition. In *Proc. ICASSP'92*, volume 1, pages 13–16, 1992.

[JCGV99] L. Josifovski, M. Cooke, P. Green, and A. Vizinho. State based imputation of missing data for robust speech recognition and speech enhancement. In *Proc. EUROSPEECH'99*, pages 2837–2840, Budapest, 1999.

[JEM99] A. Janin, D. Ellis, and N. Morgan. Multi-stream speech recognition: ready for prime time? In *Proc. EUROSPEECH'99*, Budapest, 1999.

[JH56] R. Jakobson and M. Halle. *Fundamentals of Language*. Janua Linguarum 1, The Hague, 1956.

[JH96] J.C. Junqua and J.P. Haton. *Robustness in Automatic Speech Recognition*. Kluwer Academic, 1996.

[JJNH91] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, and G.E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[Jun93] J.-C. Junqua. The lombard reflex and its role on human listerners and automatic speech recognizers. *J. Acoust. Soc. Amer.*, 93(1):510–524, 1993.

[JW89] J.-C. Junqua and H. Wakita. A comparative study of cepstral lifters and distance measures for all-pole models of speech in noise. In *Proc. ICASSP'89*, pages 476–479, 1989.

[KA98] N. Kumar and A.G. Andreou. Heteroscedastic discriminant analysis and reduced rank hmms for improved speech recognition. *Speech Communication*, 25(4), 1998.

[KAO94] T. Kurita, H. Asoh, and N. Otsu. Nonlinear discriminant features constructed by using outputs of multilayer perceptron. In *Proceedings of the IEEE International Symposium on Speech, Image Processing and Neural Networks*, pages 417–420, April 1994.

[KB00] K. Kirchhoff and J.A. Bilmes. Combination and joint training of acoustic classifiers for speech recognition. In *Proc. ISCA ITRW Workshop on automatic speech recognition (ASRU 2000)*, 2000.

[KFS00] K. Kirchhoff, G.A. Fink, and G. Sagerer. Conversational speech recognition using acoustic and articulatory input. In *Proc. ICASSP'2000*, 2000.

[KGG89] B. Koo, J. Gibson, and S. Gray. Filtering of colored noise for speech enhancement and coding. In *Proc. ICASSP'89*, pages 349–352, 1989.

[KHDM98] J. Kittler, M. Hatef, R.P.W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Machine Intelligence*, 20(3), March 1998.

[Kin98] B. Kingsbury. *Perceptually Inspired Signal-processing Strategies for Robust Speech Recognition in Reverberant Environments*. PhD thesis, University of California, Berkeley, 1998.

[Kir99] K. Kirchhoff. *Robust Speech Recognition Using Articulatory Information*. PhD thesis, University of Bielefeld, 1999.

[Kuh94] T. Kuhn. *Die Erkennungsphase in einem Dialogsystem*. PhD thesis, Universität Erlangen-Nürnberg, 1994.

[LB91] P. Lockwood and J. Boudy. Experiments with a non-linear spectral subtraction (nss), hidden markov models and the projection for robust speech recognition in cars. In *Proc. EUROSPEECH'91*, pages 79–82, 1991.

[LBB+98] S. Lawrence, I. Burns, A. Back, A.C. Tsoi, and C.L. Giles. *Neural Networks: tricks of the trade*, chapter Neural Network Classification and Prior Class Probabilities, pages 299–314. Lecture Notes in Computer Science. Springer, 1998.

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Comm.*, 28:84–95, 1980.

[LC97] R.P. Lippmann and B.A. Carlson. Using missing feature theory to actively select features for robust speech recognition with interruptions , filtering and noise. In *Proc. EUROSPEECH'97*, pages 37–40, 1997.

[Lee90] K.F. Lee. Context-dependent phonetic hidden markov models for speaker independent continuous speech. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 1990.

[Leo84] R.G. Leonard. A database for speaker independent digit recognition. In *Proc. ICASSP'84*, volume 3, 1984.

[Lev66] V. I. Levensthein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966.

[Lip87] R.P. Lippmann. An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing Magazine*, April 1987.

[LMP87] R.P. Lippmann, E.A. Martin, and D.B. Paul. Multi-style training for robust isolated-word speech recognition. In *Proc. ICASSP'87*, pages 705–708, Dallas, USA, April 1987.

[LOB78] J. Lim, A. Oppenheim, and L. Braida. Evaluation of an adaptive comb filtering method for enhancing speech degraded by white noise addition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(4):354–358, 1978.

[LT71] H. Lane and B. Tranel. The lombard sign and the role of hearing in speech. *Journal of Speech and Hearing Research*, 14:677–709, 1971.

[Mar89] J.N. Marcus. Significance test for comparing speech recognizer performance using small test sets. In *Proc. EUROSPEECH'89*, volume 2, pages 465–468, 1989.

[MB95] N. Morgan and H. Bourlard. Neural networks for statistical recognition of continuous speech. *Proc. of the IEEE*, 83(5):742–770, May 1995.

[ME91] N. Merhav and Y. Ephraim. Hidden markov modelling using a dominat state sequence with application to speech recognition. *Computer Speech and Language*, 5(4):327–339, 1991.

[MHB99] Andrew Morris, Astrid Hagen, and Hervé Bourlard. The full combination sub-bands approach to noise robust hmm/ann based asr. In *Proc. EUROSPEECH'99*, Budapest, Hungary, September 5–10 1999.

[MHC03] J. Marí-Hilario and F. Class. A comparative study of some feature reduction algorithms on the AURORA 2000 and the daimlerchrysler in-car asr task. In *Proc. EUROSPEECH'03*, pages 3101–3104, Geneva, September 2003. ISCA.

[MHGB99] Andrew Morris, Astrid Hagen, Herv Glotin, and Herv Bourlard. Multi-stream adaptive evidence combination for noise robust asr. IDIAP-RR 26, IDIAP, 1999.

[MM99] N. Mirghafori and N. Morgan. Sooner or later: exploring asynchrony in multi-band speech recognition. In *Proc. EUROSPEECH'99*, Budapest, 1999.

[MN98] D. Macho and C. Nadeu. On the interaction between time and frequency filtering of speech parameters for robust speech recognition. In *Proc. ICSLP'98*, volume 4, pages 1487–90, 1998.

[Moo77] R.K. Moore. Evaluating speech recognizers. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 25(2), April 1977.

[MRGS98] P.J. Moreno, B. Raj, E. Gouvea, and R.M. Stern. Multi-variate gaussian-based cepstral normalization for robust speech recognition. In *Proc. ICASSP'95*, volume 1, pages 141–144, May 1998.

[Mul69] Z. Muljačić. *Fonologia generale e fonologia della lingua italiana.* Il Mulino, 1969.

[NO99] Hermann Ney and S. Ortmanns. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, September 1999.

[NO00] H. Ney and S. Ortmanns. Progress in dynamic programming search for lvcsr. *Proc. of the IEEE*, 88(8):1224–1240, August 2000.

[PFB88] P.J. Price, W. Fisher, and J. Bernstein. A database for continuous speech recognition in a 1000 word domain. In *Proc. ICASSP'88*, volume 1, pages 651–654, New York, 1988.

[PL95] E. Paulus and M. Lehning. Die evaluierung von spracherkennungssystemen in deutschland. Technical Report 70, Verbmobil, TU Braunschweig, 1995.

[PMP01] J. Psutka, L. Mueller, and J.V. Psutka. Comparison of mfcc and plp parametrizations in the speaker independent continuous speech recognition task. In *Proc. EUROSPEECH'01*, Aalborg, 2001.

[Pow85] M.J.D. Powell. Radial basis functions for multi-variable interpolation: a review. Technical Report DAMPT/NA12, Dept. of Applied Mathematics and Theoretical Physics, University of Cambridge, 1985.

[RHW86] D. Rummelhart, G. Hinton, and R.J. Williams. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, chapter Learning internal representations by error propagation, pages 318–362. MIT Press, Cambridge, MA, 1986.

[RHWR96] W. Reichl, S. Harengel, F. Wolfertstetter, and G. Ruske. Neural networks for nonlinear discriminant analysis in continuous speech recognition. Verbmobil-Report 111, Techn. Universität München, April 1996.

[RJ93] L. Rabiner and B-H Juang. *Fundamentals of Speech Recognition*. Signal Processing series. Prentice-Hall, 1993.

[RLS94] A. Rosenberg, C.-H. Lee, and F. Soong. Cepstral channel normalization techniques for hmm-based speaker verification. In *Proc. ICSLP'94*, pages 1835–1838, 1994.

[RMB91] S. Renals, N. Morgan, and H. Bourlard. Probability estimation by feed-forward networks in continuous speech recognition. Technical report, ICSI, 1991.

[RMB+94] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in hmm speech recognition. *IEEE Trans. Speech and Audio Processing*, 1994.

[Rog94] G. Rogova. Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781, 1994.

[Roh76] V.K. Rohatgi. *An Introduction to Probabilitiy Theory and Mathematical Statistics*. John Wiley & Sons, 1976.

[Ros00] R. Rosenfeld. Two decades of statistical language modelling: Where do we go from here. *Proc. of the IEEE*, 88(8):1270–1278, August 2000.

[RR95] W. Reichl and G. Ruske. A hybrid rbf-hmm system for continuous speech recognition. In *Proc. ICASSP'95*, pages 3335–3338, Detroit, May 1995.

[RW98] G. Rigoll and D. Willett. Ann/hmm hybrid for continuous speech recognition with a discriminant nonlinear feature extraction. In *Proc. ICASSP'98*, 1998.

[Sch96] J. Schürmann. *Pattern Classification: a Unified View of Statistical and Neural Approaches*. John Wiley & Sons, Inc., 1996.

[SEK+00] S. Sharma, D. Ellis, S. Kajarekar, P. Jain, and H. Hermansky. Feature extraction using non-linear transformation for robust speech recognition on the AURORA database. In *Proc. ICASSP'2000*, Istambul, 2000.

[SH03] S. Sivadas and H. Hermansky. In search of target definition in tandem feature extraction. In *Proc. EUROSPEECH'03*, Geneva, September 2003. ISCA.

[Shi00] M. Shire. *Discriminant Training of Front-End and Acoustic Modelling Stages to Heterogeneous Acosutic Environments for multi-stream Automatic Speech Recognition*. PhD thesis, University of California, Berkeley, USA, 2000.

[SKKW03] M. Schafföner, M. Katz, S.E. Krüger, and A. Wendemuth. Improved robustness of automatic speech recognition using a new class definition in linear discriminant analysis. In *Proc. EUROSPEECH'03*, pages 2841–2844, Geneva, September 2003.

[SL92] E. Singer and R.P. Lippmann. A speech recognizer using radial basis function neural networks in an hmm framework. In *Proc. ICASSP'92*, pages 629–632, San Francisco, 1992.

[SPGC00] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen. Maximum likelihood discriminant feature spaces. In *Proc. ICASSP'2000*, 2000.

[SRR00] J. Stadermann, J. Rottland, and G. Rigoll. Tied posteriors: A new hybrid speech recognition technology with generic capabilities and high portability. In *ITRW ASR2000 Workshop on Automatic Speech Recognition: Challenges for the New Millenium*, Paris, France, September 2000. ISCA.

[SSN87] Q. Summerfield, A. Sidwell, and T. Nelson. Auditory enhancement of changes in spectral amplitude. *J. Acoust. Soc. Amer.*, 81(3):700–708, March 1987.

[TG96] K. Tumer and J. Ghosh. Anlysis of decision boundaries in linearly combined neural classifier. *Pattern Recognition*, 29:341–348, 1996.

[TH97] S. Tibrewala and H. Hermansky. Multi-band and adaption approaces to robust speech recognition. In *Proc. EUROSPEECH'97*, pages 2619–2622, 1997.

[TRP00] M. Thomae, G. Ruske, and T. Pfau. A new approach to discrimiantive feature extraction usign model transformation. In *Proc. ICASSP'2000*, pages 1615–1618, 2000.

[Val95] Valtcho Valtchev. *Discriminative Methods in HMM-based Speech Recognition*. PhD thesis, University of Cambridge, 1995.

[VM90] A. P. Varga and R. K. Moore. Hidden markov model decomposition of speech and noise. In *Proc. ICASSP'90*, pages 845–848, 1990.

[WF74] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.

[WKM98] C.J. Wellekens, J. Kangasharju, and C. Milesi. The use of meta-hmm in multistream hmm training for automatic speech recognition. In *Proc. ICSLP'98*, pages 2991–2994, 1998.

[WKMG98] S.L. Wu, B.E.D. Kingsbury, N. Morgan, and S. Greenberg. Integrating information from syllable-length time scales into automatic speech recognition. In *Proc. ICASSP'98*, 1998.

[WN82] J. Woodward and T. Nelson. An information theoretic measure of speech recogntion performance. In David Pallett, editor, *Proc. of Workshop on Standardization for Speech I/O Technology*, Naval Air Development Center, Warminster, PA, 1982.

[WRBB95] R.M. Warren, K.R. Riener, J.A. Bashford, and B.S. Brubaker. Spectral redundancy: Intellegibility of sentences heard through narrow spectral tilts. *Perception and Psycophysics*, 2(57):175–182, 1995.

[YC98] S.J. Young and L.L. Chase. Speech recognition evaluation: a review of the u.s. csr and lvcsr programmes. *Computer Speech and Language*, 12(4):263–279, 1998.