Hierarchical Connectionist Acoustic Modeling for Domain-Adaptive Large Vocabulary Speech Recognition

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
von der Fakultät für Informatik
der Universität Karlsruhe (Technische Hochschule)
genehmigte

Dissertation

von

Jürgen Fritsch
aus Offenburg

Tag der mündlichen Prüfung:	25. Oktober 1999
Erster Gutachter:	Prof. Dr. A. H. Waibel
Zweiter Gutachter:	Dr. A. J. Robinson

"The most beautiful thing we can experience is the mysterious. It is the source of all true art and science. He to whom this emotion is a stranger, who can no longer pause to wonder and stand wrapped in awe, is as good as dead"

Albert Einstein.

Abstract

This thesis presents a new, hierarchical framework for connectionist acoustic modeling in large vocabulary statistical speech recognition systems. Based on the divide and conquer paradigm, the task of estimating HMM state posteriors is decomposed and distributed in the form of a tree-structured architecture consisting of thousands of small neural networks. In contrast to monolithic connectionist models, our approach scales to arbitrarily large state spaces. Phonetic context is represented simultaneously at multiple resolutions which allows for scalable acoustic modeling. We demonstrate that the hierarchical structure allows for (1) accelerated score computations through dynamic tree pruning, (2) effective speaker adaptation with limited amounts of adaptation data and (3) downsizing of the trained model for small memory footprints.
The viability of the proposed hierarchical model is demonstrated in recognition experiments on the *Switchboard* large vocabulary conversational telephone speech corpus, currently considered the most difficult standardized speech recognition benchmark, where it achieves state-of-the-art performance with less parameters and faster recognition times compared to conventional mixture models.
The second contribution of this thesis is an algorithm that allows for domain-adaptive speech recognition using the proposed hierarchical acoustic model. In contrast to humans, automatic speech recognition systems still suffer from a strong dependence on the application domain they have been trained on. Typically, a speech recognition system has to be tailored to a specific application domain to reduce semantic, syntactic and acoustic variability and thus increase recognition accuracy. Unfortunately, this approach results in a lack of portability as performance typically deteriorates unacceptably when moving to a new application domain.
We present Structural Domain Adaptation (SDA), an algorithm for hierarchically organized acoustic models that exploits the scalable specificity of phonetic context modeling by modifying the tree structure for optimal performance on previously unseen application domains. We demonstrate the effectiveness of the SDA approach by adapting a large vocabulary conversational telephone speech recognition system to (1) a telephone dictation task and (2) spontaneous scheduling of meetings. SDA together with domain-specific dictionaries and language models allows to match the performance of domain-specific models with only 45-60 minutes of acoustic adaptation data.

Zusammenfassung

Die vorliegende Arbeit präsentiert einen neuen, hierarchischen Ansatz für die konnektionistische akustische Modellierung in statistischen Spracherkennungssystemen für große Wortschätze. Basierend auf dem Teile-und-Herrsche Paradigma werden a-posteriori Wahrscheinlichkeiten von HMM Zuständen in einer verteilten, in Form eines Baumes strukturierten Architektur mit Hilfe mehrerer Tausend kleiner neuronaler Netze geschätzt. Im Gegensatz zu monolithischen konnektionistischen Architekturen skaliert der vorgestellte Ansatz auf beliebig große Zustandsräume. Phonetische Kontexte werden dabei simultan in mehreren Auflösungen repräsentiert wodurch skalierbare akustische Modellierung ermöglicht wird. Es wird gezeigt, daß die hierarchische Architektur (1) beschleunigte Evaluation mittels dynamischem Pruning, (2) effektive Sprecheradaptation mit nur geringen Mengen an Adaptionsdaten und (3) nachträgliche Verkleinerung eines trainierten Modells erlaubt.

Die Leistungsfähigkeit des vorgeschlagenen hierarchischen Modells wird anhand von Erkennungsexperimenten mit dem *Switchboard* Korpus bestehend aus spontansprachlichen Telefonkonversationen, dem derzeit schwierigsten standardisierten Spracherkenner Bechmark, demonstriert. Die vorgeschlagene Architektur erzielt dabei eine Erkennungsleistung vergleichbar zu den derzeit leistungsfähigsten Systemen, benötigt dazu jedoch deutlich weniger Parameter und Rechenzeit.

Der zweite Beitrag dieser Arbeit ist ein Algorithmus der domänen-adaptive Spracherkennung mit der vorgeschlagenen hierarchischen Architektur ermöglicht. Heutige Spracherkennungssysteme leiden immer noch an einer starken Abhängigkeit von der Anwendungsdomäne für die sie trainiert wurden. Typischerweise muß ein Spracherkennungssystem auf eine bestimmte Anwendungsdomäne hin zugeschnitten werden um die semantische, syntaktische und akustische Variabilität so weit wie möglich einzuschränken und dadurch die Erkennungsleistung zu verbessern. Unglücklicherweise führt ein solcher Ansatz zu einem Mangel an Portabilität, ersichtlich daran, daß die Erkennungsleistung stark einbricht, wenn das System auf einer neuen, andersartigen Domäne angewendet wird.

Wir präsentieren Strukturelle Domänenadaption (SDA), einen Algorithmus für hierarchisch organisierte akustische Modelle, der die Skalierbarkeit der Spezifizität der Kontextmodellierung ausnutzt um die Baumstruktur des Modells an die Gegebenheiten in einer neuen Anwendungsdomäne anzupassen, um die Erkennungsleistung zu optimieren. Die Effektivität des Algorithmus wird anhand zweier Adaptionsexperimente demonstriert. Dabei wird ein auf der Switchboard Domäne trainiertes System auf (1) eine Telefon-Diktierdomäne und (2) eine spontansprachliche Dialogdomäne portiert. SDA zusammen mit domänen-spezifischen Wörterbüchern und Sprachmodellen erlaubt dabei eine Erkennungsleistung, die der Leistung domänen-spezifischer Erkenner entspricht, dabei jedoch nur 45-60 Minuten an Adaptationsdaten aus der jeweiligen Zieldomäne benötigt.

Acknowledgements

First, I want to thank my thesis advisor Alex Waibel for supporting and guiding me and my research right from the start, for enabling me to work in an inspiring research environment and most importantly, for his unflagging efforts to provide our research group with the funding that made it all possible.
Also, I wish to thank Tony Robinson, the co-advisor of my thesis. I can hardly think of someone better suited to this task than him. His work on connectionist acoustic modeling has always been a great source of inspiration to me. I'm thankful for his thorough comments that have greatly improved this thesis.
I am very grateful to Michael Finke without whom many things would not have been possible. He designed an outstanding speech recognition toolkit that allowed me to investigate my ideas in a world class competitive research environment. He supported me throughout my PhD and I'd be happy if he had half as much fun working with me as I had working with him.
I would also like to thank Detlef Koll for good collaboration in almost two years of joint work, for lots of fruitful discussions and for giving me critical reviews on my work.
I would like to thank everyone who has been involved in the Switchboard project in our labs, particularly Michael, Torsten Zeppenfeld, Petra Geutner, and Klaus Ries.
Many thanks to everyone who helped proofreading my thesis: Detlef, Michael, Rainer Stiefelhagen and Corinna Habeck. Also, I would like to thank Michael, Torsten, Thomas Polzin and Monika Woszczyna for their hospitality during my numerous visits to Pittsburgh.
I wish to thank all other colleagues in the Interactive Systems Labs, both at the University of Karlsruhe and at Carnegie Mellon University. There are too many to name them all individually, but I would like to mention Ivica Rogina and Hermann Hild who introduced me to the fascinating field of automatic speech recognition several years ago.
Special thanks to Silke Dannenmaier for administrative support and Frank Dreilich for maintaining the computing facilities which were crucial for carrying out my research.
I would also like to thank my parents Ingeborg and Günther and all the rest of my family for their unconditional support.
Finally, I wish to express deep gratitude to Corinna for her love, sacrifice and encouragement and for always being there.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the past years, research in speech recognition systems has improved the state-of-the-art significantly, such that a wide range of new, speech enabled applications have become possible. Consequently, the market in speech technology and applications develops rapidly. For instance, high quality, speaker-independent continuous-speech *dictation systems*, which formerly have only been available in research labs and for specific application domains, now are available for the general purpose mass market and will soon be integrated into computer operating systems. Another emerging application of speech technology is in *Interactive Voice Response (IVR)* enabled call centers, where people now can call fully automated information systems and retrieve selective information by communicating with the system in a natural dialog instead of by hitting the touch tone buttons of the telephone. A very interesting application of speech technology emerges in the field of *multimedia information retrieval*, where speech recognition systems are increasingly being used to categorize and transcribe radio and TV broadcast news for the purpose of indexing and content classification. Such technology appears to be a major factor in managing, accessing and filtering the huge amounts of information spilled out by the mass media.

1.1 Motivation

Despite all these promising and exciting applications, speech recognition technology still struggles with a lot of unresolved problems. For instance, speech recognition systems have to be tailored to specific application domains in order to at least approach performance comparable to humans. As a consequence, performance drops unacceptably when the system is applied to domains different from the originally targeted domain. Therefore, there currently is no universal speech recognition system available that, for any given language, would work in any environment, recognizing an arbitrary vocabulary in an arbitrary application domain. Rather, a typical speech

recognition system requires that detailed operating conditions are met in order to achieve optimal performance:

- Specification of the type of microphone to be used
- Specification of recording conditions (e.g., quiet office)
- Specification of a finite recognition vocabulary
- Specification of an application domain (e.g., dictation of financial newspaper articles, transcription of telephone conversations)

Due to the statistical nature of current speech recognition technology based on Hidden Markov Models (HMM), which implies that system parameters are learned from a large but finite set of training patterns, a restriction to a specific application domain appears to be absolutely necessary for achieving reasonable performance. The resulting lack of robustness and universality in most of the components of such systems has been identified as a major weakness of today's speech recognition technology.
While domain specific vocabularies, pronunciation dictionaries and language models can typically be obtained and exchanged easily in order to accommodate a switch to a new domain, the exchange or adaptation of the acoustic model of a speech recognizer requires considerably more effort. The task of the acoustic model is to estimate the probability of acoustic observations (suitably parameterized representation of an acoustic waveform) given a sequence of words. Typically, an acoustic model consists of a set of HMMs with associated mixture density probability models. Mismatches in the acoustic model are not only caused by changes in the recording conditions, the type of microphone or the speaker characteristics, as one might think at first glance. Basic acoustic units such as phones are modeled in various alternative realizations, depending on their phonetic context. This strategy has become standard practice in large vocabulary speech recognition and improves recognition accuracy considerably but renders the acoustic model highly dependent on domain specific components such as the vocabulary and the language model. As a result, we typically observe a mismatch in the specificity of context modeling in addition to an acoustic mismatch in a new target domain. As the specificity of context modeling can not be altered easily in conventional acoustic models because of a lack of structure and scalability, the standard approach is to build and train a new domain-specific acoustic model to replace the existing one as soon as the phonetic characteristics of the target domain differ significantly from those of the training domain. Unfortunately, this approach is time-, labour- and cost-intensive as it requires large amounts of transcribed acoustic data.

As an alternative to conventional acoustic modeling based on mixture densities, several researchers have independently developed a methodology for incorporating connectionist models (based on neural networks) into a statistical speech recognition system. Such systems are often called hybrid speech recognition systems since they combine discriminatively trained connectionist acoustic models with the standard HMM framework. In the most simple setting, a single artificial neural network is applied to the estimation of posterior phone or state probabilities. While offering better discrimination, faster evaluation and a smaller number of parameters, monolithic connectionist acoustic models are difficult to scale to context-dependent modeling and have therefore been used primarily for monophone modeling. However, the fact that state posteriors and state priors are both explicitly available offers attractive potential for adapting these models to domains different from the training domain.

1.2 Approach

This thesis presents a new, hierarchical framework for connectionist acoustic modeling that, among other beneficial properties, allows to dynamically adapt the specificity of context modeling to new, previously unseen application domains. The tree-structured model offers all the advantages of conventional connectionist acoustic models while offering a variety of beneficial new properties such as scalability to any number of HMM states and fast evaluation through dynamic tree pruning. The model is demonstrated to be effective in modeling up to 24000 states with connectionist estimators, achieving performance comparable to standard mixture based acoustic models while being smaller in size and faster to decode.

We motivate, introduce and evaluate the proposed hierarchical connectionist architecture as an alternative to standard mixture based modeling in large vocabulary conversational speech recognition. Experimental evaluation of architectural aspects is performed on the *Switchboard* corpus, currently a major focus in the speech research community. Switchboard contains more than 170 hours of telephone quality recordings of human-to-human conversations over the public telephone network. As such it exhibits strong variations in recording quality and background noise. Even worse, the conversational nature of the recordings implies a high proportion of disfluencies such as false starts, hesitations, interjections, etc. As one can imagine, Switchboard is a comparably hard speech recognition domain. Today's state-of-the-art systems yield performance in the range of 30-40% word errors.

The hierarchical structure of the proposed model together with its connectionist framework for estimating both state posteriors and state priors leads to the second major contribution of this thesis: An algorithm for dynamically adapting the structure, the size and the predictors of a trained hierarchical connectionist acoustic

model for efficient adaptation of a speech recognition system to a previously unseen domain. In *Structural Domain Adaptation (SDA)*, as the algorithm is called, a comparably small amount of transcribed data from the new domain is used to estimate the prior distribution of the HMM states of the original model in the new domain. Typically, this distribution is quite different from the one obtained on the original training corpus, due to differences in vocabulary and language model. Using an estimate of the state prior distribution on the new domain, we adapt the priors in each node of the modeling tree by propagating the state priors through the tree structure. Typically, certain branches of the modeling tree will be pruned as they lead to HMM states with very low observation counts. In addition, SDA allows to further prune the resulting tree structure according to the observation counts, for instance to downsize the acoustic model for small memory footprint and/or faster evaluation. Thus, the hierarchical architecture together with the algorithm for structural adaptation represent a versatile tool for domain-adaptive acoustic modeling.

We evaluate structural domain adaptation of our hierarchical model using two quite different application scenarios. The baseline to both experiments is a model trained on the Switchboard corpus. In the first scenario, we adapt this model's structure to a domain consisting of read newspaper articles, the Wall Street Journal (WSJ) domain. In contrast to the majority of this corpus, we selected a subset consisting of telephone quality speech in order to keep acoustic differences small. In the second scenario, we port the Switchboard model to a domain called English Spontaneous Scheduling Task (ESST), consisting of spontaneous human-to-human conversations. This corpus is recorded with high-quality microphones and exhibits a comparably small and restricted vocabulary. Using the SDA algorithm, we demonstrate that the Switchboard trained hierarchical connectionist model can be adapted effectively to the unseen domains using only 45-60 minutes of acoustic adaptation data. The resulting structurally adapted systems match the performance of domain-specific systems trained on several hours of data.

1.3 Outline

On a global level, this thesis is divided into two major parts. Chapters 2 to 4 are of introductory nature, summarizing specific aspects about automatic speech recognition that are of relevance to the remainder of the thesis. The following chapters 5 to 10 are then devoted to the original contributions of this thesis.

Chapter 2 introduces the reader to the field of statistical speech recognition. Rather than presenting all the bells and whistles of the state-of-the-art, this review of the basic concepts in automatic speech recognition is meant to provide the necessary background for readers unfamiliar with this discipline. Therefore, it only briefly

touches certain aspects that are less relevant for the content of later chapters. Also, this chapter is restricted to the presentation of standard technology, namely the one build around continuous density Hidden Markov Models. Phonetic context modeling, a very important modeling technique now found in any large vocabulary speech recognition system is reviewed. The focus here is to introduce the reader to the side effects of context modeling: on the one hand, it significantly improves performance; on the other hand it introduces a strong dependence on the specific training domain which can dramatically decrease robustness towards unseen domains. As this thesis presents a solution to this problem, this part might be regarded as both a review and a motivation for later chapters.

Chapter 3 is devoted to connectionist acoustic modeling and hybrid architectures. Here, it is shown how classifier neural networks can be integrated into the HMM formalism in order to take advantage of properties such as better class discrimination and faster model evaluation. The chapter presents and discusses architectures that have been used for hybrid speech recognition in the past.

Chapter 4 is devoted to the Switchboard large vocabulary conversational speech recognition (LVCSR) corpus. This widely used corpus serves as a benchmark for the architectures and algorithms presented in this thesis.

Chapter 5 introduces *hierarchical* connectionist acoustic modeling as a conceptual framework for tree-structured, scalable acoustic models. We theoretically motivate the derivation of this divide-and-conquer based architecture which is grounded on statistical factoring of posterior state probabilities. We discuss properties of the resulting tree based classifiers focusing on design algorithms for inducing tree structures. We show how such a model can be integrated into the classical HMM framework for the purpose of acoustic modeling. Furthermore, we analyze the viability of feed forward classifier neural networks for the task of estimating conditional posterior probabilities in tree nodes. The resulting hierarchical acoustic model, which we call a Hierarchy of Neural Networks, is then integrated into a state-of-the-art speech recognizer and evaluated on the Switchboard LVCSR corpus.

In Chapter 6, we experimentally analyze dynamic posterior based pruning of the model introduced in chapter 5. This technique is very efficient in avoiding to evaluate posterior probabilities of unlikely states, thereby reducing the computational load of acoustic model evaluation significantly. We show how this simple technique allows to efficiently trade-off recognition accuracy against decoding speed.

Chapter 7 presents an algorithm for optimally adapting a hierarchical acoustic model to the acoustic characteristics of specific speakers, an important prerequisite for many applications of speech recognition (e.g., dictation). In contrast to existing acoustic models such as those based on mixture densities, the proposed model does not require additional model parameter tying mechanisms such as regression class trees to achieve efficient speaker adaptation even with small amounts of data. We demonstrate how

the hierarchical structure of the model itself can be exploited for speaker adaptation and evaluate the resulting algorithm on Switchboard.
Building on the material from chapter 5 and 7, chapter 8 introduces Structural Domain Adaptation (SDA), an algorithm for adapting a hierarchical acoustic model to unseen domains. For that purpose it uses a combination of tree pruning and node adaptation which also adjusts the size of the model to the new domain. We present details of the SDA algorithm and evaluate it by porting a Switchboard trained model to two unseen domains of quite different nature, namely Wall Street Journal (WSJ) data and English Spontaneous Scheduling Task (ESST) data. We demonstrate how SDA applied to the hierarchical Switchboard model allows to efficiently and effectively adapt the recognizer to the new domains, requiring only 45-60 minutes of speech from those domains.
Chapter 9 is devoted to a related hierarchical architecture that has been developed as part of this thesis, so called *mixture trees*. Here, the emphasis was on hierarchically structuring the components of a standard, likelihood based acoustic model, in order to take advantage of the scaling and adaptation properties found for tree structured connectionist acoustic models. We derive an EM algorithm for estimating the parameters of such a model and evaluate it on the Switchboard corpus. Also, we compare it with the connectionist counterpart presented in earlier chapters.
Chapter 10 presents strategies for static and dynamic combination of multiple, possibly heterogeneous acoustic models in an attempt to improve recognition accuracy over each one of the contributing models. In contrast to existing frame-level combination approaches, we present an approach that achieves a reduction in word error rate through a dynamic combination of a conventional likelihood based model and the proposed hierarchical connectionist model.
Finally, chapter 11 summarizes the main contributions of this thesis and concludes with a discussion of possible future work.

Chapter 2

Statistical Speech Recognition

This chapter presents the main concepts of the state-of-the-art in statistical speech recognition. It introduces *Hidden Markov Models (HMM)* and their application to automatic speech recognition. Since the focus of this thesis lies in acoustic modeling, we restrict this presentation to aspects relevant to later chapters such as context-dependent modeling and the resulting domain dependence of acoustic models, and only briefly touch topics such as preprocessing and language modeling. For further details on specific aspects that could not be included in this chapter, the author refers the reader to the excellent reviews in [Rabiner '89, Huang et al. '90, Rabiner & Juang '93, Young '96, Jelinek '97]. Readers already familiar with the basic statistical framework of automatic speech recognition based on hidden Markov models may want to skip this chapter.

2.1 Overview

The basic unit of interest in statistical speech recognition is the posterior probability of word sequences $W_1, \ldots, W_N$ given a sequence of acoustic observation vectors $\mathbf{x_1}, \ldots, \mathbf{x_M}$ and a set of model parameters $\mathbf{\Theta}$

$$p(W_1, \ldots, W_N | \mathbf{x}_1, \ldots, \mathbf{x}_M, \mathbf{\Theta}).$$

The sequence of acoustic observation vectors consists of a condensed and suitably transformed and preprocessed representation of the actual sampled speech waveform which contains a lot of redundancy. Fig. 2.1 depicts how uttered word sequence and acoustic realization are linked by this fundamental probability density.
On a rather high level of description, a statistical speech recognition system consists of the following parts:

- A suitable framework for modeling the above probability.

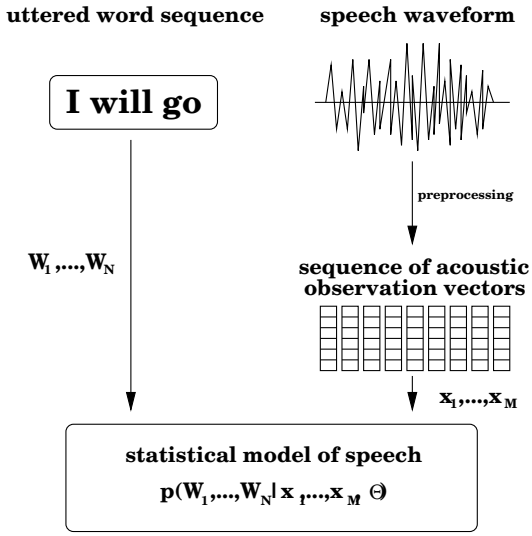

Figure 2.1: Overview: statistical speech recognition

- A method for estimating the parameters of the model. This is called the *estimation problem.*
- A method for decoding/searching the most likely word sequence, given some acoustic observation. This is referred to as the *recognition problem.*

From the early beginnings of speech recognition research, the single most important modeling framework that has been applied to statistical speech recognition has been the concept of a hidden Markov model (HMM). Today, this technique for modeling temporal sequences has evolved and been refined substantially in the context of speech recognition. Consequently, HMM based speech recognition systems dominate the field. Many alternative modeling frameworks have been shown to be just instances or special cases of HMMs. We will discuss HMMs in detail in section 2.3.1.
Training or estimation of such a model consists of finding model parameters $\mathbf{\Theta}$ that maximize the above posterior probability on a certain amount of training data

$$\hat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} \prod_{t=1}^{T} p(W_1^{(t)}, \ldots, W_{N(t)}^{(t)} | \mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_{M(t)}^{(t)}, \mathbf{\Theta}).$$

Here, the term training data refers to a collection of T training sentences, each one consisting of the uttered word sequence and the corresponding sequence of acoustic observation vectors.
When applying a trained model to the problem of speech recognition, we seek to find the sequence of words that maximizes the posterior probability for a given sequence of acoustic observation vectors and fixed model parameters $\mathbf{\Theta}$

$$\hat{W_1}, \ldots, \hat{W_N} = \arg\max_{W_1,\ldots,W_N} p(W_1, \ldots, W_N | \mathbf{x}_1, \ldots, \mathbf{x}_M, \mathbf{\Theta}).$$

Bayes' rule allows to factor the posterior probability of word sequences as follows:

$$p(W_1, \ldots, W_N | \mathbf{x}_1, \ldots, \mathbf{x}_M) = \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_M | W_1, \ldots, W_N)\ P(W_1, \ldots, W_N)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_M).}$$

To avoid unnecessary confusion, we have omitted the explicit dependence on $\mathbf{\Theta}$. This rule allows to separate the estimation process into the so called *acoustic model (AM)* consisting of terms that depend on the acoustic observations $\mathbf{x_1}, \ldots, \mathbf{x_M}$ and the *language model (LM)* consisting of terms that depend only on the sequence of words $W_1, \ldots, W_N$. Since the term in the denominator does not depend on the sequence of words, it can be omitted in the search for the most likely word sequence leading to the following simplified maximization problem

$$\hat{W_1}, \ldots, \hat{W_N} = \arg\max_{W_1,\ldots,W_N} p(\mathbf{x}_1, \ldots, \mathbf{x}_M | W_1, \ldots, W_N)\ P(W_1, \ldots, W_N).$$

In the remainder of this chapter, we will discuss some of the issues in statistical speech recognition in more depth.

2.2 Preprocessing

As the raw speech waveform contains a lot of redundancy, speech recognition systems usually employ some form of preprocessing to periodically extract relevant information about speech sounds in form of so called acoustic feature vectors from the speech signal. Although there are many different preprocessing techniques, most of them are based on short time spectral analysis or linear prediction [Rabiner & Schafer '78]. Fig. 2.2 depicts the basic principle in preprocessing speech waveforms. Typically, a Hamming window,

$$h(t) = 0.54 - 0.46\cos(\frac{2\pi t}{D}) \qquad \text{for} \qquad t \in [0, D],$$

of duration $D = 10 \ldots 40$ msec is used to extract a short segment of speech from which a representation such as the Fourier power spectrum is computed. The window is

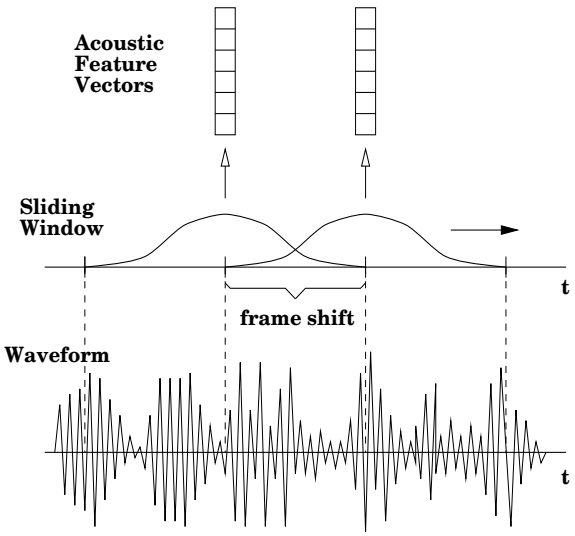

Figure 2.2: Preprocessing: short time spectral analysis

shifted in discrete steps of 5-20 msec, thereby allowing to compute short time power spectra at a rate of 50-200 frames per second.

Usually, the computation of the power spectrum is just the first step in a whole series of transformations and normalizations. For instance, a very popular preprocessing strategy consists of computing Mel-scale Frequency Cepstral Coefficients (MFCCs) [Hunt et al. '80, Davis & Mermelstein '80]. In that case, the power spectra are first transformed into the Mel-scale [Davis & Mermelstein '80], a perceptually motivated logarithmic frequency scale that emphasizes low frequency components. Next, the cosine transformation is applied to the Mel-scale spectra, resulting in so called cepstra. MFCCs are often modified to include a non-linear warping of the frequency axis in order to compensate different vocal tract lengths across different speakers [Cohen et al. '95]. Furthermore, in order to increase the robustness against different microphones and recording conditions the cepstra are often normalized for mean zero and unit variance which is called cepstral mean compensation [Beattie & Young '92]. The resulting feature vectors are sometimes further transformed using principal component analysis (PCA) [Jolliffe '86] or linear discriminant analysis (LDA) [Haeb-Umbach & Ney '92] to reduce the correlation among coeffi-

cients, increase class separability and/or to reduce the final feature dimensionality. No matter what specific sequence of transformations is being used, all preprocessing techniques aim at extracting highly condensed representations of speech from the waveform to be recognized while preserving all the information necessary for discriminating the different speech sounds in later stages.

2.3 Acoustic Modeling

By applying Bayes' rule to factor the estimation process into acoustic model and language model, we have separated the vector of model parameters into parameter subsets Θ^{AM} and Θ^{LM}, respectively. The task of acoustic modeling in statistical speech recognition is to estimate the subset $\mathbf{\Theta^{AM}}$ of acoustic model parameters which maximize

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_M | W_1, \ldots, W_N, \mathbf{\Theta^{AM}}).$$

Words W_i are modeled as sequences (or graphs) of *phone models*. The mapping from words to phone models is usually accomplished by means of a *pronunciation dictionary*. Phone models in turn are modeled by hidden Markov models in order to capture their temporal and acoustic variability.

2.3.1 Hidden Markov Models for Speech Recognition

A first-order hidden Markov model (HMM) is a probabilistic automaton

$$\lambda = \{S, \pi, A, B, V\}$$

consisting of the components

S, the set of HMM states, $S = \{s_1, \ldots, s_n\}$

π, the probability distribution over the states, where π_i is the probability that state s_i is initial.

A, the matrix of transition probabilities, where a_{ij} is the probability that state s_j follows state s_i.

B, the set of emission probability densities $B = \{b_1, \ldots, b_n\}$, where $b_i(\mathbf{x})$ models the conditional probability of observing/emitting feature vector $\mathbf{x}$ in state s_i.

V, the set of observed features which can be discrete (*discrete HMM*) or continuous (*continuous density HMM*). In the case of speech recognition, continuous density HMMs operate on a continuous multivariate representation of the speech signal while discrete HMMs operate on a finite set of discrete symbols that are obtained from the continuous feature space by means of a vector quantizer [Gersho & Gray '92]. Continuous density HMMs typically outperform discrete HMMs in speech recognition due to a better resolution of the acoustic feature space.

An HMM models a stochastic state-based process, starting at some initial state. At each time step, a new feature vector is generated (emitted) according to the current state's emission probability density followed by a transition to a new state according to the current state's transition probability distribution. HMMs get their name from the fact that the sequence of states generating the observable sequence of feature vectors is hidden.
In the context of HMMs, there are 3 well known problems [Rabiner '89], all of which have solutions in form of efficient algorithms:

- **Evaluation Problem:** Given a sequence of observation vectors and an HMM, what is the probability that the sequence has been generated by the HMM? Using a dynamic programming approach, it can be shown that this problem can be efficiently solved in time $O(n^2T)$, where n is the number of states and T is the length of the sequence. The corresponding algorithm is called *Forward algorithm*.

- **Decoding Problem:** Given a sequence of observation vectors and an HMM, what is the most likely sequence of HMM states for generating the observed sequence? Again, there is an efficient solution to this problem in time $O(n^2T)$ via a dynamic programming approach. In the case of the decoding problem, the resulting algorithm is called *Viterbi algorithm*.

- **Optimization Problem:** Given a sequence of observation vectors and an HMM topology, estimate the parameters of the HMM so as to maximize the likelihood of the sequence being generated by the HMM. There is no analytical solution to this problem. However, there is an efficient iterative method, the *Expectation Maximization (EM) algorithm* [Dempster et al. '77] which can be applied. In the context of speech recognition, the specific form of this algorithm is often called Forward-Backward or Baum-Welch algorithm.

For the purpose of speech recognition, a specific form of HMMs, namely first-order n-state left-right HMMs, are being applied to model basic speech units such as phones

and/or syllables (see Fig. 2.3). The assumption here is that speech is a sequential process exhibiting great variability in the realization and duration of specific phones. Furthermore, the asymptotic complexity of the Forward and Viterbi algorithms typically reduces to $O(nT)$ in a left-right HMM as there is only a small constant number of valid local transitions from each state.

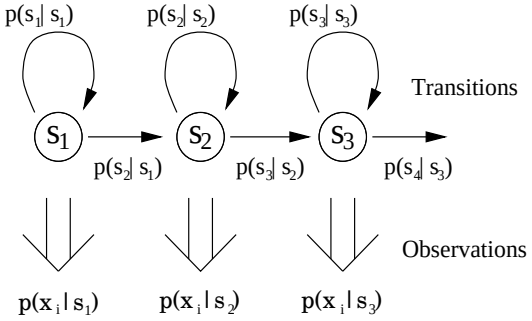

Figure 2.3: First-order 3-state left-right HMM

Modeling the inventory of phones in a specific language using the HMM model of Fig. 2.3, we can identify the acoustic model parameters of a speech recognizer to be the collection of all HMM parameters, $\Theta = \lambda$. Fig. 2.4 shows the process of converting a sequence of words into

1. a pronunciation graph (containing pronunciation variants) and
2. an HMM state graph,

which allows us to formulate the problem of recognizing words from speech via the standard HMM framework. In this framework, where word sequences are represented as directed acyclic graphs of HMM states, the likelihood of an acoustic observation can be rewritten as (omitting the dependence on $W_1, \ldots, W_N$ of the right hand side for simplicity)

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_M | W_1, \ldots, W_N) = \sum_{s_1, \ldots, s_M} p(\mathbf{x}_1, \ldots, \mathbf{x}_M | s_1, \ldots, s_M)\ P(s_1, \ldots, s_M)$$

where the summation extends over all possible state sequences $s_1, \ldots, s_M$ in the HMM model for the word sequence $W_1, \ldots, W_N$. In the Viterbi approximation, the above likelihood is approximated by the probability of the most likely state sequence

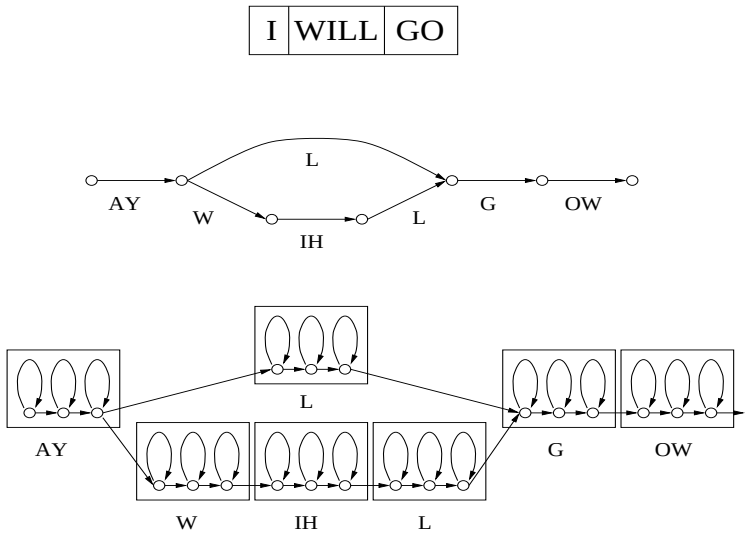

Figure 2.4: Typical hidden Markov model in speech recognition

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_M | W_1, \ldots, W_N) \approx \max_{s_1, \ldots, s_M} p(\mathbf{x}_1, \ldots, \mathbf{x}_M | s_1, \ldots, s_M)\ P(s_1, \ldots, s_M)$$

Given a specific state sequence, the likelihood of the acoustic observations given that sequence and the sequence prior probability can be factored as follows

$$\begin{aligned} p(\mathbf{x}_1, \ldots, \mathbf{x}_M | s_1, \ldots, s_M) &= \prod_{i=1}^{M} p(\mathbf{x}_i | \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}, s_1, \ldots, s_M) \\ P(s_1, \ldots, s_M) &= \prod_{i=1}^{M} P(s_i | s_1, \ldots, s_{i-1}) \end{aligned}$$

When applying first-order hidden Markov models to the estimation of such likelihoods one makes two simplifying assumptions:

- Independence of Observations:

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_M | s_1, \ldots, s_M) = \prod_{i=1}^{M} p(\mathbf{x}_i | s_1, \ldots, s_M)$$

- First-order Assumption (Observations depend only on the current state, transitions depend only on the previous state instead of on the whole history of states):

$$\begin{aligned} p(\mathbf{x}_1, \ldots, \mathbf{x}_M | s_1, \ldots, s_M) &= \prod_{i=1}^{M} p(\mathbf{x}_i | s_i) \\ P(s_1, \ldots, s_M) &= \prod_{i=1}^{M} P(s_i | s_{i-1}) \end{aligned}$$

Clearly, these assumptions do not hold for speech where successive feature vectors often exhibit high correlation and co-articulation effects can influence the realization of phones over several 100 msecs. Nevertheless, first-order hidden Markov models are widely used to model speech units, mostly because of the availability of efficient estimation and decoding algorithms. Also, many techniques have been developed over time which extenuate the effects resulting from the above assumptions.

2.3.2 Emission and Transition Modeling

Mainstream speech recognition systems follow the above approach by modeling emission probability distributions $p(\mathbf{x}|s_i)$ and transition probabilities $P(s_i|s_{i-1})$ separately and independently. Emission probability distributions are usually modeled using mixture densities from the exponential family, such as the mixture of Gaussians

$$\begin{aligned} p(\mathbf{x}|s_i) &= \sum_{j=1}^{n} \gamma_j N_j(\mathbf{x}|s_i) \\ N_j(\mathbf{x}|s_i) &= \frac{1}{\sqrt{(2\pi)^d |\Sigma_{ij}|}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_{ij})\Sigma_{ij}^{-1}(\mathbf{x} - \mu_{ij})^t \right\} \end{aligned}$$

where the γ_j denote *mixture coefficients* and the N_j Gaussian *mixture component densities* in a d-dimensional space with mean vectors μ_{ij} and covariance matrices Σ_{ij}. Often, full covariance Gaussians (1) can not be estimated reliably due to data sparsity, (2) in total require more than the available amount of memory, or (3) are too expensive to evaluate. In such cases, one typically assumes diagonal covariance Gaussians:

$$N_j(\mathbf{x}|s_i) = \frac{1}{\sqrt{(2\pi)^d \prod_{k=1}^{d} \sigma_{ijk}^2}} \exp\left\{ -\frac{1}{2} \sum_{k=1}^{d} \frac{(x_k - \mu_{ijk})^2}{\sigma_{ijk}^2} \right\}$$

Transition probabilities on the other hand are modeled by simple multinomial probabilities since they are conditioned on a discrete variable only and not on the input vector[1].
The advantage of this approach is a decoupled estimation process that separates temporal and acoustic modeling. As such, it allows to easily vary HMM state topologies after training in order to modify temporal behavior. For instance, minimum duration constraints in phone models can easily be enforced by expanding the model from a single to multiple states with tied observation probabilities [Robinson et al. '96].
However, the disadvantage of the above approach is a mismatch in the dynamic range of emission and transition probabilities. The reason is that transition probabilities are modeled separately as multinomial probabilities, constrained by the requirement to sum to one. This leads to a dominant role of emission probabilities with transition probabilities hardly influencing overall system performance [Bengio '96].

2.4 Phonetic Context Modeling

So far we have assumed that only a single HMM is used to model each monophone (see Fig. 2.4). Since the English language can be modeled by approximately 40-50 monophones, one might get the impression that only that number of HMM models need to be trained.

2.4.1 From Monophones to Triphones to Polyphones

However in practice, one observes an effect called co-articulation that causes large variations in the way specific monophones actually sound, depending on their phonetic context [Chow et al. '86]. Usually, explicit modeling of phones in phonetic context yields substantial gains in recognition accuracy [Lee '88]. However, it is not immediately clear how to achieve robust context-dependent modeling. Consider, for example, so called *triphone* models. A triphone essentially represents the realization of a specific monophone in a specific context spanning one phone to the left and right. For instance, in the HMM state graph of Fig. 2.4 the two occurrences of monophone L correspond to two different triphone models, namely L(AY,G) and L(IH,G)[2]. Assuming an inventory of 50 monophones, the number of (theoretically) possible triphones is $49 * 50 * 49 = 120050$. Many of these triphones will occur rarely or never in actual speech due to the linguistic constraints in the language. Separate modeling of all triphones therefore does not make sense as it leads to poor gener-

[1] It should be noted that it is possible to enhance the HMM formalism such that it allows to condition transition probability distributions on observation vectors.

[2] Here, the first argument denotes the left and the second argument the right neighboring phone.

alization due to unreliable parameter estimates. The problem becomes even more evident when generalizing triphones to so called *polyphones* by allowing dependence on a wider context, and not just the immediate left and right neighboring phones. To avoid this data sparsity problem, one can either apply smoothing techniques based on interpolation of polyphone models with more robust ones, or introduce a mechanism for sharing parameters across different polyphones models. The latter approach in conjunction with decision trees has become the most popular context modeling technique and will be discussed next.

2.4.2 Phonetic Decision Trees

Decision trees [Safavian & Landgrebe '91] can be applied to cluster observed polyphones into generalized context classes according to acoustic and phonetic similarity [Bahl et al. '91]. Typically, a separate CART [Breiman et al. '84] like decision tree is constructed for each HMM state of each monophone by top-down clustering of all observed polyphonic contexts of the respective monophone state. Through the use of categorical questions about specific attributes at each internal node, decision trees allow to generalize to unseen classes. This property is essential for modeling polyphonic contexts in speech recognition where decision trees allow to generalize to previously unseen polyphonic contexts that might occur during decoding.
For phonetic context modeling, a finite and meaningful set of categorical questions about phonetic contexts has to be defined. The most straight-forward questions are those relating to a specific neighboring monophone. For illustration, consider Table 2.1 containing 6 words and their phonetic transcription. We examine context modeling for the monophone AX which is contained in all 6 words. Note that the transcriptions in Table 2.1 have been arbitrarily aligned around the phone AX for easy comparison of phonetic contexts. Even when restricting context modeling to immediate neighboring phones, five of the six occurrences of the monophone AX in Table 2.1 correspond to different triphones[3].
Categorical questions for building a phonetic decision tree for monophone AX could contain simple questions for specific monophones. For instance, consider the question 'Is monophone R at position +1 ?', in other words, 'Is monophone R an immediate right neighboring phone?'. This question induces two sets of AX-polyphones, namely the ones that answer 'yes' and the ones that answer 'no'. While some other monophone questions make sense in this example, there are others which are useless (e.g., 'Is monophone AX at position +1 ?') since all polyphones would generate the same answer.
In addition to questions about specific monophones, questions about specific phonetic classes such as vowels, consonants, liquids, and fricatives are frequently used.

[3] Only 'agglomerate' and 'boomerang' share the same triphone at 'AX'.

Word	**Phonetic Transcription**								
advisory	AE	D	V	AY	Z	**AX**	R	IY	
agglomerate	AX	G	L	AA	M	**AX**	R	EY	T
boomerang			B	UW	M	**AX**	R	AE	NG
brilliant	B	R	IH	L	Y	**AX**	N	T	
devouring		D	IH	V	AW	**AX**	R	IH	NG
indicative	IH	N	D	IH	K	**AX**	T	IH	V
Position	-5	-4	-3	-2	-1	**0**	+1	+2	+3

Table 2.1: Word transcriptions to illustrate phonetic context modeling

Furthermore, context modeling may not be limited to within-word phonetic context but may include cross-word context. In that case, questions about the existence of word boundaries are quite useful, if such information is available.
Fig. 2.5 shows a typical decision tree for clustering the polyphonic variations of a particular state of monophone model AX. During construction of a phonetic decision tree for a specific state of a specific monophone, an objective function is evaluated at each node for each question to determine the question that yields the greatest gain when context classes are split according to that question. Assuming that appropriate statistical models (e.g., Gaussians) have been estimated for each polyphone observed in some training corpus, we can for instance take split likelihood gain as our objective function that scores the goodness of splits:

$$G(N, N_L, N_R) = \Big(\sum_{\mathbf{x} \in N_L} \log p_L(\mathbf{x}) + \sum_{\mathbf{x} \in N_R} \log p_R(\mathbf{x}) \Big) - \sum_{\mathbf{x} \in N} \log p(\mathbf{x})$$

where N is the node in question, N_L and N_R are the left and right child nodes and $p()$, $p_L()$ and $p_R()$ are the statistical models for the node and its left and right child nodes. Split likelihood gain measures how much the likelihood of the data in a specific node can be increased by splitting the data in two sets according to a phonetic question and modeling the data in each set separately. In case of often used simple D-dimensional diagonal covariance Gaussian models, split likelihood gain simplifies to

$$G(N, N_L, N_R) = n \sum_{k=1}^{D} \log \sigma_k^{2(N)} - \Big(n_L \sum_{k=1}^{D} \log \sigma_k^{2(N_L)} + n_R \sum_{k=1}^{D} \log \sigma_k^{2(N_R)} \Big)$$

where n, n_L and n_R are the number of samples in node N, left child N_L, and right child N_R, respectively. $\sigma_k^{2(N)}$, $\sigma_k^{2(N_L)}$, and $\sigma_k^{2(N_R)}$ are the k-th diagonal covariance coefficients of the Gaussians for node N, left child N_L and right child N_R, respectively. Using an objective function such as split likelihood gain, decision trees can be grown by iteratively splitting nodes until the gain falls below some predetermined

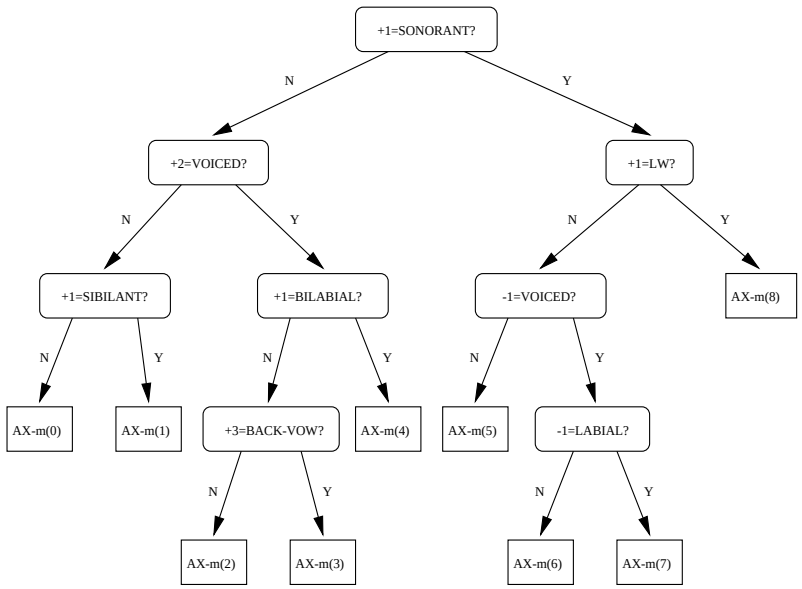

Figure 2.5: Phonetic context modeling using decision trees. Shown is a decision tree modeling phonetic contexts of the middle state (3-state HMM) of monophone AX.

threshold. Alternatively, trees can be grown up to a predetermined number of leaf nodes. Usually, the optimal number of leaf nodes is determined experimentally by recognition runs on independent validation sets.

The collection of leaf nodes of phonetic decision trees for all monophone states represents the recognizer's set of distinctly modeled context-dependent sub-phonetic units. Since each of these units models several actual HMM states, they are often called *tied states*. Typically, a large vocabulary continuous speech recognizer models thousands (up to 20000 and more) of tied states via phonetic decision trees. See Appendix B for a distribution of allophonic variation in a decision tree clustered Switchboard model for 24000 tied states. Context-dependent phonetic modeling has been reported to decrease the word error rate of speech recognition systems by up to 50% [Schwartz et al. '85, Chow et al. '86]. In our own experiments, we have observed a 40% reduction in word error rate when going from context-independent to context-dependent modeling.

2.4.3 Domain Dependence of Context Models

Phonetic context modeling using decision trees has emerged to a standard HMM modeling technique for large vocabulary speech recognition that has been adopted by almost all current state-of-the-art speech recognition systems. However, while context modeling reduces word error rates consistently, it significantly increases dependence on the acoustic characteristics and the vocabulary, phonetic dictionary and language model of the training domain. The a-priori distribution of within-word context-dependent phone models depends on both the phonetic transcription of the words in the training dictionary and the relative frequency of these words in the training corpus. Often, the words that constitute the recognition dictionary differ vastly across domains. As a result, context models obtained from data in one domain differ from those obtained on some other domain and performance of speech recognition systems in cross-domain applications drops significantly due to the mismatch in coverage of phonetic contexts.

In addition, cross-word phonetic context modeling, which improves performance over within-word context modeling, introduces yet another dependency on the training domain. By allowing context models to span across word boundaries, cross-word context models additionally depend on the relative frequency of word pairs and word triples (in case of single phone words) in the training domain. Such statistics are captured by the language model of a speech recognizer (see next section) and are known to differ significantly across domains.

As a consequence of the above mentioned dependencies, context models are typically constructed specifically on data from a specific target domain, selecting size and structure of phonetic decision trees for optimal recognition performance on data from that target domain. By focusing on a specific target domain, improved performance is achieved at the cost of reduced robustness and lack of portability to other domains. Often, context-dependent acoustic models are rebuild from scratch, if it becomes necessary to port a trained speech recognition system to some other, previously unseen domain of significantly different acoustic, phonetic and linguistic characteristics. While other domain dependent components of a speech recognition system such as dictionary and language model can be obtained relatively easily for a new domain, the construction of context-dependent acoustic models requires large amounts of transcribed acoustic data which renders porting efforts time-, labour- and cost-intensive.

Through the application of a scalable, hierarchical architecture, this thesis presents a solution to the problem of domain-dependence of context modeling that does not require expensive reconstruction of the acoustic model when switching to a new domain.

2.5 Language Modeling

The task of a language model in statistical speech recognition is to estimate the probability of word sequences, $P(W_1, \ldots, W_N)$, which can be factored as follows:

$$P(W_1, \ldots, W_N) = P(W_1)\, P(W_2|W_1) \prod_{i=3}^{N} P(W_i|W_{i-1}, \ldots, W_1).$$

In statistical n-gram modeling, one simplifies the above expression by reducing the conditioning on the full history of words to the last $n-1$ words. For instance, in 3-gram (trigram) modeling, one approximates the language model probability according to

$$P(W_1, \ldots, W_N) \approx P(W_1)\, P(W_2|W_1) \prod_{i=3}^{N} P(W_i|W_{i-1}, W_{i-2}).$$

Unfortunately, the usefulness of standard n-gram modeling is restricted to small values of n due to the exponential growth of the number of n-grams. Assuming a recognition vocabulary of M distinct words, the total number of n-tuples that theoretically need to be modeled by an n-gram equals M^n, a number that even for moderate sizes of vocabularies of a few thousand words quickly exceeds the storage and computational resources of todays computers. On the other hand, many of the M^n n-tuples never occur in any text corpus due to the grammatical regularity of language. Estimation of n-gram probabilities therefore requires smoothing techniques, typically a combination of *discounting* and *backing-off* (e.g., [Kneser & Ney '95]) in order to obtain robust probability estimates from raw n-tuple counts. In the case of trigram modeling, discounting means that the trigram counts of the more frequently occurring trigrams are reduced and the resulting excess probability mass is redistributed amongst the less frequently occurring trigrams. Backing-off is applied when there are too few trigrams to form any estimate at all and involves replacing the trigram probability by a scaled bigram probability.
Despite of the restricted context width of 2-4 words, statistical n-gram language models have proven to be quite effective. Furthermore, variable-length and category-based n-gram models [Niesler & Woodland '95], cache models [Jelinek et al. '91] and trigger models [Lau et al. '93] allow to robustly increase the context width beyond 4 words. Although many other language modeling techniques have been proposed over the years, n-gram models still dominate the field.

2.6 Decoding

The so called decoder represents the heart of any speech recognition system. Its task is to find the most likely sequence of words for any given acoustic input, where each

word is modeled by a sequence (sometimes a graph) of HMM states. As already mentioned, the decoding problem for HMMs possesses an efficient solution in form of the Viterbi algorithm. However, when decoding large vocabulary continuous speech with n-gram language models, an exact solution becomes intractable due to the very large number of competing sentence hypotheses.
The most popular solution is the application of a form of heuristic pruning to a time-synchronous Viterbi decoder which is then called Viterbi beam search. At any time step, partial hypotheses are extended by all possible successor states but are kept for future consideration only when their score stays within a certain threshold (the beam) relative to the score of the current best hypothesis. This way, only a very small fraction of the actual search space has to be examined, leading to a manageable computational complexity. Unfortunately, such a search process is no longer guaranteed to find the most probable hypothesis. Search errors are introduced when the globally best hypothesis gets pruned during decoding because of a temporarily bad local score.

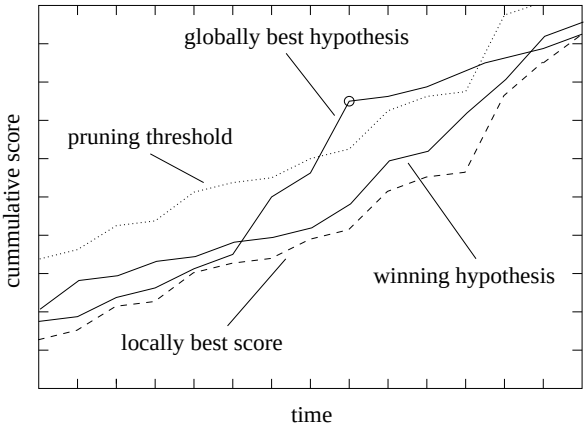

Figure 2.6: Decoding errors with Viterbi beam search

This fact is illustrated by Fig. 2.6. Here, lower scores correspond to more likely hypotheses. The lowest curve indicates the score of the locally best hypotheses for each time step. Above this curve, there is the equidistant pruning threshold curve. The score traces of two hypotheses have been included into the plot: (1) the curve for the globally best (most probable) hypothesis and (2) the curve for the 'winning' hypothesis found by this particular instance of a beam search. Note that although

the final score of the globally best hypothesis is better than the final score of the winning hypothesis, the globally best hypothesis never reaches the end of the utterance. Instead, it gets pruned at the time mark indicated by a circle since its partial score temporarily exceeds the pruning threshold. This search error can be omitted by increasing the pruning beam width such that the correct hypothesis stays below the pruning threshold curve at all times.
Large vocabulary speech recognition systems typically operate at the two endpoints of a continuous spectrum of beam widths. Research & evaluation systems have to use very large beam widths in order to reduce the probability of decoding errors. However, the increase in performance comes at the cost of high decoding time (often over one hundred times slower than real time). On the other hand, applications such as large vocabulary dictation require decoding times of almost real time in order to be usable. Among other techniques applied in this case, decoding beams have to be tightened considerably. Of course, pruning errors become more likely, resulting in a loss of performance. The trade-off between recognition accuracy and decoding speed is illustrated in Fig. 2.7 for various decoding beam widths of a typical large vocabulary speech recognition system.

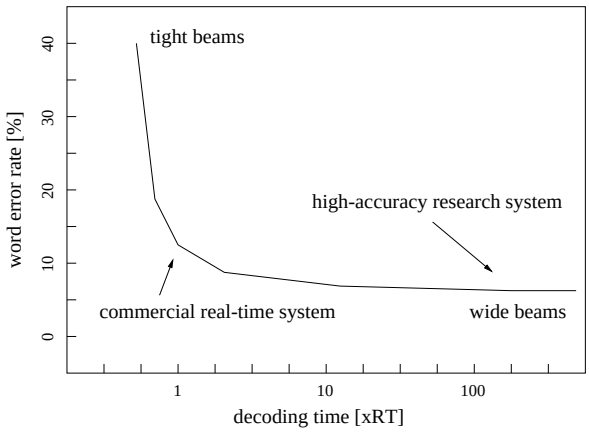

Figure 2.7: Effect of varying decoder pruning beam width

An alternative approach to the problem of searching the most probable word hypothesis in a speech recognition system is based on so called best-first stack decoding [Paul '92, Renals & Hochberg '99] which is related to the A^* algorithm used for heuristic search in artificial intelligence. These search algorithms are time asyn-

chronous – the best scoring path or hypothesis, irrespective of time, is chosen for extension and this process is continued until a complete hypothesis is determined. The asynchrony of operation requires that a suitable heuristic is employed in comparing the scores of competing hypotheses. Stack decoding has several potential advantages over Viterbi decoding: (1) The language model is decoupled from the acoustic model and is not used to generate new recognition hypotheses, (2) It is easy to incorporate non-Markovian knowledge sources (e.g., long-span LMs) without massively expanding the state space, and (3) The Viterbi assumption is not embedded in the search and thus a full maximum-likelihood search criterion may be used with little or no computational overhead. Disadvantages of the approach include sensitivity to the choice of heuristic and the possibility of repeated computation.
It should be noted that today's decoder technology has become quite complex due to cross-word context-dependent phonetic modeling, tree-structured pronunciation lexica, and look-ahead techniques. It is beyond the scope of this thesis to go into all these details. The interested reader is referred to [Odell '95, Young '96, Ravishankar '96].

Chapter 3

Connectionist Acoustic Modeling

This chapter starts with a critical view on standard HMM based acoustic modeling in speech recognition. We reveal the major weaknesses of traditional HMM modeling that, together with the reborn interest in connectionist models of cognitive processes in the eighties, led to the formulation of an alternative paradigm for acoustic modeling. We motivate and introduce connectionist acoustic modeling, giving a review on the techniques and architectures that have been investigated. We close this chapter with a discussion of some shortcomings of connectionist acoustic modeling.

3.1 Drawbacks of Standard Modeling

By standard modeling, we refer to the statistical framework based on Hidden Markov Models presented in chapter 2. More specifically, standard modeling refers to the application of Gaussian mixture models for HMM observation probability estimation. Recognition systems based on such models offer powerful learning and decoding algorithms along with flexible modeling of temporal aspects which is why they have attracted so much interest in the speech recognition community. Practically all existing speech recognition systems are build around this modeling paradigm. However, in order to take advantage of the representational power of HMMs, algorithms must explicitly or implicitly make simplifying assumptions about the time series being modeled. Some of these assumptions are obviously unrealistic and violated when modeling speech with HMMs. Nevertheless, this suboptimal model is generally accepted because it can be used more effectively than any alternative. Given the strong base of mathematical tools for statistical speech recognition with HMMs, modifying only a few aspects of the existing approach at a time seems more appropriate than starting from scratch. Following is a list of shortcomings that have been identified with standard modeling:

- **Independence Assumption:** Successive observation vectors are considered independent and therefore uncorrelated. This is a poor match to most kinds of speech segments. Diphthongs (e.g., AY,EY,OY) and glides (e.g., W,Y), for instance, exhibit strong non-stationary behavior.

- **First-Order Assumption:** Observation vectors depend only on the current HMM state and transitions depend only on the previous HMM state instead of on the whole history of states. In contrast, speech is characterized by strong co-articulation effects, e.g., observation vectors are influenced by the previous phonetic state(s).

- **Poor Discrimination:** HMM training algorithms are based on Maximum Likelihood (ML) which assumes correctness of the models. As we just argued, correctness of the models must be questioned due to first-order and independence assumptions. More importantly, ML implies poor discrimination since ideally, minimization of the word error rate should be based on minimizing a-posteriori word or sentence probabilities.

- **Distributional Assumptions:** For practical as well as computational reasons, observation probability distributions in large vocabulary conversational speech recognition systems are almost always modeled by mixtures of diagonal covariance Gaussians. The diagonal covariance assumption neglects correlations between individual coefficients of observation vectors.

- **Architectural Structure:** Standard mixture based modeling of observation probability distributions results in an independent and unstructured set of models. Missing structure is not problematic in terms of performance or accuracy of modeling. However, many algorithms in speech recognition such as speaker adaptation and fast acoustic match require to structure the acoustic model according to some acoustic similarity criterion. If such structure were built inherently into the model, the above mentioned algorithms could be realized much easier. Also, the missing structure prevents us from scaling the model in terms of the number of modeled HMM states.

Almost all of the above mentioned assumptions and shortcomings have been addressed by researchers over the years. The first two assumptions are inherent to the HMM model being used and can only be addressed by some sort of add-on correction mechanism or by moving to an entirely different model. However, the latter three assumptions/shortcomings can be addressed by replacing the set of mixture density models that approximate the HMM state observation probabilities by a more suitable

model. This approach allows the system to benefit from the excellent temporal modeling properties of HMMs while investigating alternative forms of acoustic modeling of speech.

3.2 Discriminative Modeling

A detailed treatment of discriminative modeling requires to establish a globally discriminant training criterion based on the posterior probability of sequences of words. Globally discriminant approaches have been investigated (e.g., [Valtchev '95]) but usually suffer from high computational complexity which is why simplifying assumptions are often made in order to apply these approaches to large vocabulary speech recognition tasks.
To avoid the computational pitfalls of globally discriminative modeling and still improve discrimination, locally discriminative modeling has been proposed by several researchers (e.g., [Bourlard & Morgan '94]. In locally discriminative modeling, the training criterion is based on the posterior probability distribution over the set of acoustic HMM states for a specific acoustic feature vector. In other words, rather than discriminating words in a sentence, we aim at discriminating the basic speech units in each frame of speech data. Before discussing the potential benefits and advantages of such modeling, we first have to elaborate on how locally discriminative modeling can be integrated into HMMs. After all, the HMM formalism requires to model state likelihoods for each frame of acoustic data. Using Bayes' rule we can satisfy this constraint:

$$p(\mathbf{x}|s_i) = \frac{p(s_i|\mathbf{x})}{p(s_i)}\, p(\mathbf{x})$$

Instead of directly estimating the state likelihoods $p(\mathbf{x}|s_i)$ for each state s_i given an input feature vector $\mathbf{x}$, we can take a detour that allows us to include the state posteriors $p(s_i|\mathbf{x})$. Estimators for the latter can be trained using the Maximum A Posteriori (MAP) rather than the Maximum Likelihood (ML) training criterion. However, as seen in the above expression, the integration of a MAP estimator requires to divide the estimates of the posterior state probabilities by their prior probabilities and to multiply the outcome by the unconditional probability of observing the feature vector $\mathbf{x}$. Fortunately, there is no need to estimate $p(\mathbf{x})$ when applying the above rule to speech recognition. It merely adds an offset to the acoustic scores for each acoustic frame that is independent of the HMM state and therefore does not influence the outcome of a Viterbi style search for the most likely state/word sequence. Therefore, locally discriminative modeling in the HMM framework requires only to divide estimates of the posterior state probabilities by their prior probabilities. The result-

ing quantity $\hat{p}(\mathbf{x}|s_i)$ can directly be used as HMM state emission probability and is usually called *scaled likelihood*:

$$\hat{p}(\mathbf{x}|s_i) = \frac{p(s_i|\mathbf{x})}{p(s_i)}$$

The reader may ask: What is the point of going through this detour when we finally derive essentially the same probability as we would with a conventional Gaussian mixture based estimator? There are at least the following potential advantages:

- **Improved Discrimination:** Estimators of the posterior probabilities are trained according to MAP in contrast to the ML based likelihood estimators. In MAP based modeling, the emphasis is on modeling class boundaries while in ML modeling, the emphasis is on accurately modeling each class' distribution. ML based estimators are in danger of wasting a lot of their parameter resources in modeling a distribution in regions where no other classes compete. MAP estimators on the other hand focus their parameter resources at class boundaries in order to maximize discrimination between classes.

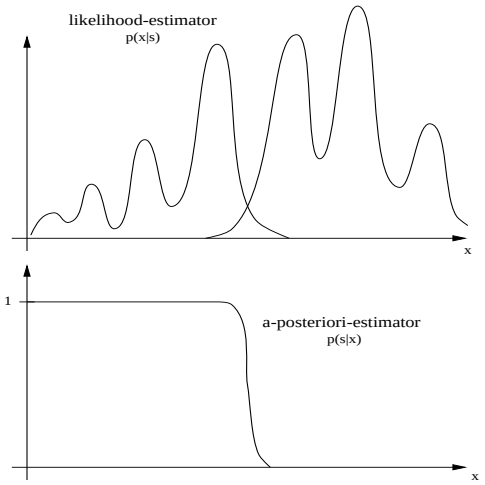

Figure 3.1: Likelihood estimators vs. a-posteriori estimators

Fig. 3.1 illustrates this behavior for a two class problem. The upper graph shows relatively complex class distributions that the likelihood based estimators seek

to capture as accurately as possible. However, as can be seen in the lower plot, discrimination between the two classes can be achieved with a very simple MAP estimator since there is almost no overlap between the classes (the MAP estimator for the second class is simply $1 - p(s|x)$ and is not included in the plot).

- **Smaller Number of Parameters:** Fig. 3.1 also reveals another problem with likelihood estimators. The waste of parameter resources in modeling the distribution in regions where it would not be necessary typically requires more parameters and more complex estimators than in the case of discriminative a-posteriori estimators.
  It was shown [Bourlard & Morgan '94, Morgan & Bourlard '95], that locally discriminative acoustic models can indeed achieve the same performance with less parameters when compared to standard HMM modeling.

- **Explicit Control over Class Priors:** In case of a mismatch of class priors between training and test corpus, for instance caused by a significantly different vocabulary, locally discriminant models can be adapted effectively since the class priors are explicitly available.

In addition, depending on the type of estimator being used, locally discriminant models offer easy integration of additional knowledge sources and reduced assumptions about the type of emission probability distribution. In the case of neural network estimators, which we will discuss next, the assumption of independence of observations can be weakened considerably by taking a window of frames around the current time frame as input to the estimator, instead of just the current time frame. This way, important contextual information can be incorporated into the probability estimation process.

3.3 Connectionist Acoustic Modeling

Relying on distributed internal representations for solving classification and regression tasks, connectionist architectures, also known as (artificial) neural networks [Rumelhart & McClelland '86, Bishop '95a, Ripley '96], created considerable interest in the speech recognition community [Lippmann '89, Waibel & Lee '90, Waibel '91]. As neural networks were found to be excellent tools for classifying speech units such as phones, they have primarily been applied to simple speech recognition problems such as classification of static patterns. Neural networks for classification of complete temporal sequences have not been successful for continuous speech recognition where the number of possible word sequences are practically infinite. However, within the

HMM framework, connectionist architectures have proven to be viable and sometimes superior alternatives as acoustic models for the estimation of (scaled) state likelihoods.
A few years back, it was shown (e.g., [Bridle '90]) that the outputs of appropriately trained classifier neural networks approximate class posterior probabilities. A proof of this property can be found in Appendix A. In fact, classifier neural networks were found to be both efficient and versatile tools for approximating posterior probabilities. Although there are other estimators for posterior probabilities such as polynomial classifiers, neural networks have become the single most important architecture for locally discriminant acoustic modeling. Neural network based locally discriminant models are usually called *connectionist acoustic models* and speech recognition systems based on these models are often termed *hybrid NN/HMM systems.*
A wide variety of neural network models has been investigated for the purpose of estimating posterior state probabilities. Following is a list of the most popular architectures that have been applied to connectionist acoustic modeling:

- **Multi Layer Perceptrons (MLP):** MLPs arguably are the most frequently applied neural network models for connectionist acoustic modeling (e.g., [Morgan & Bourlard '90, Bourlard & Morgan '94, Morgan & Bourlard '95, Tebelskis '95]) due to a simple topology and an efficient training algorithm based on gradient descent (error backpropagation). Since MLPs were also applied extensively in the architecture proposed in this thesis, we will present this type of neural network in more detail.

  Fig. 3.2 depicts the structure of a typical feed-forward classifier MLP for connectionist acoustic modeling consisting of fully interconnected layers (each unit in the hidden and output layer receives activation from all units in the previous layer). Although MLPs can consist of several hidden layers in addition to an input and an output layer, those with a single hidden layer were found to be sufficient for successful classification of speech units and are theoretically capable of modeling the same class of functions as networks with more hidden layers, provided there are enough units in the hidden layer. Each unit in the hidden and output layer computes a nonlinear function of its input vector $\mathbf{x}$ consisting of a linear *activation function* followed by a non-linear *transfer function.* While all MLP units use the following projective kernel

  $$a_i(\mathbf{x}) = \sum_{k=1}^{N} w_{ik} x_k + b_i,$$

  as activation function, where the w_{ik} are weights and the b_i are the unit biases[1], transfer functions are different for different layers of the network. Hidden units

[1] Often, input vectors are implicitly extended by a constant coefficient of 1 which allows to write

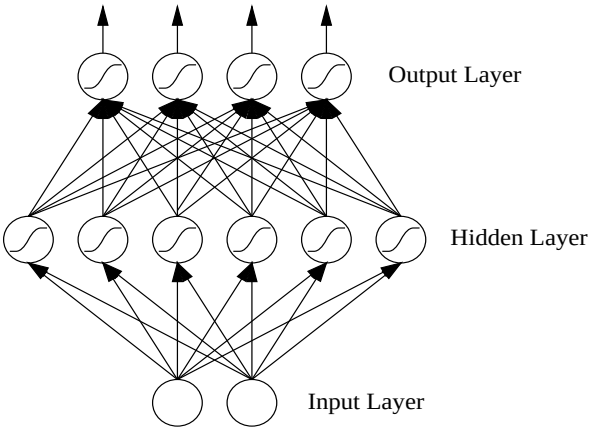

Figure 3.2: Multi layer perceptron (MLP)

are mostly equipped with the sigmoid transfer function, yielding the following output function

$$y_i(\mathbf{x}) = \frac{1}{1+\exp(-a_i(\mathbf{x}))} = \frac{1}{1+\exp(-\sum_{k=1}^{N} w_{ik}x_k - b_i)}$$

Alternatively, the *tanh* function which is just a symmetric version of the sigmoid function is applied to hidden unit activations. The type of transfer function used for units in the output layer depends on the learning task and should not be chosen arbitrarily. In the case of connectionist acoustic modeling, MLPs are used for multi-way classification of speech units such as monophone HMM states. In statistical theory, multi-way soft classification is modeled by a multinomial probability density. It can be shown [Jordan & Jacobs '94, Jordan '95], that the matching network transfer function (canonical link) for this probability model is the *softmax* [Bridle '90] function:

$$y_i(\mathbf{x}) = \frac{\exp(a_i(\mathbf{x}))}{\sum_{j=1}^{M}\exp(a_j(\mathbf{x}))} = \frac{\exp(\sum_{k=1}^{N} w_{ik}x_k + b_i))}{\sum_{j=1}^{M}\exp(\sum_{k=1}^{N} w_{jk}x_k + b_j))}$$

where M is the number of output units, N is the number of hidden units and x_k is the output of the k-th hidden unit. The softmax function together with

the bias as part of the weight vector $\mathbf{w}$

a cross-entropy error function (which essentially is equivalent to the log of a multinomial probability density) constitute the optimal choice for output units of a classifier neural network with respect to statistical interpretation of network outputs and efficiency of parameter estimation. For these reasons, we adhere to the above theory and use classifier MLPs with softmax outputs throughout this thesis. Nevertheless, it should be noted that other transfer functions can and have been applied to MLP based classification as well, particularly when statistical interpretability of network outputs is not required.

- **Recurrent Neural Networks (RNN):** RNNs are comparable to MLPs except that they contain additional recurrent connections that feed activations from the outputs of a particular layer back to the inputs of that layer. Such recurrent connections effectively allow for improved modeling of temporal dynamics, which was found to be advantageous in speech recognition [Robinson & Fallside '91, Robinson '94], where there are considerable correlations between adjacent feature vectors. The *Abbot* system [Hochberg et al. '95], arguably the first connectionist speech recognition system that has achieved competitive performance on large vocabulary speech recognition tasks has been built around a recurrent neural network [Robinson et al. '96] and is regularly participating in the annual DARPA Broadcast News evaluations [Cook et al. '97a, Cook & Robinson '98, Cook et al. '99].

  An interesting extension of RNNs that allows to simultaneously train and optimally combine recurrent neural networks for forward and backward time directions, the bidirectional recurrent neural network [Schuster & Paliwal '97], has also been applied successfully to the task of connectionist acoustic modeling.

  However, even though there is a generalization of backpropagation called Backpropagation Through Time (BPTT) available for recurrent networks, training of RNNs is considerably more expensive than training of MLPs and requires presentation of training patterns in their correct sequential order. Furthermore, by using a window of frames around the current training frame as input pattern vectors, contextual information can be incorporated into MLPs to some extent as well.

- **Radial Basis Function Networks (RBF):** In contrast to the projective kernel used as activation function in MLPs, RBFs make use of the following radial kernel in hidden units

$$a_i(\mathbf{x}) = \sum_{k=1}^{N} \frac{(w_{ik} - x_k)^2}{\sigma_i^2}$$

with parameters w_{ik} (cluster means) and σ_i^2 (cluster variances). Also, an exponential transfer function is used in hidden units which yields the following output

$$y_i(\mathbf{x}) = \exp(-a_i(\mathbf{x})) = \exp(-\sum_{k=1}^{N} \frac{(w_{ik} - x_k)^2}{\sigma_i^2})$$

Usually, RBFs [Moody & Darken '89, Renals '89, Poggio & Girosi '90] consist of a single hidden layer and a linear output layer which renders these models very similar to Gaussian mixture densities. As such, RBFs can be initialized more efficiently than MLPs by applying unsupervised clustering algorithms for obtaining the parameters of the hidden layer. Also, RBFs with linear output layer need not be trained with iterative gradient descent optimization methods. Rather, a more efficient two step process of (1) clustering hidden unit means and selecting appropriate variances, and (2) estimating output weights analytically using minimum least squares procedures is typically applied [Moody & Darken '89]. Although RBFs are primarily used with linear output layers for the purpose of regression, they can also be applied to classification tasks [Yee '92]. However, due to their localized activations in the hidden layer, RBFs are better suited to regression tasks and it is often found that networks with projective kernels such as MLPs outperform RBFs on classification tasks. Due to their close relationship to mixture densities, RBFs have raised considerable interest in the speech recognition community (e.g., [Ney '91, Renals et al. '91]) and even classifier RBFs were applied successfully to the task of connectionist acoustic modeling in speech recognition systems [Fritsch '96].

In addition to the above connectionist architectures, Time-Delay Neural Networks (TDNN) [Waibel et al. '87, Waibel '89, Hild & Waibel '93], modular ensembles of TDNNs [Waibel et al. '88, Waibel '88, Waibel '89], and Hierarchical Mixtures of Experts (HME) [Jordan & Jacobs '94] have been used to build state-of-the-art connectionist speech recognition systems [Fritsch '96, Fritsch et al. '96].
Initially, connectionist acoustic modeling was applied to context-independent phonetic modeling, using a single, sometimes very large neural network with 40-60 output units for jointly estimating posterior probabilities of typically about 40-60 HMM states representing the phones modeled by the system. In a number of studies, it was shown that such monolithic connectionist acoustic models can outperform classical acoustic models based on Gaussian mixture densities, provided they both use the same number of parameters and the same input features. However, performance of Gaussian mixture models can be improved over connectionist acoustic models by switching to context-dependent phonetic modeling. In contrast to connectionist

modeling, the increased number of HMM states in context-dependent systems pose no conceptual problem to traditional mixture modeling since each state's emission probability distribution is modeled independently, using a separate mixture model for each state. Connectionist acoustic models based on a monolithic neural network lack such scalability and are not directly suitable for context-dependent phonetic modeling since joint estimation of state posteriors requires the output layer of the network to consist of as many units as there are HMM states. Successful training of classifier neural networks becomes increasingly difficult if not impossible with larger and larger output layers. Other ways of incorporating context-dependency into the connectionist acoustic modeling framework therefore had to be found.

3.4 Connectionist Context Modeling

In context-dependent acoustic modeling, we have to estimate HMM state likelihoods $p(\mathbf{x}|s_l)$ for $1 \leq l \leq N$ just as in the context-independent case, except that the total number of HMM states N is significantly larger. Noteworthy, each state s_l does not only correspond to a specific phone and position in the underlying atomic HMM model but also to a specific context class (e.g., diphone, triphone). In classical context-dependent acoustic modeling based on Gaussian mixture densities, knowledge about underlying phone, position and context class identities of individual states is not required as each state simply gets its own mixture density.
However, for connectionist acoustic modeling, it is advantageous to rewrite the state likelihoods by making the knowledge about underlying phone ω_i, context class c_j and, in case of multi-state HMM topologies, position π_k in the atomic HMM of state s_l explicit:

$$p(\mathbf{x}|s_l) = p(\mathbf{x}|\omega_{i(s_l)}, c_{j(s_l)})$$

with $i(s_l) \in \{1, \ldots, I\}$ and $j(s_l) \in \{1, \ldots, J\}$ for single-state HMMs and

$$p(\mathbf{x}|s_l) = p(\mathbf{x}|\omega_{i(s_l)}, c_{j(s_l)}, \pi_{k(s_l)}).$$

for K-state HMMs ($k(s_l) \in \{1, \ldots, K\}$). In the locally discriminant framework, we apply Bayes' rule to express state likelihoods in terms of state posteriors and priors as already described earlier. We first take a look at context-dependent modeling with single-state HMM topologies. Bayes' rule yields

$$p(\mathbf{x}|\omega_i, c_j) = \frac{p(\omega_i, c_j|\mathbf{x})\, p(\mathbf{x})}{P(\omega_i, c_j)}.$$

As usual, $p(\mathbf{x})$ can be omitted, resulting in scaled likelihoods

$$\hat{p}(\mathbf{x}|\omega_i, c_j) = \frac{p(\omega_i, c_j|\mathbf{x})}{P(\omega_i, c_j)}.$$

Straight-forward application of a single network for estimating $p(\omega_i, c_j|\mathbf{x})$ requires $I \times J$ output nodes. However, context-dependent posteriors $p(\omega_i, c_j|\mathbf{x})$ and priors $P(\omega_i, c_j)$ can be decomposed into smaller, easier to solve subtasks that require only networks with I and J output nodes using statistical factoring [Morgan & Bourlard '92]. According to the definition of conditional probability, there are two different ways of factoring, yielding considerably different context-dependent connectionist architectures:

1. **Factoring Contexts:**

$$\hat{p}(\mathbf{x}|\omega_i, c_j) = \frac{p(\omega_i, c_j|\mathbf{x})}{P(\omega_i, c_j)} = \frac{p(c_j|\mathbf{x})\, p(\omega_i|c_j, \mathbf{x})}{P(c_j)\, P(\omega_i|c_j)}$$

2. **Factoring Phones:**

$$\hat{p}(\mathbf{x}|\omega_i, c_j) = \frac{p(\omega_i, c_j|\mathbf{x})}{P(\omega_i, c_j)} = \frac{p(\omega_i|\mathbf{x})\, p(c_j|\omega_i, \mathbf{x})}{P(\omega_i)\, P(c_j|\omega_i)}$$

In both cases, the original posterior state probability $p(\omega_i, c_j|\mathbf{x})$ has been decomposed into a product of an unconstrained posterior probability and a conditional posterior probability. The first one can be estimated with a neural network just like in the context-independent case. The second, conditional posterior probability can be estimated in various ways. Viewing a feed-forward classifier neural network as an estimator of the left side of a conditional, given the right side as input, the input layer of such a network can be extended by adding binary nodes that sparsely encode the value of the discrete dependent variable.
Alternatively, conditional posterior probabilities can be estimated using a set of neural networks, one for each possible value of the discrete dependent variable. Each one of these networks has to be trained only on data corresponding to the specific value of the discrete dependent variable for which it was build. Consider for instance the case of (1.) factoring contexts. The unconstrained posterior probability $p(c_j|\mathbf{x})$ can be estimated with a neural network with J output units, one for each context class. The conditional posterior $p(\omega_i|c_j, \mathbf{x})$ can be estimated with a set of J neural networks N_j, one for each context class, estimating

$$p_j(\omega_i|\mathbf{x}) = p(\omega_i|c_j, \mathbf{x}) \qquad \forall j \in \{1, \ldots, J\}$$

The conditional dependence of $p_j(\omega_i|\mathbf{x})$ on c_j is realized by training each network only on data corresponding to its context class c_j. While each of these networks estimates phone posteriors as in the context independent case, they all do so for a different phonetic context. Since they represent specialized versions of a network for context-independent connectionist modeling, it is advantageous to initialize the parameters of all the context-specific networks with the parameters of a trained context-independent network. This way, training of the networks is accelerated and context-dependent estimates are regularized which avoids overfitting in cases of little available context-specific training data.

In the case of (2.) factoring phones, the unconstrained phone posterior $p(\omega_i|\mathbf{x})$ is identical to the one estimated in the context-independent case. In contrast to the case of factoring contexts, context-independent modeling is transparently embedded into the context-dependent architecture, allowing to easily switch between the two modes of operation. Furthermore, factoring phones allows to apply phonetic decision trees to induce a variable, robust and data-dependent number of generalized context classes for each phone. The conditional posteriors $p(c_j|\omega_i, \mathbf{x})$ can be estimated by a set of I phone-specific neural networks N_i such that

$$p_i(c_j|\mathbf{x}) = p(c_j|\omega_i, \mathbf{x}) \qquad \forall i \in \{1, \ldots, I\}$$

Again, the conditional dependence on the phone ω_i is realized by restricting the training set of each phone-specific network to data corresponding to the respective phone. The resulting context-dependent connectionist architecture consists of expert networks for discriminating the context classes separately for each phone while the architecture resulting from factoring contexts consists of expert networks for discriminating the phones separately for each context class. Both approaches have been applied successfully, yielding improved performance over context-independent connectionist acoustic modeling [Franco et al. '94, Franco et al. '97, Fritsch et al. '97, Kershaw et al. '95, Kershaw '97].

The factored priors in the denominator of the expressions for scaled likelihoods are determined according to relative frequencies in the training set. Conditional priors can be obtained in a similar way as conditional posteriors by restricting the training set on which relative frequencies are computed to data corresponding to the value of the discrete dependent variable.

Finally, we note that state posteriors for multi-state HMM topologies can be decomposed analogously to the single-state case. The additional variable π_k, indicating position in the multi-state HMM, simply adds another degree of freedom in the order of factoring. However, not all of the 6 possible ways of factoring the state posteriors yield reasonable configurations as has been investigated in [Fritsch '96].

3.5 Problems with Connectionist Modeling

Connectionist acoustic models possess a wide range of properties (locally discriminant, more compact, faster evaluation, explicit class priors, etc.) that are missing in traditional mixture density based models. Nevertheless, wide-spread use of standard connectionist acoustic models in large vocabulary speech recognition systems has been hindered because of the following problems:

- **Lack of Scalability:** As already mentioned, monolithic connectionist acoustic models scale poorly with respect to the number of HMM states that are modeled. A classification task involving n classes requires a classifier neural network with n output nodes. Unfortunately, the number of output nodes in a neural network can not be increased arbitrarily. In [Cohen et al. '92], the authors report a decrease in speech recognition performance when increasing the number of output units in a monolithic network from 69 (context-independent) to 200 (context-dependent). In fact, connectionist acoustic models were observed to perform best when applied to the level of speech monophones instead of on the level of subphonetic HMM states. We have shown how the technique of factoring context-dependent state posteriors allows for decomposition of an otherwise oversized classification problem into a sequence of two or three considerably smaller classification problems. Although factoring posteriors has opened the door to context-dependent connectionist modeling, some of the resulting classification tasks may still be too large for accurate estimation of posteriors, especially when phonetic decision trees are used to induce variable amounts of context classes for each phone.

- **Non-Uniform Priors:** It has been observed that classifier neural networks generate poor estimates of posterior class probabilities for infrequent classes that occur rarely in the training set. Estimates of posteriors for frequent classes tend to be overestimated by the network while estimates of posteriors for very infrequent classes often vanish [Lawrence et al. '98].

  For optimal results, the classes to be discriminated by the network should be distributed uniformly in the data used for training the neural network. Unfortunately, real-world classification problems typically exhibit quite irregular, non-uniform prior distributions. For instance, the following plot (Fig. 3.3) depicts the distribution of prior probabilities of English phones as estimated on the Switchboard LVCSR corpus (see Chapter 4).

  The least frequent phones (ZH,OY,EN) are about 20 times less probable than the most frequent phones (S,T,N).

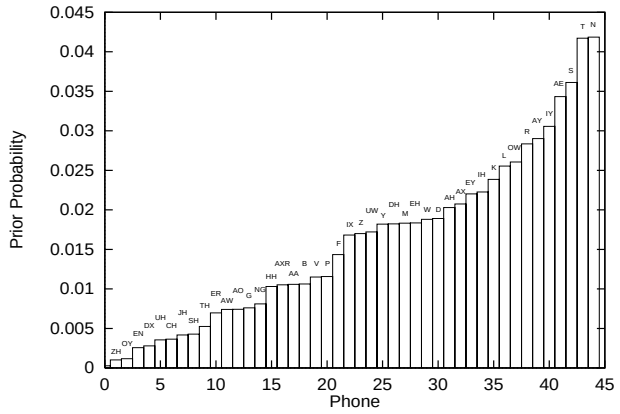

Figure 3.3: Non-uniformity of phone prior distribution on Switchboard

- **Computational Cost of Training:** The amount of computation required for training the neural networks in a connectionist acoustic model can be orders of magnitude more than is required for classical HMM training. This is a consequence of locally discriminant modeling, in which all parameters are potentially modified for each training pattern, irrespective of the class the pattern belongs to. In traditional, non-discriminatively trained Gaussian mixture models, only the parameters of a single mixture (out of several thousand) are affected by any training pattern. Furthermore, maximum likelihood training of Gaussian mixture based acoustic models can easily be distributed among several computers, speeding up training times significantly. In contrast, connectionist acoustic models require on-line training of typically very large neural networks which can not be distributed as easily. Rather, efficient training requires dedicated hardware which is not the case for conventional models. Furthermore, even when using dedicated parallel hardware for the training of neural networks, researchers have reported training times of up to several weeks for their largest connectionist acoustic models.

We will present a new architecture for connectionist acoustic modeling in chapter 5 that, in addition to providing structural benefits not found in traditional models, avoids each one of the above pitfalls that have prevented wide-spread application of connectionist acoustic models so far.

Chapter 4

The Switchboard Corpus

In order to assess and compare the performance of speech recognition systems, a variety of standardized speech recognition benchmarks and corpora have been collected over the years. Large structured collections of speech and associated transcriptions are essential to progress in automatic speech recognition. Due to the many approximations and heuristics, superiority or inferiority of algorithms and architectures can not only be justified theoretically but must be assessed on widely used benchmark corpora. In the following, we present the Switchboard corpus [Godfrey et al. '92], which has been used for experimental evaluation of the architecture and algorithms proposed in this thesis.

4.1 Overview

Switchboard is a large multi-speaker corpus of conversational American English telephone speech and text collected automatically over T1 lines at Texas Instruments (TI). It includes about 2500 conversations by 500 different speakers from every major dialect region in the United States. Originally designed for speaker identification and topic spotting, it is now being used primarily for evaluating large vocabulary conversational speech recognition (LVCSR) systems for band-limited (telephone quality) speech.
Overall, the corpus contains about 250 hours of speech and nearly 3 million words of text. The conversations were recorded as two separate but synchronized data streams with 8kHz sampling rate and μ-law encoding, one for each speaker. The isolation of the two speakers is limited by the long distance telephone network's echo cancelling performance, but is generally better than 20 decibel. However, some recordings exhibit heavy cross-talk with both speakers audible on one recording side. Although adaptive filters can be used to reduce such cross-talk, it is nevertheless regarded as a potential problem for speech recognition algorithms.

Participating subjects were asked to lead a natural conversation for about 5 minutes whereby the automatic collection system suggested one out of 70 topics such as 'air pollution', 'care of the elderly', 'football', 'music', 'child care', 'taxes', etc. Various demographic information about the participating speakers was gathered and stored. This includes their age, sex, level of education and geographically-defined dialect area where they grew up. According to this information, about 54.9% of the speakers were male, 45.1% female. About 90% of the speakers had an education of college level or above. The following two tables give information about age and dialect distribution in the corpus:

Dialect Region	Percentage
South Midland	29.4%
Western	16.1%
North Midland	14.6%
Northern	14.2%
Southern	10.6%
New York City	6.2%
Mixed	4.9%
New England	4.0%

Age	Percentage
20–29	26.4%
30–39	33.7%
40–49	21.1%
50–59	16.4%
60–69	2.4%

Table 4.1: Dialect region and age distribution in Switchboard

The relatively high percentage of speakers from the 'South Midland' area is attributable to the fact that a lot of Texas Instruments employees participated and the company is located in this area. The speech in the Switchboard corpus is fully transcribed, and the transcription conventions documented. Court reporters produced most of the verbatim transcripts, following a manual prepared specifically for the project.

4.2 Characteristics

Switchboard is a spontaneous, conversational telephone speech corpus. As such, it exhibits a variety of phenomena that render automatic speech recognition a very difficult problem:

- **Speaking Style:** disfluencies, linguistic incoherence, false starts, interruptions, repetitions, emotions (mostly laughter), bad grammar

- **Pronunciation Effects:** highly variable speaking rate, reduced pronunciations (going to → gonna), co-articulation, sloppy speech (whatcha gonna do 'bout it?)
- **Telephone Channel:** reduced bandwidth, signal degradation, high variation in channel quality, reverberations, echos, cross-talk, static noise
- **Ambient Noise:** music, television, kids crying, cars passing by, kitchen noise, etc.

Furthermore, conversational speech exhibits an extremely non-uniform distribution of words.

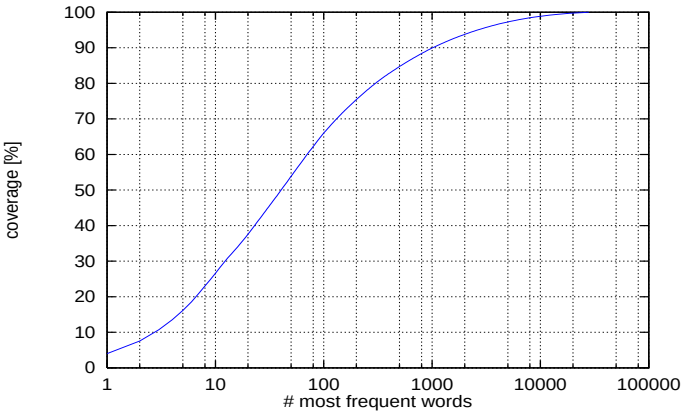

Figure 4.1: Word coverage – Switchboard corpus

The Switchboard corpus contains about 28000 distinct words. Fig. 4.1 shows corpus coverage with respect to most frequent words. According to this analysis, the 100 most frequent words account for roughly 65% of the corpus. The 1000 most frequent words account for ca. 90% of the corpus. Among the most frequent words are 'I', 'THE', 'AND', 'YOU', 'THAT', 'TO', 'A', 'OF' and 'IT'. 90 of the 100 most frequent words are composed of only a single syllable. There is a large diversity of phonetic pronunciation of these short, frequent words. For instance, the word 'AND' has been found in 87 different phonetic pronunciations where the most common pronunciation represents just 16% of all occurrences. The high variation in phonetic realization

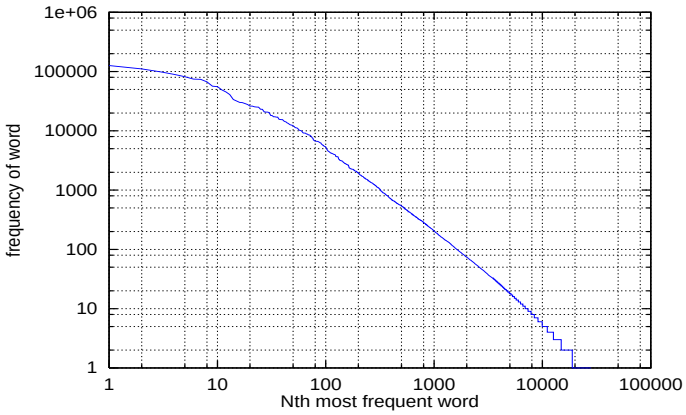

Figure 4.2: Word frequencies – Switchboard corpus

renders the recognition of these words rather difficult. Even worse, since these words are very common, their mis-recognition influences overall performance severely.
Fig. 4.2 shows word frequencies for the Switchboard corpus. While the 100 most frequent words have frequencies that would allow separate word models for each one of them, more than 20600 words (73.6%) occur less than 10 times in the corpus.

4.3 LVCSR Evaluations

The Defense Advanced Research Projects Agency (DARPA) together with the National Institute for Standards and Technology (NIST) performs annual official evaluations (denoted 'Hub-5E') of large vocabulary conversational speech recognition (LVCSR) systems on the Switchboard and Callhome corpora. Among the sites that have participated in such evaluations are BBN, Dragon Systems, SRI, Boston University, Cambridge University and Carnegie Mellon University. In 1997, the author participated as a member of the Interactive Systems Labs, a group of researchers from University of Karlsruhe and Carnegie Mellon University which scored first on that year's Switchboard test set [Finke et al. '97]. The following table presents the best results in terms of word error rate in recent years' evaluations. Unfortunately, results are not directly comparable since NIST selects a new test set of varying difficulty each year. Switchboard-I denotes the original corpus, while Switchboard-II denotes

a relatively new, not yet fully transcribed additional corpus that has been used in recent evaluations.

year	test set	word error rate
1995	Eval-95 from Switchboard-I	48.0%
1996	Eval-96 from Switchboard-I	38.8%
1997	Eval-97 from Switchboard-II	35.1%
1998	Eval-98 from Switchboard-II	36.7%

Table 4.2: Best performances in official Switchboard evaluations

While there has been a lot of progress in speech recognition technology on the Switchboard corpus, the word error rate still hovers in the thirties, falling far short of human capabilities (4% word errors according to [Lippmann '97]) on this data.

4.4 Thesis Relevance

The Switchboard corpus offers unique features that make it attractive for evaluating the architecture and algorithms proposed in this thesis. It is one of the largest existing speech corpora and arguably the most difficult one for today's speech recognition technology with lots of open questions and great potential for improvements in modeling. With respect to thesis relevance, the Switchboard corpus

- is ideal for building a robust baseline recognizer for general American English and for domain adaptation experiments due to its acoustic and linguistic variability.
- offers a high degree of phonetic variability which requires detailed phonetic context modeling to achieve competitive performance. On the one hand, this allows to demonstrate the scalability of the proposed hierarchical connectionist acoustic model to arbitrarily large HMM state spaces. On the other hand, this leads to very specific context models ideally suited for structural adaptation experiments on domains with less variability in phonetic context.
- represents a large vocabulary corpus. With a phonetic dictionary of about 30000 distinct word forms from a large variety of topics, a Switchboard recognizer covers most of the words of smaller domains.

While there are speech corpora with even larger vocabularies such as the Wall Street Journal (WSJ) and the Broadcast News (BN) domains, Switchboard is rather unique in its combination of conversational speaking style, large vocabulary and reduced bandwidth acoustic quality.

Chapter 5

Hierarchical Connectionist Acoustic Modeling

This chapter presents a hierarchically organized connectionist architecture for probabilistic classification and its application to acoustic modeling in automatic speech recognition [Fritsch '97, Fritsch & Finke '98a, Fritsch & Finke '98b]. We introduce hierarchical soft classifiers and discuss their theoretical properties with respect to statistical interpretation which allows for integration into the HMM framework. Induction of suitable tree structures is discussed in detail, focusing on clustering algorithms and distance metrics developed specifically for acoustic modeling. We present *Hierarchies of Neural Networks* – hierarchical classifiers that rely on feed-forward neural networks for local conditional posterior probability estimation. Thousands of neural networks have to be optimized when training such a model for connectionist acoustic modeling. We present efficient training techniques that allow to train hierarchical models much faster than most existing connectionist acoustic models. Finally, we present experiments and recognition results using the proposed architecture on the Switchboard corpus.

5.1 Hierarchical Classifiers

Consider the task of classifying patterns $\mathbf{x}$ as belonging to one of N classes ω_k, where N is assumed to be very large ($N > 5000$). Applying the principle of *divide and conquer*, the task of discriminating between thousands of classes can be broken down into a hierarchical structure of many considerably smaller classification tasks. This idea underlies the approaches to decision tree architectures [Breiman et al. '84, Quinlan '86, Safavian & Landgrebe '91]. Decision trees classify input patterns by asking categorical questions at each internal node. Depending on the answer to these questions a single path is followed to one of the child nodes and the process repeats

until a leaf node is reached and a winner class label is emitted. However, decision tree classifiers are restricted to hard decisions. No information about the confusability of a specific input pattern is available. Rather, we are often interested in the posterior class probabilities $p(\omega_k|\mathbf{x})$ as a measure of the degree of class membership. The optimum choice in the Bayes' sense then is to pick the class with maximum a-posteriori probability[1]. Furthermore, it is sometimes required to supply a measure of the degree of membership for all potential classes to a superordinate decision making process as, for instance, in statistical speech recognition. Adhering to the divide and conquer approach but generalizing the decision tree framework, the statistical method of factoring posteriors can be applied to decompose the class posteriors hierarchically. We call the resulting architecture a *soft classification tree*.

5.1.1 Hierarchical Decomposition of Posteriors

For now, we assume, that optimal posterior probabilities are available. Let S be the set of classes ω_k to be discriminated. Consider we have a method at our disposition which gives us a partitioning of S into M disjoint and non-empty subsets S_i such that members of S_i are almost never confused with members of S_j $(\forall j \neq i)$. A particular class ω_k will now be a member of S and exactly one of the subsets S_i. Therefore, we can rewrite the posterior probability of class ω_k as a joint probability of the class and the corresponding subset S_i and factor it according to

$$\begin{aligned} p(\omega_k|\mathbf{x}) &= p(\omega_k, S_i|\mathbf{x}) \qquad \text{with} \qquad \omega_k \in S_i \\ &= p(S_i|\mathbf{x})\, p(\omega_k|S_i, \mathbf{x}). \end{aligned}$$

Thus, the global task of discriminating between all the classes in S has been converted into (1) discriminating between subsets S_i and (2) independently discriminating between the classes ω_k remaining within each of the subsets S_i. This two-stage process can be interpreted as corresponding to a tree-structured architecture (see Fig. 5.1). In this tree structure, the root node (first level) performs coarse classification between the subsets S_i, while the second level nodes perform classification among the classes ω_k within each subset S_i. Each base class ω_k is represented by a leaf node in the tree. By subdividing subsets S_i further and hierarchically repeating the process of factoring conditional posteriors, we can build larger, deeper tree structures.
In the limit, a N class classification problem can be decomposed into a binary tree structure consisting of $N-1$ nodes, each modeling a binary classification problem (see Fig. 5.2). Note however, that the branching factor does not have to be binary

[1] Assuming that equal risks are assigned to all classes.

or constant for all nodes in the classification tree but that it might be subject to optimization during the tree design phase.
In order to compute the posterior probability for a specific class, we have to follow the path from root node to the leaf corresponding to the class in question, taking the product of all the conditional posteriors along the way. Both the design of the tree structure (*divide*) and the estimation and multiplication (*conquer*) of conditional posteriors at each node are important aspects in this architecture, that have to be considered thoroughly because in practice, only approximations to the conditional posteriors are available [Schürmann & Doster '84].

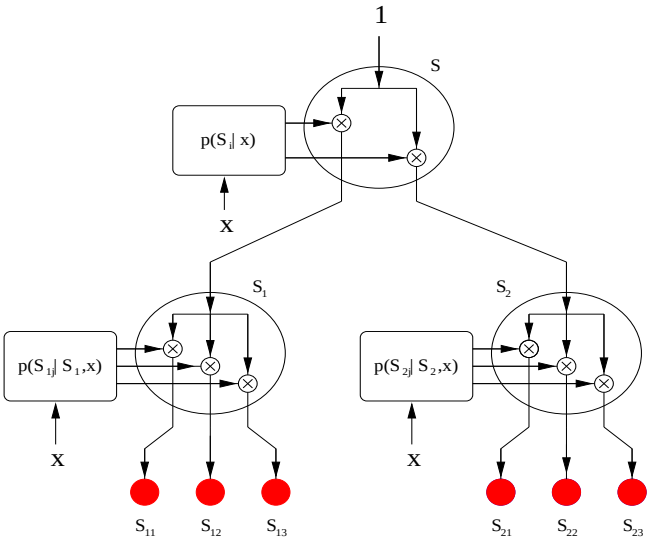

Figure 5.1: Hierarchical decomposition of posteriors

5.1.2 Properties

In addition to being applicable to classification tasks involving thousands of classes, hierarchical soft classifiers possess a variety of other interesting properties that make this kind of model particularly interesting for connectionist acoustic modeling.

- **Mass Distribution:** The presented architecture can be interpreted as a probability mass distribution device. At the root node, an initial probability mass

of 1 is fed into the architecture. At each node, the incoming probability mass is multiplied by the respective conditional posterior probabilities and fed into the child nodes. Eventually, the probability mass is distributed among all the leaves (classes) rendering their posterior probabilities. In contrast, decision trees represent hard-switching devices, where only a single path from root node to one of the leaves is considered.

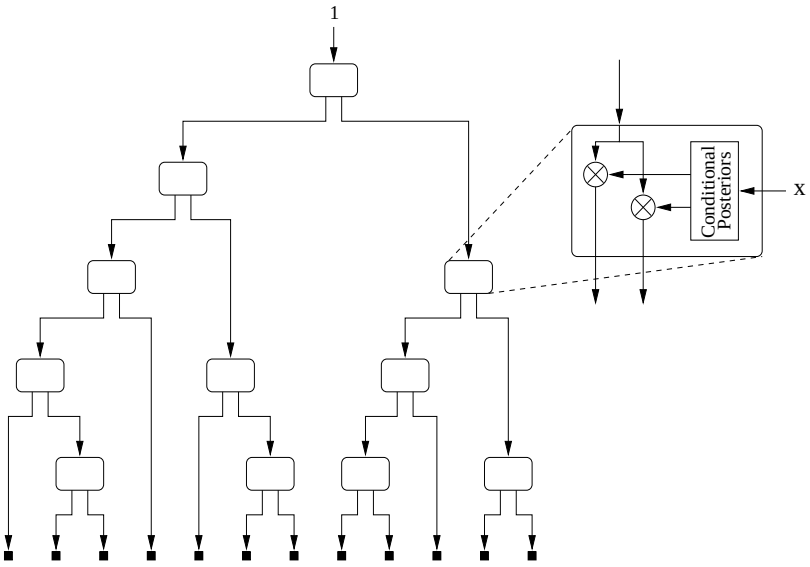

Figure 5.2: Binary tree structure for computing class posteriors

- **Fault Tolerance:** If one of the nodes in a classification tree, for example the root node, fails to provide good estimates of conditional posteriors, a hard decision tree will produce many classification errors due to the greedy local decisions. In contrast, such shortcomings will influence the decision process less dramatically in a soft classification tree as classification decisions are being delayed until the tree is fully evaluated and the complete posterior probability distribution is available at the leaf nodes. More general, Fig.5.3 demonstrates that greedy local choices as performed in a hard decision tree do not necessarily lead to the maximum a-posteriori leaf node (the one chosen by a soft classification tree).

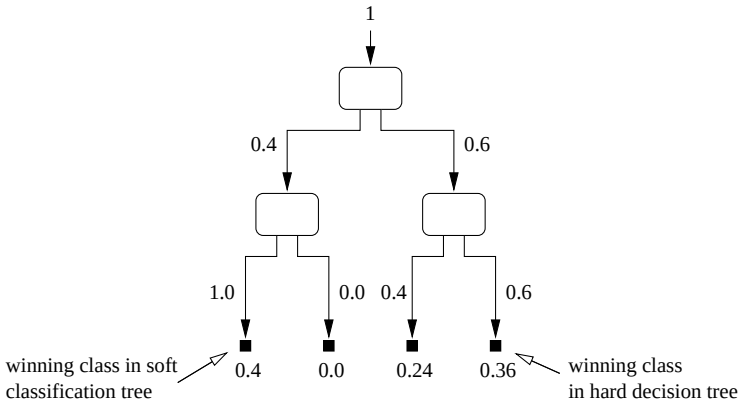

Figure 5.3: Difference between soft classification tree and hard decision tree

- **Cross Sectioning:** A very interesting and important aspect of soft classification trees is the sum-to-unity property observable in any horizontal cross section at any level of the tree (see Fig. 5.4. For any cross section, the partial posteriors computed down to the specific tree level sum up to 1 and constitute a valid posterior probability distribution. Thus, a soft classification tree can be cut off at any tree level and still be used to compute posterior probabilities for a reduced number of classes. This is equivalent to merging the original classes according to the tree topology up to the level of cross sectioning. The resulting classification task will be less specific and often easier to solve than the original one.

- **Pruning:** Related to the sum-to-unity property of cross sections is the property that partial posteriors computed on a path from the root node to a leaf are decreasing monotonically. This in turn allows to close paths whenever the partial posterior falls below a suitable threshold, thereby pruning whole subtrees with classes that would otherwise receive posteriors smaller than the threshold. This property yields the possibility to smoothly trade-off classification accuracy against computational complexity. In the limit, when only a single path with highest conditional posterior probability is followed, the soft classification tree resembles a decision tree.

The above properties together with the fact that soft classification trees are suitable to any size of classification task render this kind of model an optimal choice for

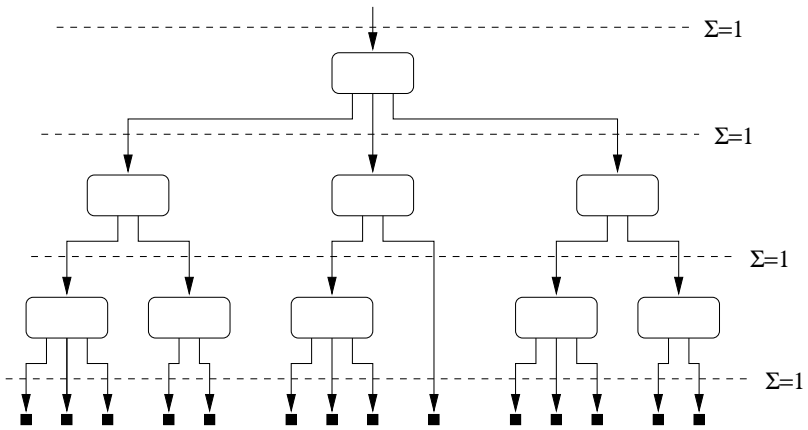

Figure 5.4: Sum-to-unity property of cross sections

locally discriminant acoustic modeling. However, in order to be applicable in a speech recognition system, the following issues have to be addressed:

- Given a set of HMM states to be modeled, how can we construct a suitable tree structure for hierarchical acoustic modeling?
- How can we estimate the local conditional posterior probabilities required at each tree node?
- How can a hierarchical acoustic model be trained efficiently, given the huge amount of training data typically required for speech recognition?
- How can a hierarchical classifier be integrated into an HMM based speech recognition system?

The remainder of this chapter is therefore devoted to the analysis of the above issues and to the specific algorithmic and architectural solutions that we have developed in order to realize a speech recognition system based on a hierarchical acoustic model.

5.2 Tree Construction

When it comes to the design of soft classification trees, or equivalently to the design of hierarchical decompositions of class posteriors, the choice of algorithm depends

mostly on the number of initial classes. In our specific case, we are seeking a tree structure on top of a set of HMM states, that results in an effective hierarchical decomposition of the state posteriors. As the set of base classes in this case consists of the set of decision tree clustered HMM states, we might consider to adopt the structure of the context clustering decision trees for the purpose of hierarchically factoring the state posteriors. However, the set of decision trees typically constitutes a bad choice for hierarchical acoustic modeling for the following reasons:

- In the standard approach, phonetic decision trees are grown independently for each monophone. One of the motivations for this restriction is reducing the computational complexity of tree growing. However, there is considerable evidence for acoustic similarities between allophonic variants of different monophones which suggest to not enforce any such constraint when the tree structure is to be exploited for hierarchical acoustic modeling.

- Phonetic decision trees typically exhibit a strong imbalance due to the categorical questions used for splitting nodes. While the yes-branch of such questions contains only few examples, the no-branch contains the majority of examples and therefore is subject to excessive further splitting, causing the imbalance. In contrast, a more balanced tree is desirable for hierarchical soft classification of HMM states in order to exploit the divide-and-conquer principle most effectively.

Instead of adopting the most likely suboptimal phonetic decision tree structure, we will investigate approaches at constructing alternative tree structures specifically for the purpose of hierarchical soft classification of HMM states. To that end, we postulate the following design criteria for tree structures that are to be used for hierarchical acoustic modeling.

1. Acoustic similarity of child nodes should be smallest for the root node and should increase monotonically towards the bottom of the tree. This is to ensure that the complexity of the local classification tasks increases from top to bottom. As the quality of estimates of the local conditional posterior probabilities at the root node influences the accuracy of posterior probabilities of all leaf nodes, we want to have a classification task as easy as possible at the root node. Further down the tree, the accuracy of estimates of the local conditional posteriors becomes less and less critical.

2. Balanced trees should be favored, such that an approximately equal number of nodes have to be traversed to evaluate any leaf node.

3. The a-priori distribution of child nodes should be close to a uniform distribution for any tree node to allow for the training of accurate estimators of the local conditional posterior probabilities.

We will first discuss optimal tree structures before we will turn to locally optimal algorithms required when dealing with the large number of classes typically encountered in context-dependent acoustic modeling.

5.2.1 Optimality

The optimal soft classification tree for a given task and given type and structure of estimators for the conditional node posteriors is the one which results in minimum classification error. If all the node classifiers would compute the true conditional posteriors, the tree structure would have no influence on the classifier performance because any kind of factoring (through any kind of tree structure) yields an accurate decomposition of the class posteriors. However, in practice, approximation errors of node classifiers render the choice of tree structure an important issue. For small numbers of classes, the optimal tree can in principle be found by exhaustively training and testing all possible partitionings for a particular node (starting with the root node) and choosing the one that gives the highest recognition accuracy. However, even if restricting the tree structure to binary branching nodes and balanced partitionings, the number K of partitionings that have to be examined at the root node

$$K = \binom{N}{N/2}$$

quickly brings this algorithm to its limits, even for a moderate number of classes N. Therefore, we have to consider heuristics to derive potentially sub-optimal tree structures. For example, one valid possibility is to assume that the achievable accuracy of approximations to the true posteriors is related to the separability of the corresponding sets of classes.

5.2.2 Prior Knowledge

Following the above mentioned guideline, prior knowledge about the task in question can often be applied to hierarchically partition the global set of classes into reasonable subsets. The goal is to partition the remaining set of classes in a way that intuitively maximizes the separability of the subsets. For example, given a set of phones in a speech recognizer, a reasonable first partitioning would be to build subsets consisting of voiced and unvoiced phones. In larger speech recognition systems where we have to distinguish among multiple context-dependent phone states, prior knowledge such

as state and context identity can be used as splitting criterion (see Fig. 5.5). In tasks such as speaker or writer identification, features such as gender or age are potential candidates for splitting criteria.

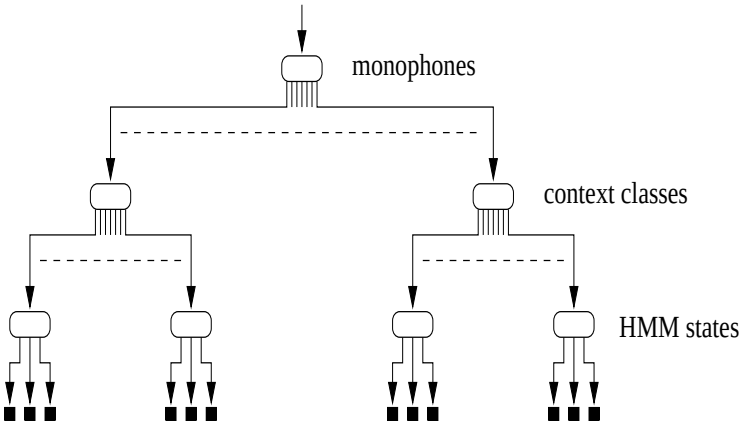

Figure 5.5: Three level tree structure for context-dependent hierarchical acoustic modeling constructed based on prior phonetic knowledge

The advantage of such knowledge driven decompositions is a fast tree design phase which is a clear superiority of this approach when dealing with large numbers of classes. However, this method for the design of hierarchical classifiers is subjective and error prone. Two experts in a specific field might disagree strongly on what constitutes a reasonable hierarchy. Furthermore, it is not always the case that *reasonable* partitionings yield good separability of subsets. Expert knowledge can be misleading.

5.2.3 Confusion Matrices

In case the number of classes is small enough to allow the training of a single classifier, the design of a soft classification tree can be based on the confusion matrix of the trained monolithic classifier. Indicating the confusability of each pair of classes, the confusion matrix yields relatively good measures of the separability of pairs of classes. This information can be exploited for designing a tree structure using a clustering algorithm. For instance, we can define the following (symmetric) distance measure between two disjunct sets of classes S_k and S_l

$$d(S_k, S_l) = -\sum_{\omega_i \in S_k} \sum_{\omega_j \in S_l} C(\omega_i, \omega_j | \mathcal{T}) + C(\omega_j, \omega_i | \mathcal{T})$$

where $C(\omega_i, \omega_j | \mathcal{T})$ denotes the number of times class ω_i is confused with class ω_j as measured on a set of labeled patterns $\mathcal{T}$. The distance $d(S_k, S_l)$ can now be used as a replacement for the usual Euclidean distance measure in a standard bottom-up clustering algorithm. Unfortunately, once the number of classes increases to several thousand, training of a monolithic classifier becomes increasingly difficult.

5.2.4 Cluster Methods

Assuming that separability of classes correlates with approximation accuracy of estimators for the posterior class probabilities, we can go further and assume that separability of classes can be measured by a suitable distance between a class representative or the class conditional distributions in feature space. Examples of distance measures include, for instance, the Euclidean distance between class means or the Mahalanobis distance between the classes second order statistics. Irrespective of the chosen distance measure, the goal always is to group the set of classes in a way that results in maximum inter- and minimum intra-group distances. Solutions to this problem are known as clustering algorithms (e.g., [Duda & Hart '73]). We will investigate this class of algorithms in more detail in the following section and develop efficient solutions for designing soft classification trees.

5.3 Bottom-Up vs. Top-Down Clustering

Considering the large amount of HMM states that we intend to model with hierarchical classifiers, efficiency and practicality of a tree growing algorithm is of foremost importance. It is for this reason, that we are focusing on cluster methods for constructing the proposed hierarchically organized acoustic models. In the following, we develop specific solutions for clustering hierarchical classifiers based on parametric models (single Gaussians) for the HMM states and suitable distance measures between such densities, comparing the two major types of hierarchical cluster methods, namely agglomerative (bottom-up) and divisive (top-down) algorithms. Agglomerative procedures start with n singleton clusters as the leaf nodes and construct a tree structure bottom-up by successively merging clusters. In contrast, divisive procedures start with all the samples in one cluster (the root node) and construct a tree structure top-down by successively splitting clusters. We will see that both approaches exhibit pros and cons with respect to our application.

5.3.1 Distance Measures

As already mentioned, we represent each initial class (HMM state) probabilistically using a single Gaussian density. In contrast to more simple representations, for instance consisting of just a mean vector, the additional second order statistics allow for more accurate distance measures. Gaussian densities for each HMM state can easily be obtained from the training data using maximum likelihood estimation.
In our work, we have investigated two distance measures based on second order statistics for hierarchical clustering, namely one based on information divergence and one based on split likelihood gain.

- **Symmetric Information Divergence:**
  Information divergence, also known as Kullback-Leibler (KL) divergence [Kullback & Leibler '51], measures the amount of information that is lost, when approximating a continuous probability density p_i with some other continuous density p_j. In its basic form, it is defined as

$$KL(p_i, p_j) = \int_{\mathbf{x}} p_i(\mathbf{x}) \log \frac{p_i(\mathbf{x})}{p_j(\mathbf{x})} d\mathbf{x}$$

  It measures how closely p_j resembles p_i. For increasing similarity, the KL divergence approaches zero, which is obtained only in case of $p_i = p_j$. Unfortunately, the standard form of the KL divergence is not symmetric and hence can not be used directly as a distance measure. However, a symmetric version of the KL distance can easily be derived:

$$d(p_i, p_j) = KL(p_i, p_j) + KL(p_j, p_i) = \int_{\mathbf{x}} (p_i(\mathbf{x}) - p_j(\mathbf{x})) \log \frac{p_i(\mathbf{x})}{p_j(\mathbf{x})} d\mathbf{x}$$

  In our case, the densities p_i and p_j model the distribution of acoustic vectors in HMM states s_i and s_j and are parameterized by normal densities. The above KL divergence thus measures the amount of dissimilarity between HMM states s_i and s_j. One can show (e.g., [Tou & Ganzales '74]) that the symmetric information divergence between two normal densities amounts to

$$\begin{aligned} d(N_i, N_j) &= \frac{1}{2} tr\{(\mathbf{\Sigma}_i - \mathbf{\Sigma}_j)(\mathbf{\Sigma}_j^{-1} - \mathbf{\Sigma}_i^{-1})\} \\ &+ \frac{1}{2} tr\{(\mathbf{\Sigma}_i^{-1} + \mathbf{\Sigma}_j^{-1})(\mu_i - \mu_j)(\mu_i - \mu_j)^t\} \end{aligned}$$

To reduce the computational load of a clustering algorithm that utilizes this distance measure, we typically restrict the Gaussian covariances to diagonal matrices, resulting in the following simplified distance measure

$$d(N_i, N_j) = \frac{1}{2} \sum_{k=1}^{D} \frac{\sigma_{ik}^2 + (\mu_{ik} - \mu_{jk})^2}{\sigma_{jk}^2} + \frac{\sigma_{jk}^2 + (\mu_{ik} - \mu_{jk})^2}{\sigma_{ik}^2} - 2$$

- **Split Likelihood Gain:**

  In contrast to symmetric information divergence, split likelihood gain measures the gain in likelihood obtained by splitting the data of a specific cluster into two halves and modeling each half separately. Therefore, this measure is primarily used for divisive clustering. We have already introduced split likelihood gain in our overview of phonetic context modeling in chapter 2 where it served as the optimization criterion in selecting phonetic questions for divisive growing of phonetic decision trees. Assuming perfect models for the distribution of our data vectors, there would be no gain from separately modeling parts of a given density. However in practice, we can only approximate the true distribution of our data, for instance by means of second order statistics. Consequently, considerable gains in likelihood can be obtained by splitting the data as is illustrated in Fig. 5.6.

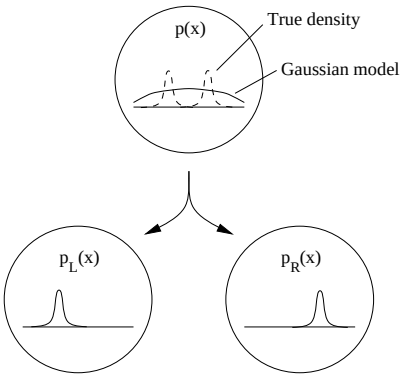

Figure 5.6: Split likelihood gain with Gaussian models

Split likelihood gain can also be considered a measure of the dissimilarity of the classes resulting from a split. In the limit of identical distributions for the data

in each class, its value converges to zero as there is no gain from modeling the classes separately. As already stated in chapter 2, split likelihood gain is defined as the difference in log likelihood between the child nodes and the parent node.

$$d(p, p_L, p_R) = \left(\sum_{\mathbf{x} \in N_L} \log p_L(\mathbf{x}) + \sum_{\mathbf{x} \in N_R} \log p_R(\mathbf{x}) \right) - \sum_{\mathbf{x} \in N} \log p(\mathbf{x})$$

Let $N(\mathbf{x}; \mu, \sigma^2)$ denote a diagonal covariance Gaussian density modeling the data distribution before the split and $N_L(\mathbf{x}; \mu_L, \sigma_L^2)$, $N_R(\mathbf{x}; \mu_R, \sigma_R^2)$ denote diagonal covariance Gaussian densities modeling the data distribution after the split. Hereby, split likelihood gain simplifies to the following, computationally efficient expression:

$$d(N, N_L, N_R) = n \sum_{k=1}^{D} \log \sigma_k^2 - \left(n_L \sum_{k=1}^{D} \log \sigma_{Lk}^2 + n_R \sum_{k=1}^{D} \log \sigma_{Rk}^2 \right)$$

where n, n_L and n_R denote the number of samples observed in the parent and child distributions, respectively.

5.3.2 Agglomerative Clustering

Our initial approach to hierarchically clustering HMM states for tree-structured connectionist acoustic modeling is based on agglomerative clustering. Starting with a set of leaf nodes representing the HMM states to be clustered, we successively create new tree nodes by merging existing ones according to their acoustic similarity, thereby constructing a binary tree structure in a bottom-up fashion. We model the data distribution at the initial leaf nodes by diagonal Gaussian densities and measure their dissimilarity using the symmetric information divergence introduced earlier.
In a straight-forward approach, we would merge the statistics of two child nodes to form a new, single Gaussian density for the parent node. However, as the available amount of data increases exponentially towards the top of the tree, the complexity of the corresponding distribution will also increase considerably. A single, diagonal Gaussian density must be considered a poor approximation of the distribution in the upper region of the tree. In order to improve the modeling accuracy, we developed an extension [Fritsch et al. '97] of standard agglomerative clustering which forms mixture densities as models of the data distribution in parent nodes by successively merging the initial Gaussian densities, using the within-cluster a-priori probabilities of the Gaussians as the mixture weights (see Fig. 5.7). This way, none of the initial information is lost during clustering, given that we generalize the distance measure to the information divergence between mixtures of Gaussians.

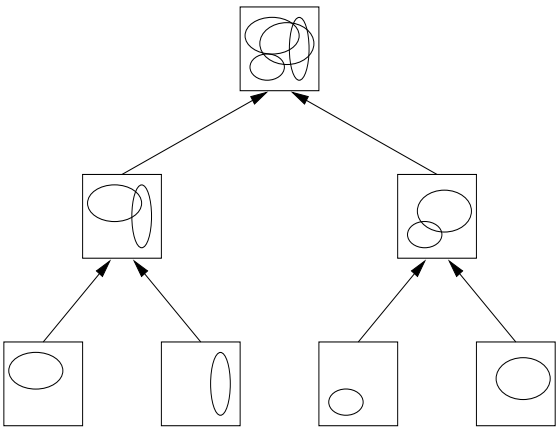

Figure 5.7: Forming Gaussian mixtures for improved agglomerative clustering

A computationally feasible approximation to the otherwise not analytically solvable integral expression for the KL divergence between Gaussian mixtures can be obtained by assuming linearity of the symmetric information divergence $d(S_k, S_l)$ between two clusters S_k and S_l:

$$\begin{aligned} d(S_k, S_l) &= d\left(\sum_{s_i \in S_k} p(s_i|S_k)\ N_i\ ,\ \sum_{s_j \in S_l} p(s_j|S_l)\ N_j\right) \\ &\approx \sum_{s_i \in S_k} \sum_{s_j \in S_l} p(s_i|S_k)\ p(s_j|S_l)\ d(N_i, N_j) \end{aligned}$$

Here, $p(s_i|S_k)$ and $p(s_j|S_l)$ denote the within-cluster a-priori probabilities of the states s_i in the cluster S_k and the states s_j in the cluster S_l. N_i and N_j denote the Gaussian models for states s_i and s_j, respectively.
Finally, Fig. 5.8 details the agglomerative clustering algorithm as we have used it for clustering tree structures for a given set of HMM states. The computational complexity of this algorithm is $O(n^3)$[2], where n is the number of HMM states. In our initial experiments, we were applying this algorithm to construct tree structures

[2]Under the assumption that we only have $O(n)$ memory available. If we would have $O(n^2)$ memory available, the computational complexity could be reduced to $O(n^2)$ using a priority queue for maintaining the distances between all pairs of states.

for context-independent modeling, typically involving less than 200 HMM states. For illustration purposes, Fig. 5.9 shows a dendrogram of a typical agglomerative clustering run on a relatively small set of only 56 HMM states corresponding to the set of single-state monophone HMMs in a context-independent Switchboard system. The model set consists of 44 standard English phones along with 7 noise sounds (marked with a plus), 4 phones modeling interjections (marked with an ampersand) and silence (SIL).

Agglomerative Clustering Algorithm

1. Initialize algorithm with n clusters S_i, each containing
   (1) a parametric model of the state-conditional likelihood
   for the corresponding state s_i, e.g., a diagonal Gaussian
   (2) a count C_i, indicating the frequency of state s_i in the training set.
2. Compute within-cluster priors $p(s_i|S_k)$ for each cluster S_k based on the counts C_i
3. Compute the symmetric divergence measure $d(S_k, S_l)$ between all pairs of clusters S_k and S_l.
4. Find the pair of clusters with minimum divergence, S_k^* and S_l^*
5. Create a new cluster $S = S_k^* \bigcup S_l^*$ containing all states from S_k^* and S_l^* plus their respective counts. The resulting parametric model is a mixture of Gaussians where the mixture coefficients are the state priors
6. Delete clusters S_k^* and S_l^*
7. While there are at least 2 clusters remaining, continue with 2.

Figure 5.8: Agglomerative clustering algorithm based on information divergence

Interestingly, the agglomerative clustering algorithm identifies clusters of phones that correspond roughly to well known linguistic classes, such as stop consonants, nasals and fricatives. The top level split separates silence, breathing and noise sounds (lower subtree) almost perfectly from speech sounds (upper subtree). Remarkable phone clusters that emerge early on during clustering consist of (IX,IH,IY,Y), (JH,CH,SH,ZH), (Z,S,F), (ER,AXR,R), (T,D,P,K). After these initial experiments with context-independent systems, we switched to context-dependent models with thousands of states as required for state-of-the-art performance in large, complex domains such as Switchboard. Fig. 5.10 shows a tiny part of the dendrogram for

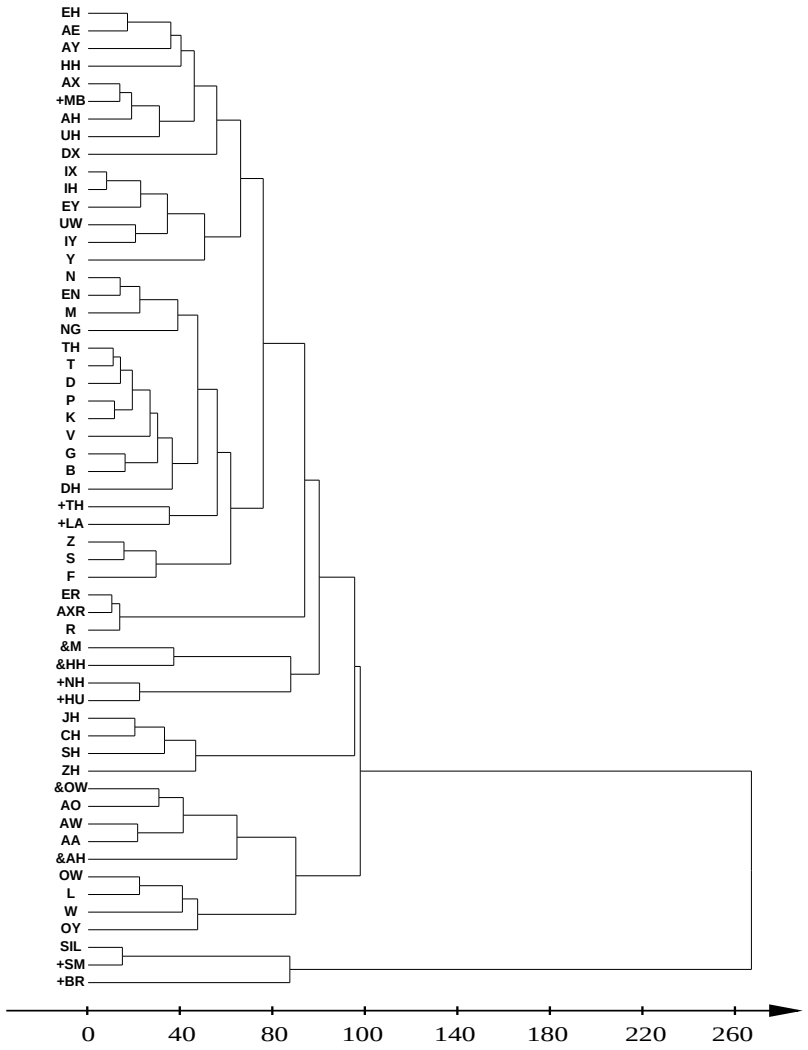

Figure 5.9: Agglomerative clustering of context-independent phones

unconstrained agglomerative clustering of 5000 context-dependent HMM states.

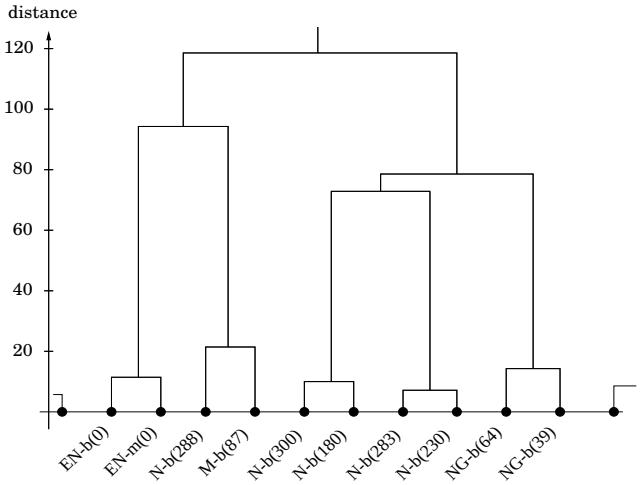

Figure 5.10: Agglomerative clustering of context-dependent HMM states

Note, that the algorithm groups together HMM states belonging to different monophones right from the start[3]. This observation confirms our earlier claim that the acoustic similarity of HMM states can sometimes be larger between allophonic variants of different monophones than between allophonic variants of the same monophone.
The basic agglomerative clustering algorithm as presented so far exhibits certain weaknesses that become most prominent when increasing the number of HMM states to be clustered. First of all, the algorithm tends to produce very imbalanced trees. The upper curve in Fig. 5.11 shows the average depth of leaf nodes for different numbers of leaf nodes. For comparison, we have included a curve that gives the depth of leaf nodes in a balanced binary tree for the same number of leaf nodes[4]. For 5000 HMM states (leaf nodes), the average depth of leaf nodes of a tree constructed by agglomerative clustering already reaches 80 with a standard deviation of over 50. As stated in the beginning of this section, such imbalanced trees are problematic and

[3] The state names consist of an initial monophone name, followed by the identifier for the position in a three state left-right HMM (b,m,e), followed by an identifier for the specific allophonic variant of that state (in brackets).

[4] This curve constitutes a lower bound for the average depth of leaf nodes in any binary tree

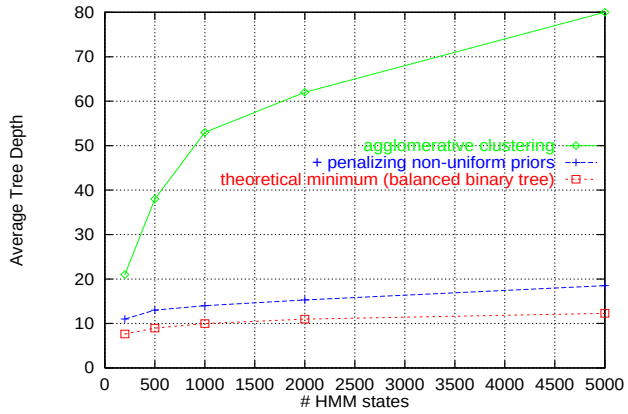

Figure 5.11: Unconstrained agglomerative clustering leads to imbalanced trees

undesirable for the task of hierarchically estimating HMM state posteriors. Thus, we incorporated an additional penalty term into the distance measure for clustering which enforces balanced trees by favoring a uniform distribution of the a-priori probabilities of child nodes. More specifically, we chose the negative entropy of the distribution $\{p, 1-p\}$ of a-priori child node probabilities

$$-H(p) = p\ \log(p)\ +\ (1-p)\ \log(1-p)$$

as the penalty term and incorporated it additively into our distance measure using an empiric weighting factor α as follows:

$$d(S_k, S_l)^\star = d(S_k, S_l) - \alpha H(p(S_k))$$

As can be seen in Fig. 5.11 ($\alpha = 100$), the additional penalty term allows to cluster balanced trees using the agglomerative algorithm. In Figs.5.12 and 5.13, we have investigated the effect of the additional penalty term in more detail.
For four different systems with 5000, 1000, 500 and 200 decision tree clustered, context-dependent HMM states, we have clustered soft classification trees using the agglomerative clustering algorithm with additional entropy penalty term. The effect of varying the weight α on the average depth of leaf nodes can be seen in Fig. 5.12. For increasing α, we obtain more and more balanced trees, until for $\alpha \geq 50$, this process eventually saturates. Particularly for systems with large numbers of HMM

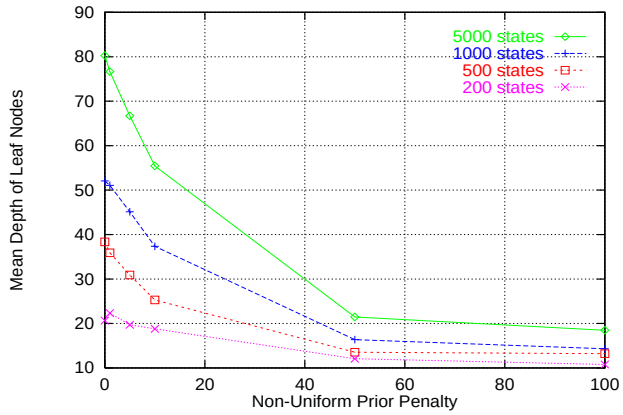

Figure 5.12: Average depth of leaf nodes when penalizing non-uniform priors during agglomerative clustering of binary trees

states such as the 5000 state system, the additional penalty term greatly improves tree balance.

However, balancing our agglomerative cluster trees is only an indirect effect of the additional entropy based penalty term. In fact, the original purpose of introducing this penalty term was to favor a uniform prior distribution of child nodes at each tree node. With respect to the estimation of posterior probabilities of child nodes, a highly non-uniform prior distribution typically leads to poor estimates for the infrequent child node, especially when a neural network is trained to estimate these posteriors. We will discuss this aspect in more detail in a later section on aspects of connectionist estimators for conditional posterior probabilities in our architecture. The effect of the additional penalty term on the prior distribution of child nodes can be seen in Fig. 5.13 for the same state sets and cluster runs already depicted in Fig. 5.12. This time, however, we have plotted the average normalized entropy of the a-priori distribution of child nodes vs. varying values of α. The normalized entropy has a range of $[0, 1]$, with 0 corresponding to one of the priors being zero and 1 corresponding to a perfectly uniform prior distribution. As expected, the incorporation of the additional penalty term not only leads to more balanced trees but is also effective in increasing the average normalized node entropy and thereby allows to control the prior distribution and to enforce uniform priors.

The second disadvantage of the basic agglomerative clustering algorithm is the re-

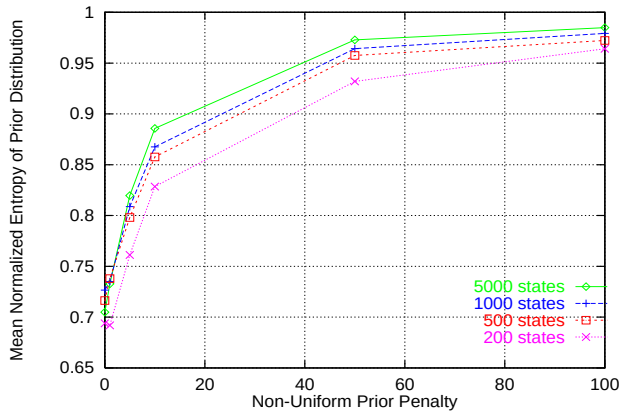

Figure 5.13: Average entropy of node prior distributions when penalizing non-uniform priors during agglomerative clustering of binary trees

striction to binary trees. A binary tree requires $n-1$ local estimators, one for each of $n-1$ internal nodes, given n leaf nodes. In our application, detailed context-modeling can easily lead to several thousand HMM states requiring the same number of local estimators for a binary hierarchy. Trees with a larger branching factor would allow to decrease the number of local estimators (at the cost of creating more complex classification tasks) and to decrease the average depth of the soft classifier tree. Extending the basic agglomerative clustering algorithm to allow for larger branching factors b essentially increases the computational complexity exponentially according to $O(n^{b+1})$, which is unfeasible in practice. However, we can construct b-ary trees from binary trees in a post-processing step by applying the following greedy bottom-up node merging algorithm:

1. Create a list P of tree nodes that initially consists of all leaf nodes
2. Determine a list Q of tree nodes that constitute root nodes of the largest subtrees containing $2 < n \leq b$ nodes from P.
3. For each node q in Q: Find the set of nodes $P_q \in P$ in the corresponding subtree and make them leaf nodes of a new node q' that replaces q. Set $P = (P \backslash P_q) \cup q'$. Throw away all other nodes in the subtree of q.
4. While P contains more than 2 nodes, go to step 2.

In addition to criteria such as tree balance, uniformity of priors and possible branching factors, computational complexity and scalability also are important aspects of a tree construction algorithm. The relatively high computational complexity of the proposed agglomerative clustering algorithm diminishes its applicability to clustering more than about 1000 HMM states. However, allowing for the following simplification of the proposed algorithm, we can significantly reduce its computational complexity: Consider only the distances between the cluster with smallest count and all other clusters in step 3 of algorithm 5.8 (this reduces the complexity from $O(n^3)$ to $O(n^2)$). This way, low probability states are grouped together early in the clustering process, increasing cluster mass rapidly such that later decisions will be based on reasonably reliable cluster models. Furthermore, this strategy naturally leads towards balanced trees.

In summary, agglomerative clustering based on information divergence is a viable strategy for constructing hierarchical soft classifiers for connectionist acoustic modeling. However, we had to modify and extend the basic algorithm in order to make the algorithm more efficient, more flexible and to enforce balanced trees and uniform priors.

5.3.3 Divisive Clustering

As an alternative to agglomerative clustering, we have investigated divisive (top-down) clustering. In divisive clustering, we start with a single cluster containing all the HMM states and successively split clusters until only clusters containing a single HMM state remain. Top-down approaches have the advantage, that if most interest is on the upper levels of the resulting tree structure, they are more likely to produce informative clusterings. In the binary case, a divisive method has to consider $2^{n-1}-1$ partitions of n states into two non-empty sets at the first step. In general, this is computationally unfeasible, so we have to apply reasonable heuristics such that we only have to consider a small proportion of these partitions. In our case, we seek a division into two clusters that maximizes their dissimilarity, measured by means of the split likelihood gain distance measure introduced earlier. There are various heuristic approaches to find a division that yields a dissimilarity as close as possible to the maximum (e.g., variants of k-means clustering). As k-means is not directly applicable to split likelihood gain, we developed an iterative method for divisive clustering of HMM states (see Fig. 5.14).

Note, that this instance of a divisive algorithm allows to construct trees with arbitrary branching factor, not just binary trees as is the case with the standard agglomerative clustering algorithm. Furthermore, experimental evaluation of the above algorithm revealed, that the cost of constructing b-ary trees depends only linearly on b, even though the theoretical number of possible legal partitionings grows exponentially with

increasing b. Note also the similarity of the given divisive algorithm to the one that is typically used for growing phonetic decision trees for context modeling. However, a finite set of questions allows to limit the number of splits to be considered for growing phonetic decision trees to a few hundred, whereas the algorithm in Fig. 5.14 relies on a greedy optimization heuristic.

Divisive Clustering Algorithm

1. Initialize algorithm with a single cluster, containing sufficient Gaussian statistics (accumulators) for each one of the HMM states s_i to be modeled
2. Randomly select b states from the current cluster and use their statistics as initial models for b new child nodes
3. For all remaining states in the current cluster: Find the child node for which the split likelihood criterion is maximized when adding the state's statistics and put it into that node
4. Randomly select a child node and a state in that node: Compute the difference in split likelihood gain when removing the state from the selected node and putting it into each one of the other child nodes
5. If split likelihood gain can be increased, move the selected state from the selected child node into the child node which gives maximum gain and go back to step 4
6. For each child node: If there are more than b states left, continue with step 2; otherwise create leaf nodes for the remaining states

Figure 5.14: Divisive clustering algorithm for constructing b-ary trees based on split likelihood gain

In addition to being computationally more efficient than the agglomerative counterpart and allowing to directly construct trees with branching factors $b \geq 2$, the above divisive cluster algorithm offers yet another advantage in that it creates more balanced trees. Fig. 5.15 shows how the average depth of leaf nodes in trees clustered with the divisive algorithm decreases with increasing branching factor. The curves are given for different numbers of leaf nodes (HMM states).

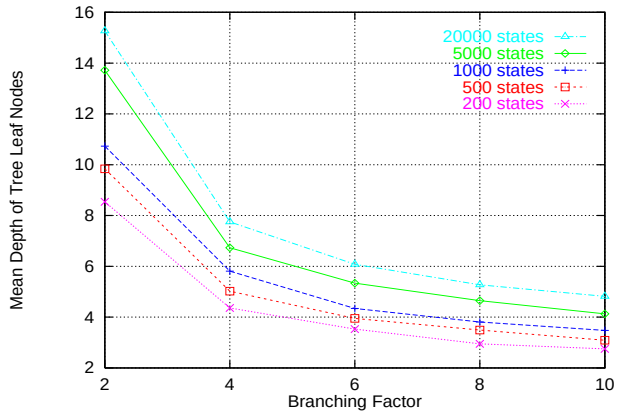

Figure 5.15: Average tree depth vs. branching factor for divisive clustering

Note that even a binary tree clustered for as many as 20000 HMM states is almost perfectly balanced without the use of any explicit penalty term in the distance measure as was required for agglomerative clustering. This behavior can be attributed to the fact that the split likelihood gain distance measure already favors uniform splits to some extent as it is highly dependent on the distribution of model counts.

In order to be able to visually compare the shape and quality of clustered trees, we next applied the divisive algorithm to a small set of context-independent HMM states, similar to the one used in Fig. 5.9. Fig. 5.16 depicts the resulting dendrogram. Note, that the dendrogram is plotted on a log-scale since split likelihood gain correlates with the amount of data being split (the model counts) which decreases exponentially due to the splits being applied during clustering. Clusters of phones similar to the ones found in agglomerative clustering can be identified in Fig. 5.16, e.g., (JH,CH,SH,ZH), (Z,S,F) and (M,N,NG). However in divisive clustering, the specific value of the distance measure at which a split occurred is more indicative of the amount of data (sum of counts of the corresponding HMM states) in the cluster than of the acoustic similarity. In contrast, the agglomerative clustering algorithm allows for better analysis and comparison of the acoustic similarity of states and clusters across the tree since the applied distance measure is independent of the amount of data used to estimate the cluster statistics.

Although the divisive clustering algorithm does not suffer from the tree imbalance problem of standard agglomerative clustering, we still investigated the effect of explic-

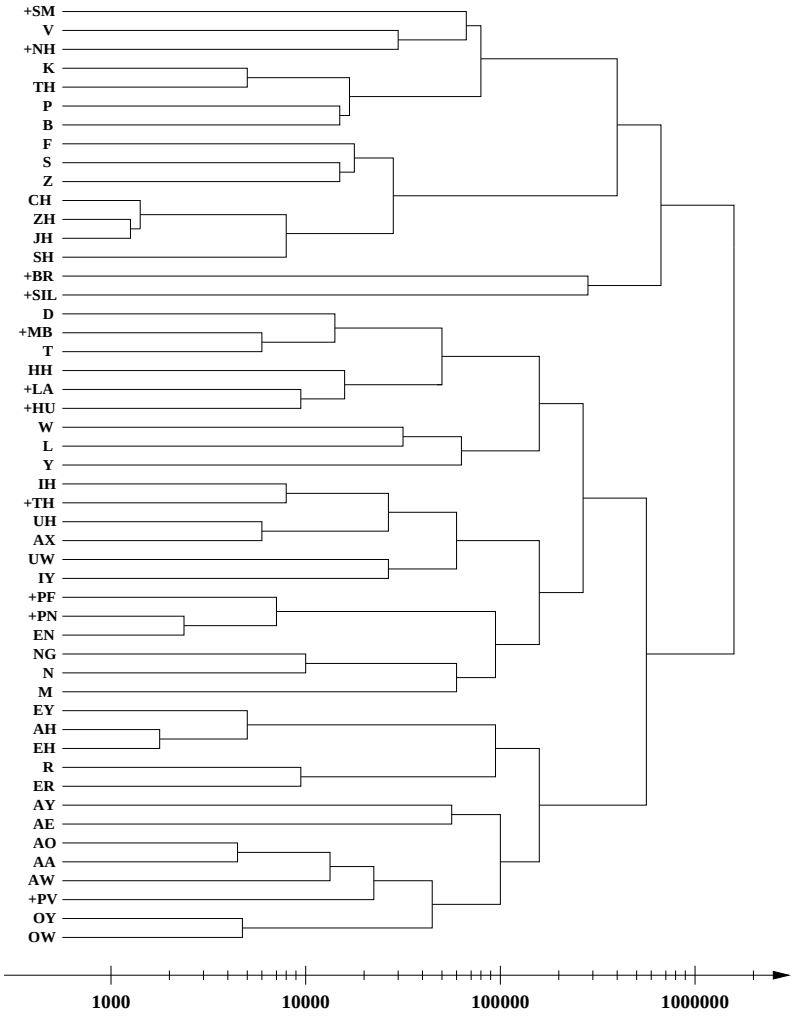

Figure 5.16: Divisive clustering of context-independent phones

itly enforcing uniform priors at each tree node during clustering. For that purpose, we modified the basic divisive algorithm in steps 3 and 4 by verifying that

$$b \min_{i=1,\ldots,b} p_i \geq \phi \qquad \phi \in [0,1]$$

is satisfied before adding or relocating any state in any child node (the p_i's are the child node priors). A value of $\phi = 0$ corresponds to the original divisive algorithm where no restrictions are imposed. Larger values of ϕ slowly enforce a uniform a-priori distribution until for $\phi = 1$, only a perfectly uniform prior distribution will be allowed. Of course, $\phi = 1$ is not a reasonable value in practice. The algorithm will fail to enforce the constraint as perfectly uniform prior distributions can normally not be realized. Fig. 5.17 and 5.18 give results for clustering runs with the extended divisive algorithm. As already observed in Fig. 5.15, the basic divisive algorithm already creates reasonably balanced trees. Consequently, the additional constraint on the prior distributions hardly reduces the average depth of leaf nodes as can be seen in Fig. 5.17. Only the curve for 5000 states shows a significant decrease in average depth of leaf nodes for increasing prior penalty ϕ.

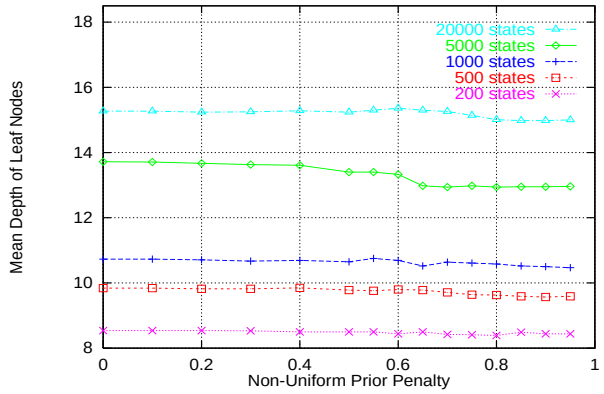

Figure 5.17: Average depth of leaf nodes when penalizing non-uniform priors during divisive clustering of binary trees

However, there is a measurable effect when examining the average node prior distribution (see Fig. 5.18). The average normalized entropy of prior distributions can be increased significantly for $\phi \approx 0.8$. Not surprisingly, the average entropy decreases

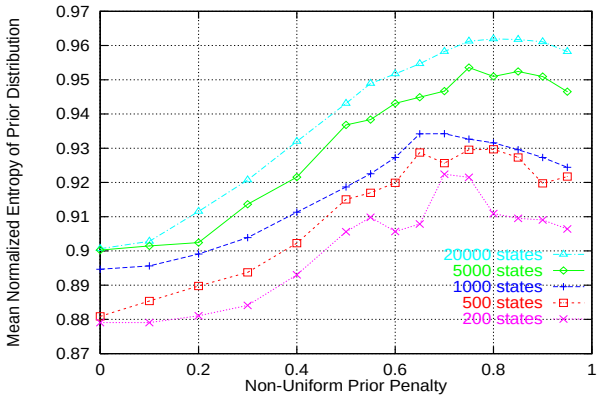

Figure 5.18: Average entropy of node prior distributions when penalizing non-uniform priors during divisive clustering of binary trees

again for larger values of ϕ as it becomes harder, if not impossible to enforce the uniformity constraint at some nodes. However, comparing Fig. 5.18 with Fig. 5.13, we can see that enforcing uniform priors is not nearly as important in the divisive as it is in the agglomerative algorithm.

5.3.4 Discussion

In summary, divisive clustering is computationally more efficient than agglomerative clustering and offers the attractive advantage of being more flexible in that it allows to directly construct trees with arbitrary branching factors. In addition, the trees resulting from divisive clustering are more balanced and the algorithm does not necessarily require any intervention to enforce tree balance as is the case with agglomerative clustering. Table 5.1 compares the main features of the two clustering algorithms presented in the previous two sections.

While these considerations lead us to a preference towards the divisive algorithm, it should be noted that this preference results mainly from a computational complexity point of view. In fact, agglomerative clustering often yields linguistically more meaningful tree structures. In their extended versions, both algorithms are capable of generating balanced tree structures that hierarchically represent acoustic similarity of HMM states - a prerequisite for effective hierarchical acoustic modeling.

Agglomerative Clustering	Divisive Clustering
emphasis on lower levels	emphasis on upper levels
better representation of acoustic similarity	more influenced by prior probabilities
binary trees	arbitrary b-ary trees
locally optimal	based on heuristics
explicit balancing required	yields balanced trees
computationally expensive	comparatively fast

Table 5.1: Comparison between agglomerative and divisive clustering algorithms

5.4 Local Probability Estimation

Once a suitable tree structure has been constructed using one of the methods presented in the previous section, it remains to provide estimators for the local conditional a-posteriori probabilities at each tree node. In this section, we discuss the various issues that have to be addressed in order to ensure that accurate estimates of the conditional posteriors can be obtained. In particular, we address the following issues:

- How to estimate conditional posteriors (5.4.1)
- What kind of connectionist model to apply (5.4.2)
- How to obtain suitable target vectors for training (5.4.4)
- How to determine the model complexity (5.4.5)
- What kind of learning algorithms to apply (5.4.6)

5.4.1 Estimation of Conditional Posteriors

Fig. 5.19 shows the task of estimating local conditional a-posteriori probabilities at a specific tree node. Given a certain input feature vector $\mathbf{x}$, the local estimator has to provide a-posteriori probabilities $p(S_i|S,\mathbf{x})$ for each one of the child nodes S_i, conditioned on the current node S. Of course, the estimates of the conditional a-posteriori probabilities have to satisfy

$$p(S_i|S,\mathbf{x}) \geq 0 \quad \forall i \qquad \text{and} \qquad \sum_i p(S_i|S,\mathbf{x}) = 1$$

in order to represent a valid posterior probability distribution.

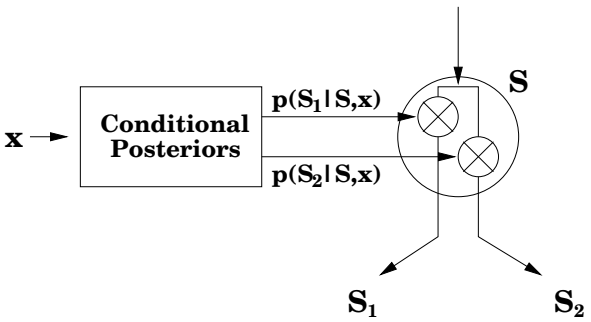

Figure 5.19: Local Probability Estimation

As already discussed in chapter 3, classifier neural networks have proven to be excellent tools for estimating posterior class probabilities, when trained appropriately. A wide variety of monolithic network architectures has been applied to the task of directly estimating HMM state posteriors, an approach that is commonly called connectionist acoustic modeling with the resulting speech recognition systems called hybrid NN/HMM systems. In the case of our hierarchical architecture, we have decomposed the task of jointly estimating HMM state posteriors into a tree structured, modular ensemble of smaller, localized tasks, namely to estimate conditional a-posteriori probabilities for child nodes in the tree. In order to apply neural network models to the estimation of these probabilities, we first have to solve the problem of estimating *conditional* posteriors using a classifier neural network.
Each of the modularized estimation tasks is associated with a particular tree node and is furthermore independent of all others. As the posteriors to be estimated are conditioned on the particular tree node the task is associated with, the conditional dependence on this tree node can be realized by restricting the training set for the local neural network estimator to training patterns of HMM states that are located within the subtree with the specific tree node as root node. Fig. 5.20 illustrates this technique for a three layer, binary hierarchy.
Associated with each leaf node (HMM state) is a particular set of training patterns[5], labeled with the index of the corresponding leaf node. Associated with each internal tree node is a neural network estimator for the conditional posterior probabilities of all its child nodes. The networks at the nodes in the lowest level of the tree are trained on the patterns of all their direct child nodes, which are at the same time

[5] Assuming, for now, that the underlying HMMs are trained according to the Viterbi algorithm that implies a one-to-one mapping between HMM states and training patterns.

leaf nodes. The training set for the networks at the nodes one level above consists of the combination of the training sets of the corresponding child nodes, which is approximately twice as big in a binary hierarchy. This process continues further up the hierarchy, with the nodes' training sets roughly doubling at each tree layer, until we reach the root node. The training set at the root node consists of all patterns of all leaf nodes, i.e. the complete training set from all HMM states. By restricting the training sets in the above described manner, we have set the basis for training the local neural networks to estimate the desired conditional posterior probabilities.

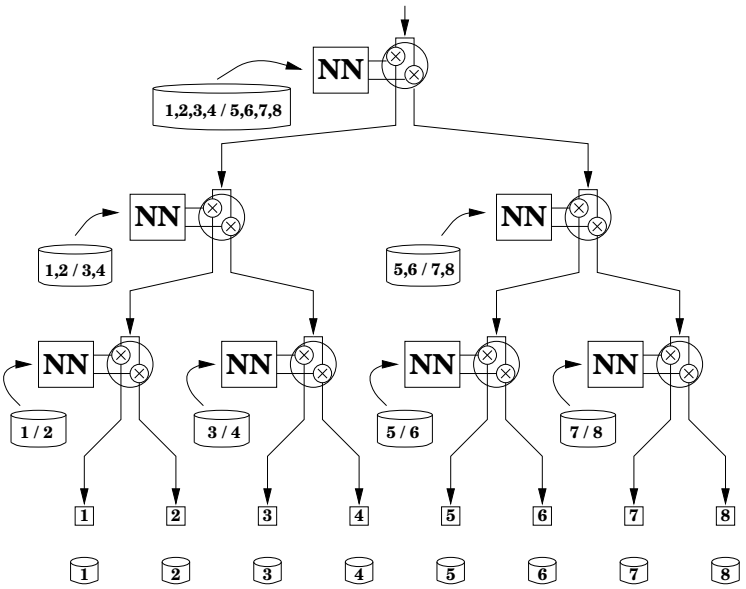

Figure 5.20: Hierarchical distribution of HMM state training data

It should be noted that there are other, non-connectionist approaches to the task of estimating a-posteriori probabilities, for instance polynomial regression [Schürmann '96]. However, as we will shortly see, connectionist models have the distinct advantage that particularly classification models are well understood in terms of statistical interpretation [McCullagh & Nelder '89] and can be realized such that they intrinsically adhere to the constraints of an a-posteriori probability distribution. Polynomial regression models, in contrast, require post-processing in form of confidence mapping to achieve

this property.
In the remainder of this thesis, we refer to a soft classification tree equipped with neural networks at each internal tree node for the estimation of the relevant conditional a-posteriori probabilities as a *Hierarchy of Neural Networks (HNN)* [Fritsch '97, Fritsch & Waibel '98]. When applied to the task of estimating posterior probabilities of HMM states in a hybrid, locally discriminative speech recognition system, we call our architecture a *Hierarchical Connectionist Acoustic Model*. We finally note that the Neural Tree model proposed in [Stromberg et al. '91] is architecturally similar to the Hierarchy of Neural Networks. However, Neural Trees represent a specific form of standard decision trees in which neural networks are used for making hard local decisions. Instead of computing a posterior probability distribution over HMM states, Neural Trees make hard decisions about the potentially correct HMM state (as do most of the decision tree models) and are therefore only of limited use for acoustic modeling in large vocabulary speech recognition.

5.4.2 Feed-Forward Classifier Networks

We chose a simple feed-forward architecture, the Multi-Layer Perceptron (MLP) with a single, non-linear hidden layer of problem-dependent size, a non-linear softmax output layer and fully interconnected layers without shortcuts as the sole connectionist model for the estimation of local conditional posterior probabilities in a Hierarchy of Neural Networks. Fig. 5.21 depicts the structure of such a model, to be used in a binary HNN.
The units in the hidden layer compute the weighted sum of their inputs which includes a bias unit with a constant activation of 1 and passes the result through a `tanh` shaped squashing function, a symmetric version of the commonly used sigmoid activation function (see also section 3.3). The nodes in the output layer also compute a weighted sum of their inputs which consist of the activations in the hidden layer. Again, an additive bias vector is included before the final network outputs y_k are computed through a softmax activation function. The softmax activation function has been chosen because in the terminology of generalized linear models, it represents the (inverse) canonical link to a multinomial probability model for a likelihood based objective function (cross-entropy) for multi-way classification [McCullagh & Nelder '89, Jordan & Jacobs '94, Jordan '95]. As we will shortly see, the network has to be trained as a multi-way classifier[6] in order to approximate a-

[6] In the case of a binary HNN, the networks in the tree structure have to be trained for binary classification which, according to the generalized linear model theory, involves a Bernoulli probability model and a single sigmoid output node. However, one can show that softmax for two output nodes is equivalent to a single node sigmoid. This allows to unify our approach and to represent both binary and multi-way classification using the same model.

posteriori probabilities and choosing the canonical link instead of an arbitrary squashing function at the network's output layer together with the appropriate objective function allows for statistical interpretation and in addition simplifies first-order training algorithms such as error backpropagation [Rumelhart et al. '86], as parts of the derivatives cancel out.

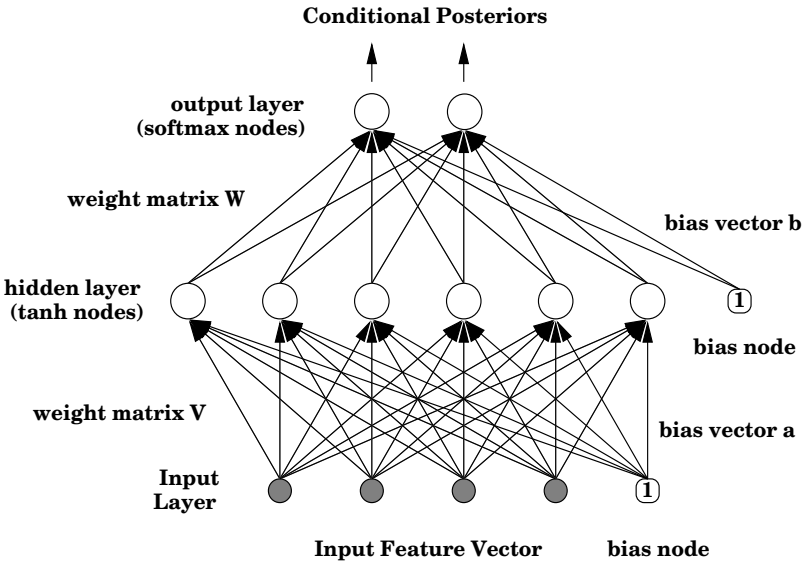

Figure 5.21: Multi layer perceptron (MLP) with a single hidden layer for local estimation of conditional posteriors in a binary HNN

Overall, the network jointly computes the following functions $y_k(\mathbf{x})$ at its output units which will be interpreted as estimates of an a-posteriori probability distribution over the feature space, from which $\mathbf{x}$ is taken:

$$y_k(\mathbf{x}) = \frac{\exp(z_k(\mathbf{x}))}{\sum_l \exp(z_l(\mathbf{x}))} \qquad z_k(\mathbf{x}) = \sum_j w_{kj} \tanh\left(\sum_i v_{ji} x_i + a_j\right) + b_k$$

where the v_{ji} denote the weights from unit i in the input layer to unit j in the hidden layer (subsumed in the weight matrix V), a_j denotes the bias weight for hidden unit j (subsumed in the bias vector a), the w_{kj} denote the weights from unit j in the

hidden layer to unit k in the output layer (subsumed in the weight matrix W) and b_k denotes the bias weight for output unit k (subsumed in the bias vector b).
Typically, we include the bias vectors a and b into the weight matrices V and W, respectively, and extend the input and hidden layer activation vectors by an additional constant of 1, which allows to formulate the network function more compact as

$$\mathbf{y} = \text{softmax}(W \tanh(V\mathbf{x}))$$

The number of input units in this architecture is fixed for all networks in the tree and is given by the dimensionality of the input feature space. The number of output units is equal to the branching factor at the tree node for which the network computes a-posteriori probabilities, i.e. the number of child nodes. Each unit in the output layer represents a particular child node of the corresponding tree node. The size of the hidden layer can be chosen arbitrarily and constitutes the single degree of freedom in terms of varying model complexity. We prefer this standard MLP architecture over more complex models for the following reasons:

- According to the universal approximation theorem that goes back to a theorem by the Russian mathematician Kolmogorov, any continuous n-variate function (e.g., the posterior probabilities in our case) can be approximated to an arbitrary degree of accuracy by an MLP with a single hidden layer of appropriate, finite size containing non-linear squashing functions. According to this, a single non-linear hidden layer is sufficient for general function approximation and the corresponding MLP constitutes the simplest architecture that exhibits this property.

- MLPs have been used extensively and successfully for the estimation of posterior probabilities in the past, especially in the field of connectionist acoustic modeling.

- The simplicity of the MLP and its layered architecture allows for efficient online training using the error backpropagation algorithm. Efficiency of parameter estimation is an important aspect of hierarchical connectionist acoustic models as the tree structures can contain several thousand nodes, requiring to train thousands of neural networks.

- The process of optimizing the network size in terms of generalization performance is simplified by the fact that there is only a single parameter (the number of hidden units) for controlling the model complexity.

5.4.3 Local Training Targets

Estimation of a (conditional) posterior probability density with the proposed type of neural network requires, in addition to an appropriate non-linearity at the output layer, that the network is trained as a pattern classifier, minimizing a suitable objective function for 1-out-of-N target vectors. In the case of our hierarchical connectionist architecture, such target vectors are obtained according to Fig. 5.22.

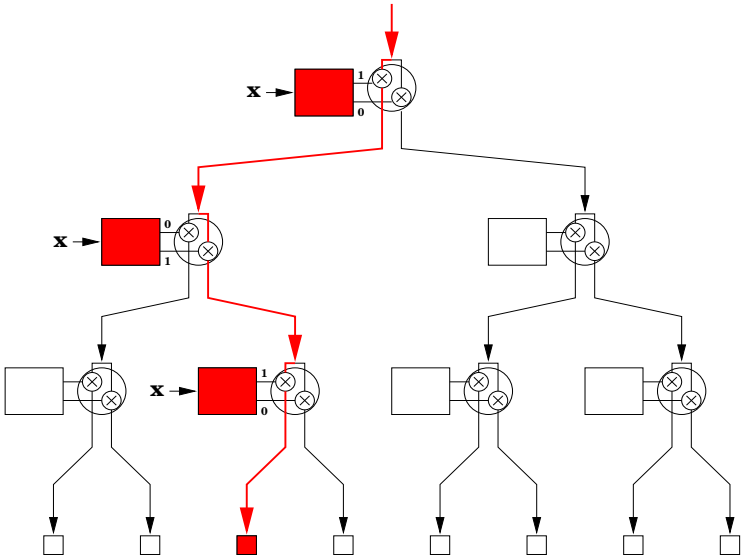

Figure 5.22: Local training targets for Viterbi based HMM training

Under the Viterbi assumption, an acceptable simplification in training HMMs for speech recognition, there is exactly one HMM state that is considered responsible for generating each pattern vector. In our tree structure, the Viterbi assumption implies that only a single leaf node has to be considered for each training pattern. As a consequence, only the networks at nodes on the path from the root node to the currently active leaf node receive training information for the corresponding training pattern. Local training targets for the networks at the nodes along that path consist of vectors of 0s with a single 1 at the position that corresponds to the next child node on the path. Depending on the training mode, the parameters of the local

classifiers are either updated after each training pattern (on-line training), after a certain number of patterns (batch training), or after all training patterns have been presented to the architecture (off-line training).

5.4.4 Model Selection

When applying connectionist models (or any other model) to the task of supervised pattern classification based on a finite training set, it is well known that the complexity of the model, i.e. the number of parameters employed, must be chosen carefully in order to avoid overfitting and achieve generalization to unseen data. Typically, there is a certain operating point for a given number of training patterns, where classification performance on an independent validation set is optimal. While classifiers with a smaller number of parameters are not capable of capturing the full complexity of the classification task, those with a larger number of parameters overfit to the training set and exhibit poor generalization to unseen data. This effect is known as the bias/variance-dilemma or -trade-off [Geman et al. '92, Tibshirani '96].
Basically, the prediction error of a learner can be decomposed into a sum of a bias (measuring how accurate the learner predicts the training data) and a variance component (measuring how much the learners prediction errors vary over different test sets), plus an additional term that quantifies the difficulty of the learning problem. Increasing the number of parameters reduces the bias of the predictor but at the same time increases the variance, while decreasing the number of parameters decreases the variance but increases the bias. The problem of selecting the optimum model size is usually addressed by one of the following approaches:

1. A-priori, knowledge based selection of model size
2. Iterative selection of model size (several trials)
3. Regularization [Girosi et al. '95]

We first investigate a-priori selection of the model size. Ideally, the complexity of local node classifiers should be selected so as to maximize generalization ability of the complete hierarchy. Generalization, on the other hand, is influenced by three factors: (1) size and distribution of the training set, (2) model complexity and (3) classification complexity of the specific task at hand. Obviously, we can not influence the latter of these factors. Furthermore, in the context of our architecture, we assume that the size of the training set for each tree node is fixed by the tree topology, once the hierarchy has been designed. Therefore, we have to choose the model complexity of the estimator at each node based on available training data and difficulty of classification task. The following Fig. 5.23 shows the amount of training data available on

average at each level in a typical binary hierarchical connectionist acoustic model for 6000 HMM states[7]. In accordance with the intuition already gained from Fig. 5.20, the number of available training patterns increases exponentially from the bottom of the tree to the top.

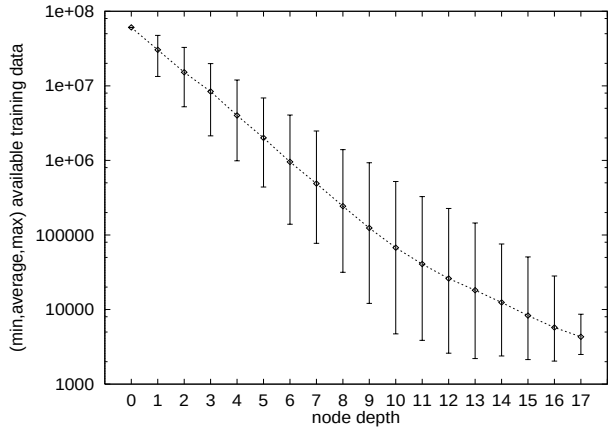

Figure 5.23: Available training data in different depths of HNN tree

The overall number of parameters in the type of neural network that we are using depends linearly on the number of hidden units. According to [Baum & Haussler '89] and with some approximations, a rule of thumb is to choose the number of hidden units M to satisfy

$$M < N\,\epsilon$$

where N denotes the number of available training patterns and ϵ is the expected error rate on the test set. In our case, the variation in the number of training patterns in the different nodes is expected to dominate the above formula. Therefore, a reasonable initial strategy is to set the number of hidden units proportional to b^{-d}, where b is the branching factor of the tree structure and d is the depth of the node. As long as the tree is approximately balanced in terms of the prior distribution of child nodes, this strategy leads to hidden layers with size proportional to the number of available training patterns. However, as an exponential increase in the number of hidden units

[7] The tree has been designed from the full Switchboard training set, using the agglomerative clustering algorithm from section 5.3.2, penalizing non-uniform priors.

(when going from the bottom of the tree to the top) quickly leads to unfeasibly large networks and the gains from making large networks even larger seem not to be worth the additional complexity, it is advisable to limit the number of hidden units to some predefined maximum.
The second approach to model selection is more thorough but at the same time considerably more expensive. In our case it consists of training a set of MLPs with different numbers of hidden units, e.g., 16, 32, 64, etc., for each tree node and comparing their performance on a previously unseen validation set, finally selecting the network which gives maximum accuracy and discarding all others. The disadvantage of this approach, of course, is that a large number of networks will be trained in vain and the overall training time increases significantly. Nevertheless, we have investigated this approach for building a hierarchical connectionist acoustic model for the Switchboard domain. We report results of these experiments in the evaluation section.
The effectiveness of the third approach, regularization, has been demonstrated for many other connectionist architectures [Girosi et al. '95] but remains to be investigated in the context of the proposed model in future work.

5.4.5 Optimization Algorithms

Even though the majority of individual networks in our tree structure have to be trained only on small proportions of the full training corpus, there are several hundred if not thousand such classification tasks to be processed in order to obtain a completely trained hierarchy (see next section). Therefore, we consider only stochastic learning/optimization algorithms that compute approximations to the gradients and update the parameters of a network either after each single training pattern or after a small batch of pattern vectors instead of after presentation of the complete training set.
Another important issue with our tree structured architecture is memory requirements. A feasible optimization algorithm has to be conservative in its memory requirements if we want to train all neural networks within a hierarchy while passing through the data. For instance, assuming that we have 256 MBytes available and our hierarchical acoustic model consists of about 4000 networks, an optimization algorithm must not take more than 64 KBytes of memory per network. Assuming furthermore that the local networks in our hierarchy typically consist of about 2000 weights, each taking 4 Bytes, the available memory for an optimization algorithm allows only to store the equivalence of about 8 times the vector of weights. Thus, we are limited to first-order optimization algorithms. A second order algorithm such as the Newton-Raphson method would require to store the Hessian of the objective function with respect to the vector of weights, a 2000 x 2000 matrix, for each network,

summing up to a total of about 256 GBytes in the scenario given above.
Consider a particular network at a particular node in an HNN tree. Let $\mathcal{T}$ denote the training set available for estimating the parameters of the network. The training set consists of pairs of input and target vectors:

$$\mathcal{T} = \{(\mathbf{x_1}, \mathbf{t_1}), \ldots, (\mathbf{x_n}, \mathbf{t_n})\}$$

The cross-entropy error function simplifies as follows under the assumption of 1-out-of-N target vectors:

$$\begin{aligned} E &= -\sum_{i=1}^{n}\sum_{j=1}^{b} t_{ij} \log y_j(\mathbf{x}_i) \\ &= -\sum_{i=1}^{n} \log y_{c(i)}(\mathbf{x}_i) \qquad \text{for } t_{ic(i)} = 1 \text{ and } t_{ij} = 0 \ \forall j \neq c(i) \end{aligned}$$

where $y_j(\mathbf{x})$ represents the function computed by the network at its j-th output unit. The goal of training is to minimize the above error function with respect to the weights in the neural network estimator. For the experiments presented in this thesis, we have investigated the following two optimization algorithms for training the networks in an HNN:

Stochastic Gradient Descent

In its standard formulation, gradient descent updates the vector of weights $\mathbf{w}_t$ at time t according to

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}_t \quad \text{with} \quad \Delta\mathbf{w}_t = -\eta \frac{dE}{d\mathbf{w}}(\mathbf{w}_t)$$

with scalar learning rate η. The on-line version of gradient descent, often called *stochastic* gradient descent computes a stochastic estimate of the true gradient from only a few training vectors in order to update the parameters of the network more often than just after presentation of the whole training set. As mentioned before, on-line operation of the optimization algorithm is crucial for achieving acceptable training times in speech recognition applications. It has frequently been observed that convergence of gradient descent can be sped up significantly by introducing a so-called *momentum* term:

$$\Delta\mathbf{w}_t = -\eta \frac{dE}{d\mathbf{w}}(\mathbf{w}_t) + \alpha\Delta\mathbf{w}_{t-1}$$

using a scalar momentum factor α. Although simple gradient descent is surprisingly effective in training neural networks, there is one noticeable disadvantage with respect

to our architecture: The learning parameters (learning rate η and momentum factor α) have to be tuned separately for the training of each network in a Hierarchy of Neural Networks as different networks require different parameter settings for optimal learning. Due to the large number of networks in such a hierarchy, we have to find ways of automatically tuning these parameters in order to obtain a reasonable training algorithm for the full HNN tree.

Scaled Conjugate Gradients

To avoid problems with learning parameters that have to be tuned specifically for each network in an HNN, we have investigated a second optimization algorithm that allows to train a neural network without requiring any crucial learning parameter. We adopted an algorithm called Scaled Conjugate Gradients (SCG) [Møller '93] which is a variant of the standard conjugate gradients algorithm that does not require a time consuming line search. In fact, the SCG algorithm contains no critical user dependent parameters and enables a fully automatic network training.
In the SCG algorithm, the vector of weights $\mathbf{w}_t$ at time t is updated according to the following iterative optimization rule:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \Delta\mathbf{w}_t \quad \text{with} \quad \Delta\mathbf{w}_t = \alpha_t \mathbf{p}_t$$

with the scalar α_t computed as follows

$$\alpha_t = \frac{\mathbf{p}_t^T \mathbf{g}_t}{\mathbf{p}_t^T \mathbf{s}_t}$$

where $\mathbf{g}_t$ denotes the negative gradient of E at time t with respect to the weights $\mathbf{w}$:

$$\mathbf{g}_t = -E'(\mathbf{w}_t) = -\frac{dE}{d\mathbf{w}}(\mathbf{w}_t)$$

and the conjugate gradients $\mathbf{p}_t$ computed recursively as follows

$$\begin{aligned} \mathbf{p}_0 &= \mathbf{g}_0 \\ \mathbf{p}_{t+1} &= \mathbf{g}_{t+1} - \beta_t \mathbf{p}_t \end{aligned}$$

with the scalar β_t according to

$$\beta_t = \frac{|\mathbf{g}_{t+1}|^2 - \mathbf{g}_{t+1}^T \mathbf{g}_t}{\mathbf{p}_t^T \mathbf{g}_t}.$$

The SCG algorithm differs from the standard conjugate gradients algorithm in the expression for the vector $\mathbf{s}_t$ which is approximated as follows to avoid computing the Hessian with respect to the weights:

$$\mathbf{s}_t = E''(w_t)\mathbf{p}_t \approx \frac{E'(\mathbf{w}_t + \epsilon\mathbf{p}_t) - E'(\mathbf{w}_t)}{\epsilon} + \lambda_t \mathbf{p}_t$$

The constant ϵ is uncritical as long as it is kept small enough and the scaling factor λ_t is adjusted automatically by the SCG algorithm depending on the positive definiteness of the Hessian (see [Møller '93] for further details).
Although the SCG algorithm theoretically requires 'true' gradients, i.e. off-line mode of operation, we have obtained satisfactory results with stochastic gradients computed from about 100 training patterns which again is crucial for our application. It should be noted though, that the SCG algorithm routinely fails to converge when using considerably less than the above mentioned 100 pattern vectors for computing estimates of the gradients.

5.4.6 Approximation Accuracy

While we have discussed how we can accurately estimate local conditional posterior probabilities in a hierarchical classifier using small classifier neural networks, it is not immediately clear whether the final estimates computed at the leaf nodes accurately approximate the real a-posteriori probability distribution. Inevitably, the local node classifiers can only produce estimates of the true conditional posteriors. Final class posteriors at the leaf nodes are computed by multiplying these local estimates in a specified manner. How do local approximation errors influence the global approximation error in a Hierarchy of Neural Networks?
A common way to empirically verify a classifiers ability to approximate posterior class probabilities is to compute a histogram for the probability of a classifier output $y_i(\mathbf{x})$ belonging to the correct target class. Formally, we estimate $P(i = c|y_i(\mathbf{x}))$, where $y_i(\mathbf{x})$ is the output of the classifier for class i, given an input feature vector $\mathbf{x}$ and c is the index of the target class. The plot in Fig. 5.24 was computed from the outputs of a trained Hierarchy of Neural Networks classifier for 8000 tied HMM states (see section 5.7.6 for details) fed with 500000 pattern vectors. A classifier that produces perfect a-posteriori probabilities would yield a histogram curve that follows the diagonal from $(0, 0)$ to $(1, 1)$. The closer a histogram curve follows that diagonal in practice, the more accurate are the classifier's approximations to the true posteriors.
As we can see from Fig. 5.24, our experiment yields an almost perfect diagonal, demonstrating that a Hierarchy of Neural Networks can indeed be trained to produce accurate estimates of posterior probabilities for a large number of classes. It appears that local approximation errors (if present) do not amplify but cancel out during the top-down computation of state posteriors in a hierarchical connectionist acoustic model.

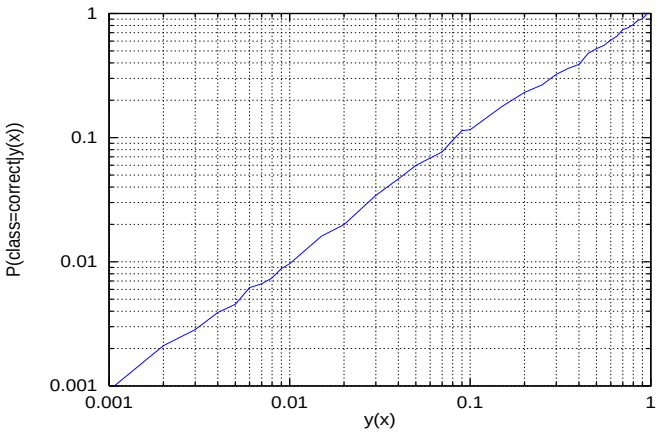

Figure 5.24: Empiric Validation of Posterior Approximation Property of Hierarchical Classifier for 8000 HMM States

5.5 Global Training Techniques

The training of connectionist acoustic models is typically reported to be orders of magnitude more expensive then training of comparable conventional acoustic models based on mixture densities. In the past, dedicated parallel processing hardware has often been necessary in order to train the sometimes very large connectionist acoustic models in reasonable times[8]. This is considered one of the major drawbacks of the connectionist approach to acoustic modeling.

Although the individual networks in our hierarchical connectionist model are comparatively small, training of the overall architecture is also computationally quite expensive due to the very large number of networks that typically constitute such a hierarchy. However, in contrast to monolithic connectionist models, training of our HNN model can still be realized efficiently on standard hardware as each network in an HNN tree can be trained independently of all others. The set of tree nodes with associated networks can be distributed among several standard computers and trained independently.

We have developed two parallelizable training techniques for HNN trees which will be described in the following. The first one is based on jointly training all the networks in an HNN tree while passing through the training data. In contrast, the second

[8] Where 'reasonable' quite often translates to 'several days'!

technique is based on extracting separate training sets for each one of the networks in an HNN tree using a single pass through the training data and then independently training all the networks in parallel on their respective training set.

5.5.1 Joint Training

Consider the process of jointly training all the networks in a Hierarchy of Neural Networks. We sequentially process the input training patterns and determine for each pattern the networks involved in estimating the posterior probability of the associated HMM state according to a Viterbi alignment (see Fig. 5.22). These networks are located on the path from root node to the specific leaf node representing the target HMM state. The associated training pattern is presented to each one of these networks together with 1-out-of-N target vectors that are constructed according to Fig. 5.22. At each time step, different networks will receive the respective training pattern depending on the target HMM state[9]. Whenever a network in the HNN tree has accumulated a certain predefined amount of training patterns (the batch size), we trigger a parameter update using one of the optimization algorithms presented in the previous section and start to accumulate training patterns again, possibly keeping local state information for the optimization algorithm (such as the previous gradient for computing momentum terms in subsequent updates).
There is no communication or synchronization required between the individual tree nodes. Thus, the entire training scheme can be parallelized and distributed easily among several processors. We simply keep the entire HNN tree structure on each processor but instantiate only disjunct sets of nodes with networks. After training, we merge the networks from each training process into a single complete HNN tree structure. As no communication is required, the speed-up obtained from distributing joint training scales almost linearly with the number of available computers[10].
In principle, the individual nodes in an HNN tree can be distributed in any fashion. However, we certainly prefer configurations that result in sufficiently balanced computational load during training. Fig. 5.25 presents a strategy that aims at optimal load balance by grouping all nodes in every tree level and assigning a different processor to each such node cluster. At each time step during training, a single network in each cluster receives the current training vector according to the path from root node to the leaf node representing the current target HMM state. Assuming that all networks in the HNN get updated after having received a globally constant amount of training patterns, the presented strategy in fact achieves optimal load balance.

[9] Of course, the network at the root node will receive all training patterns, irrespective of the target HMM states.

[10] In practice, disk I/O can become a bottleneck since all parallel processes have to access the speech waveform data (or the precomputed feature vectors) during training.

In case we apply stochastic gradient descent to the task of jointly training the individual networks in an HNN, we have to tune the learning parameters specifically for each network. This is important because networks in different levels of the HNN tree receive vastly different amounts of training data. For instance, the network at the root node typically requires a comparatively small learning rate as it receives several million training patterns (in our application). In contrast, the networks at the bottom of the tree receive only a few thousand training patterns and therefore require considerably larger learning rates to guarantee fast convergence. Thus, we assign individual learning rates to each network in an HNN but keep a global momentum factor (typically $\alpha = 0.9$).

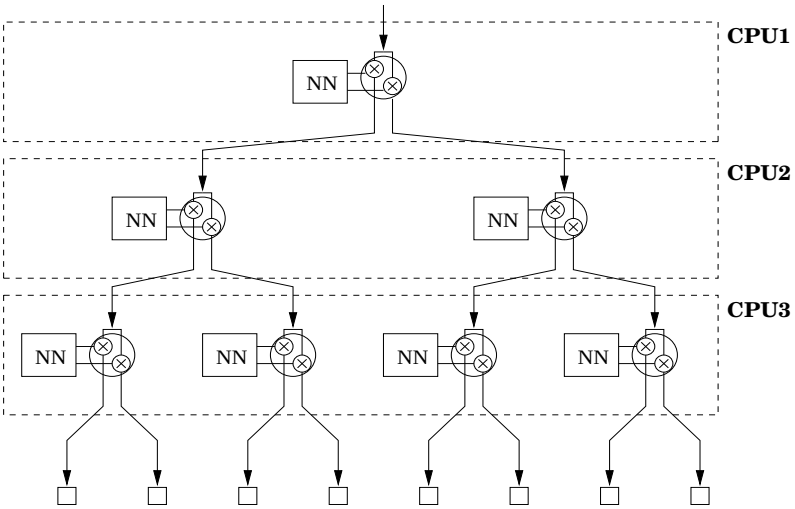

Figure 5.25: Distributing joint training of HNN nodes on several CPUs

The local learning rates η_i are initialized with a single global learning rate η_G. During the process of training the hierarchy, the local learning rates are adapted individually with the global learning rate functioning as an upper bound for the local learning rates to avoid divergence of the optimization process due to excessively large local learning rates. Furthermore, the global learning rate η_G is annealed according to the following rule to stabilize gradient descent towards the end of training:

$$\eta_G^{(t+1)} = \eta_G^{(t)} * \kappa_G$$

Typically, we use an initial global learning rate $\eta_G^{(0)}$ between 0.001 and 0.01 and a global annealing factor κ_G of $0.999 \ldots 0.9999$ applied after each 10000 training patterns.
In order to accommodate the different learning speeds of the node classifiers due to the different amounts of available training data, we control individual learning rates using the following measure of correlation between successive gradient vectors $g^{(t-1)}$ and $g^{(t)}$:

$$\gamma^{(t)} = \arccos\left(\frac{g^{(t)} g^{(t-1)}}{|g^{(t)}|\ |g^{(t-1)}|}\right)$$

$\gamma^{(t)}$ measures the angle between the gradients $g^{(t-1)}$ and $g^{(t)}$. Small angles indicate high correlation and therefore steady movement in weight space. Therefore, we increase the learning rate linearly up to the current maximum (as determined by initial learning rate, annealing factor and number of updates performed) whenever $\gamma^{(t)} < 90°$ for several updates. Large angles, on the other hand, indicate random jumps in weight space. We therefore decrease the learning rate exponentially whenever $\gamma^{(t)} > 90°$ for several consecutive updates. In summary, we obtain the following update rule for local learning rate η_i of network i:

$$\eta_i^{(t+1)} = \min\left\{\eta_G^{(t+1)}, \left\{\begin{array}{c}\eta_i^{(t)} + \delta \\ \eta_i^{(t)} * \kappa \\ \eta_i^{(t)}\end{array}\right\}\right\} \qquad \text{if } \left\{\begin{array}{c}\frac{1}{M}\left(\sum_{k=0}^{M} \gamma^{(t-k)}\right) < 90° - \epsilon \\ \frac{1}{M}\left(\sum_{k=0}^{M} \gamma^{(t-k)}\right) > 90° + \epsilon \\ \text{else}\end{array}\right\}$$

with linear increase $\delta = 0.001 \ldots 0.01$ and exponential annealing factor $\kappa = 0.5 \ldots 0.9$. The number of batch updates M controls smoothing of γ whereas ϵ controls the influence of the global learning rate. For $\epsilon \to 90°$, local learning rates are forced to follow the global learning rate, whereas low values of ϵ allow local learning rates to develop individually. Typical values that have been used in our experiments are $M = 10$ and $\epsilon = 20°$. We finally note that the above learning rate adaptation scheme is very similar in spirit to the delta-bar-delta learning rule proposed in [Jacobs '88]. However, in contrast to our scheme the delta-bar-delta rule is based on the sign of the product instead of on the angle between successive gradient vectors[11].

5.5.2 Independent Training and Sampling

In addition to the joint training technique presented above, we have developed an alternative training scheme that, although requiring considerable amounts of temporary disk space, allows for faster and more convenient training of the local neural

[11] Furthermore, the delta-bar-delta rule assumes a separate learning rate for each network weight.

networks in an HNN. Furthermore, this approach allows us to apply more sophisticated optimization algorithms of higher order since the individual networks are trained sequentially on each processor.
Independent training proceeds in three stages (see Fig. 5.26). In an initial step, we process the complete training database, determine the relevant nodes in the hierarchy for each training pattern and store for each one of these nodes the pattern vector together with the corresponding target vector in a node-specific training data set for subsequent network training. Instead of storing all the pattern vectors relevant to each node, we sample only a subset of these patterns for the nodes in the upper levels of the tree. This saves substantial amounts of storage space and is considered uncritical (in terms of the accuracy of the resulting estimators) since there is a lot of redundancy in the corresponding training sets. In our experiments on the Switchboard corpus, consisting of roughly 60 million training patterns, we required about 3-5 GBytes of disk space for storing the partially sub-sampled training sets for all the tree nodes of a typical HNN.

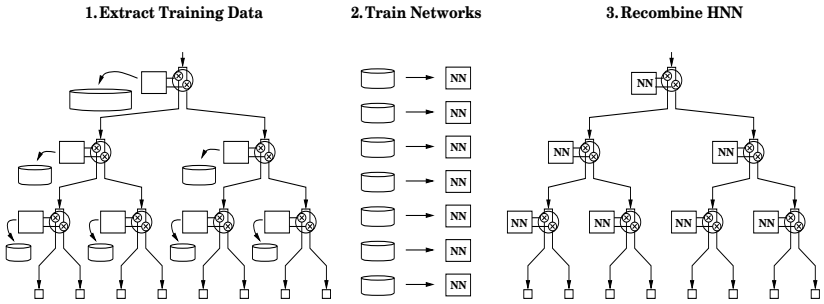

Figure 5.26: Data sampling and independent training of HNNs

In a second step, we sequentially train neural networks for each one of the extracted node-dependent training sets. All kinds of sophisticated, memory-intensive optimization algorithms can be applied in this stage as we do not have to keep the complete hierarchy in memory as in the case of joint training. Furthermore, independent training can easily be distributed among several processors without requiring any code changes, achieving linear speed-up and optimal load balancing without any effort. We simply partition the set of networks to be trained appropriately and let each processor train a separate subset of networks.
In the final step of sequential training, we have to plug in the trained networks into the corresponding nodes of the HNN tree in order to obtain the completely

trained hierarchical connectionist model. One of the big advantages of independent training is that we can efficiently experiment with different network sizes or different optimization algorithms once the training sets have been extracted and stored on disk. A potential disadvantage is the large amount of required temporary disk space. Using the independent training technique, we were able to train medium sized HNNs consisting of around 1000 nodes on the full Switchboard corpus in less than 24 hours using 8 Pentium-II/400Mhz CPU's. This is comparable to the training time required for a conventional non-connectionist model using the same number of CPU's. In summary, our hierarchical connectionist acoustic model does not suffer from the excessively long training times typically reported for monolithic connectionist architectures.

5.6 Integration into HMM Framework

We now turn our attention to aspects concerning the integration of the presented hierarchical connectionist model into the standard HMM framework found in nearly all of today's large vocabulary continuous speech recognition systems.

5.6.1 Model Integration

We consider the case of integrating our hierarchical connectionist acoustic model into a decision-tree clustered context-dependent HMM speech recognizer. Fig. 5.27 gives an overview of the relevant parts of the resulting hybrid NN/HMM speech recognition system. A sequence of raw sub-phonetic HMM states, e.g., a triphone HMM, is translated into a sequence of more robust tied HMM states by means of the appropriate phonetic decision trees. This part of the recognizer is identical for both conventional as well as connectionist acoustic models. In conventional models, we assign a separate Gaussian mixture model to each leaf node in all phonetic decision trees for estimating emission probabilities for the corresponding tied HMM state. In contrast, a hierarchical connectionist acoustic model estimates these HMM emission probabilities within a single tree structure where there is a one-to-one mapping between the leaf nodes of the HNN tree and the set of leaf nodes of all decision trees. We distinguish the following three phases of building a context-dependent large vocabulary connectionist speech recognizer from initial state alignments of the training data[12]:

- **Tree Building:** Once the phonetic decision trees have been grown for a particular application domain, we construct a Hierarchy of Neural Networks model

[12] These alignments might either be obtained by uniform segmentation or by running the Viterbi algorithm on reference word transcriptions using some other, previously trained acoustic model

for the set of leaf nodes of all decision trees. If the HNN is to be constructed using one of the cluster algorithms presented in section 5.3, we first have to estimate the required Gaussian models for each leaf node in each decision tree according to the given initial state alignments.

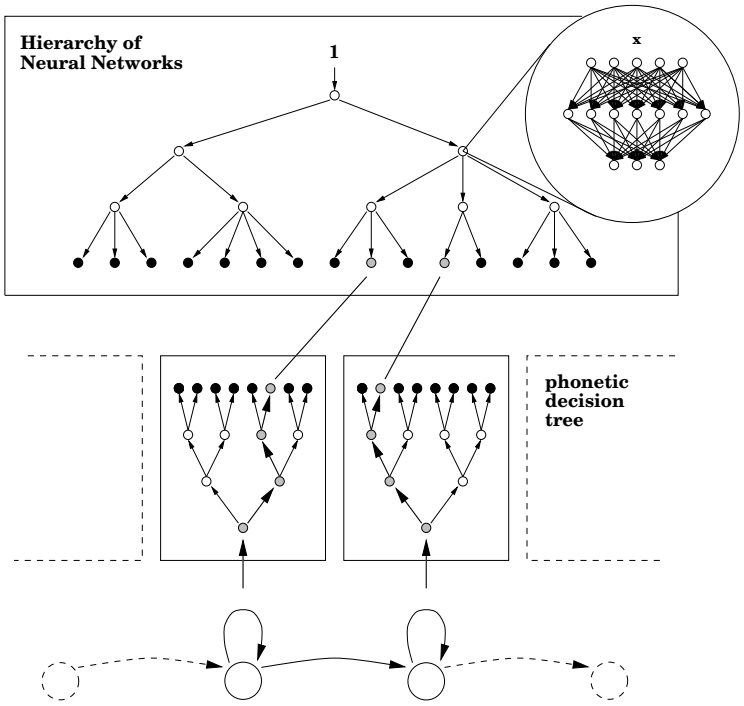

Figure 5.27: Integration of a hierarchical connectionist acoustic model into a decision tree clustered HMM recognizer

- **Training:** In Viterbi/label training, there is a one-to-one mapping between target HMM state and feature vector for each time step. Using the appropriate phonetic decision tree, we can determine the HNN leaf node that corresponds to the target HMM state and assign the current feature vector to the nodes on the path from that leaf node to the HNN root node for subsequent training of

the corresponding neural network estimators. In principle, we could also apply forward-backward HMM training but the computational complexity of training the hierarchical connectionist acoustic model then increases several-fold with hardly any gains in recognition performance.

- **Decoding:** Typically, a frame synchronous Viterbi beam search accesses the acoustic model by requesting emission probabilities for a list of currently active tied HMM states for each time frame. This list translates into a list of HNN leaf nodes, for which we compute posterior probabilities by top-down evaluation of the relevant nodes in the tree.

5.6.2 Incorporating Priors

So far, we have been concerned with the estimation of HMM state posteriors $p(s_i|\mathbf{x})$ through a tree-structured connectionist model. In the HMM framework however, we are required to provide estimates of the HMM state emission probabilities $p(\mathbf{x}|s_i)$ also referred to as state likelihoods. Application of Bayes' rule yields an expression for converting state posteriors into state likelihoods:

$$p(\mathbf{x}|s_i) = \frac{p(s_i|\mathbf{x})}{p(s_i)} p(\mathbf{x})$$

The last term $p(\mathbf{x})$ can be omitted in frame-synchronous decoding as already mentioned in section 3.2. According to this, a scaled likelihood can be computed by simply dividing the estimates of the state posteriors by the state priors:

$$\hat{p}(\mathbf{x}|s_i) = \frac{p(s_i|\mathbf{x})}{p(s_i)}$$

In our hierarchical connectionist acoustic model, we have decomposed the state posteriors into products of local conditional node posteriors. The posterior probability of a specific HMM state is computed by multiplying all the estimates of local conditional posteriors on the path from root node to the leaf node representing the HMM state (see Fig. 5.28).
Using the naming conventions from Fig. 5.28, a specific HMM state posterior is computed according to

$$p(s_i|\mathbf{x}) = \prod_{k=0}^{D(s_i)-1} p(N_i(k+1)|N_i(k), \mathbf{x})$$

given a tree structure and local estimators for the conditional posterior probabilities ($D(s_i)$ denotes the depth of the leaf node that represents the state s_i). Now, the

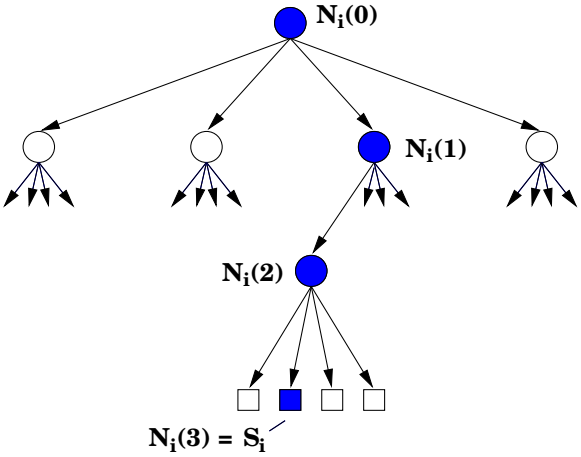

Figure 5.28: Top-down computation of state posteriors in a hierarchical connectionist acoustic model

same tree structure can be used to decompose the HMM state prior probabilities into a product of conditional prior probabilities:

$$p(s_i) = \prod_{k=0}^{D(s_i)-1} p(N_i(k+1)|N_i(k))$$

Thus, in addition to having a separate neural network based estimator for the conditional posterior probabilities at each tree node, we need to estimate conditional prior probabilities of child nodes at each tree node. This can simply be done by counting the occurances of child nodes for each tree node and normalizing these counts to relative frequencies. Interestingly, decomposing the priors as well as the posteriors allows to rewrite the expression for the scaled likelihood of HMM states such that we can compute it from local scaled likelihoods at each node in the HNN tree:

$$\hat{p}(\mathbf{x}|s_i) = \prod_{k=0}^{D(s_i)-1} \frac{p(N_i(k+1)|N_i(k),\mathbf{x})}{p(N_i(k+1)|N_i(k))}$$

Thus, as a byproduct of computing scaled likelihoods for an HMM state (represented by a particular leaf node), we also obtain scaled likelihoods for the tree nodes along the path down to that HMM state.

$$\hat{p}(\mathbf{x}|N_i(j)) = \prod_{k=0}^{j-1} \frac{p(N_i(k+1)|N_i(k),\mathbf{x})}{p(N_i(k+1)|N_i(k))} \quad \text{with} \quad 0 \leq j \leq D(s_i)$$

Both the partial posterior and partial prior computed down to a specific tree node represent valid probabilities. In fact, they model the posterior and prior probabilities of the acoustic unit emerging from the union of HMM states reachable from that node. We gradually refine the estimates of the state posterior and prior probabilities on our way from the root node down to a leaf node by increasing the acoustic resolution from broad phonetic classes down to single HMM states. This multi-scale representation and computation of the probabilities of acoustic units is one of the main properties and advantages of the hierarchical approach presented in this thesis and will be exploited for various purposes in later chapters.
Another interesting point to note is that the hierarchical decomposition of prior probabilities according to the tree structure of an HNN and their explicit utilization in the modular computation of scaled likelihoods offers possibilities for soft structural modifications absent in any conventional connectionist acoustic model. By altering the local conditional priors at a specific tree node, we can re-weight the contribution of the subtrees emerging from its child nodes. We can even softly pinch off certain tree branches completely without having to explicitly remove these branches. The explicit availability of state priors in connectionist acoustic models is considered to be advantageous since it allows to adapt the model to differences in the prior distributions between training and test set. In monolithic connectionist models however, the modification of priors does not influence the structure or behavior of the model itself. In contrast, the tree structure of our hierarchical connectionist acoustic model represents just a hull of possible structures that are shaped by the actual local conditional priors. This property of our model opens the door to structural model adaptation and modification of the specificity of context modeling. We will detail this aspect of the model and its application to domain-adaptive speech recognition in chapter 8. In summary, it is most important to note that priors are an essential and powerful part of our hierarchical model that allow to dynamically control its structure, rather than just an add-on correction mechanism required by the HMM formalism.

5.6.3 Embedded Training

It is possible to use so-called embedded Viterbi training to iteratively optimize both the alignment of the training data, i.e. the segmentation into words, phones and HMM states, and the parameters of a connectionist acoustic model [Franzini et al. '90]. With respect to this technique, our hierarchical connectionist model does not differ

from conventional, monolithic connectionist models and the same algorithm can be applied:

1. Compute initial state labels for entire training set
2. Train HNN model on current state labels
3. Relabel training set using the current HNN model by performing Viterbi alignments
4. Unless some stopping criterion is fulfilled, go to step 2.

Of course, we need to provide reasonable initial state labels and we must have a consistent criterion for stopping the above iterative process. Provided that we start from accurate initial labels, e.g., Viterbi alignments with some other trained acoustic model, recognition accuracy typically saturates quickly, requiring only 2-3 iterations of embedded training.

5.7 Evaluation on Switchboard

The performance of hierarchical connectionist acoustic models in large vocabulary speech recognition systems has been evaluated in experiments on the Switchboard telephone speech corpus. We detail the architectures that we have constructed and trained and compare their performance on this difficult but standard benchmark task.

5.7.1 General Setup

All of the experiments with hierarchical connectionist acoustic models were performed in more or less the same general speech recognition setup which is described in the following. We were mostly using the Janus Recognition Toolkit (JanusRTk) [Finke et al. '97, Zeppenfeld et al. '97], a state-of-the-art statistical speech recognition toolkit very well suited for research and development due to its object-oriented modular structure and its tight coupling with the Tcl/Tk scripting language. For some of the more recent experiments, we were using a new, completely rewritten large vocabulary speech recognition toolkit [Finke et al. '99] which is particularly well suited for modeling conversational speech as, for instance, found in the Switchboard corpus. Irrespective of the recognition toolkit used, the basic recognizer components (feature preprocessing, phones set, phonetic dictionary, language model, etc.) were identical such that recognition results are directly comparable. Following is an itemized description of the main components:

- **Preprocessing:** First, we applied a Hamming-windowed short-time spectral analysis using a 256-point Fast Fourier Transformation (FFT) to the raw 8 kHz audio data. The analysis window has a length of 160 samples and is shifted by 80 samples, resulting in a preprocessing rate of 100 frames per second. The power spectrum is then frequency warped using a piecewise linear transformation to compensate different vocal tract lengths. After transforming the power spectra into a 30-dimensional log Mel-scale, 13-dimensional cepstra are computed by applying a discrete cosine transformation. The resulting Mel-Frequency Cepstral Coefficients (MFCC) are normalized by cepstral mean subtraction for each conversation side. After adding the average log power of the analysis window, 42-dimensional vectors consisting of the 14-dimensional static features and their first and second order time derivatives were computed. The final 32-dimensional feature vectors were obtained by applying a truncated LDA transformation. We also experimented with 39 dimensional feature vectors consisting of 13 MFCCs and their first and second order time derivatives without applying the LDA and obtained similar results.

  Conventional context-independent connectionist acoustic models are often built on much higher dimensional pattern vectors consisting of a window of multiple MFCC vectors extending several frames into the past and the future in order to capture a higher amount of acoustic context and thereby improve modeling accuracy. In contrast to such implicit incorporation of acoustic context, our hierarchical model allows for more effective explicit modeling of phonetic context which is why standard preprocessing appears to be sufficient. Furthermore, we are interested in compact features as the number of parameters in our distributed and modular hierarchical architecture depends linearly on the feature dimensionality.

- **Phonetic Modeling:** The set of monophones consists of 1 silence model, 1 garbage mumble phone for modeling unknown words, 6 noise models (breathing, human noise, non-human noise, lip smack, throat cleaning and laughter) and 48 speech phones (containing 4 special phones for modeling interjections). In phonetic context modeling, phonetic decision trees were grown only for the 48 speech phones and for 2 of the noises (laughter and mumble). Silence and all other noises were modeled context-independently. For clustering pentaphone context decision trees, a set of around 100 phonetic questions was asked in a window of +/- 2 phones within words and +/- 1 phone across words. Phones at word boundaries and three different stress levels are marked with special tags which can be queried in addition to the standard phonetic features.

- **Pronunciation Modeling:** Accurately modeling pronunciation variability in conversational speech is an important component for automatic speech recogni-

tion. Pronunciation modeling in our Switchboard system is based on the work reported in [Finke '96, Finke & Waibel '97b]. The pronunciation dictionary contains 15000 unique words with an average of 2 pronunciation variants per word, yielding a total of 30000 entries. 59% of the words are represented by just a single pronunciation variant. Some of the remaining words are represented by up to 50 different pronunciation variants. The pronunciation variants are generated from the baseform pronunciations by a decision tree based approach and associated pronunciation weights are learned from the training corpus. In addition, the phonetic dictionary was enhanced by 1756 pronunciation variants of the 262 most frequently occurring word-tuples (e.g., GOING-TO) and -triples (e.g., A-LOT-OF), so-called multi-words. This allows to capture cross-word pronunciation effects much better (e.g., GOING-TO $\rightarrow$ GONNA). In a procedure called Flexible Transcription Alignment (FTA) [Finke & Waibel '97a], the acoustic training data is aligned against an artificially enriched training transcription represented as a directed acyclic graph. The graph models a variety of conversational effects by allowing for multiple pronunciations, optional multi-words, optional filler words, optional begin and end words etc. Using the Viterbi algorithm, the best matching sequence of words is extracted and aligned for subsequent training of the acoustic model. This way, FTA improves the quality of transcriptions which was shown to yield significant gains in recognition accuracy.

- **Language Modeling:** For language modeling, we use a three-way non-linear interpolation of Kneser-Ney [Kneser & Ney '95] back-off trigram models. The three models were trained on the Switchboard (3M words), Callhome (200k words) and Broadcast News (130M words) corpora, respectively. Where noted, we have used the Switchboard language model by itself in order to simplify and speed up decoding.

- **Decoding:** A state-of-the-art time-synchronous Viterbi beam search decoder using a phonetic prefix-tree organized lexicon was used for generating word lattices and first best hypotheses with context-dependent acoustic models. A standardized interface between the decoder and potential acoustic models allows for easily switching between conventional mixture based acoustic models and the hierarchical connectionist acoustic models presented in this thesis.

For further details on specific aspects of the speech recognition system used for our experiments on the Switchboard domain, the reader is referred to [Finke et al. '97, Zeppenfeld et al. '97]. All recognition results were obtained on a subset of the official 1996 Switchboard evaluation test set consisting of the first 30 seconds of speech from

all 40 speakers in that set, which amounts to 4550 words contained in a total of 20 minutes of speech.

5.7.2 Manually Constructed vs. Clustered HNNs

First, we compare manually constructed against automatically clustered Hierarchies of Neural Networks (HNN). An HNN for 10000 decision tree clustered tied HMM states was manually constructed using knowledge about phonetics and HMM topologies involved. Fig. 5.29 depicts the structure of this model and the strategy applied in decomposing the task into a hierarchical model.

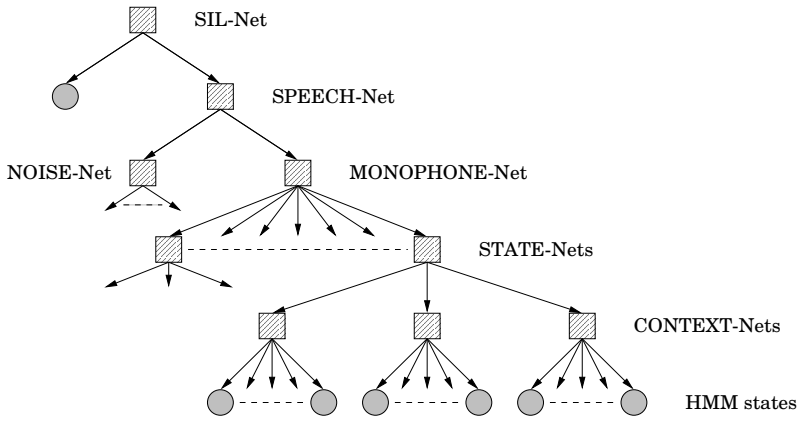

Figure 5.29: Manually constructed HNN for 10000 HMM states

The root consists of a binary node for discriminating between the silence phone and all other phones. In the second layer, another binary node is used to discriminate between noise phones and speech phones. In the third layer, we use a 7-ary node for discriminating the noise phones and a 48-ary node for discriminating the speech monophones. In the fourth layer, we use 3-ary nodes for discriminating between the begin-, middle- and end- states of the atomic 3-state left-right HMM topologies used for modeling speech and noise phones. In the final fifth layer, we discriminate between the individual contextual variations of each sub-phonetic unit as provided by the context-clustering decision trees. The complete tree contains a total of 209 internal nodes equipped with single hidden layer MLPs for estimating the local conditional posteriors. Note that the branching factor of the individual nodes varies considerably

in the tree (Fig. 5.30). While the roughly 50 state nets perform 3-way classification tasks, some of the context nets perform classification tasks involving more than 200 classes.

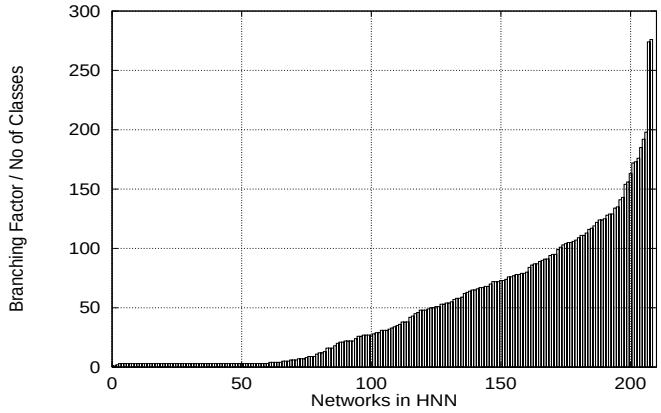

Figure 5.30: Branching factors of individual nodes in manually constructed HNN

To make the distribution of branching factors more uniform, the monophone node could be decomposed further according to phonetic classes such as plosives, fricatives, vowels, etc. However, we would also have to decompose the context nodes in the final layer of the HNN tree. Although this could be accomplished by cloning the structure of the corresponding context decision trees, such proceeding would lead to a highly imbalanced HNN tree which is undesirable for the reasons stated in section 5.2.
The manually constructed HNN was compared against a bottom-up clustered hierarchy for 6000 tied HMM states. We chose a system with 6k HMM states instead of the above 10k HMM states in order to reduce the number of parameters in the clustered HNN to a value comparable to the manually constructed HNN. As clustered HNNs benefit from more tree nodes, a comparison between 10k systems would not be fair. In bottom-up clustering, we applied an additional penalty term to enforce tree balance and avoid non-uniform priors. The resulting binary HNN was compactified to a 10-ary HNN using the node merging algorithm presented in section 5.3.2. After experimenting with various tree branching factors, we found that values in the range 4-10 yield tree structures that represent a good compromise between resolution and compactness. Binary trees are disadvantageous because they are expensive to evaluate and ineffective in their use of parameters as they are comparatively deep and

contain the largest amount of internal tree nodes. HNN trees with branching factors considerably larger than 10, on the other hand, are difficult to train as the complexity of the local learning tasks increases with increasing branching factor.

depth	# nodes = # networks	# hidden units/network
0	1	256
1	6	128
2	43	128
3	145	64
4	326	64
5	298	32
6	143	32
total	962	

Table 5.2: Overview of bottom-up clustered 10-ary HNN for 6k HMM states

The final tree for modeling 6k HMM states has height 7 and consists of 962 internal tree nodes. A set of 962 single hidden layer MLPs was assigned to the tree nodes and model complexity was controlled by increasing the number of hidden units from 32 at the bottom of the tree to 256 at the top of the tree. The overall number of parameters of this model amounts to 2.1 million, which compares to about 2 million parameters contained in the manually constructed HNN. Table 5.2 gives details for the clustered tree and Table 5.3 gives recognition results obtained with the manually constructed and the clustered HNN trees. Even though modeling considerably less HMM states, the clustered HNN achieves a significantly better recognition rate than the manually constructed HNN.

acoustic model	# states	# params	word error rate
manually constructed HNN	10000	2.0 M	37.3 %
bottom-up clustered HNN	6000	2.1 M	35.8 %

Table 5.3: Performance of manually constructed vs. clustered HNN

We attribute the difference in performance to the differing tree topologies. In contrast to the manually constructed HNN tree, the bottom-up clustered HNN tree exhibits small and comparatively uniform average branching factors that allow to robustly train estimators of conditional posterior probabilities. Some of the local classification tasks in the manually constructed tree may not be performed accurately due to an excessively large number of classes involved.

5.7.3 Scalability

In order to demonstrate the scalability of the hierarchical connectionist modeling framework with respect to the amount of phonetic context modeling we give results obtained with three different clustered HNN models: (1) a tree similar to the one in Fig. 5.9 for 3-state context-independent modeling of the 56 monophones, (2) the tree for 6k decision tree clustered context-dependent HMM states from the previous section and (3) a tree for 24k decision tree clustered context-dependent HMM states. To our knowledge, it has never before been attempted to construct a connectionist acoustic model for such a high degree of context modeling and it would not make much sense to apply a monolithic architecture to this task. However, the hierarchical decomposition of posteriors used in our hierarchical connectionist model allows to apply this model even to as many as 24000 HMM states and benefit from the increased acoustic and phonetic resolution.
During the construction phase for the 24k tree, we carefully experimented with different values for the non-uniform prior penalty in order to obtain an even more compact tree structure than in the case of 6k HMM states. The resulting tree structure has height 5 and contains a total of 4046 internal tree nodes with a maximum branching factor of 10. Again, we were assigning a set of 4046 single hidden layer MLPs as estimators for the local conditional posteriors to the tree nodes. Also, the number of hidden units was increased from the bottom to the top of the tree, this time however using values ranging from 16 to 128. The resulting hierarchical connectionist acoustic model contains a total of 3.1 million parameters distributed over the 4046 neural networks (see Table 5.4).

depth	**# nodes = # networks**	**# hidden units/network**
0	1	128
1	10	128
2	77	64
3	524	32
4	3434	16
total	4046	

Table 5.4: Overview of bottom-up clustered 10-ary HNN for 24k HMM states

Table 5.5 gives recognition error rates obtained for all three HNN models. Obviously, context-dependent modeling improves performance enormously compared to context-independent modeling. Furthermore, modeling 24000 instead of only 6000 context-dependent HMM states reduces the word error rate by 2.5% absolute.

acoustic model	# states	# params	word error rate
context-independent HNN	154	0.8 M	56.4 %
context-dependent HNN	6000	2.1 M	35.8 %
context-dependent HNN	24000	3.1 M	33.3 %

Table 5.5: Scalability of hierarchical connectionist acoustic modeling framework

These results show that the hierarchical connectionist modeling framework scales well to excessive amounts of context modeling and that this property allows to significantly improve performance on Switchboard compared to context independent modeling.

5.7.4 Joint Training

The HNN trees for 6k, 10k and 24k HMM states were trained on Viterbi state alignments from a mixture of Gaussians system using stochastic on-line gradient descent and the joint training technique presented in section 5.5.1. A randomly selected set of 100 utterances was excluded from the training set and used as a validation set for determining early stopping. The three plots in Fig. 5.31 show the evolution of various performance measures on the validation set during training of the largest model built, namely the HNN tree for 24k HMM states. Each vertical line corresponds to a full pass through the available training data consisting of 2500 conversation sides with a total of 87000 utterance segments.
From top to bottom, the plots show (1) the normalized log likelihood of the validation data according to the given state alignments and the model trained so far, (2) the average of the local normalized log likelihood over all the 4046 classifier neural networks in the tree and (3) the average of the normalized mis-classification errors, again averaged over all the 4046 tree nodes. All three curves level off after about 3 passes through the training data, demonstrating that even such a large model can be trained to convergence in very few training iterations. Furthermore, the fact that the normalized log likelihood levels off on the validation set instead of starting to decrease again at some point, indicates that the hierarchical model is very robust to overfitting effects on the Switchboard domain. In contrast to the training of monolithic connectionist models on smaller tasks, explicit regularization is not required in our case. Obviously, the large amount of training data[13] allows for excellent generalization and early stopping is not necessary. We attribute this behavior to the following aspects:

- Training data can be considered very noisy, since a large variety of different speakers and recording conditions have been considered during collection of

[13] The full Switchboard training corpus consists of roughly 60 million patterns

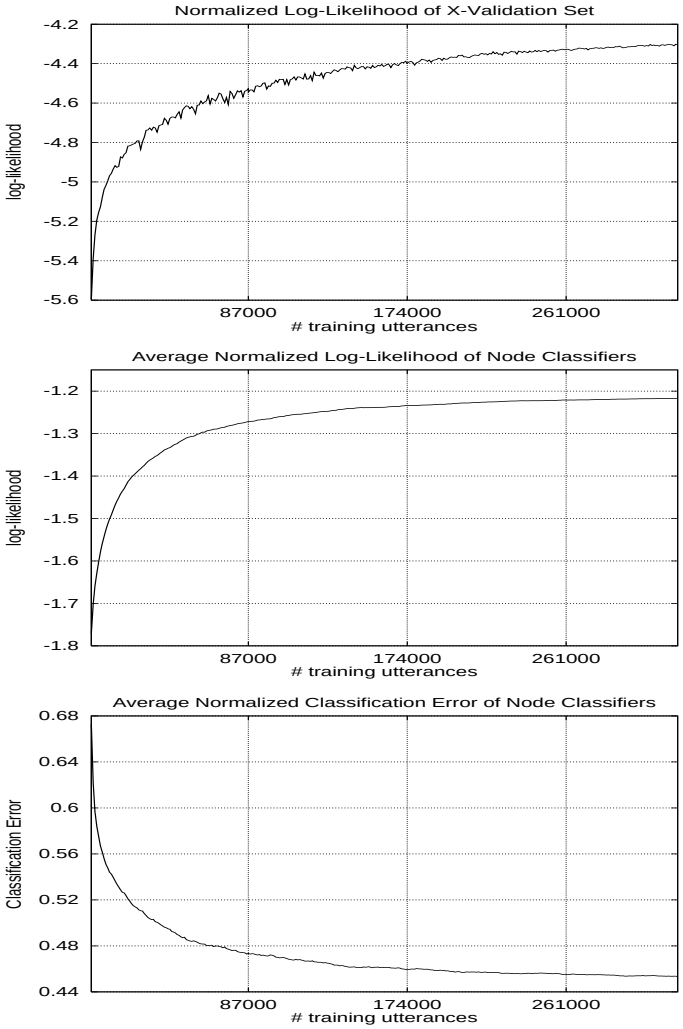

Figure 5.31: Monitoring performance on validation set during joint training of HNN architecture for 24k HMM states

the corpus. Training with noisy data is similar to regularization and therefore improves generalization [Bishop '95b].

- Some of the 3434 networks at the nodes in the lowest level of the 24k HNN tree do not receive enough training samples to generalize well to unseen new data. Although all of these networks together constitute 85% of the total number of networks in the tree, they contribute just as one out of 5 (height of the tree) networks to any particular posterior probability. The networks in the upper part of the hierarchy have the largest influence on the posteriors computed by the tree. For those networks, the very large amount of available training data guarantees that the validation set error approaches the training set error which is an indicator for good generalization performance.

5.7.5 Comparison to Conventional Models

We compare the performance of our largest hierarchical connectionist model to the performance of a state-of-the-art system with a conventional, mixtures of Gaussians based acoustic model. For this purpose, we had available the best performing system [Finke et al. '97] on the Switchboard part of the official 1997 DARPA Hub-5E evaluation. As the author was participating in the group of researchers that developed that system, a direct comparison of the two modeling paradigms within the same general system setup is possible. We report recognition error rates for two systems that differ only in the model for estimating HMM emission probabilities (Table 5.6).

acoustic model	**# params**	**word error rate**	**decoding time**
Hierarchy of Neural Networks (HNN)	3.1 M	34.4%	90 xRT
Mixtures of Gaussians (CMU-ISL/Hub-5E 97)	6.6 M	31.5%	300 xRT

Table 5.6: Comparison between hierarchical connectionist and conventional acoustic models on 1997 development test set

In contrast to earlier experiments, these results were obtained with a single (Switchboard) trigram language model. The underlying context clustering decision trees were constructed for the mixtures of Gaussians model and define a set of 24000 tied pentaphone HMM states. They were adopted without modifications for hierarchical connectionist modeling. For decoding, we have used large beams to minimize the number of pruning errors caused by the heuristic search strategy as the focus of this experiment was on comparing the acoustic models. Tightening the search beam yields faster decoding times for both models but also increases the word error rate. Also, it

should be noted that the HNN model has a slight disadvantage because it was trained only on state alignments generated with the mixtures of Gaussians model.
Considering that the mixtures of Gaussians model (a) contains more than twice the number of parameters, (b) went through several iterations of embedded Viterbi training and (c) was heavily optimized on the above test set during the development of the evaluation system, the hierarchical connectionist modeling framework yields performance competitive to the best current state-of-the-art systems[14] while decoding is more than 3 times faster for decoding beams that minimize the number of pruning errors.

5.7.6 Local Model Selection

In another experiment, we were comparing a-priori determined model size against automatic local model selection. For that purpose, we constructed a 4-ary HNN tree for 8000 tied HMM states using the top-down divisive clustering algorithm. The resulting model tree has height 9 and was equipped with single hidden layer MLPs as shown in column 3 of Table 5.7 for the baseline model. In automatic local model selection, we used the same tree structure but trained a set of MLPs with 4, 8, 16, 32, 64 and sometimes even 128 hidden units for each tree node and selected the one which gave minimum error on an independent validation set.

depth	# nodes = # networks	# hidden units per network	
		baseline	model selection
0	1	64	max 128
1	4	64	max 128
2	12	64	max 128
3	23	64	max 128
4	76	32	max 128
5	256	32	max 64
6	984	32	max 64
7	2188	16	max 64
8	330	16	max 64
total	3866		

Table 5.7: Overview of top-down clustered 4-ary HNNs for 8k HMM states

In order to be able to easily train and test several different MLPs for each tree node, we used the independent instead of the joint training technique for HNN training. For

[14] The official evaluation results on the 1997 Switchboard evaluation test set ranged from 35.1% (achieved by the CMU-ISL/Hub-5E 97 system used in the above comparison) to 42.9% [Martin et al. '97]. Note that the results in Table 5.6 were obtained on a different test set.

this purpose, we extracted and stored a predetermined number of feature vectors (up to 150000) for each node in the HNN tree in one pass through the available training data. Once the training data was extracted, neural networks for each tree node were trained sequentially on the corresponding training set. Fig. 5.32 shows the mean and standard deviation of the optimal number of hidden units in each tree level as found by local model selection.

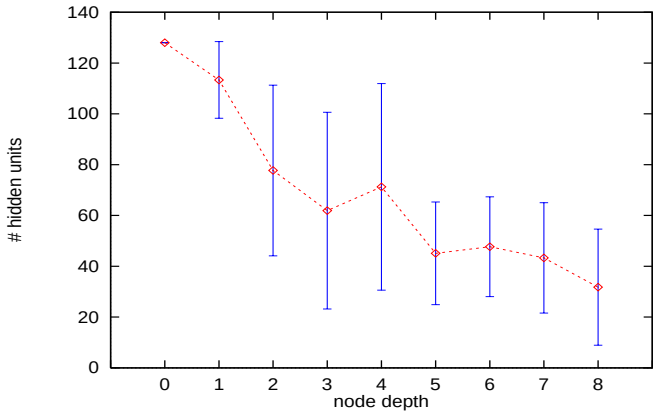

Figure 5.32: Automatic local model selection (see text)

As expected, the average number of hidden units decreases with increasing tree depth. However, there are some nodes even in the upper levels of the tree where networks with only 4 hidden units yield best performance. On the other hand, some nodes at the bottom of the tree are equipped with networks with 64 hidden units by local model selection.

acoustic model	# states	# params	word error rate
baseline HNN	8000	2.7 M	38.6 %
local model selection	8000	3.6 M	37.8 %

Table 5.8: Effect of local model selection on recognition performance

Finally, Table 5.8 gives recognition results for the baseline HNN and the HNN resulting from local model selection. Again, these results were obtained on our standard

test set with a single (Switchboard) trigram language model. Local model selection increases the number of parameters from 2.7 to 3.6 million and decreases the word error rate by 0.8% absolute.

5.7.7 Embedded Viterbi Training

In all the experiments reported so far, the hierarchical connectionist models were trained on state alignments that were generated with a conventional mixture of Gaussians model. In this experiment, we investigated whether we can improve performance by re-training on state alignments computed with the hierarchical connectionist model itself. This procedure is commonly called iterative embedded Viterbi training. We used the HNN model for 8k HMM states from the previous section as our baseline model. Table 5.9 gives word error rates for the baseline and one iteration of embedded Viterbi training.

acoustic model	**trained on state labels from**	**word error rate**
baseline HNN	mixture of Gaussians system	38.6 %
embedded training	baseline HNN	37.6 %

Table 5.9: Performance gain through embedded Viterbi training

Re-aligning the training data with the connectionist model followed by re-training improved performance by 1.0% absolute. This results shows that state labels optimized with one particular acoustic model are not necessarily optimal for training some other acoustic model.

Chapter 6

Fast Model Evaluation

This chapter presents a technique for efficiently evaluating the hierarchical connectionist acoustic model presented in the previous chapter. Based on exploiting the hierarchical structure by means of dynamic tree pruning, it allows to accelerate the evaluation of posterior state probabilities in hierarchies of neural networks considerably. While dynamic tree pruning represents a technique for trading-off recognition speed against accuracy, we present experimental results that indicate that the evaluation of a hierarchical connectionist acoustic model can be sped up by a factor of almost 10 with hardly any increase in word error rate. Furthermore, dynamic tree pruning can be realized by adding a single line of code as the potential for fast evaluation is inherent to the architecture. In contrast, conventional acoustic models require additional structures for determining relevant subsets of HMM states to be evaluated and do not provide comparably high speed-ups.

6.1 Real-Time Speech Recognition

Today, automatic speech recognition technology still is comparably demanding in terms of memory and processing speed requirements. For instance, state-of-the-art research systems for large vocabulary conversational speech recognition on the Switchboard domain were reported to require 200-300 MBytes of RAM and to run in 100-300 times real-time (xRT)[1] on standard hardware [Martin et al. '97]. More recently, there have been substantial efforts in speeding up research systems which led to a new Spoke condition in the Broadcast News evaluation for systems that run at about 10xRT and faster [DAR '98]. For commercial applications, a speech recognition system often has to cope with limited resources and has to fulfill real-time constraints in order to be useful.

[1] 100xRT means that it takes 100 seconds to decode an utterance of 1 sec duration.

Regardless of the specific type of acoustic model being employed, the approaches to speeding up a statistical speech recognition system always follow the same general pattern. Fig. 6.1 illustrates this process in terms of the distribution of computations into two broad classes: (1) evaluating the acoustic model and (2) decoding (where decoding contains evaluating the language model).

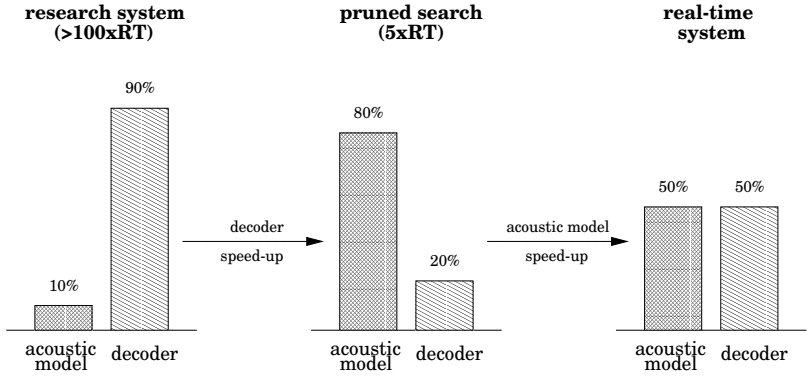

Figure 6.1: From research to real-time systems: qualitative analysis of proportion of time spent in acoustic model evaluation vs. actual decoding

As the qualitative analysis of Fig. 6.1 shows, the evaluation of the acoustic model constitutes only a very small proportion of the overall computations in a typical research system. Most of the time is spent in decoding word hypotheses as pruning beams are kept large to avoid search errors. The first step in speeding up a Viterbi beam search based decoder therefore always is to tighten the pruning beams which vastly reduces overall computations. Small increases in word error rate typically have to be tolerated in this step as search errors are introduced. The middle plot in Fig. 6.1 shows the implications of tight decoding beams on the distribution of computations. While the time spent in actual decoding has been reduced significantly, the proportion of time spent in evaluating acoustic model scores all of a sudden dominates the overall running time and can consume even more than 80% of the total amount of computation in a speech recognition system. Applying some kind of technique for fast, approximative acoustic model evaluation becomes crucial for achieving automatic speech recognition in real-time with an approximately uniform distribution of computations among acoustic model evaluation and actual decoding. Consequently, there has been a large body of work on techniques for speeding up the evaluation of conventional mixture densities based acoustic models (e.g., [Watanabe

et. al. '94, Fritsch et. al. '95, Fritsch & Rogina '96, Knill et. al. '96, Ravishankar '96]). Some of these approaches are based on applying a tree structure to quickly and dynamically determine a significantly reduced set of HMM states with potentially high emission probabilities. Only the likelihoods of states in this reduced set are fully evaluated, the likelihoods of all others are approximated. Speed-ups in the evaluation of the acoustic model on the order of a factor of 3-5 with virtually no or only a modest increase in word error rate have been reported using such techniques. However, the computation required to construct the additional structures for determining reduced sets of HMM states and the additional memory required to store these structures sometimes limits the usefulness of these techniques in practice.
Consider now the hierarchical connectionist acoustic model presented in the previous chapter. This model already is organized in a tree structure which can be exploited for fast, approximative evaluation without a need for additional structure as we will shortly see. In addition, improved local discrimination of HMM states in the hierarchical connectionist model allows to speed-up model evaluation more aggressively than in the case of conventional models.

6.2 Dynamic Tree Pruning

For any given time frame, the scores of HMM states with high posterior probability of emitting the current feature vector have to be evaluated with high accuracy as they are most likely to influence the result hypothesis of a Viterbi beam search. However, the majority of HMM states exhibit comparably small posterior probabilities of emitting the current feature vector. It is sufficient to efficiently compute approximations of the scores of these states which allows to save a large proportion of overall computations. The tree-structured top-down computation of posterior probabilities in a hierarchical connectionist acoustic model allows to implement this idea in form of *dynamic tree pruning* [Fritsch & Finke '98a]. A similar pruning technique has been proposed by [Waterhouse & Robinson '95, Waterhouse '97] for fast approximative evaluation of hierarchical mixtures of experts [Jordan & Jacobs '94].
The posterior probability of an HMM state in a Hierarchy of Neural Networks is computed as the product of the conditional node posteriors along the path from root node to the specific leaf node representing the HMM state (see Fig. 5.28):

$$p(s_i|\mathbf{x}) = \prod_{k=0}^{D(s_i)-1} p(N_i(k+1)|N_i(k), \mathbf{x})$$

where $D(s_i)$ is the depth of the leaf node, $\mathbf{x}$ is the current feature vector and the $N_i(k),\ \forall k = \{1, \ldots, D(s_i)\}$ denote the tree nodes along the path from root to s_i. As each of the conditional node posteriors in the above product fulfills the constraint

$$0 \leq p(N_i(k+1)|N_i(k), \mathbf{x}) \leq 1$$

the top-down computation of $p(s_i|\mathbf{x})$ yields monotonically decreasing *partial* posterior probabilities

$$\begin{aligned} p_j^\star(s_i|\mathbf{x}) &= \prod_{k=0}^{j-1} p(N_i(k+1)|N_i(k), \mathbf{x}) \qquad \forall j \in \{1, \ldots, D(s_i)\} \\ p_j^\star(s_i|\mathbf{x}) &\leq p_{j-1}^\star(s_i|\mathbf{x}) \end{aligned}$$

with $p(s_i|\mathbf{x}) = p_{D(s_i)}^\star(s_i|\mathbf{x})$. The monotonicity of partial posteriors implies that the posterior probability of HMM states in a subtree can never become larger than the partial posterior computed down to the root node of that subtree. This allows to efficiently identify subtrees that contain HMM states with posterior probability less than a given threshold Θ. As the score of these dynamically determined low-probability states does not have to be computed with full accuracy, we can stop evaluating conditional posteriors on our way down the tree according to the following rule

$$\text{if} \quad p_j^\star(s_i|\mathbf{x}) < \Theta, \quad \text{stop top-down evaluation}$$

Fig. 6.2 illustrates dynamic tree pruning for the case of a binary tree. For the specific feature vector in this example, only the shaded nodes have to be evaluated. All others lie on paths with partial probability smaller than Θ. The dashed boxes represent the subtrees that are not evaluated. The posterior probabilities of all HMM states in such a dashed box are tied and approximated by some function of the partial posterior at the associated root node.
Several strategies for assigning posteriors to HMM states in pruned subtrees have been investigated in this thesis. Although the speed-up in evaluating the acoustic model stand-alone is identical for all these strategies (depending only on the pruning threshold Θ), the effect on sentence decoding in a complete recognition system are very different. Consider the case that the partial posterior $p_j^\star(s_i|\mathbf{x}) < \Theta$ for some $j < D(s_i)$:

- **Partial Posterior Pruning (PPP):**
  In partial posterior pruning, we assign the partial posterior computed down to the node where pruning occurs to the HMM states in the corresponding subtree:

  $$p(s_i|\mathbf{x}) = p_j^\star(s_i|\mathbf{x})$$

  This partial posterior is an upper bound on the posterior probabilities of the HMM states in the subtree and therefore overestimates the true posteriors. For that reason, PPP might even slow down decoding and be counterproductive for very small Θ.

- **Uniform Posterior Pruning (UPP):**
  In uniform posterior pruning, we scale the partial posterior by a factor $\gamma = 1/N$, where N is the number of HMM states in the pruned subtree:

  $$p(s_i|\mathbf{x}) = \gamma \, p_j^\star(s_i|\mathbf{x})$$

  This rule distributes the partial posterior uniformly among all HMM states in the pruned subtree and thereby ensures that the hierarchical connectionist acoustic model computes a valid overall posterior probability distribution.

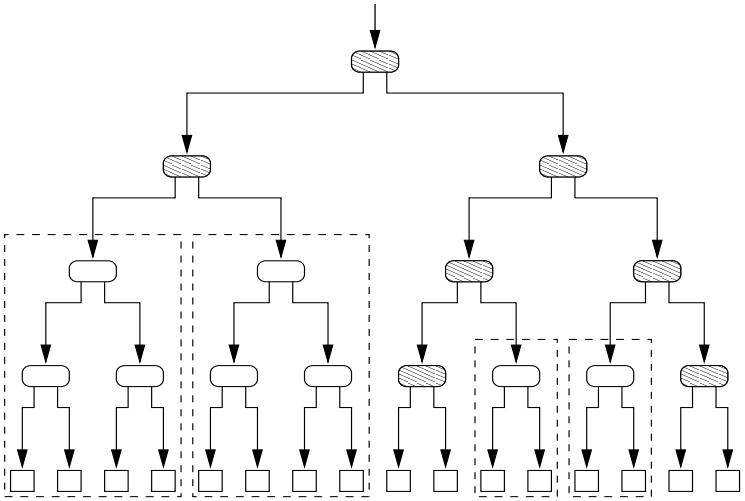

Figure 6.2: Dynamic tree pruning

- **State Deactivation Pruning (SDP):**
  In state deactivation pruning, we effectively deactivate the HMM states in the pruned subtree by setting

  $$p(s_i|\mathbf{x}) = 0$$

  This strategy indirectly speeds up decoding significantly as partial hypotheses that end with one of the pruned states get pruned immediately as their score falls out of the decoder's pruning beam. We have termed this technique state deactivation pruning as it is similar in spirit to phone deactivation pruning [Renals '96]. However, in contrast to phone deactivation pruning which

was proposed in the context of a monolithic connectionist model, SDP in a hierarchical acoustic model additionally yields significant savings in score computation as only parts of the hierarchy have to be evaluated. It should be noted though, that phone deactivation pruning may also yield savings in score computation if the monolithic, context-independent acoustic model is augmented with context-dependent modules as in [Kershaw et al. '95].

In the remainder, we present experiments and results of dynamic tree pruning in hierarchical connectionist acoustic models. We evaluate the above three strategies in terms of their effect on recognition speed and word error rate.

6.3 Experimental Evaluation

All of the experiments with dynamic tree pruning have been carried out on the Switchboard corpus, using a recognition setup identical to the one used for the experiments in chapter 5. We first analyze the effect of dynamic tree pruning on the hierarchical connectionist architecture in isolation and then take a look at the effects on decoding speed and word error rate in a complete large vocabulary conversational speech recognition system.

6.3.1 Pruning Hierarchies of Neural Networks

For the experiments reported here, we have selected two of the HNN acoustic models constructed in the previous chapter, one for 8000 tied HMM states, the other one for 24000 tied HMM states. We first take a look at the effect of dynamic tree pruning on the average percentage of tree nodes that have to be evaluated in the HNN. Fig. 6.3 shows the impact of the pruning threshold Θ on the amount of computations required in the acoustic model. The outcome of this experiment is independent of the pruning strategy as the acoustic model was evaluated stand-alone (without subsequent decoding).
The baseline percentage (no pruning) for these curves is 65% (not 100%) as this is a typical average number of HMM states for which the decoder requests emission probabilities for each frame. The percentage of nodes that have to be evaluated is roughly halved for pruning thresholds of $\Theta \approx 10^{-5.5}$ for the model with 8000 leaf nodes and $\Theta \approx 10^{-6.5}$ for the model with 24000 leaf nodes. A speed-up of about 10 in computation of acoustic scores can be achieved by setting $\Theta \approx 10^{-3}$ for the 8k model and $\Theta \approx 10^{-4}$ for the 24k model.
Next, we take a look at the percentage of HMM states for which the posterior probabilities are computed with full accuracy, i.e. for which the partial posteriors are computed completely down to the leaf nodes (see Fig. 6.4).

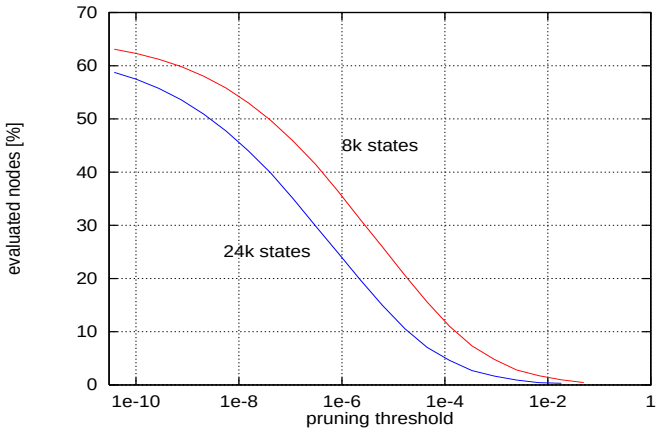

Figure 6.3: Effect of dynamic tree pruning on percentage of evaluated tree nodes

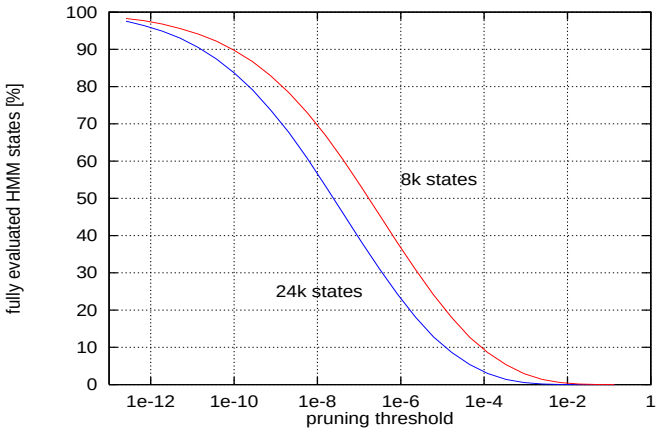

Figure 6.4: Effect of dynamic tree pruning on percentage of HMM states fully evaluated

This time we consider only those HMM states for which an estimate of the emission probability is requested by the decoder, thus the baseline for no pruning is 100%. Dynamic tree pruning starts to reduce the number of fully evaluated HMM states for $\Theta \geq 10^{-13}$. The curves in Fig. 6.4 are very similar to those in Fig. 6.3, however one can make an interesting observation. The effect of pruning with a given threshold Θ is stronger in case of the 24k model compared to the 8k model. As a consequence, speed-ups are larger for the 24k model than for the 8k model, given equal pruning thresholds. Of course, speed-up by itself means nothing if not set in relation to the effect of pruning on the accuracy with which posterior probabilities are estimated in an HNN. The following Fig. 6.5 depicts how the average (negative logarithmic) posterior probability of the correct model (along Viterbi alignments of the validation set) is influenced by the dynamic tree pruning threshold Θ.

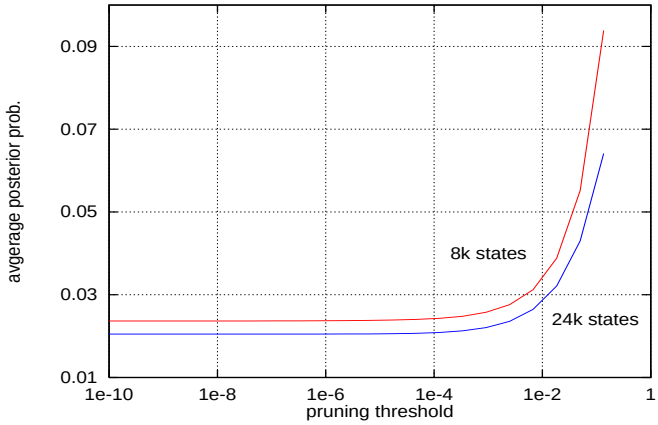

Figure 6.5: Effect of dynamic tree pruning on posteriors

Here, we have applied partial posterior pruning (PPP) to assign probabilities to pruned HMM states. PPP was chosen because pruning errors show up clearly in form of overestimation of posteriors with this pruning strategy. This allows to easily find useful operating points for the threshold Θ. Up to a pruning threshold of about $\Theta = 10^{-4}$, the posterior probabilities along the alignments of the correct hypotheses are hardly influenced by dynamic tree pruning. Above this however, pruning starts to influence the probability of the correct HMM states. As already discussed before, PPP overestimates the true posterior probabilities which is experimentally confirmed by the plots in Fig. 6.5. Still, for $\Theta = 10^{-4}$ we can achieve speed-ups of a factor of

10 and more in evaluating the state posteriors with almost no effect on the score of the correct hypotheses.

6.3.2 Pruning during Decoding

We now investigate the effect of dynamic tree pruning on a complete large vocabulary conversational speech recognition system. To this end, we use the HNN acoustic model for 24000 HMM tied states in the recognition system described in section 5.7. The baseline system with no pruning in the hierarchical connectionist acoustic model and wide decoding beams runs in 145 times real-time (xRT) on a 300 MHz Sun UltraSparc and achieves a word error rate of 34.4% on a subset of 12 speakers taken from the 1996 Switchboard evaluation test set.

As a first step, we tighten the decoding beams until performance starts to decrease due to search errors. Tighter decoding beams allow us to speed-up the recognition system to roughly 90xRT with a small increase in word error rate to 34.8%. For even tighter beams the word error rate increases considerably. In contrast to other less difficult domains, the conversational style of speaking and the poor quality of telephone channels in the Switchboard domain leads to diffuse acoustic models and a comparably high amount of confusion during decoding. That in turn limits the recognition speed obtainable by tightening the decoding beams such that real-time operation without significant losses in recognition accuracy appear impossible on today's standard hardware. However, we next show that applying dynamic tree pruning to the hierarchical connectionist acoustic model yields considerable savings in both the evaluation of the acoustic model and in decoding.

Consider first the impact of dynamic tree pruning on the decoding time. Fig. 6.6 shows a plot of the real-time factors obtained with dynamic tree pruning for all three pruning strategies introduced earlier. As mentioned earlier, the baseline speed for no pruning is a decoding time of roughly 90 times real-time. As expected, the required decoding time decreases with increasing pruning threshold Θ[2]. Furthermore, SDP yields the largest gains in recognition speed, followed by UPP. PPP on the other hand yields comparably small gains in recognition speed especially for high pruning thresholds. Note that the different gains in recognition speed obtained by these three pruning strategies reflect the differences in their ability to indirectly prune the search space and thereby speed-up decoding. The gains in evaluating the acoustic model itself are identical for all three methods.

Of course, the gains in recognition speed obtained by dynamic tree pruning must be contrasted with the impact on the recognition error rate in order to determine appropriate values for the pruning threshold Θ and to assess and compare the quality

[2] Again, note that higher pruning thresholds correspond to smaller values on the x-axis in these plots, due to the negative logarithmic transform applied

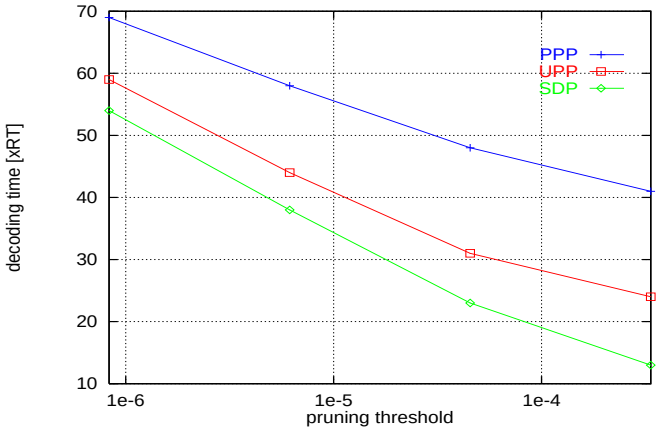

Figure 6.6: Decoding time vs. pruning threshold

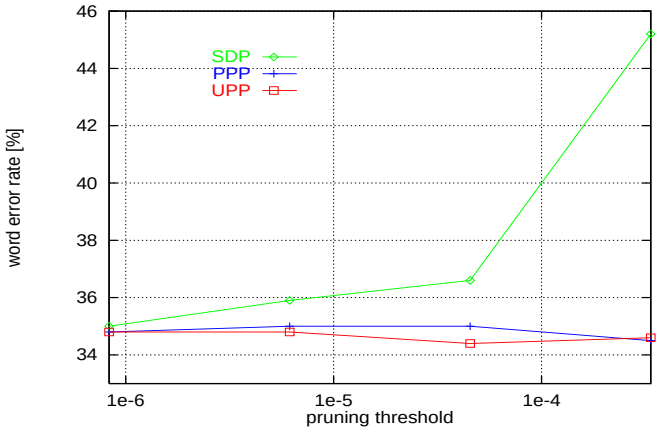

Figure 6.7: Word error rate vs. pruning threshold

of each one of the above pruning strategies. Fig. 6.7 shows how different pruning thresholds affect the recognizer's word error rate for all three pruning strategies. This plot reveals that although SDP yields high gains in recognition speed, it also causes significant increases in word error rate. In contrast, PPP and UPP exhibit a more gentle influence on recognition accuracy. In fact, for the range of pruning thresholds shown in Fig. 6.7 the recognition accuracy is not affected adversely at all. It is interesting to analyze the effects of dynamic tree pruning by means of a combined plot of recognition speed and accuracy. Fig. 6.8 depicts the trade-off between recognition speed and accuracy induced by dynamic tree pruning in a single graph. Here, we have investigated a bigger range of pruning thresholds from $10^{-9} \leq \Theta \leq 10^{-3}$ in order to make the trade-off more obvious.

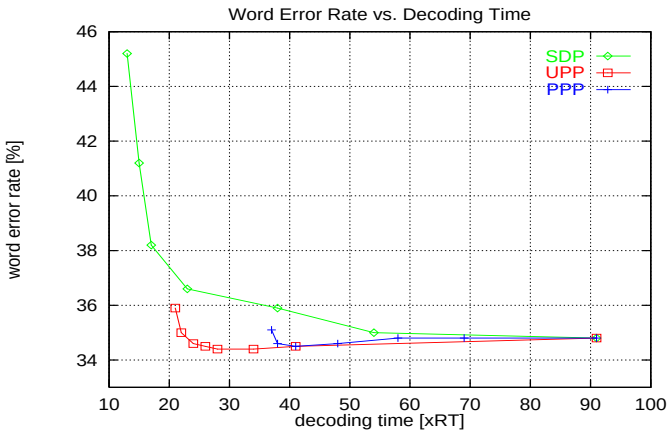

Figure 6.8: Word error rate vs. decoding time for varying pruning threshold in dynamic tree pruning

6.4 Discussion

Based on the above experiments and results, the usefulness and applicability of the proposed pruning strategies appear as follows:

- **Partial Posterior Pruning (PPP):**
  PPP yields comparably small gains in recognition speed and furthermore slows

down decoding for high pruning thresholds such that the usefulness of this technique is rather restricted.

- **Uniform Posterior Pruning (UPP):**
  UPP yields gains in overall recognition speed of up to a factor of 6 over the wide beam baseline system without negative effects on the recognition accuracy. Because of its moderate impact on recognition accuracy for high pruning thresholds, UPP should be regarded as the standard pruning strategy for dynamic tree pruning.

- **State Deactivation Pruning (SDP):**
  SDP yields considerably higher gains in recognition speed than those obtainable by UPP (up to a factor of 11). However, such high gains decrease the recognition accuracy noticeably. Therefore SDP should only be applied with comparably small pruning thresholds Θ or in cases where the speed-ups obtained by UPP are not sufficient for the particular application.

The following table 6.1 summarizes our results for speeding up a large vocabulary conversational speech recognition research system in terms of word error rate and associated decoding times. The baseline word error rate of 34.4% can be maintained while speeding up the system by a factor of 6 using uniform posterior pruning (UPP).

condition	**word error rate (%)**	**decoding time (xRT)**
baseline	34.4	145
tight decoding beams	34.8	91
moderate dynamic tree pruning	34.6	24
aggressive dynamic tree pruning	45.1	13

Table 6.1: Summary of results for fast model evaluation on Switchboard

Allowing a 30% relative increase in the word error rate, we can even speed-up the system by a factor of 13 using the more aggressive state deactivation pruning (SDP). Note that these results have been obtained with a large and complex evaluation system using the largest and most accurate hierarchical connectionist acoustic model (24000 tied states) that we have build so far. The techniques presented in this chapter allow us to reduce the turn-around times during the development of evaluation systems significantly from 145 times real-time to 24 times real-time without a loss in recognition accuracy.

Chapter 7

Speaker Adaptation

This chapter presents an algorithm for effectively adapting the parameters of a speaker-independent hierarchical connectionist acoustic model to the characteristics of a specific speaker. In contrast to existing acoustic models such as those based on mixture densities, the proposed hierarchical connectionist model does not require additional model parameter tying mechanisms such as regression class trees for effective adaptation of the model to specific speakers in the case of limited amounts of adaptation data. Rather, we benefit from the multi-level tree-structured representation of HMM states in our hierarchical connectionist model which inherently realizes parameter tying according to acoustic similarity.

7.1 Introduction

In some very rare cases, it is adequate to train the acoustic model of a speech recognition system on data from a single speaker, yielding a so-called *speaker-dependent* system. In most cases however, we are more interested in *speaker-independent* models that do not require data from a potential user during training. The parameters of a speaker-independent acoustic model are trained on data from several hundred different speakers in order to achieve robustness to unseen speakers. Unfortunately, this strategy not only increases robustness but also degrades overall system performance because of an increase in model variance. Fig. 7.1 illustrates this for two hypothetic acoustic models A and B. The incorporation of distributions from several speakers increases the variance of A and B compared to a single speaker and thereby increases their overlap which makes it harder to distinguish the two classes.
Generally, the error rate of a speaker-independent system is about twice as high as that of a speaker-dependent system. To close the gap in performance in cases where speaker-independent modeling is unavoidable as for example in commercial dictation systems, various methods for *speaker adaptation* have been proposed as a means for

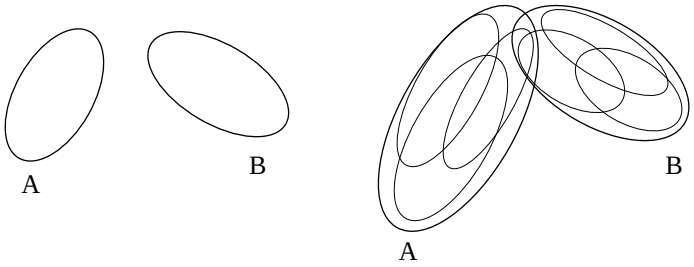

Figure 7.1: Speaker-dependent (left) vs. speaker-independent (right) models

improving the performance of such models on a specific speaker. In any case, speaker adaptation requires some acoustic data from the target speaker, so called adaptation data. We distinguish the following two principle strategies for speaker adaptation:

- **Feature based adaptation:** The preprocessed acoustic data of the target speaker is normalized by applying some kind of transformation such that the performance of the acoustic model improves on that data. A popular example of this strategy is vocal tract length normalization (VTLN) which attempts to normalize speech spectra for differing lengths of the vocal tract. In other approaches, speech cepstra are transformed by a general linear or affine map which is obtained by maximum likelihood estimation on the adaptation data. Feature based adaptation typically yields only moderate gains in accuracy as a single (linear) transformation of the input features does not allow to capture the characteristics of different speakers.

- **Model based adaptation:** Here, we follow the opposite strategy. Instead of transforming the input features such that the probability that our model has generated it is maximized, we transform the model to fit the data. At first glance, we might claim that there is no real difference between these two approaches. However, in model based adaptation, one typically applies different transformations to different HMM states or even to different component densities in a mixture model and thereby takes into consideration the complex variation in the acoustic realization of different polyphones across speakers. The most popular example for this strategy is Maximum Likelihood Linear Regression (MLLR) [Leggetter & Woodland '94] which applies linear (actually affine) transformations to the means (and potentially also to the variances) of Gaussians in a mixture densities based acoustic model. In fact, MLLR has evolved

to now being *the* standard technique for speaker adaptation in mixture based acoustic modeling.

Interestingly, it has been found that the gains obtained from feature and model based adaptation are nearly additive [Pye & Woodland '97], suggesting that both approaches cover different aspects of the variation among speakers. Depending on whether reference transcriptions are available with the acoustic adaptation data or not, we furthermore distinguish between supervised and unsupervised speaker adaptation:

- **Supervised adaptation:** For each adaptation utterance both the acoustic data (e.g., cepstra) and the reference word transcription are available (as is the case in training). Using the Viterbi algorithm, a state alignment can be generated that assigns HMM states to acoustic pattern vectors for each time frame. For instance, supervised adaptation is typically incorporated into a dictation system in form of an enrollment phase where the user has to read adaptation sentences that are provided by the system before he/she is allowed to use the system.

- **Unsupervised adaptation:** Only the acoustic data is available for each adaptation utterance. In order to obtain the state alignments required for most adaptation algorithms, the adaptation utterance is first decoded with the speaker-independent acoustic model, yielding a sentence hypothesis. This sentence hypothesis (although probably containing erroneous words) is then aligned with the adaptation data by applying the Viterbi algorithm[1]. If available, estimates of word confidence can be used to mask portions of the sentence that are considered unreliable by the recognizer. Unsupervised adaptation does not require user cooperation in form of an enrollment phase but can be applied while the recognizer is in use. However, unsupervised adaptation yields lower gains in recognition accuracy than supervised adaptation.

The approach to speaker adaptation that we present in the remainder of this chapter falls into the category of model based adaptation algorithms. We present and evaluate it in the context of unsupervised speaker adaptation on the Switchboard domain but it can just as well be applied to supervised adaptation as we will demonstrate in chapter 8.

[1] Often, a Viterbi decoder already provides a state alignment of the sentence hypothesis.

7.2 Limited Amounts of Adaptation Data

An important aspect of speaker adaptation that has to be addressed by any adaptation algorithm is data sparsity resulting from limited amounts of available adaptation data. A typical state-of-the-art acoustic model for large vocabulary speech recognition models several thousand distinct HMM states. For instance, consider a mixture density based acoustic model for 8000 HMM states trained on the Switchboard corpus. Fig. 7.2 depicts the coverage of this set of HMM states for various amounts of adaptation data (assuming a preprocessing rate of 100 frames per second).

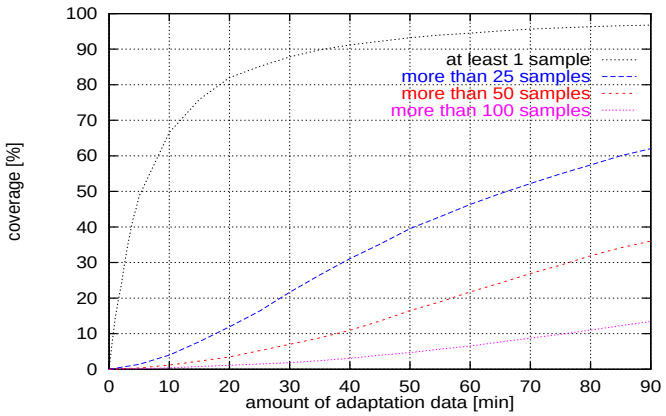

Figure 7.2: Data sparsity problem in speaker adaptation

About 5 minutes of adaptation data yield at least a single pattern vector (sample) for half of the HMM states. In order to cover 90% of all HMM states with at least a single pattern vector, over 35 minutes of adaptation data is required. However, observing a single pattern vector clearly does not allow to estimate an MLLR adaptation transformation. If we demand that more than 100 pattern vectors (samples) be observed per state, even a full hour of adaptation data only yields a coverage of 6.5% of all HMM states.

In practice, the amount of available adaptation data per speaker often is much lower. The Switchboard corpus, for instance, consists of telephone conversations between two speakers with an average duration of about 6 minutes and a maximum duration of about 10 minutes. Fig. 7.3 depicts a histogram plot showing the distribution of the amount of speech available per conversation side (speaker) in the corpus. There is a

sharp peak at around 3 minutes of data per speaker. For that amount of data, almost two thirds (65%) of the 8k HMM states in the above mentioned acoustic model will not be observed at all. Clearly, some kind of transformation tying must be introduced such that the large proportion of unobserved models can also benefit from these small amounts of adaptation data. Furthermore, tying is crucial for accumulating enough data for robust estimation of the parameters of an adaptation transformation.

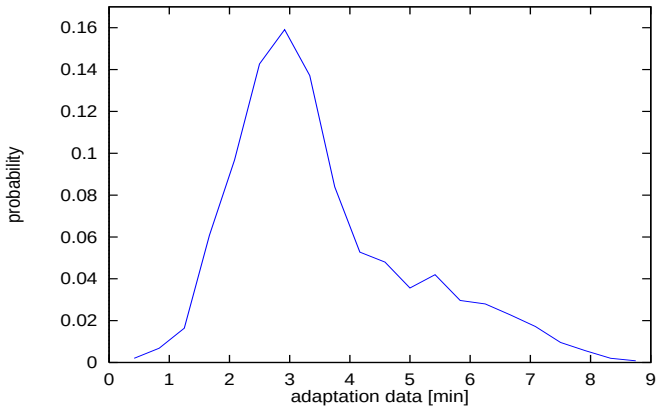

Figure 7.3: Histogram plot of amount of available data for 20000 Switchboard conversation sides

The standard approach to transformation tying uses a precomputed regression class tree for assigning a small, data dependent number of MLLR transformations (corresponding to the leaf nodes of the regression class tree) to the set of component densities of all mixtures. A regression class tree is computed by top-down clustering the set of component densities according to acoustic similarity down to a certain number of leaf nodes that depends on the amount of available adaptation data (see Fig. 7.4). The more adaptation data we have available, the deeper the regression class tree. At each leaf node of the regression class tree, a single MLLR adaptation transformation is estimated from the joint data of all component densities tied to that leaf node and then applied to transform the parameters of the tied component densities.
This way, it is possible to adapt even the component densities in HMM states that have not been observed in the adaptation data. However, such regression class tree based MLLR requires to compute and store the additional tying structure as the

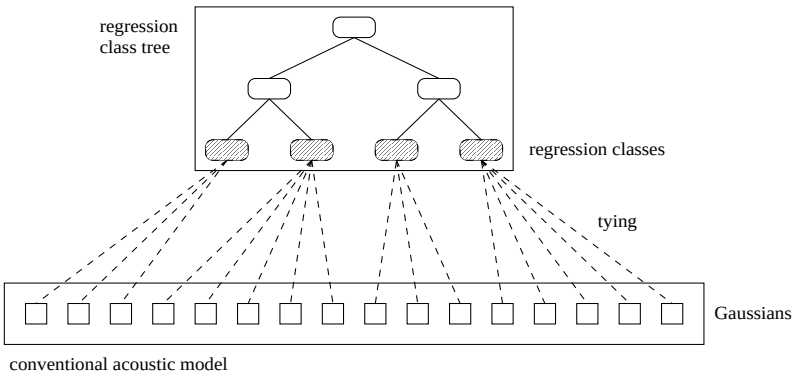

Figure 7.4: Transformation tying in regression tree based MLLR of conventional acoustic models

conventional acoustic model itself exhibits no structure at all. Interestingly, the tying structure that is missing in conventional acoustic models is readily available in a hierarchical connectionist acoustic model. Consequently, this kind of model inherently supports effective adaptation with limited amounts of adaptation data and the corresponding adaptation algorithm turns out to be much simpler and requires no additional structure.

7.3 Adaptation Algorithm for HNN Models

In our implementation of a speaker adaptation algorithm for the hierarchical connectionist acoustic model presented in this thesis, we exploit the multi-level state tying inherent to this tree structured model. When presenting training or adaptation data to the hierarchy, the available amount of data at each node increases from the bottom to the top of the tree. The root node of an HNN tree receives all data presented to the acoustic model and its estimates of posterior and prior probabilities are contributing to all the HMM states. Thus, the root node realizes the highest level of parameter sharing in this model and therefore is our primary candidate for model adaptation. Depending on the amount of available adaptation data, we might also adapt the tree nodes in the level below the root node and so on.
Following the above lines of thought, we can formulate a general method for speaker adaptation in the hierarchical acoustic model (Fig. 7.5). It consists of the three steps counting, node selection and node adaptation. First, the available amount

of adaptation data is computed for each tree node. The single free parameter in the algorithm is the *adaptation threshold* C_{min} which defines how many samples are considered to form 'enough data' for adapting the parameters of a node. Based on this threshold, we select tree nodes for adaptation in the second step.

HNN Speaker Adaptation Algorithm

1. **Counting:**
   (a) Compute state labels for the adaptation data using Viterbi alignments of either the correct transcripts (supervised adaptation) or the word hypotheses generated by the recognizer (unsupervised adaptation).
   (b) For each HMM state: count the number of adaptation samples assigned to that state according to the state alignments.
   (c) Assign these state counts to the corresponding leaf nodes in the HNN tree. Compute HNN node counts $C(N_i)$ for all nodes N_i in a bottom-up fashion:
   $$C(N_i) = \sum_{N_j \in CHILDREN(N_i)} C(N_j)$$
2. **Node Selection:**
   (a) Determine a reasonable minimum number of adaptation samples required to adapt a HNN tree node, the *adaptation threshold* C_{min}.
   (b) Select all HNN nodes N_i for adaptation that satisfy $C(N_i) \geq C_{min}$.
3. **Adaptation of Selected Nodes:**
   (a) Adapt local estimator of conditional a-posteriori probabilities
   (b) Adapt estimates of the prior probabilities of child nodes

Figure 7.5: Outline of speaker adaptation algorithm for hierarchical connectionist acoustic model

Finally, we adapt the parameters of the selected nodes based on the available adaptation data. It is very important to note that both the local estimator of posterior probabilities (a neural network in our case) *and* the estimates of child prior probabilities need to be adapted in each selected node as we use the model to compute scaled likelihoods. Nodes that receive less than C_{min} samples of adaptation data are not adapted by the algorithm. Although this might potentially lead to a mismatch between the adapted and the unadapted nodes in the tree structure, the benefit of

(1) improved discrimination in the selected and adapted tree nodes, and of (2) the significant amount of sharing of these nodes among the HMM states is expected to compensate such an effect.

7.3.1 Node Selection

For a given constant adaptation threshold C_{min}, different numbers of tree nodes will be selected for different speakers, depending on the amount of available adaptation data. This is exactly the behavior we desire in speaker adaptation: With increasing amount of adaptation data the number of tree nodes subject to adaptation increases until eventually all tree nodes are adapted to the characteristics of a particular speaker. In practical applications of speaker adaptation however, the amount of available adaptation data typically is very limited and allows to adapt only a small proportion of all tree nodes. Figs. 7.6 and 7.7 depict the situation for small and medium amounts of adaptation data.

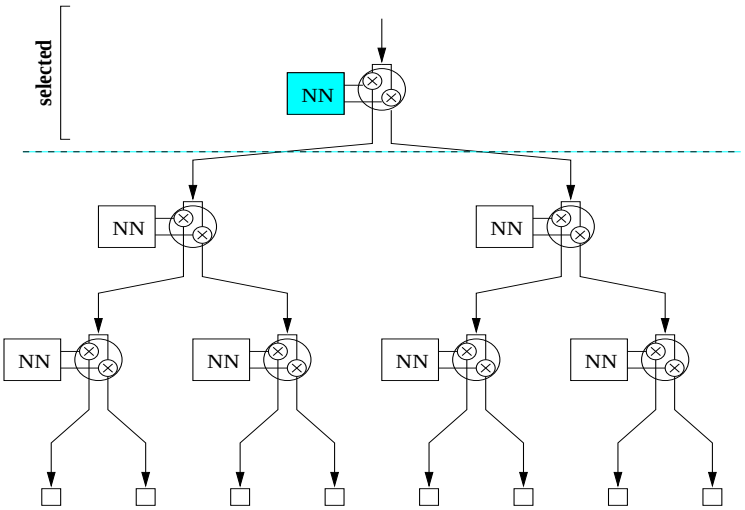

Figure 7.6: Adaptive selection of HNN nodes: small amount of adaptation data

For our investigation of speaker adaptation of hierarchical connectionist acoustic models on the Switchboard speech corpus, we chose the previously mentioned HNN model

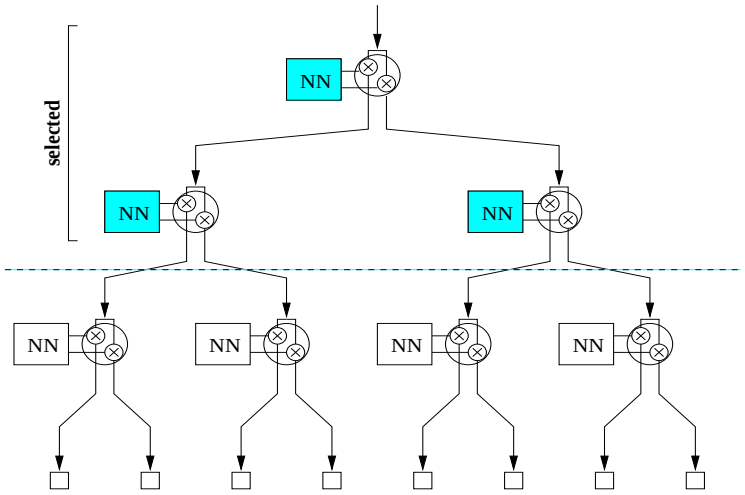

Figure 7.7: Adaptive selection of HNN nodes: medium amount of adaptation data

for 8000 tied HMM states and a set of 20 representative speakers from the 1996 evaluation test set. The amount of available adaptation data for each speaker varies between 1 minute and 7 seconds to 3 minutes and 16 seconds. Fig. 7.8 depicts the minimum, mean and maximum number of tree nodes selected for adaptation on that test set, depending on the value of the adaptation threshold C_{min}.
Assuming for instance, that $C_{min} = 2000$ yields enough data for robustly adapting HNN tree nodes, our algorithm selects between 2 and 7 tree nodes for adaptation.

7.3.2 Node Adaptation

Before we can answer the question of how to set the value of C_{min}, we first discuss methods for adapting the parameters of a particular HNN tree node as the choice of adaptation method will determine the amount of adaptation data required. As mentioned above, adaptation of HNN tree nodes requires to adapt the neural network that estimates local conditional posterior probabilities and furthermore to adapt the estimates of the prior probabilities of child nodes. Severe mismatches between posteriors and priors will lead to degraded performance if we only adapt one of the two distributions in a connectionist acoustic model.
Let's first consider the task of adapting the neural network for estimating local con-

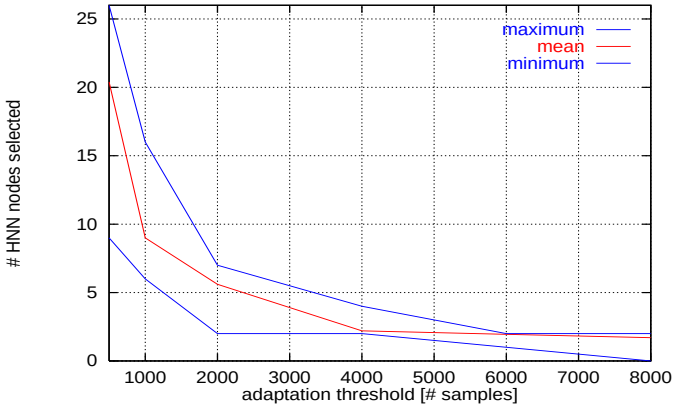

Figure 7.8: Min/mean/max number of HNN nodes subject to adaptation for different adaptation thresholds over 20 Switchboard test set speakers

ditional posterior probabilities at a specific tree node. The estimates produced by the network reflect the posterior probability distribution on the training set. Given a small set of adaptation samples from a specific speaker, we want to take account of the fact that the posterior distribution for that speaker might differ from the learned speaker-independent distribution. Depending on the number of available adaptation samples, we might consider the following techniques for adapting a network's parameters. In all cases, we have to withhold a small proportion of the adaptation data to be used as a validation set during network training. Otherwise, we will overfit the relatively small amounts of adaptation data, resulting in poor generalization.

- **Train new network:** If we have available a comparably large amount of adaptation data, we can train a new neural network with randomly initialized weights to estimate the local posterior probabilities on the adaptation data and simply replace the existing network in the corresponding HNN tree node. However, we must be aware of the fact that we will discard all information gained from speaker-independent training for the particular node that is subject to adaptation.

- **Retrain old network:** In this variant of adaptation, we continue to train the existing speaker-independently trained neural network on the adaptation data available at the corresponding HNN tree node. As we monitor the networks

performance on a withheld validation set during training, we can guarantee that the performance of the adapted network will be at least as good as the performance of the speaker-independent network with which we have started. Thus, this technique implicitly regularizes the adaptation step and prevents overfitting of small adaptation data sets.

- **Train additional linear front-end layer:** In cases where a HNN tree node obtains very little adaptation data or where the corresponding neural network is too large for retraining, we might want to keep the parameters of the network fixed and add an additional layer in front of the network's input layer which linearly transforms the input pattern vectors. This way, only a relatively small amount of parameters have to be estimated from the adaptation data which increases the robustness and generalization performance of the resulting adapted network. Adding a linear front-end layer for adaptation purposes is best suited to the relatively large networks used in traditional, monolithic connectionist acoustic models where it has been applied successfully (e.g., [Neto et al. '95]). For smaller networks, it will not be as effective as network retraining since a linear front-end layer can not fully capture the typically non-linear mapping from speaker-independent to speaker-dependent feature space.

Compared to traditional monolithic connectionist acoustic models, the networks used in our hierarchies of neural networks are much smaller which normally allows to apply retraining of the relevant speaker-independent networks in order to achieve effective speaker adaptation. Adaptation of local prior probabilities can be accomplished by simply re-estimating them on the adaptation data. As the available adaptation data at a particular HNN node must be sufficient for retraining the local neural network, it will be more than sufficient for re-estimating the priors.

7.4 Adaptation Experiments

In the following, we present results of applying the proposed adaptation algorithm to the task of unsupervised speaker adaptation of a hierarchical connectionist acoustic model on 20 representative speakers from the 1996 Switchboard evaluation test set. For that purpose, a speaker-independent Hierarchy of Neural Networks acoustic model for 8000 context-dependent tied HMM states has been trained on the full Switchboard training corpus, consisting of data from more than 500 different speakers. As we investigate unsupervised speaker adaptation, we first have to run the recognizer with the speaker-independent acoustic model and generate sentence hypotheses and state alignments for the acoustic adaptation data. Fig. 7.9 shows the amount of adaptation data available for each of the 20 adaptation speakers.

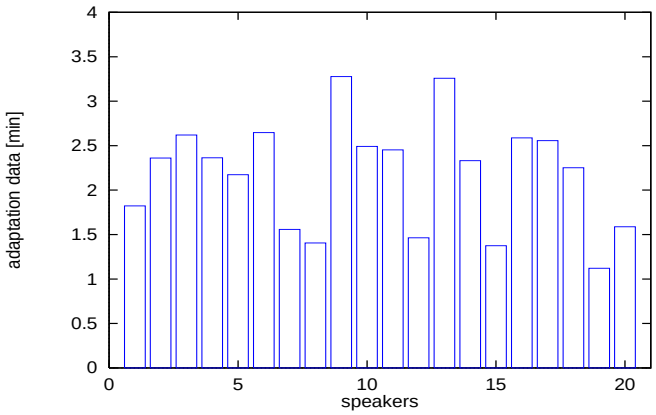

Figure 7.9: Available adaptation data for 20 Switchboard test set speakers

The baseline performance of the speaker-independent system on the set of these 20 adaptation speakers is 36.8% word error. Thus, on average every third word is decoded falsely, resulting in erroneous word and state alignments. We did not attempt to mask the false segments by applying some kind of word confidence measure as has become popular now, but used all of the obtained state alignments for adaptation.

7.4.1 Node Selection

For each speaker, we compute sample counts for all HNN tree nodes from the state alignments and select nodes subject to adaptation based on a pre-determined adaptation threshold C_{min}. We have experimented with

$$C_{min} \in \{500, 1000, 2000, 4000, 6000, 8000\}.$$

The following table 7.1 gives an overview of the number of adaptation samples (frames) available for each speaker and the number of tree nodes selected for adaptation based on the different values of C_{min}. Note how the differing global amounts of available adaptation data lead to significantly different numbers of selected tree nodes for each speaker. Furthermore, the value of C_{min} must be carefully selected such that it does not exceed the available amount of adaptation data as in the case of $C_{min} = 8000$ for speaker 'sw4338-A'. Otherwise, none of the tree nodes will be selected and the model can not be adapted.

7.4.2 Neural Network Adaptation

Having selected tree nodes for adaptation according to the adaptation threshold, we proceed with the adaptation of the local neural networks. We opted for the method of retraining the existing speaker-independent networks on the available adaptation data. For each selected tree node, 10% of the corresponding adaptation data were withheld as the validation set for monitoring performance. The remaining 90% were used for gradient-descent based training. Training was stopped as soon as the performance on the validation set ceased to improve.

Speaker ID	**# adapt. frames**	**# HNN nodes subject to adaptation for various values of C_{min}**					
		500	**1000**	**2000**	**4000**	**6000**	**8000**
sw3157-A	10933	18	8	5	2	2	2
sw3157-B	14164	24	9	7	2	2	2
sw3264-A	15719	25	10	7	2	2	2
sw3380-A	14182	24	9	7	2	2	2
sw3494-B	13044	23	8	6	2	2	2
sw3538-A	15881	24	11	7	2	2	2
sw3538-B	9348	17	7	4	2	2	2
sw3822-A	8433	9	7	3	2	2	1
sw3824-B	19658	25	16	7	4	2	2
sw3835-A	14950	25	9	7	2	2	2
sw3927-A	14715	25	9	7	2	2	2
sw3940-B	8782	10	8	2	2	2	1
sw4073-B	19546	26	12	7	4	2	2
sw4093-A	13990	23	9	7	2	2	2
sw4093-B	8247	12	7	3	2	2	1
sw4141-A	15525	25	10	7	2	2	2
sw4178-A	15341	25	9	7	2	2	2
sw4322-A	13510	23	8	6	2	2	2
sw4338-A	6726	10	6	2	2	1	0
sw4373-B	9524	15	8	4	2	2	1
Average:	**13111**	**20.4**	**9.0**	**5.6**	**2.2**	**2.0**	**1.7**

Table 7.1: Adaptation data and number of adapted nodes for 20 Switchboard test set speakers

As an example, we take a closer look at the network at the root node of the HNN tree. Fig. 7.10 shows for each test speaker the classification error rate of this network on the speaker's validation data before and after adaptation. In all but 3 cases, the classification error rate of the network at the root node could be improved by retraining it on the adaptation data. As can be seen from table 7.1, the adaptation

algorithm typically selects more than just the root node for adaptation. Thus, even if the performance of the root node could not be improved by retraining on the adaptation data, some other node further down the tree might be.

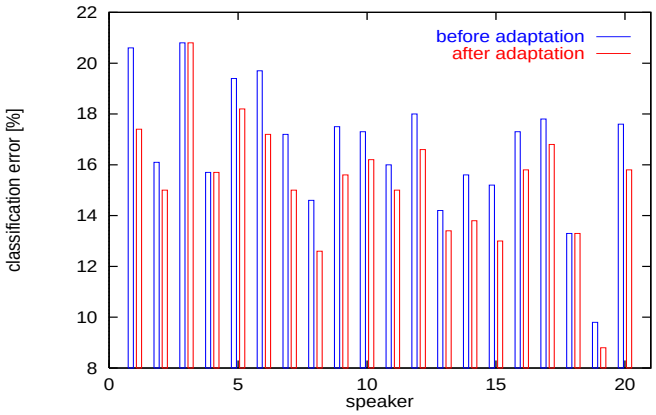

Figure 7.10: Classification error of root node classifiers before and after adaptation

Let's now take a look at how the local improvements in conditional posterior probability estimation at the adapted nodes influence the estimation of HMM state posteriors in the complete hierarchy. Fig. 7.11 shows the average negative log posterior probability of the HMM state assigned to an adaptation sample for different numbers of adapted tree nodes in the hierarchical connectionist acoustic model. The estimates are averaged over all data of all 20 adaptation speakers. As expected, the average negative log posterior decreases (the posterior probability increases) with increasing number of adapted nodes. The plot in Fig. 7.11 suggests to adapt even more than an average of 20 nodes in the HNN tree as the average posterior probability of the adaptation data is expected to rise even further. However, we must be careful and not jump to conclusions imprudently. The state alignments that we are scoring were obtained from erroneous sentence hypotheses generated by the decoder as we operate in unsupervised adaptation mode. Thus, an increase in the number of adapted tree nodes will only increase the probability of the falsely decoded hypotheses - not the probability of the unknown correct transcription that we really seek to increase. In practice, we have to measure the word error rate for different values of C_{min} in order to find an optimal number of adapted tree nodes.

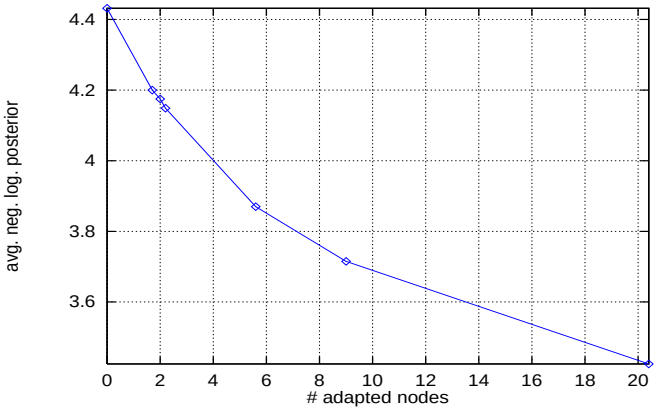

Figure 7.11: Average negative log posterior probability of adaptation data for varying number of adapted tree nodes

7.4.3 Prior Adaptation

As stated before, the estimates of child prior probabilities at selected tree nodes are adapted by re-estimating these probabilities on the adaptation data. This is accomplished by simply normalizing the counts for each child node to get relative frequencies. To demonstrate that there really are differences in the distributions of prior probabilities between the speaker-independent training set and adaptation sets of different speakers, we again take a closer look at the root node of the HNN tree. The root node in our 8k HNN has four child nodes. Fig. 7.12 plots the symmetric KL distance (information divergence) between the speaker-independent prior distribution and the speaker-adapted prior distributions for each speaker.

Although comparably small, there are measurable differences between these prior distributions. As the KL-distances themselves are not easily interpretable, we also plot the actual prior distributions for the speaker-independent baseline and the speakers with the smallest (sw4141-A / #16) and largest (sw4338-A / #19) KL distances in Fig. 7.13.

Primarily the prior probability of the first child node seems to vary strongly between different speakers. In the case of speaker sw4338-A, this prior has more than doubled compared to the speaker-independent estimates. This observation confirms the importance of prior re-estimation as an essential part of speaker adaptation in our

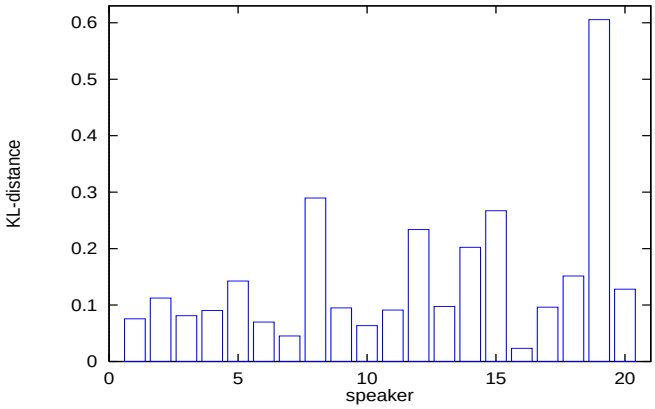

Figure 7.12: KL-divergence of prior distributions at root node before and after adaptation

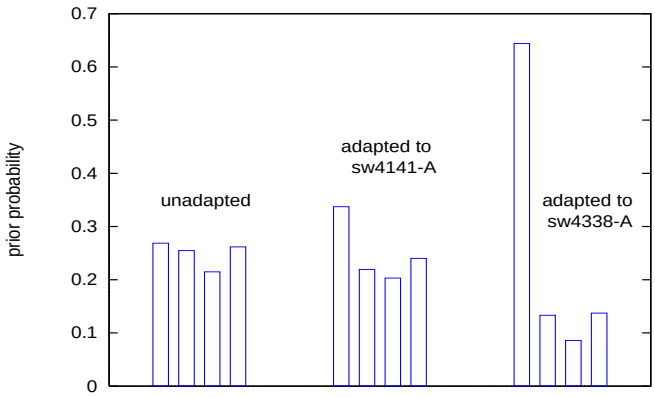

Figure 7.13: Comparison of unadapted and adapted prior distributions at the root node

hierarchical connectionist acoustic model.

7.4.4 Recognition Results

Finally, we present recognition results for using the speaker-adapted hierarchical models for re-decoding the corresponding speaker's data. Table 7.2 gives individual results for all 20 test speakers and the different adaptation thresholds investigated in this study.

Speaker ID	**adapt. data**	**unadapted baseline**	**word error rates for various values of C_{min}**					
			500	**1000**	**2000**	**4000**	**6000**	**8000**
sw3157-A	1'49"	39.3%	38.6%	38.6%	38.6%	38.6%	38.6%	38.6%
sw3157-B	2'21"	43.3%	41.3%	42.3%	43.3%	44.2%	44.2%	44.2%
sw3264-A	2'37"	31.8%	29.4%	29.4%	31.2%	32.7%	32.7%	32.7%
sw3380-A	2'21"	25.0%	23.0%	23.0%	22.2%	24.4%	24.4%	24.4%
sw3494-B	2'10"	41.8%	34.9%	33.6%	34.9%	35.6%	35.6%	35.6%
sw3538-A	2'38"	27.6%	21.9%	25.7%	25.7%	21.9%	21.9%	21.9%
sw3538-B	1'33"	37.9%	36.5%	34.8%	40.0%	39.1%	38.3%	38.3%
sw3822-A	1'24"	47.2%	43.8%	44.9%	46.1%	46.1%	46.1%	46.1%
sw3824-B	3'16"	37.9%	36.8%	36.8%	38.5%	36.8%	36.8%	36.8%
sw3835-A	2'29"	37.5%	35.9%	36.7%	38.3%	35.9%	35.9%	35.9%
sw3927-A	2'27"	39.8%	31.2%	32.1%	32.1%	33.0%	33.0%	33.0%
sw3940-B	1'27"	53.8%	48.7%	47.4%	50.0%	50.0%	50.0%	51.3%
sw4073-B	3'15"	39.3%	36.1%	36.1%	35.2%	33.6%	34.4%	34.4%
sw4093-A	2'19"	34.8%	31.5%	29.3%	29.3%	29.3%	29.3%	29.3%
sw4093-B	1'22"	25.2%	24.4%	22.8%	23.6%	22.8%	22.8%	29.1%
sw4141-A	2'35"	22.8%	22.8%	21.3%	22.1%	22.1%	22.1%	22.1%
sw4178-A	2'33"	35.5%	30.1%	28.0%	28.0%	30.1%	30.1%	30.1%
sw4322-A	2'15"	37.7%	30.8%	31.5%	30.8%	33.1%	33.1%	33.1%
sw4338-A	1'07"	55.5%	51.8%	52.6%	51.1%	51.1%	55.5%	55.5%
sw4373-B	1'35"	31.8%	26.4%	27.1%	26.4%	26.4%	26.4%	34.1%
Total:	–	**36.8%**	**33.4%**	**33.3%**	**33.9%**	**33.9%**	**34.1%**	**35.0%**

Table 7.2: Results of unsupervised speaker adaptation for 20 Switchboard test set speakers

The last row gives word error rates for the unadapted baseline and all adapted systems averaged over all 20 test speakers. As expected, the overall performance improves with increasing number of adapted tree nodes. However, there usually is a trade-off between increasing the number of adapted parameters and generalization performance in unsupervised adaptation algorithms. With increasing number of parameters, performance typically first improves (as we have observed here too) but then starts to

degrade again as we gradually allow the model to perfectly reproduce the erroneous transcriptions of the adaptation data. It appears that the algorithm for adapting hierarchical connectionist acoustic models does not suffer from this overfitting effect. Performance of adapted models seems to level off with increasing number of adapted parameters. The difference between the results for $C_{min} = 1000$ and $C_{min} = 500$ are not statistically significant.
Concerning this kind of behavior, the robustness of our adaptation algorithm can be attributed to the fact that we are not seeking to optimize the likelihood of all the available adaptation data, as for example is the case in MLLR, but the posterior probability over a smaller, withheld validation set as is common practice for avoiding overfitting in the training of neural networks. As we start training on the adaptation data with the speaker-independent network parameters and do not allow for a decrease in the performance on the validation set during adaptation, the effective number of adapted parameters is smaller than what we would assume from the selected number of tree nodes. In fact, with decreasing C_{min}, we only *allow* the adaptation algorithm to adapt more networks in the HNN tree - we do not *force* it to really adapt all the selected networks. As a result the adaptation algorithm is less dependent on finding an optimum value for C_{min}.
Finally, Fig. 7.3 summarizes the recognition results we have obtained with unsupervised adaptation on Switchboard data. Here, we have included both the adaptation threshold used for selecting tree nodes for adaptation and the resulting average number of selected tree nodes.

C_{min}	**average**	
	# selected nodes	**word error rate**
unadapted	–	36.8%
8000	1.7	35.0%
6000	2.0	34.1%
4000	2.3	33.9%
2000	5.6	33.9%
1000	9.0	33.3%
500	20.4	33.4%

Table 7.3: Summary of results for unsupervised speaker adaptation

Compared to the unadapted baseline, unsupervised speaker adaptation on an average of about 2 minutes of adaptation data yields an average relative reduction in word error rate of 9.5%. While this is comparable to what has been reported for regression tree based MLLR adaptation of conventional acoustic models, the hierarchical structure of the connectionist model presented in this thesis allows for more natu-

ral integration of speaker adaptation. No additional structures such as regression class trees are required for dealing with small amounts of adaptation data as the tree structured model itself realizes the required parameter sharing.

Chapter 8

Structural Domain Adaptation

One of the most interesting applications of hierarchical connectionist acoustic models is in domain-adaptive speech recognition. We present Structural Domain Adaptation (SDA) [Fritsch et al. '98a, Fritsch et al. '98b], an approach for efficiently and effectively downsizing and adapting the structure of a hierarchical connectionist acoustic model for the purpose of porting a large vocabulary conversational speech recognition system to a previously unseen application domain. We motivate why structural as well as acoustic adaptation is beneficial in addition to the adaptation of the vocabulary and the language model of a speech recognition system. We demonstrate how SDA allows to build domain-adaptive speech recognition systems that match the performance of domain-specific systems with only moderate requirements regarding the amount of acoustic adaptation data.

8.1 Motivation

It is well known that statistical speech recognition systems are highly dependent on the characteristics of the data they are trained on. To obtain reasonable performance, one has to focus on a specific application domain in order to restrict the variability of both the acoustic and the linguistic training data. Typical aspects of relevance are

- quality of acoustic data (sampling rate, microphone, AD converters)
- recording conditions (indoor/outdoor/telephone, background noise)
- type of speech (read/spontaneous/conversational, isolated/continuous)
- vocabulary and phonetic transcription of words
- a-priori probability of words and word sequences

In addition to the large variability in acoustic realization of elementary speech units, the typically finite size of the recognition vocabulary and the language model require to restrict the application domain of a speech recognition system. The resulting lack of universality stands in sharp contrast to what we all experience in human speech recognition. Following is a list of the most popular application domains for which speech recognition systems are currently built:

- Personal speech-to-text (dictation) systems
- Interactive Voice Response (IVR) systems for automated call centers
- Transcription of broadcast news for building searchable multimedia databases for information retrieval
- Command & control systems
- Car navigation systems

As long as statistical speech recognition systems are being used in matched conditions, meaning that the application domain matches the training domain, performance can be expected to be similar to what has been observed on a validation set during training. However, if there are considerable acoustic or linguistic mismatches caused either by deploying a lab-trained system to the field or by applying a system to an unseen, new application domain, performance often drops unacceptably (e.g., [Thomson '97]).
In the following, we experimentally examine this problem by means of the Switchboard domain (spontaneous conversational telephone speech) as the baseline training domain and the following two rather different target domains:

- **Wall Street Journal (WSJ):** This domain is representing a dictation task that consists of read financial newspaper articles. For the purposes of this thesis, we are using a less known subset of the WSJ corpus consisting of telephone data (from DARPA's 1993 WSJ Spoke 6 evaluation) in order to match the recording conditions between Switchboard and WSJ. Still, there are large differences in type of speech, vocabulary and language between these two corpora. The focus in choosing this particular subset of the WSJ domain was on investigating the problem of porting a conversational speech recognition system to a dictation domain.
- **English Spontaneous Scheduling Task (ESST):** This domain is consisting of high-quality (16 KHz) recordings of spontaneous conversations for scheduling meetings. Although the differences in type of speech are less serious between

this corpus and the Switchboard corpus, there are significant differences in size of vocabulary, language model and quality of acoustic data. The focus in choosing this domain was on investigating the problem of porting a conversational speech recognition system to a domain of much smaller and more specific vocabulary.

Table 8.1 gives recognition results obtained in in-domain vs. out-of-domain conditions on the above two corpora. These results impressively demonstrates the domain-dependence of statistical speech recognition systems. The first row gives the word error rate for a speech recognition system that was trained and optimized specifically for the WSJ domain and tested on data from the same domain. The second row gives the word error rate for another system, this time trained and optimized specifically for the ESST domain and tested again on in-domain data. In contrast, the last row gives word error rates for a system trained and optimized on the Switchboard domain and tested without modifications[1] on data from the WSJ and the ESST domains, respectively.

recognizer training domain	**word error rate on WSJ domain**	**word error rate on ESST domain**
WSJ	12.5%	–
ESST	–	19.5%
Switchboard	45.4%	55.3%

Table 8.1: In-domain vs. out-of-domain performance of speech recognition systems

In our scenario the out-of-domain word error rate is roughly 3 times higher than what is achievable with dedicated recognizers in matched conditions. In addition, the out-of-domain performance is way too poor in both cases to allow for any reasonable application.
From the results above it is obvious that universal, domain-independent speech recognition is not available with today's technology. However, it is possible to at least adapt or exchange the relevant components of a recognizer using some data from a new domain and thereby reduce the mismatch between training and application domain. Whereas new domain-specific vocabularies and phonetic dictionaries can be obtained quickly and inexpensively [Geutner et al. '97], and new domain-specific language models require only that large amounts of text data are available, the adaptation of the acoustic model is considerably more expensive and time- and labour-consuming as it also requires the availability of transcribed acoustic data.

[1] Except that 16 KHz data was downsampled to 8 KHz as required by the Switchboard preprocessing frontend.

In addition, an analysis of state-of-the-art decision tree clustered context-dependent acoustic modeling reveals that there are two different aspects that cause a dependence on the training domain:

- Estimators of HMM emission probabilities depend on the acoustic characteristics of the training domain.
- Specificity of context modeling as represented by number and identity of phonetic context classes depends on vocabulary, phonetic dictionary and language model of the training domain.

It is important to note that the phonetic transcription of words *and* their a-priori probabilities in the training corpus affect the outcome of word internal phonetic context clustering. Additionally, cross-word phonetic context modeling is affected by the probabilities of word pairs (and word triples in the case of single-phone words) in the training corpus. Previous approaches to domain adaptation (e.g., [Siu et al. '99]) have only addressed the first of the above items by means of some sort of supervised acoustic adaptation. The second item is mostly ignored, as size and specificity of conventional acoustic models can not easily be modified due to the flat, independent representation and evaluation of emission probabilities. This is quite disadvantageous in terms of memory and computational requirements in cases where a considerably smaller amount of context modeling is sufficient in the target domain.
Consider for instance the extreme case of porting a large vocabulary conversational speech recognition (LVCSR) system (e.g., trained on the Switchboard corpus) to a ten word vocabulary digit recognition task. The very specific acoustic model of the LVCSR system will typically consist of several thousand context-dependent HMM states and require over 20 MBytes of RAM and considerable amounts of computation during decoding. Simple acoustic adaptation might be effective in reducing the word error rate to some extent but the model will still be too large and detailed to be used for simple ten word digit recognition. Instead, a completely new acoustic model is typically clustered and trained from scratch, which requires large amounts of transcribed acoustic data.
In the remainder of this chapter, we will present a technique that, in addition to acoustic adaptation, allows us to adapt the size and structure of hierarchical connectionist acoustic models to smaller requirements regarding the specificity of phonetic context modeling. In contrast to acoustic adaptation of conventional acoustic models, our approach addresses both of the discussed aspects of domain dependence and allows to adapt the size and re-use parts of a trained hierarchical model for any new domain, even in cases such as the ten digit task described above. We will show that domain-specific performance can be achieved with domain-specific model size and only small requirements regarding the amount of transcribed acoustic adaptation data.

8.2 Quantifying Domain Mismatches

First however, we analyze and quantify the differences between application domains that cause the large discrepancy in performance between in-domain and out-of-domain application of speech recognition systems in more detail, using Switchboard (SWB) as the baseline training domain and WSJ and ESST as the target application domains.

8.2.1 Vocabulary and Language Model

We compare the vocabulary and language model used in the SWB recognizer with vocabularies and language models built specifically for the WSJ and ESST domains, respectively. The vocabulary used in the SWB domain consists of 15000 unique words. In contrast, the vocabulary used in the WSJ domain consists of only 5000 unique words and the vocabulary used in the ESST domain consists of only 2850 unique words. Even though both of the target domains exhibit a much smaller vocabulary, the vocabulary of the SWB domain does not cover all of the words in the target domains (see Table 8.2).

domain	**out-of-vocabulary rate**
WSJ	7.4%
ESST	0.9%

Table 8.2: Out-of-vocabulary rates of SWB vocabulary on WSJ and ESST test sets

On the test sets used for our experiments the *Out-Of-Vocabulary (OOV)* rate for the SWB vocabulary is 7.4% for WSJ and 0.9% for ESST. Particularly in the case of WSJ, the OOV words are expressions and proper names specific to the domain of financial news. A general rule of thumb is that each OOV word causes between 1 and 2 word errors. Thus, the mismatch in vocabulary explains a significant part of the increase in word error rate of the SWB recognizer at least in the case of WSJ data. A much larger contribution to the mismatch however is caused by differences in the domain-specific language models. Standard n-gram language models learn to predict characteristic word sequences that are specific to the training domain. They typically perform poorly on texts from a different, previously unseen domain. We measure the difficulty of a recognition task relative to a given statistical language model and a test set consisting of a sequence of words $w_1, \ldots, w_n$ by the *perplexity*

$$PP = \hat{P}(w_1, \ldots, w_n)^{(-\frac{1}{n})}$$

where $\hat{P}()$ denotes the probability of a word sequence as estimated by the language model. The perplexity measures the average number of words between which the recognizer must decide when transcribing a word of spoken text, relative to the given language model. The maximum in perplexity is given by the size of the vocabulary. All other things being equal, we are interested in a language model that minimizes the perplexity. Table 8.3 gives perplexities computed for different domain-specific language models on the specific test sets from WSJ, ESST and SWB that were used for the experiments in this chapter. Obviously, there are great differences in perplexity between in-domain and out-of-domain usage.

language model training domain	**perplexity on WSJ**	**perplexity on ESST**	**perplexity on SWB**
WSJ	*68*	251	148
ESST	1607	*23*	500
SWB	757	205	*71*

Table 8.3: In-domain vs. out-of-domain perplexity of various language models on test sets used in this thesis

For the out-of-domain recognition results reported in Table 8.1, we have used the SWB language model for recognizing speech from WSJ and ESST. In these two cases, the perplexity on the test sets increased by a factor of 11 from 68 to 757 on WSJ and by a factor of 9 from 23 to 205 on ESST!
To measure the effect of mismatches in vocabulary and language model on the word error rate of a recognizer, we repeated the experiments from Table 8.1, this time however with domain-specific vocabularies and language models. Table 8.4 gives results for these experiments.

recognizer training domain	**word error rate on WSJ domain**	**word error rate on ESST domain**
WSJ	12.5%	–
ESST	–	19.5%
SWB	17.2%	28.3%

Table 8.4: In-domain vs. out-of-domain performance of speech recognition systems when using domain-specific vocabularies and language models

Interestingly, 62% (WSJ) and 49% (ESST) of the mismatch in word error rate between in-domain and out-of-domain speech recognition can be compensated by switching to

domain-specific vocabularies and language models. It can be assumed, that obtaining domain-specific vocabularies and dictionaries requires relatively little effort. Furthermore, training domain-specific language models is simplified by an ever increasing amount of available text material, for instance on the Internet.
However, there still is a significant difference in word error rate remaining which is caused primarily by mismatches in acoustic modeling.

8.2.2 Acoustic Model

Mismatches in acoustic modeling are much harder to compensate than mismatches in vocabularies and language models. In contrast to the latter, replacing the original model with a domain-specific one is typically impracticable for acoustic models as it would require the very expensive recording and transcription of several hours of speech data in order to obtain enough training material for robustly clustering and training a completely new domain-specific acoustic model.
Before we discuss alternative, less expensive solutions to this problem, let's first analyze which factors contribute to a mismatch in acoustic modeling in out-of-domain applications of speech recognition systems. We have identified the following three types of mismatches that typically occur jointly:

1. **Acoustic mismatch:** An acoustic mismatch is caused by a wide variety of factors: different microphones, pre-amplifiers and AD converters, different sampling rates, different recording conditions, existence/nonexistence of background noise, different dialect, age or gender of speakers. All these factors lead to a difference in emission probability distributions for the basic speech units modeled by the HMM states in an acoustic model.

2. **Context specificity mismatch:** Different application domains require different amounts of phonetic context modeling, depending on difficulty, type of speech (conversational vs. read) and size of the domain vocabulary. The specificity of phonetic context modeling is determined by the number of decision tree clustered HMM states which is typically fixed a-priori for a given training domain and can not be altered easily in conventional acoustic models. In out-of-domain scenarios the acoustic model can either turn out to be too general (too small, not enough allophonic variation) or too specific (too large, overfitting, many unseen HMM states) for the target domain. Even if we manage to eliminate overfitting effects in cases where the model is too large and detailed by techniques such as parameter tying, we still have an oversized model that consumes too much memory.

3. **Prior mismatch:** The distribution of a-priori probabilities of a set of context-dependent HMM states that were clustered on some training domain varies significantly from domain to domain, mostly depending on the words in the vocabulary, their phonetic transcriptions and their unigram probabilities. The following Fig. 8.1 demonstrates this effect by means of a decision tree clustered model for 24k HMM states constructed on the SWB corpus. The baseline a-

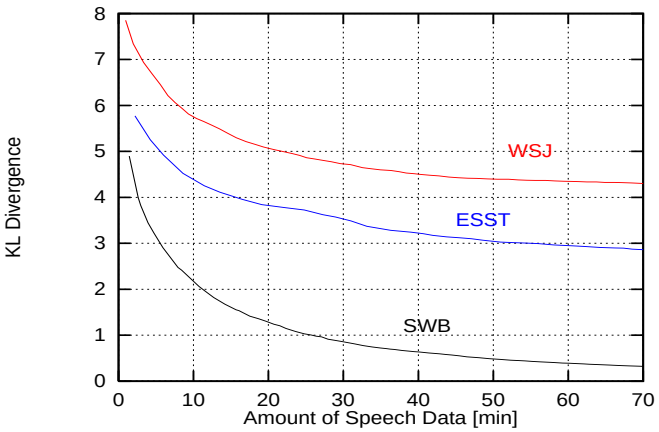

Figure 8.1: KL-divergence of a-priori HMM state distributions between the training domain (SWB) and various application domains (SWB,WSJ,ESST) for different amounts of data

priori distribution of the 24k HMM states is estimated on the full SWB training corpus. This distribution is then compared against a-priori distributions estimated from a variable amount of data from the SWB, WSJ and ESST corpora, respectively. We use information divergence (KL-distance) to compare two a-priori distributions. The smaller the KL-distance, the more similar the a-priori distributions. From left to right, the a-priori distributions become more and more stable as more data is being used for their estimation. As expected, the SWB curve approaches zero for an increasing amount of data, as the corresponding prior distribution converges against the prior distribution estimated on the full training corpus. However, the curves for WSJ and ESST level off at some offset distance considerably larger than zero after about one hour of data. Obviously, the a-priori distributions of the SWB clustered HMM states on WSJ and ESST will never converge to that of the SWB corpus, no matter

how much data we use for estimating them.

In the standard approach, only the first type of mismatches is addressed by some sort of supervised acoustic model adaptation, for instance using regression tree based MLLR with a mixture of Gaussians based acoustic model. Mismatches in the specificity of phonetic context modeling and the prior distribution of HMM states are ignored completely. These kind of mismatches are a result of the domain-dependence of phonetic decision trees used for clustering basic speech units such as phones based on their phonetic context. Size and structure of these trees are determined on the training domain and therefore depend heavily on the specific vocabulary, phonetic dictionary, and language model of the training domain. Out-of-domain mismatches in these components therefore cause mismatches in the phonetic decision trees that in turn cause mismatches in the specificity of context modeling and the prior distribution of HMM states because the leaf nodes of the phonetic decision trees define the set of distinctly modeled HMM states.
To eliminate these mismatches, we would have to replace the set of phonetic decision trees with a set of new ones constructed specifically for the target domain. Unfortunately, such an approach implies that a completely new acoustic model is trained on the target domain which is impracticable for the reasons already stated above.

8.3 The SDA Algorithm

In contrast to conventional models, the tree structure of hierarchical connectionist acoustic models together with their scalability and multi-level representation of phonetic contexts allows for efficient and effective compensation of all three kinds of mismatches in acoustic modeling (see previous section) that occur in out-of-domain applications of speech recognition systems. To demonstrate this, we have developed an algorithm called *Structural Domain Adaptation (SDA)* [Fritsch et al. '98a, Fritsch et al. '98b] that is based on the observation that it is always possible to reduce the specificity of context modeling in a trained hierarchical connectionist acoustic model by removing (pruning) irrelevant substructures from the modeling tree.
Consequently, the idea in structural domain adaptation is to construct a very detailed, highly specific hierarchical connectionist acoustic model using phonetic decision trees that yield a large amount of context resolution such that almost all potentially significant contexts of a particular language are represented in the model tree. Of course, this requires a training domain that exhibits a comparably large amount of phonetic variability and a large vocabulary. If there is no such training domain available, we can use a meta-domain composed of several different sub-domains. Starting from the resulting large and very detailed acoustic model, structural domain adaptation then compensates (1) acoustic and (2) structural mismatches in a specific target

domain in a two step process based on a moderate amount of transcribed acoustic adaptation data (typically less than an hour). Additionally, it allows to arbitrarily reduce the overall size and specificity of the acoustic model to decrease memory and computational requirements.

Acoustic mismatches are reduced by applying the adaptation algorithm presented in the previous chapter in supervised mode. Based on the amount of available adaptation data and an adaptation threshold C_{min}, this step in the algorithm dynamically determines a set of tree nodes in the hierarchical connectionist acoustic model that receive enough adaptation samples for adapting the corresponding estimators of local posterior and prior probabilities. As these nodes are located in the upper layers of the tree, their estimates are contributing to a large number of HMM states and by adapting these nodes, we effectively and robustly adapt all HMM states through the tree's tying mechanism. The algorithm automatically adjusts the number of adapted parameters to the amount of available adaptation data. Furthermore, the adaptation step in SDA adjusts the local prior probabilities of selected nodes thereby compensating a large proportion of the mismatch in the a-priori probability distribution of HMM states between training and target domain. See chapter 7 for a detailed analysis.

Structural mismatches, i.e. differing local requirements concerning the specificity of context modeling, are compensated by identifying and permanently deleting irrelevant substructures in the hierarchical model. Tree nodes are considered irrelevant if they receive less or equal than a small, empirically determined pruning threshold C_{prune} of adaptation samples in the target domain (represented by the adaptation data). If a particular tree node is rarely used in the target domain (as indicated by a small node count), we can assume that the partitioning into more specific context classes performed by this tree node no longer makes sense and can not be performed robustly in the target domain. Fortunately, the modular composition of HMM state posteriors in a hierarchical connectionist acoustic model allows to remove such nodes and thereby reduce the local specificity of context modeling without having to adjust a single parameter. Due to the over-specificity of the baseline model, certain HMM states will not be observed at all in the adaptation data available from the target domain. Tree branches leading to such 'dead' states are pruned implicitly as a result of prior adaptation but SDA will additionally remove obsolete nodes (nodes that lead only to unobserved states), if existing in the model tree.

Since we count the number of adaptation samples for each tree node as the first step of acoustic adaptation anyway, determining whether or not a node is subject to pruning can take place in conjunction with determining whether or not it is subject to adaptation. Of course, the count thresholds must satisfy $C_{prune} < C_{min}$ since tree nodes can not be adapted and pruned at the same time. The following Fig. 8.2 illustrates structural domain adaptation by means of a small balanced binary example

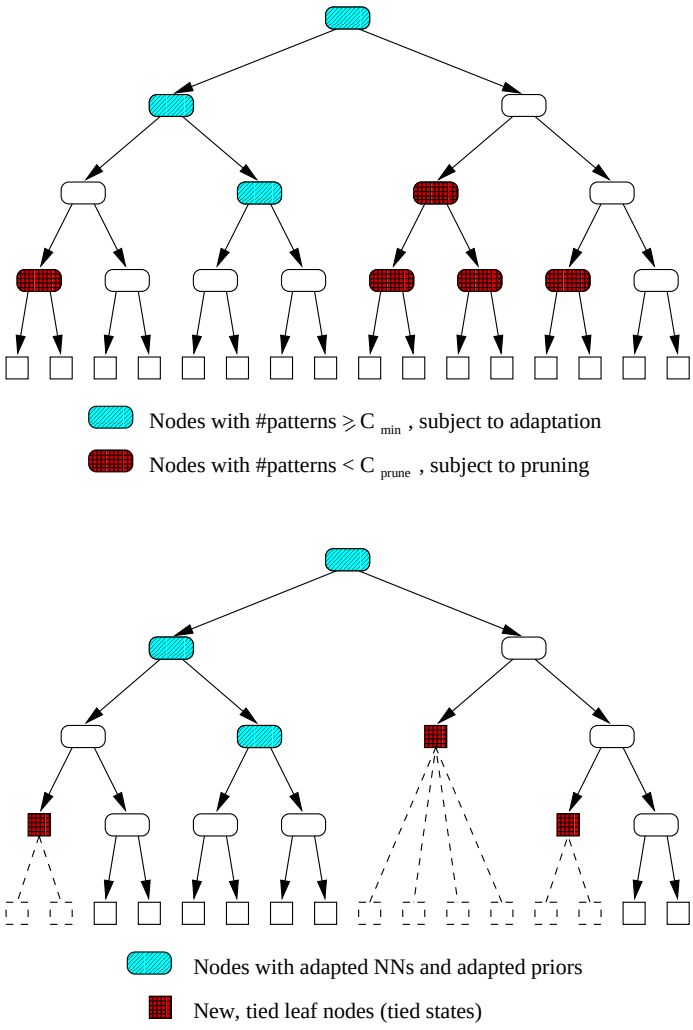

Figure 8.2: Structural domain adaptation of hierarchical connectionist acoustic models

tree. Note however that the SDA algorithm imposes no restrictions on the branching factor and the balance of the adapted model's tree structure.

Fig. 8.3 summarizes the SDA algorithm in compact form. Optimal values for the two parameters C_{min} and C_{prune} of the algorithm must be determined empirically on the available adaptation data. Increasing C_{prune} beyond the value that results in optimal recognition performance on the target domain allows for controlled downsizing of the hierarchical connectionist acoustic model, trading off recognition accuracy against memory and computational requirements.

Structural Domain Adaptation Algorithm

1. **Counting:** Compute state alignments of the adaptation data. For each HMM state (tree leaf node), count the allotted number of adaptation samples and propagate these counts up through the tree structure, thereby computing pattern counts C_i for each tree node.
2. **Acoustic Adaptation:** Select nodes subject to adaptation by checking whether $C_i \geq C_{min}$ using a pre-defined adaptation threshold C_{min}. Adapt the local estimators for conditional posterior and prior probabilities in the selected tree nodes.
3. **Structural Adaptation:** Select nodes subject to pruning by checking whether $C_i \leq C_{prune}$ using a pre-defined pruning threshold C_{prune} with $C_{prune} < C_{min}$. Remove the selected tree nodes and tie all dangling HMM states in a pruned subtree to a newly created leaf node that replaces the root of the pruned subtree.

Figure 8.3: Algorithm for structural domain adaptation of hierarchical connectionist acoustic models

8.4 Domain Adaptation Experiments

In the following, we experimentally evaluate the SDA algorithm by adapting a Switchboard recognizer to the two previously mentioned target domains. In these experiments, we focus on structural domain adaptation of the hierarchical connectionist acoustic model, assuming that domain-specific vocabularies and language models are available.

8.4.1 Baseline Switchboard Recognizer

Our baseline recognizer is built and trained on 170 hours of Switchboard conversational American English telephone speech. This training corpus is covered by roughly 30000 distinct words. The recognizer's training dictionary contains 64000 pronunciations for these 30000 words. Based on the training data and the pronunciation dictionary, a cross-word context-dependent (pentaphone) hierarchical connectionist acoustic model with a total of 24000 tied states distributed among 56 3-state left-right phonetic HMMs was constructed and trained. See Appendix B for details on the distribution of tied states among phone models. The large number of 24000 tied HMM states was chosen in order to obtain a very detailed and over-specific (in terms of phonetic context modeling) baseline model suitable for subsequent structural adaptation.

On a subset of the official Switchboard 1996 evaluation test set, the Switchboard recognizer based on the connectionist 24k HNN acoustic model achieves a word error rate of 33.3%.

8.4.2 Selection of Nodes for Adaptation and Pruning

In a first experiment, we quantify the effects of adaptation threshold C_{min} and pruning threshold C_{prune} on the number of nodes selected for adaptation and pruning, respectively. For this purpose, we consider an equal amount of adaptation data (45 min) from both target domains. For both target domains, we first compute HMM state counts and HNN tree node counts on Viterbi alignments of the available adaptation data. With respect to these counts, we compute the number of tree nodes subject to adaptation for different values of C_{min}. Fig. 8.4 shows the resulting curves on the WSJ and ESST domains (note the logarithmic scale on the abscissa). For example, approximately 100 tree nodes are selected for adaptation when using a value of $C_{min} = 1000$ frames.

Fig. 8.5 shows for different pruning thresholds C_{prune} how many tree nodes are removed in the pruning step of structural domain adaptation. Interestingly, these curves do *not* start at 0% for $C_{prune} = 0$, indicating that there are significant amounts of unobserved HMM states in both target domains for the given adaptation data. The considerably larger amount of tree nodes pruned in the case of ESST data can be attributed to the smaller phonetic variety resulting from the smaller vocabulary in that domain. Furthermore, although an equal amount of adaptation data was used for computing the curves in Fig. 8.4 and Fig. 8.5, the WSJ data yields more nodes for acoustic adaptation and less nodes for pruning, indicating a more uniform distribution of HMM states than in the case of the ESST data.

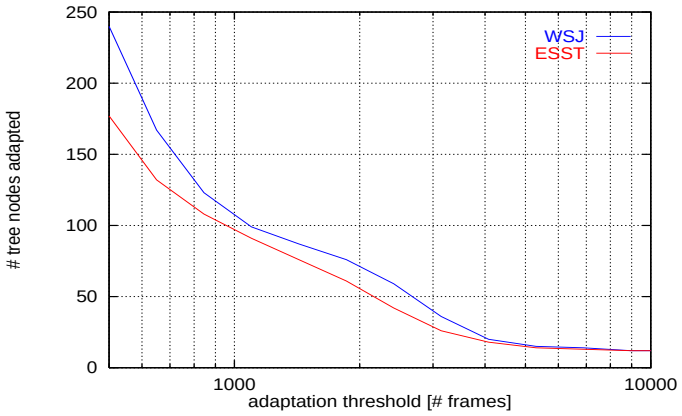

Figure 8.4: Structural domain adaptation – adaptation step

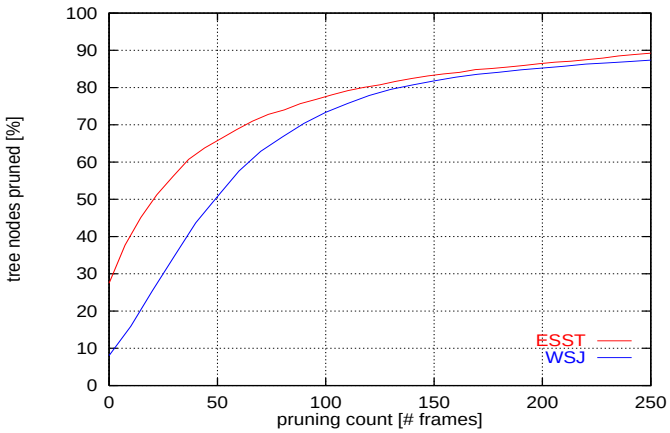

Figure 8.5: Structural domain adaptation – pruning step

8.4.3 Adapting to WSJ Domain

We now take a closer look on structural domain adaptation on the Wall Street Journal (WSJ) target domain. In our experiments, we were using the WSJ'93 Spoke 6 subset, consisting of read Wall Street Journal articles that were recorded through a telephone headset. The domain specific vocabulary is three times smaller than the vocabulary of the baseline Switchboard recognizer and consists of 5000 words. The WSJ'93 Spoke 6 subset defines separate adaptation and test sets that were adopted for our experiments as they allow us to compare our results with the official evaluation results. The following Table 8.5 provides details about these adaptation and test sets. Note that a maximum of 45 minutes of transcribed adaptation data is available for structural domain adaptation on the WSJ domain.

# adaptation speakers	10
amount of adaptation data	45.5 min
# adaptation frames	273500
# test set speakers	10
amount of test data	27 min
# words in test set	3865

Table 8.5: Adaptation and test sets for SDA on WSJ domain

The coverage of the original 24000 tied HMM states on the 45 minutes of WSJ adaptation data amounts to 93.3%. Using a domain specific vocabulary and language model and after several iterations of (1) optimization of the speaker dependent VTLN warping factors based on the first hypothesis transcripts and (2) normalization of the loudness of the input waveforms and (3) adjustment of the language model weight and the word insertion penalty for optimal performance on a held out development set consisting of parts of the adaptation data, we achieve a baseline word error rate of 14.4% on the WSJ test set with the otherwise unaltered acoustic model trained on the Switchboard corpus. Note that this tuning phase already yields a significant improvement in the word error rate.
We then applied the SDA algorithm for three different adaptation thresholds C_{min} and varying pruning thresholds ($C_{prune} \in \{0, 20, 40, 80, 160, \ldots\}$). Fig. 8.6 shows the resulting word error rates in relation to the remaining number of HNN leaf nodes (equivalent to the number of unique HMM states). For each of the three curves (corresponding to the three different values of C_{min} investigated) in Fig. 8.6, Table 8.6 gives details about the performance and size of the resulting structurally adapted HNN tree for the pruning threshold that yields the optimal word error rate.
The best overall result of 12.0% word error rate is achieved for $C_{min} = 500$ and

$C_{prune} = 40$. Although tree pruning yields only minor improvements in terms of the word error rate[2], it is very effective in reducing the size of the original model. For the optimal settings, SDA prunes the original HNN tree to 65% of its original size and improves performance by 16.7%. Using a larger value for the pruning threshold, the size of the HNN tree can be further decreased to only about 20% of its original size (4585 leaf nodes) with only a slight increase in word error rate to 12.8%.

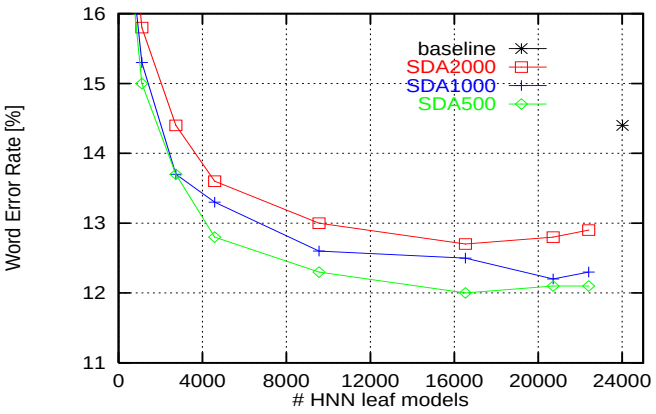

Figure 8.6: Structural domain adaptation on the WSJ domain

acoustic model	# adapted nodes	# unique HMM states	HNN tree size	word error rate	relative gain
baseline	–	24016	100%	14.4%	–
SDA2000	72	16532	65%	12.7%	11.8%
SDA1000	101	20706	84%	12.2%	15.3%
SDA500	240	16532	65%	12.0%	16.7%

Table 8.6: Results for optimal structural domain adaptation on WSJ

We compare these results to the best official evaluation result of 12.5% word error rate, achieved by a domain-specific (trained on 62 hours of band-limited WSJ0 and

[2]The rightmost point on the curves in Fig. 8.6 correspond to the absence of explicit pruning with $C_{prune} = 0$. From there, the pruning threshold increases towards the left of the plot.

WSJ1 data) and telephone-adapted recognizer on the same WSJ'93 Spoke 6 evaluation test set [ARP '94]. Clearly, the SDA approach allows to match domain-specific performance with an out-of-domain hierarchical connectionist acoustic model and only 45 minutes of acoustic adaptation data. Table 8.7 summarizes all results during the various phases from the original Switchboard system to the final structurally adapted system for the WSJ domain.

condition	# HMM states	word error rate
SWB recognizer	24016	45.4%
+ domain-specific vocab/LM	24016	17.2%
+ tuning on development set	24016	14.4%
+ optimal performance SDA	16532	12.0%
+ minimum tree SDA	4585	12.8%
WSJ recognizer	N/A	12.5%

Table 8.7: Summary of domain adaptation on WSJ

Most importantly, the SDA algorithm not only yields domain-specific recognition performance but furthermore allows for a significant decrease in the specificity of phonetic context modeling which results in a considerably smaller acoustic model.

8.4.4 Adapting to ESST Domain

In a second set of experiments, we used the SDA algorithm to adapt the Switchboard recognizer to the ESST domain. In contrast to the WSJ'93 Spoke 6 data, the entire ESST corpus is collected in 16 kHz / 16 bit using high-quality Sennheiser microphones. We therefore had to downsample the data to 8 kHz before feeding it into the Switchboard recognizer. Although ESST consists of spontaneous human-to-human dialogs, it is considerably different from the Switchboard domain in many respects. The vocabulary consists of 2850 unique words which is only 19% of the size of the Switchboard vocabulary. In contrast to Switchboard, general articulation is much clearer and there are hardly any conversational phenomena such as false starts, interjections and laughter. Finally, there is only a single topic in ESST (the scheduling of meetings) which restricts linguistic and phonetic variability considerably. We therefore expect that the ESST domain requires significantly less specificity in phonetic context modeling than is realized in our 24k states Switchboard recognizer.
For the following domain adaptation experiments, we have compiled an adaptation set of 62 minutes of speech and a test set of 18 minutes of speech. Table 8.8 gives an overview of these two data sets. Using the transcriptions available for the adaptation data, we first computed Viterbi state alignments with the original 24000 tied states

HNN model. In these alignments, the coverage of the 24000 states amounts to 74.9%. In other words, one forth of all HMM states were not observed in the 62 minutes of adaptation data.

# adaptation speakers	18
amount of adaptation data	62 min
# adaptation frames	372100
# testset speakers	14
amount of test data	18 min
# words in test set	3309

Table 8.8: Adaptation and test sets for SDA on ESST domain

In contrast to the WSJ domain, there never was an official evaluation on the ESST domain as it is an internal CMU-collected domain. However, for comparison we had a recognizer available that was built and optimized specifically for the ESST domain. On the above test set, this domain-specific mixtures of Gaussians based recognizer achieves a word error rate of 19.5% and models only 1150 tied HMM states.
In comparison, using a domain specific vocabulary and language model and the same kind of optimizations of acoustic parameters and decoding parameters already applied in the case of WSJ, we achieve a baseline word error rate of 25.5% on the ESST test set with the unaltered 24k acoustic model trained on Switchboard. Thus, the performance of the raw Switchboard acoustic model is 30% worse than that of a domain-specific one. In addition, the Switchboard model is over-sized and over-specific as it models 20 times more allophonic variations (HMM states).
We then used the ESST adaptation data to apply the structural domain adaptation algorithm to the Switchboard HNN tree. As in the case of WSJ, we applied three different adaptation thresholds ($C_{min} \in \{500, 1000, 2000\}$) and varying pruning thresholds ($C_{prune} \in \{0, 20, 40, 80, 160, \ldots\}$). Fig. 8.7 shows the resulting word error rates in relation to the remaining number of HNN leaf nodes.
Again, tree pruning in addition to acoustic adaptation yields only minor improvements in terms of the word error rate but is very effective in reducing the size of the original model, as already observed on the WSJ domain. However, there are two remarkable differences to the curves obtained on the WSJ domain. First, optimal performance is obtained for the largest instead of for the smallest value of C_{min}, meaning that it is advantageous to adapt less tree nodes on ESST than on WSJ. Secondly, the ESST domain allows for much larger pruning thresholds and thereby a considerably smaller size of the HNN tree. In fact, the hierarchical connectionist acoustic model can be pruned to only about 13% of its original size before the word error rate starts to increase noticeably.

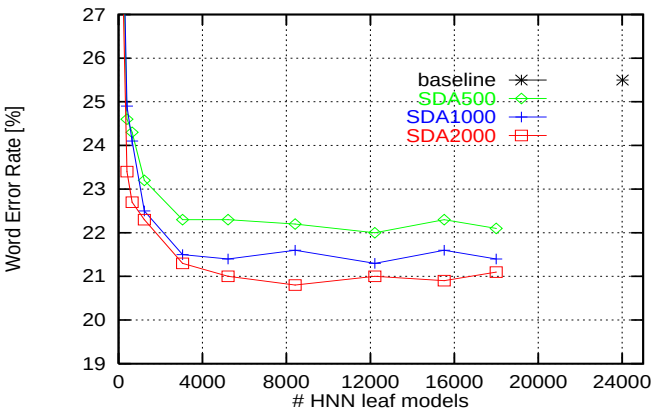

Figure 8.7: Structural domain adaptation on the ESST domain

For each of the three curves in Fig. 8.7, Table 8.9 gives details about the performance and size of the resulting structurally adapted HNN tree for the pruning threshold that yields the optimal word error rate.

acoustic model	# adapted nodes	# unique HMM states	HNN tree size	word error rate	relative gain
baseline	–	24016	100%	25.5%	–
SDA500	280	12210	49%	22.0%	13.7%
SDA1000	120	12210	49%	21.3%	16.5%
SDA2000	71	8411	34%	20.8%	18.4%

Table 8.9: Results for optimal structural domain adaptation on ESST

The best overall result of 20.8% word error rate is achieved with $C_{min} = 2000$ and $C_{prune} = 160$. For these settings, SDA prunes the original HNN tree to 34% of its original size and improves performance by 18.4%. As mentioned before, the size of the HNN tree can be further decreased to only about 13% of its original size with only a slight increase in word error rate to 21.3%. Table 8.10 summarizes all results during the various phases from the original Switchboard system to the final structurally adapted system for the ESST domain.
In comparison, the structurally adapted Switchboard system achieves performance

condition	# HMM states	word error rate
SWB recognizer	24016	55.3%
+ domain-specific vocab/LM	24016	28.3%
+ tuning on development set	24016	25.5%
+ optimal performance SDA	8411	20.8%
+ minimum tree SDA	3051	21.3%
ESST recognizer	1150	19.5%

Table 8.10: Summary of domain adaptation on ESST

close to a domain-specific ESST system which has been optimized extensively on the given test set. While the adaptation step of SDA yields most of the gain in recognition accuracy, the pruning step in addition allows to prune substantial parts of the original hierarchical connectionist acoustic model which (a) improves generalization by reducing the specificity of context modeling through state tying and (b) yields significant savings in memory requirements.

8.4.5 Comparison to Conventional Acoustic Adaptation

In a last series of experiments, we compare structural domain adaptation with conventional regression tree based MLLR adaptation of a mixtures of Gaussians acoustic model. In contrast to the SDA approach, regression tree based MLLR can only be used to acoustically adapt the existing set of context models. It does not allow to reduce the specificity of context modeling or the size of the acoustic model to reflect the differing requirements in a new target domain. By comparing the two adaptation approaches, we seek to determine whether structural adaptation in addition to acoustic adaptation as performed by the SDA algorithm is beneficial in terms of reduction of the word error rate.

For the following experiments with supervised MLLR adaptation, we were using a Switchboard-trained continuous-density mixtures of Gaussians acoustic model for the same 24000 tied HMM states used in the previous sections. The mixtures of Gaussians based model was taken from the Interactive Systems Labs 1997 Switchboard evaluation system (CMU-ISL) and achieves a baseline word error rate of 31.5% on the Switchboard test set.

Using domain-specific vocabularies and language models and some tuning of the decoding parameters, we obtain baseline word error rates of 13.3% (WSJ) and 24.8% (ESST) with the unadapted conventional acoustic model. We then used alignments of the available adaptation data to perform regression tree based MLLR adaptation of the means of the approximately 100000 Gaussian densities in the system.

To vary the number of adaptation transforms, we were experimenting with three different settings for C_{min} (2000, 4000 and 8000). Table 8.11 gives results obtained with the adapted models on the WSJ domain. The number of full rank linear adaptation transforms applied to the Gaussian means ranges from 99 down to 31.

C_{min}	# transforms	word error rate	relative gain
baseline	–	13.3%	–
8000	31	11.6%	12.8%
4000	51	11.7%	12.0%
2000	99	12.0%	9.8%

Table 8.11: Results with regression tree based MLLR adaptation on WSJ

Optimal performance was achieved using $C_{min} = 8000$ which yields a relative reduction in word error rate of 12.8% compared to the unadapted baseline. Table 8.12 gives results obtained for the same set of experiments on the ESST data.

C_{min}	# transforms	word error rate	relative gain
baseline	–	24.8%	–
8000	35	22.2%	10.5%
4000	63	22.0%	11.3%
2000	108	23.0%	7.3%

Table 8.12: Results with regression tree based MLLR adaptation on ESST

Again, three different adaptation thresholds were investigated. As the ESST adaptation set is a little larger than that of the WSJ domain, the number of adaptation transforms obtained increased slightly. The best result was achieved for $C_{min} = 4000$ which yields a relative reduction in word error rate of 11.3%.
The following Table 8.13 summarizes and compares the performance improvements obtained with the SDA and MLLR approaches. Our results show that MLLR based adaptation yields smaller relative improvements in word error rate, particularly in the case of adapting a Switchboard model to the ESST domain.
In contrast to MLLR based adaptation of a conventional architecture, the SDA approach not only compensates for mismatches in acoustic space but furthermore adapts to differing specificity of phonetic context in unseen domains by adapting node priors and by pruning defective parts in the modeling hierarchy. This way, differences in the a-priori probability of HMM states can be compensated and the resolution of context modeling can be adapted to the specific requirements in the target domain. As an

type of acoustic model	**adaptation approach**	**relative gain on**	
		WSJ domain	**ESST domain**
Hierarchy of Neural Networks	SDA	16.7%	18.4%
Mixtures of Gaussians	MLLR	12.8%	11.3%

Table 8.13: Comparison of relative gains obtained with SDA vs. with MLLR

important side effect, the SDA algorithm allows to downsize a hierarchical connectionist acoustic model and thereby reduce the memory requirements and decoding time substantially.

Chapter 9

Mixture Trees

In previous chapters, we have seen how a large vocabulary conversational speech recognition system benefits from a hierarchically organized connectionist acoustic model. By adopting the connectionist framework of estimating state posteriors instead of state likelihoods, we were able to apply hierarchical factoring to obtain a tree structured estimator with advantageous scaling properties. In this chapter, we present *mixture trees*, a different but related tree structured architecture for acoustic modeling. We demonstrate that most, but not all of the properties of a hierarchical connectionist acoustic model can also be obtained with this likelihood based model.

9.1 Hierarchically Tied Mixture Densities

We consider the task of estimating HMM state observation likelihoods for a set of N decision tree clustered states s_i using mixture densities. In a conventional continuous-density HMM setting, we model each state *independently* according to

$$p_i(\mathbf{x}) = p(\mathbf{x}|s_i) = \sum_{k=0}^{K_i-1} c_i^{(k)}\, q_i^{(k)}(\mathbf{x}) \qquad \forall i \in \{1, \ldots, N\}$$

where the $c_i^{(k)}$ are (affine) mixture weights satisfying $\sum_{k=0}^{K_i-1} c_i^{(k)} = 1$ and $c_i^{(k)} \geq 0$, and the $q_i^{(k)}(\mathbf{x})$ are mixture component densities in the space of feature vectors $\mathbf{x}$. Mixture densities are usually preferred over simple densities because of their universal approximation property.

Mixture trees are motivated both by the observation that individual mixture densities of context-dependent speech models overlap considerably in feature space and by the desire for a tree structured acoustic model with all the advantageous properties that we have discussed in previous chapters. Thus, instead of using separate sets of component densities for each mixture density, we share some of the component

densities to allow for joint modeling of the overlapping parts of the distributions. For instance, consider the set of 4 mixture densities depicted in Fig. 9.1.

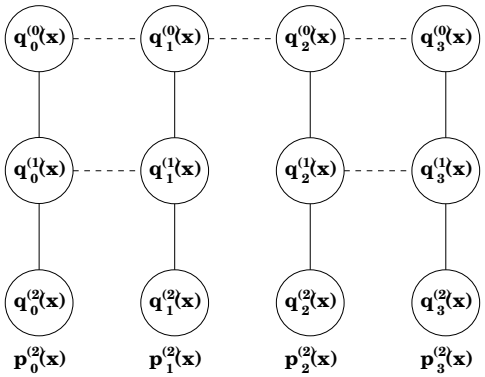

Figure 9.1: Hierarchically tying mixture densities

Each one of the densities consists of 3 vertically organized component densities. Instead of assigning 3 component densities exclusively to each mixture, we share some of them between adjacent mixtures such that the bottom level components are used exclusively, the center level components are shared between two and the top level components are shared between all four densities. However, hierarchically sharing component densities by itself does not yet yield a truly hierarchical model. We also have to find a hierarchical representation of the mixture weights that allows to represent and evaluate hierarchically tied mixture densities in a tree structure. For that purpose, we introduce (shared) mixture interpolation weights at each vertical arc that connect component densities. We call the resulting tree structured configuration a *mixture tree* [Fritsch '99b, Fritsch '99a] (Fig. 9.2).
Introducing depth d and branching factor b of a mixture tree, we rewrite the state observation likelihoods, now being estimated by the leaves of the mixture tree, as $p_i(\mathbf{x}) = p_i^{(d)}(\mathbf{x})$ and recursively define the mixture model as

$$\begin{aligned} p_0^{(0)}(\mathbf{x}) &= q_0^{(0)}(\mathbf{x}) \\ p_i^{(k)}(\mathbf{x}) &= \alpha_i^{(k)} q_i^{(k)}(\mathbf{x}) \; + \; (1 - \alpha_i^{(k)})\, p_{\lfloor i/b \rfloor}^{(k-1)}(\mathbf{x}) \end{aligned}$$

where the $q_i^{(k)}$ are (tied) component densities and the $\alpha_i^{(k)}$ are local interpolation weights, satisfying $0 \leq \alpha_i^{(k)} \leq 1$, such that the p_i are valid probability densities. An

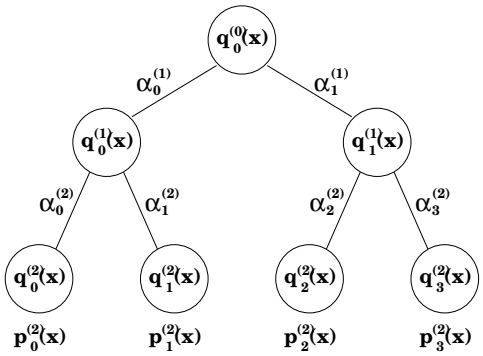

Figure 9.2: Mixture tree ($d = 2, b = 2$)

individual mixture density represented in the tree is evaluated top-down, starting at the root node. Proceeding down the tree towards the corresponding leaf node, we incrementally refine the current estimate by computing affine interpolations between the already accumulated partial mixture probability and the current local component density using interpolation weights $\alpha_i^{(k)}$. For $\alpha_i^{(k)} = 0$, the local component density does not contribute at all and the mixture likelihood up to that point is determined by all predecessors. In effect, setting $\alpha_i^{(k)} = 0$ allows to skip a component density. In contrast, setting $\alpha_i^{(k)} = 1$ corresponds to neglecting the partial mixture probability accumulated through all the predecessor nodes. Of course, such an extreme behavior is not desired as it renders the tree structure ineffective and is not expected to happen unless there is absolutely no overlap between the modeled distributions.
Hierarchically tied mixture densities, as computed recursively by a mixture tree, can be interpreted as conventional mixture densities

$$p_i(\mathbf{x}) = p(\mathbf{x}|s_i) = \sum_{k=0}^{K_i-1} c_i^{(k)}\, q_i^{(k)}(\mathbf{x}) \qquad \forall i \in \{1, \ldots, N\}$$

with tied components $q_i^{(k)}$ and component mixture weights $c_i^{(k)}$ that are computed from the affine interpolation weights according to

$$c_i^{(k)} = \alpha_i^{(k)} \prod_{j=k+1}^{K_i-1} (1 - \alpha_i^{(j)}) \qquad \forall k \in \{0, \ldots, K_i - 1\}$$

where $\alpha_i^{(0)} \equiv 1$ and K_i is the number of component densities for mixture p_i, equivalent to the depth of the corresponding leaf node in the mixture tree. Thus, the recursive

interpolation scheme that we have proposed for mixture trees can be justified by the correspondence to a set of constrained conventional mixture densities.
Also, an interesting and important aspect of mixture trees is that any node (not just the leaf nodes) computes a valid probability density which depends only on predecessor nodes. This property allows to downsize the mixture tree without having to re-estimate any of the parameters as is required with conventional acoustic models (e.g., [Hwang & Huang '98]). In fact, the partial mixture probability computed down to a specific node represents the probability of the feature vector being generated by any of the leaf nodes (states) in the corresponding subtree. This implies that the root component density $q_0^{(0)}(\mathbf{x})$ models the unconditional density $p(\mathbf{x})$ of the data.

9.2 Parameter Estimation

Assuming a maximum likelihood framework, the parameters of a mixture model have to be estimated iteratively using an Expectation-Maximization (EM) algorithm [Dempster et al. '77, Redner & Walker '84, McLachlan & Krishnan '97]. Furthermore, if the Forward-Backward algorithm is used for training the HMMs, we face two nested probabilistic models; (1) assigning HMM states to observations and (2) assigning mixture component densities within state mixtures to observations.
In the following, we will derive an EM algorithm for estimating the parameters of a mixture tree that is applied to the estimation of HMM state emission likelihoods in a statistical speech recognition system. Without sacrificing universality, we first present the EM algorithm for the case of Gaussian component densities. The resulting algorithm can easily be modified for other types of component densities, even for component densities that are mixtures themselves. We discuss both Forward-Backward and Viterbi based HMM training.
Irrespective of the kind of HMM training algorithm chosen, the E-step (Expectation) of the EM algorithm for mixture trees is identical and consists of computing posterior probabilities of mixture tree nodes for each input feature vector $\mathbf{x}$. For that purpose, it is crucial to note that the interpolation weights $\alpha_i^{(k)}$ represent the a-priori node probabilities in the tree. Thus, for each feature vector $\mathbf{x}$ in the training set, we can compute the a-posteriori node probabilities $h_i^{(k)}(\mathbf{x})$ according to

$$h_i^{(k)}(\mathbf{x}) = \frac{\alpha_i^{(k)} q_i^{(k)}(\mathbf{x})}{\alpha_i^{(k)} q_i^{(k)}(\mathbf{x}) \; + \; (1-\alpha_i^{(k)})\, p_{[i/b]}^{(k-1)}(\mathbf{x})} = \frac{\alpha_i^{(k)} q_i^{(k)}(\mathbf{x})}{p_i^{(k)}(\mathbf{x})}$$

with the exception of $h_0^{(0)}(\mathbf{x}) = 1$. The $h_i^{(k)}(\mathbf{x})$ measure the probability with which the respective node's component density contributes to the partial mixture that has been accumulated down to that node. Again, it is important to note that the node posteriors in our model depend only on parent nodes, not on any of the child nodes.

For the M-step (Maximization) of the algorithm, we have to distinguish between Forward-Backward and Viterbi Training.

9.2.1 Forward-Backward Training

In Forward-Backward HMM training, we obtain HMM state occupation probabilities $\gamma_i(\mathbf{x})$ for each HMM state s_i and each feature vector $\mathbf{x}$. In our case, state occupation probabilities translate to mixture tree leaf occupation probabilities and represent the a-priori probabilities of leaf nodes.
In the M-step, we update the mixture weights $\alpha_i^{(k)}$ and the parameters of the component densities $q_i^{(k)}$ based on the expectations for all training patterns gained in the E-step such that the likelihood of the model given the data increases. We obtain the following updates for the node parameters:

$$\begin{aligned}
\alpha_i^{(k)} &= \frac{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)\; h_i^{(k)}(\mathbf{x}_m)}{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)} \\
\mu_i^{(k)} &= \frac{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)\; h_i^{(k)}(\mathbf{x}_m)\; \mathbf{x}_m}{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)\; h_i^{(k)}(\mathbf{x}_m)} \\
\mathbf{\Sigma}_i^{(k)} &= \frac{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)\; h_i^{(k)}(\mathbf{x}_m)\; \mathbf{x}_m \mathbf{x}_m^t}{\sum_{m=1}^{M} \gamma_i^{(k)}(\mathbf{x}_m)\; h_i^{(k)}(\mathbf{x}_m)}
\end{aligned}$$

where $\mu_i^{(k)}$ is the mean vector and $\mathbf{\Sigma}_i^{(k)}$ is the covariance matrix of the Gaussian component density $q_i^{(k)}$. The node occupation probabilities $\gamma_i^{(k)}(\mathbf{x})$ can be computed in a bottom-up fashion from the state/leaf occupation probabilities $\gamma_i(\mathbf{x})$ according to

$$\begin{aligned}
\gamma_i^{(d)}(\mathbf{x}) &= \gamma_i(\mathbf{x}) \\
\gamma_i^{(k)}(\mathbf{x}) &= \sum_{j=bi}^{b(i+1)-1} \gamma_j^{(k+1)}(\mathbf{x})
\end{aligned}$$

That is, a node occupation probability is computed as the sum of all state occupation probabilities of all states (leaf nodes) in the corresponding subtree.

9.2.2 Viterbi Training

In the case of Viterbi training, a state alignment implies a one-to-one mapping between HMM states and feature vectors. Thus, for any input feature vector, there is exactly one state with state occupation probability $\gamma_i(\mathbf{x}) = 1$, all other state occupation probabilities vanish. In a mixture tree, the Viterbi assumption leads to a

single path of non-zero node occupation probabilities $\gamma_i^{(k)}(\mathbf{x}) = 1$ from root to one of the leaves for each feature vector. Therefore, node posteriors in the E-step have to be evaluated only along the path through the mixture tree that corresponds to the current pair of feature vector and HMM state. The parameter update formulas in the M-step simplify accordingly to

$$\begin{aligned}
\alpha_i^{(k)} &= \frac{\sum_{x_m \in \mathcal{T}_i^{(k)}} h_i^{(k)}(\mathbf{x}_m)}{\sum_{x_m \in \mathcal{T}_i^{(k)}} 1} \\
\mu_i^{(k)} &= \frac{\sum_{x_m \in \mathcal{T}_i^{(k)}} h_i^{(k)}(\mathbf{x}_m)\ \mathbf{x}_m}{\sum_{x_m \in \mathcal{T}_i^{(k)}} h_i^{(k)}(\mathbf{x}_m)} \\
\mathbf{\Sigma}_i^{(k)} &= \frac{\sum_{x_m \in \mathcal{T}_i^{(k)}} h_i^{(k)}(\mathbf{x}_m)\ \mathbf{x}_m \mathbf{x}_m^t}{\sum_{x_m \in \mathcal{T}_i^{(k)}} h_i^{(k)}(\mathbf{x}_m)}
\end{aligned}$$

where $\mathcal{T}_i^{(k)}$ denotes the set of training patterns that correspond to the respective tree node. In other words, $\mathcal{T}_i^{(k)}$ consists of the feature vectors with state labels corresponding to one of the leaf nodes in the subtree starting at node $N_i^{(k)}$.

9.2.3 Parameter Initialization

As with standard mixture densities, reasonable initialization of parameters is crucial for rapid convergence of the EM algorithm. In the case of hierarchically tied mixture densities with Gaussian component densities, we initialize local mixture weights according to

$$\alpha_i^{(k)} = \frac{1}{k+1}$$

which corresponds to a uniform component prior distribution for all mixtures in the mixture tree as can be seen by substituting the above expression into the expression for computing mixture weights $c_i^{(k)}$ from interpolation weights $\alpha_i^{(k)}$ in a mixture tree:

$$\begin{aligned}
c_i^{(k)} &= \alpha_i^{(k)} \prod_{j=k+1}^{K_i-1} (1 - \alpha_i^{(j)}) \qquad \forall k \in \{0, \ldots, K_i - 1\} \\
&= \frac{1}{k+1} \prod_{j=k+1}^{K_i-1} (1 - \frac{1}{j+1}) \\
&= \frac{1}{k+1} \prod_{j=k+1}^{K_i-1} \frac{j}{j+1}
\end{aligned}$$

$$= \frac{1}{k+1}\,\frac{k+1}{k+2}\,\cdots\,\frac{K_i-2}{K_i-1}\,\frac{K_i-1}{K_i}$$
$$= \frac{1}{K_i}$$

Individual Gaussian component densities are initialized using the ML estimates for the Forward-Backward weighted data observed at the corresponding tree node:

$$\mu_i^{(k)} = \frac{\sum_{m=1}^{M}\gamma_i^{(k)}(\mathbf{x}_m)\,\mathbf{x}_m}{\sum_{m=1}^{M}\gamma_i^{(k)}(\mathbf{x}_m)}$$
$$\mathbf{\Sigma}_i^{(k)} = \frac{\sum_{m=1}^{M}\gamma_i^{(k)}(\mathbf{x}_m)\,\mathbf{x}_m\mathbf{x}_m^t}{\sum_{m=1}^{M}\gamma_i^{(k)}(\mathbf{x}_m)}$$

In case of Viterbi training, this simplifies to the ML estimates for all data of all states (leaf nodes) found in the subtree of the node to be initialized.

9.2.4 Mixtures as Component Densities

Mixture component densities $q_i^{(k)}$ in a mixture tree can themselves be mixture densities, allowing for more accurate modeling of node distributions. For instance, each node's component density may be modeled as a mixture of Gaussians:

$$q_i^{(k)} = \sum_{j=0}^{M-1} c_j N_j(\mathbf{x}) = \sum_{j=0}^{M-1} \frac{c_j}{\sqrt{(2\pi)^D|\Sigma_j|}}\exp\{-\frac{1}{2}(\mathbf{x}-\mu_j)^t\Sigma_j^{-1}(\mathbf{x}-\mu_j)\}$$

The easiest way to train a mixture tree with this kind of component densities is to start with component densities that consist of single Gaussians, train the mixture tree until convergence of the EM algorithm and then apply a technique called mixture splitting [Young '94], that replaces the single Gaussians with mixtures of M Gaussians randomly positioned around the mean of the original Gaussian according to its variance. After increasing the number of components in the mixture densities of the mixture tree using this technique, we can continue to train the parameters according to the EM algorithm, however now considering the fact that we face two nested probabilistic models which requires to weight updates for the within-node component densities with the node posteriors obtained in the E-step of the EM algorithm for mixture trees.

9.3 Constructing Mixture Trees

As already mentioned, an important prerequisite for the successful application of mixture trees is that state distributions overlap in feature space, such that mixture

density components can be tied for joint modeling of the overlapping parts. While this is usually the case for context-dependent HMM models, it is not immediately clear how to group these models to form an optimal mixture tree structure. Considering the large amount of HMM states and our experience gained with hierarchical connectionist acoustic models, cluster algorithms appear to be viable solutions to the task of constructing suitable mixture trees. Also, it is in principle possible to adopt the structure of the phonetic context modeling decision trees for mixture tree modeling as already noted in section 5.2. However, due to the categorical questions asked, decision tree structures are typically highly imbalanced, a rather undesirable property of mixture trees. In fact, the following issues are particularly important aspects of mixture trees:

- **Tree Balance:** A mixture tree should be balanced to ensure that all embedded mixture densities consist of an approximately equal number of component densities. Otherwise, we allow certain HMM states to be modeled more accurately than others.

- **Branching Factor:** The number of component densities that constitute the mixture modeling a given state is determined by the depth of the corresponding tree leaf node. Assuming a balanced mixture tree, the average number of components per mixture is determined solely by the tree branching factor[1]. A binary tree maximizes this number. Furthermore, the branching factor controls the degree of component sharing and might therefore be subject to data-driven optimization.

Keeping the above constraints in mind, the agglomerative and divisive clustering algorithms that we have presented and discussed in section 5.3 are applicable to the task of mixture tree construction without modifications.

9.4 Exploiting Tree Structure

The hierarchical structure of mixture tree based acoustic models offers almost the same advantageous properties than those of hierarchical connectionist acoustic models. With the exception of techniques that require estimates of (partial) posterior probabilities such as fast evaluation by pruning against a fixed posterior threshold, the algorithms that we have developed for the connectionist hierarchical model are applicable to mixture trees as well:

[1] Assuming that the number of leaf nodes/HMM states is fixed and given a-priori

- **Structural Adaptation:** A speech recognition system based on a very detailed, highly specific mixture tree can efficiently and effectively be ported to a previously unseen domain. The differing vocabulary and language model in this domain typically induce a strong mismatch in the a-priori distribution of HMM states and in the specificity of context modeling required for optimal performance as discussed in chapter 8. Using a small amount of adaptation data for estimating a-priori state distributions in the new domain, we can identify low probability states that model phonetic contexts that are irrelevant to the new domain and prune the corresponding branches in the mixture tree. The resulting tree is structurally adapted to the unseen domain and its local estimators can additionally be adapted to the differing acoustic distributions by means of adaptation algorithms such as maximum likelihood linear regression (MLLR) [Leggetter & Woodland '94]. However, in contrast to the connectionist counterpart, branch priors that play an important role in adapting the tree structure are not explicitly available in mixture trees. Therefore, structural adaptation of mixture trees is somewhat limited in comparison to the hierarchical connectionist model.

- **Speaker Adaptation:** As discussed earlier, the structure of mixture trees represents exactly the kind of information needed for tying component densities for MLLR based adaptation when only a limited amount of adaptation data is available. As such, mixture trees can be interpreted as MLLR regression trees that have to be constructed additionally for conventional mixture models. As component densities are already tied in a mixture tree, speaker adaptation with a limited amount of adaptation data can be accomplished by simply selecting those tree nodes in the vicinity of the root node for which we observe more than a predefined amount of data and applying the usual linear transformation to the parameters of the corresponding component densities.

- **Downsizing the Tree:** In order to compute the likelihood of a specific state by means of a mixture tree, we have to follow the path from the root node to the leaf corresponding to that state, refining estimates of the likelihood at each node. Instead of traversing the tree all the way down to the leaves, we can stop computing refined likelihoods at any tree level and treat all states in the remaining subtree as a new tied state. This way, the specificity of context-dependent modeling and the number of distinctly modeled HMM states of a trained mixture tree can be reduced arbitrarily, from full context-dependent modeling all the way down to context-independent modeling and further. Thus, pruning of mixture trees allows to easily adapt recognizers to available memory and/or processor speed without having to re-train or re-cluster the acoustic model.

- **Fast Model Evaluation:** In contrast to hierarchical connectionist acoustic models, fast model evaluation can not be achieved by pruning partial scores against a fixed absolute threshold as this requires estimates of the partial posterior probabilities. However, fast evaluation of mixture trees can be achieved by evaluating the tree structure in a breadth first manner. This way, pruning can be delayed until partial scores are available for all nodes in a specific tree layer. As pruning can not be based on an absolute threshold, we may identify promising branches by rank ordering and selecting the n best nodes in each tree layer for further evaluation.

9.5 Experiments on Switchboard

For our experiments with a mixture tree based acoustic model on the Switchboard corpus, we constructed a context-dependent HMM system with a total of 8000 tied states by building about 150 phonetic decision trees, one for each state of context-independent 3 state HMM phone models. Top-down decision tree clustering was based on split likelihood gain using diagonal Gaussians to model state distributions.

9.5.1 Construction and Evaluation of Mixture Trees

We applied divisive clustering to construct a binary mixture tree for the 8000 states. Non-uniform priors were penalized during tree construction in order to obtain a balanced tree. The final mixture tree had a maximum depth of 18. Simple diagonal Gaussians were chosen as component densities in each node. After initialization according to section 9.2.3, we trained the mixture tree for 4 iterations using Viterbi state alignments of 170 hours of Switchboard data from a conventional recognizer. To improve modeling accuracy, we then replaced the Gaussian component densities in each tree node by mixtures of 8 Gaussians that were obtained from the original Gaussian by mixing-up as explained earlier. The resulting mixture tree, containing a total of 127992 Gaussians in 15999 nodes, was trained for another 6 iterations, until training data likelihood converged.

Fig. 9.3 depicts the initial as well as mean and standard deviation of the final interpolation weights α in each level of the trained mixture tree. For increasing tree depth, interpolation weights get smaller consistent with the initialization strategy and eventually level off at a mean of around 0.3. Their variance increases slightly towards the bottom of the tree which might indicate saturation of the specificity of context-modeling in some branches of the tree.

Next, we evaluated the performance of the trained mixture tree in recognition experiments. All recognition runs used a 15k vocabulary and a standard trigram language

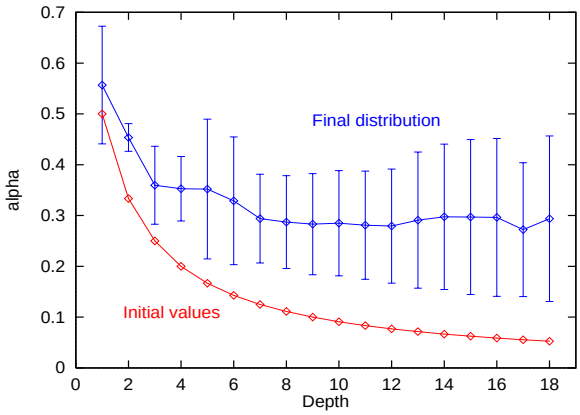

Figure 9.3: Distribution of interpolation weights in mixture tree

model trained on the Switchboard corpus. The results reported here were obtained on a test set consisting of the first 30 seconds from 12 representative speakers taken from the 1997 development test set and contained a total of 1340 words. Using the full mixture tree with mixtures of 8 Gaussians as component densities in each node, we achieved an unadapted word error rate (WER) of 36.6% on this test set.

Acoustic Model	# states	# components	# params	WER
Mixture Densities	8000	16/mixture	10.1 M	36.1 %
Binary Mixture Tree	8000	8/tree node	10.0 M	36.6 %

Table 9.1: Mixture tree vs. mixture densities based acoustic models

For comparison, a conventional model based on mixtures of Gaussians with an approximately equal number of parameters (using mixtures of 16 Gaussians for each of the 8000 states) achieves a comparable unadapted word error rate of 36.1% on the above test set. Table 9.1 summarizes the results of this comparison.

9.5.2 Downsizing of Mixture Trees

In this experiment, we investigated the effects of downsizing the mixture tree, thereby reducing the specificity and amount of context modeling. The original tree of depth

18 that models 8000 HMM states was successively reduced in size by removing the lowest tree level. This way, we obtained mixture trees with depths ranging from the original 18 down to 8. The smallest mixture tree with depth 8 lead to only 179 distinctly modeled states, corresponding roughly to the number of states in a context-independent system. We decoded the above test set for each pruned mixture tree, using the exact same decoder parameters as with the original mixture tree. Table 9.2 summarizes the results obtained with the original and the pruned trees regarding size, overall decoding speed and word error rate (WER) on the above test set.

pruning depth	# tied states	# nodes in tree	tree size [%]	speed [xRT]	WER [%]
–	8000	15999	100.0	48	36.6
17	7991	15983	99.8	48	36.6
16	7897	15795	98.7	48	36.6
15	7290	14581	91.1	45	36.7
14	5722	11445	71.5	39	37.2
13	3699	7399	46.2	31	39.4
12	2109	4219	26.3	24	40.6
11	1143	2287	14.2	19	43.8
10	619	1239	7.7	17	52.0
9	331	663	4.1	16	55.4
8	179	359	2.2	16	62.5

Table 9.2: Results with downsized mixture trees

While the speed-up in evaluating likelihoods (not shown in Table 9.2) that can be achieved by pruning the mixture tree corresponds roughly to the reduction in tree size, the speed-up for overall recognition time depends on the proportion of time spent in actual decoding which can significantly exceed the proportion of time spent in evaluation of acoustic likelihoods. The highest speed-ups can be expected for close to real-time systems. In our case, decoding with the smallest tree was about three times faster than decoding with the full tree.

Fig. 9.4 depicts a plot of word error rate vs. mixture tree size for the results summarized in Table 9.2. As expected, the performance for the smallest tree, modeling 179 distinct HMM states is comparable to what is typically reported for context-independent Switchboard systems. On the other hand, the mixture tree can be downsized to about 25% of its original size at the cost of only moderate increases in word error rate of about 4% absolute.

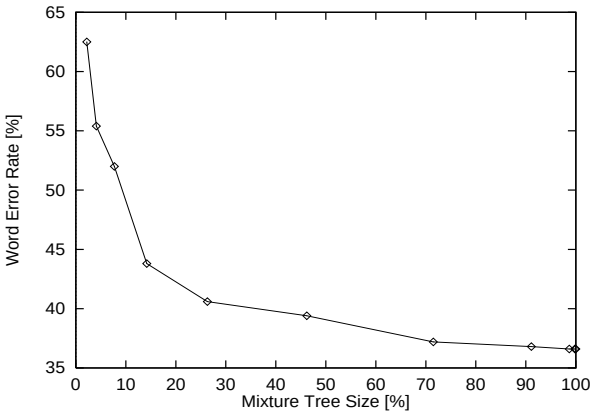

Figure 9.4: Word error rate vs. size of pruned mixture tree

9.6 Discussion

Mixture trees represent a likelihood-based alternative to hierarchical connectionist acoustic models with very similar structural properties, which can be exploited for a variety of tasks in speech recognition. As such, mixture trees offer the same advantages compared to conventional mixture models. However, although the two tree structured models share a lot of properties, they differ in the following aspects:

- **Estimation Paradigm:** While mixture trees directly estimate state conditional likelihoods as required by the HMM formalism, its connectionist counterpart separately estimates state posteriors and priors, which are then combined via Bayes' rule to form estimates of scaled likelihoods. The latter approach offers advantages with respect to the training criterion and the explicit availability of state priors and tree branch priors.

- **Training Criterion:** Mixture trees are trained according to the maximum likelihood (ML) criterion, using a variant of the EM algorithm. In contrast, a hierarchical connectionist acoustic model is trained according to the maximum a-posteriori (MAP) criterion which leads to more discriminant models.

- **Explicit Priors:** The explicit availability of priors in all levels of the tree structure allowed us to develop an algorithm for soft structural adaptation of

hierarchical connectionist acoustic models to unseen domains. By modifying the priors according to empiric distributions gained on adaptation data in the unseen domain, we were able to gradually pinch off certain undesired branches of the tree structure. As priors are not explicitly available in a likelihood based model, we can only decide upon keeping or removing such branches in a mixture tree model.

Thus, mixture trees represent an interesting tree-structured architecture for acoustic modeling with advantageous properties compared to conventional mixture models and a close relationship to the hierarchical connectionist model we have presented earlier. While both models offer scalable phonetic context modeling, the connectionist variant exhibits unique properties that render it more suitable for structural adaptation and domain-adaptive speech recognition.

Chapter 10

Acoustic Model Combination

In this chapter, we discuss methods for improving speech recognition performance by combining several information sources. It is well known that pattern classification tasks benefit from almost any kind of combination approach as long as sufficiently diverse representations or learners are involved. In the case of an automatic speech recognition system, there are three potential levels where combination may be introduced (see Fig. 10.1). First, we can combine the probability estimates of several acoustic models before they are fed into the decoder (referred to as acoustic model combination). Second, we can combine the probability estimates of several language models before they are fed into the decoder (referred to as language model combination) and third, we can run several complete and independent recognizers in parallel and combine their output word hypotheses (referred to as hypothesis combination). Of course, combination can occur simultaneously at several of these levels.

It is now common practice in evaluation systems to combine several language models estimated on different corpora by interpolating their probability estimates. Also, it has recently become popular to apply a word hypothesis combination scheme based on voting to evaluation systems, see [Fiscus '97] for details. Combination at the level of acoustic models is less commonly applied but appears to yield considerable gains for connectionist acoustic models, for instance in the Meta-Pi framework [Hampshire & Waibel '89] or when applying the mixtures of experts model [Jacobs '95] to homogeneous [Waterhouse & Cook '96, Cook et al. '97b] and heterogeneous [Fritsch & Finke '97] acoustic models. Furthermore, a variant of boosting [Cook & Robinson '96] appears to yield the greatest gains in combining several connectionist acoustic models but is computationally very demanding during training.

In our work, we have focused on the level of acoustic modeling and developed methods for the combination of heterogeneous acoustic models [Fritsch & Finke '97], e.g., conventional Gaussian mixture models and locally discriminative models such as the hierarchical connectionist acoustic model presented earlier.

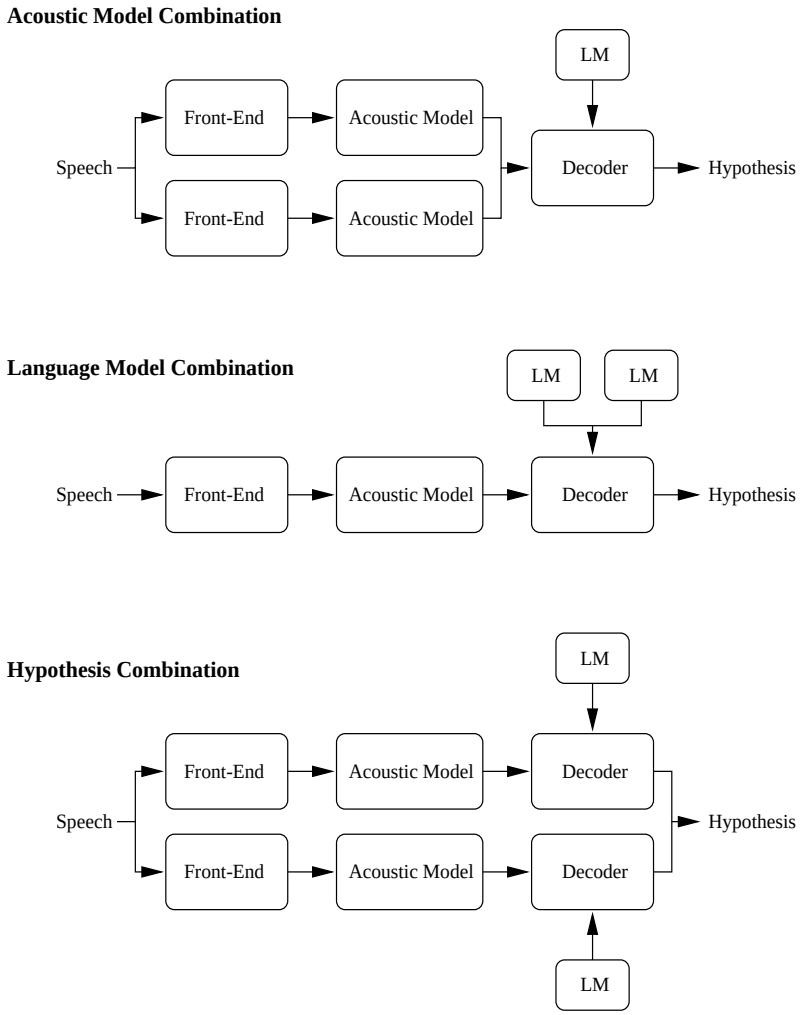

Figure 10.1: Acoustic model, language model and word hypothesis combination

The difference in training paradigms underlying these two types of models leads to significantly different probability estimates that exhibit less mutual dependence than ensembles of homogeneous models. It can be expected that heterogeneous models focus on different parts of the speech signal, resulting in greater diversity and thus greater potential gains in combination. However, heterogeneous acoustic models require some kind of normalization as they typically estimate different quantities (likelihoods vs. scaled likelihoods or posteriors).

10.1 Static Combination

In static combination approaches, one assumes that only a-priori information about the accuracy of individual acoustic models is available. As a consequence, the models are combined using constant weighting factors. Consider m different acoustic models, each estimating HMM emission probabilities $p(\mathbf{x_k}|s_i)$ for the same HMM states s_i based on different feature vectors $\mathbf{x_k}$. We can compute combined estimates of the HMM emission probabilities by applying the product rule

$$p(\mathbf{x_1}, \ldots, \mathbf{x_m}|s_i) = \prod_{k=1}^{m} p(\mathbf{x_k}|s_i)^{\gamma_k} \quad \text{with} \quad \gamma_k \geq 0 \ \text{ and } \ \sum_{k=1}^{m} \gamma_k = 1$$

where the γ_k are a-priori weights that allow to control the relative contribution of each acoustic model to the combined estimates. Typically, the a-priori weights are manually tuned for maximum performance on an independent validation set.
Alternatively, combined estimates of the HMM emission probabilities can be computed by affine interpolation, known as linear opinion pools in statistics [Jacobs '95]:

$$p(\mathbf{x_1}, \ldots, \mathbf{x_m}|s_i) = \sum_{k=1}^{m} \gamma_k \, p(\mathbf{x_k}|s_i) \quad \text{with} \quad \gamma_k \geq 0 \ \text{ and } \ \sum_{k=1}^{m} \gamma_k = 1$$

Note, that a linear opinion pool results from applying the logarithm to the product rule, which is interesting since speech recognition systems typically operate in the log domain.
Now consider the case of combining two heterogeneous acoustic models, one producing estimates of the class conditional likelihoods $p(\mathbf{x}|s_i)$, the other producing estimates of scaled likelihoods $\frac{p(\mathbf{x}|s_i)}{p(\mathbf{x})}$ by means of a locally discriminative connectionist acoustic model. In a theoretically sound framework, we would either have to treat these estimates as 'data' and apply a supra-Bayesian approach [Jacobs '95] or apply some transformation to normalize the estimates of each model to a common domain, suitable for combination. Although such normalization becomes crucial in more elaborate combination techniques, it can be omitted in practice for the simple case of

applying constant scalar weights to each acoustic model. Of course, the estimates resulting from such disregardful combination will not be interpretable as a probability distribution but the combination usually improves accuracy nevertheless.
We have been investigating static weighted combination based on the product rule using two heterogeneous acoustic models trained on the Switchboard corpus. The first one was the state-of-the-art conventional Gaussian mixture model for 24000 tied HMM states that was used in the best performing system in the 1997 Hub-5 evaluation [Finke et al. '97]. The second model was a Hierarchy of Neural Networks (HNN) consisting of roughly 4000 networks, designed for the same set of 24000 HMM states and trained on the same corpus. Fig. 10.2 depicts how the word error rate of the combined system varies for different combination weights, when decoding a subset of the 1996 evaluation set (all other components of the recognizer were kept identical).

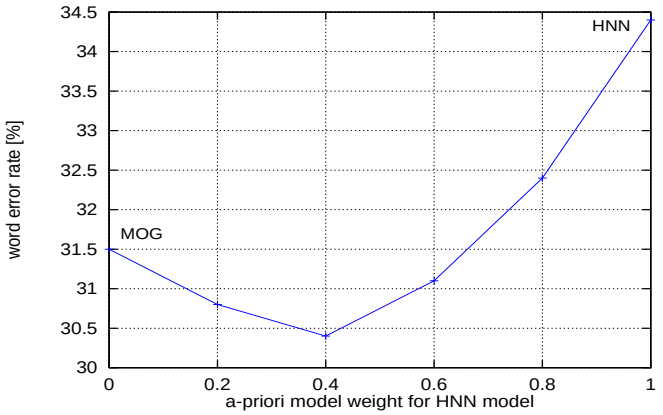

Figure 10.2: Static combination of (1) a mixtures of Gaussians model (MOG) and (2) a hierarchical connectionist acoustic model (HNN)

Table 10.1 shows the word error rates of the two models used stand-alone (left and right end of plot in Fig. 10.2) and in combination. The optimal combination weights turned out to be 0.6 for the Gaussian mixture model and 0.4 for the HNN. and resulted in a relative decrease in word error rate of 3.4% and 11.6% compared to stand-alone use of the mixtures of Gaussians and HNN model, respectively.
Up to now, we have assumed that the a-priori weighting factors are applied globally to the combination of estimates for all HMM states. Instead, one can generalize

acoustic model	word error rate
Mixture of Gaussians (MOG)	31.5 %
Hierarchy of Neural Networks (HNN)	34.4 %
MOG+HNN, product rule (0.6/0.4)	30.4 %

Table 10.1: Best result for static, log-linear combination of heterogeneous acoustic models

the above combination rules to allow for state-dependent a-priori model weights, as proposed in [Rogina & Waibel '94]:

$$p(\mathbf{x_1}, \ldots, \mathbf{x_m}|s_i) = \prod_{k=1}^{m} p(\mathbf{x_k}|s_i)^{\gamma_k(s_i)} \quad \text{with} \quad \gamma_k(s_i) \geq 0 \ \text{ and } \ \sum_{k=1}^{m} \gamma_k(s_i) = 1 \ \ \forall i$$

In contrast to the case of global weights, the considerably larger number of state-dependent weights $\gamma_k(s_i)$ can no longer be tuned manually but must be optimized on some training set, for instance using gradient ascent in log likelihood. This in turn prevents us from directly applying this rule to the case of heterogeneous models and requires to normalize the estimates of such models prior to combination.

10.2 Normalizing Heterogeneous Models

When attempting to combine likelihood estimates of conventional and posterior or scaled likelihood estimates of connectionist acoustic models using state-dependent or dynamically computed weights, we have to transformation the estimates of one or both types of models such that both provide properly normalized probability distributions, suitable for subsequent combination. In the following, we present two such normalization methods, one targeting a-posteriori probability distributions and another, computationally more efficient one based on empiric normalization of likelihoods.

10.2.1 A-Posteriori Normalization

The principle behind our first approach is to require that each acoustic model provides an a-posteriori probability distribution for the set of HMM states and to compute such a distribution as a post-processing step, if necessary. Connectionist acoustic models directly estimate the desired probability distribution and thus can be plugged into the model combination step as is. Conventional mixture models on the other hand provide estimates of the state conditioned HMM emission probabilities which

have to be converted to an a-posteriori probability distribution in order to be fed into the combination step. The task of the combination step then is to compute an aggregate a-posteriori distribution from all the incoming a-posteriori distributions. Just as with a single connectionist acoustic model, the resulting combined a-posteriori distribution has to be converted back to scaled likelihoods by dividing by state priors in order to accommodate the HMM framework. Fig. 10.3 depicts the scenario for the combination of one conventional and one connectionist acoustic model.

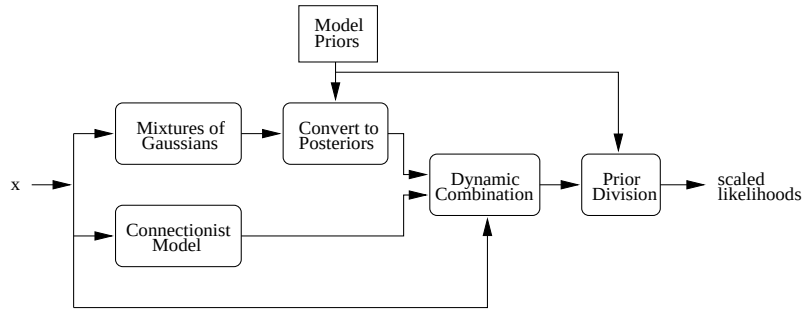

Figure 10.3: Model combination based on normalizing to a-posteriori probabilities

It remains to discuss how we can compute an a-posteriori probability distribution from the state likelihoods estimated by a conventional acoustic model. According to Bayes' rule, the a-posteriori probability of a state s_i given some input feature vector $\mathbf{x}$ can be computed according to

$$p(s_i|\mathbf{x}) = \frac{p(\mathbf{x}|s_i)\, p(s_i)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|s_i)\, p(s_i)}{\sum_j p(\mathbf{x}|s_j)\, p(s_j)}$$

which requires the states' prior probabilities $p(s_j)$ in addition to the state likelihoods $p(\mathbf{x}|s_j)$. For a conventional continuous density model with sometimes more than 20000 unique HMM states, computation of the a-posteriori probability distribution according to the above formula turns out to be computationally very expensive due to the sum of likelihoods in the denominator. As pointed out in [Willett et al. '98], there is an efficient way of computing the a-posteriori distribution in the case of heavy mixture tying, i.e. in phonetically-tied, semi-continuous and discrete HMMs. However, in the case of a fully continuous density HMM system, the required amount of computations can become prohibitive. A computationally less expensive alternative is to directly estimate $p(\mathbf{x})$ in the above formula using a mixture density with a sufficiently large number of component densities. New problems arise in that case:

how do we ensure that the resulting a-posteriori distribution is properly normalized such that the probabilities sum up to one? How can we efficiently compute a mixture density with the large number of component densities that are required for accurately estimating $p(\mathbf{x})$? Depending on the specific case, we might still end up with a computationally very expensive procedure.
In summary, normalization to a-posteriori distributions appears to be attractive in cases where we want to combine only a single conventional model with one or many connectionist models or in cases where the conventional models employ some kind of mixture tying such that we can efficiently compute the required a-posteriori distributions.

10.2.2 Empiric Normalization

We have developed an efficient technique for normalizing estimates of heterogeneous acoustic models that does not even require that the models produce probability scores at all [Fritsch & Finke '97]. All that is required is that the emitted model scores are continuous, bound to a finite interval and that all acoustic models adhere to the same interpretation of score ordering, for instance, that lower scores correspond to better acoustic matches.

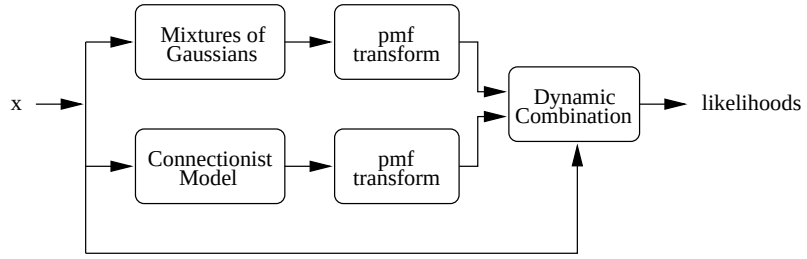

Figure 10.4: Model combination by normalizing based on empiric probability mass functions

Fig. 10.4 depicts the setup for empiric normalization. Here, the outputs of all acoustic models are normalized using a transformation based on the empiric probability mass function (pmf) of the distribution of model scores. The resulting transformed estimates are bound to the range $[0, 1]$ and represent pseudo-probabilities, e.g., a value of 0 corresponds to the worst score while a value of 1 corresponds to the best score of the corresponding acoustic model. This property is achieved consistently for all acoustic models participating in the ensemble and thus enables dynamically weighted combinations.

The normalization functions for each acoustic model are estimated as follows. First, we empirically estimate the distribution of acoustic scores for each acoustic model based on some held-out training set. This is typically done by estimating a discrete, histogram-based probability distribution. For instance, consider the earlier case of (1) a conventional Gaussian mixtures model and (2) a hierarchical connectionist acoustic model. Fig. 10.5 shows empiric distributions of acoustic scores for both models. Here, the Gaussian mixture model emits the negative logarithm of likelihoods and the connectionist model emits the negative logarithm of scaled likelihoods.

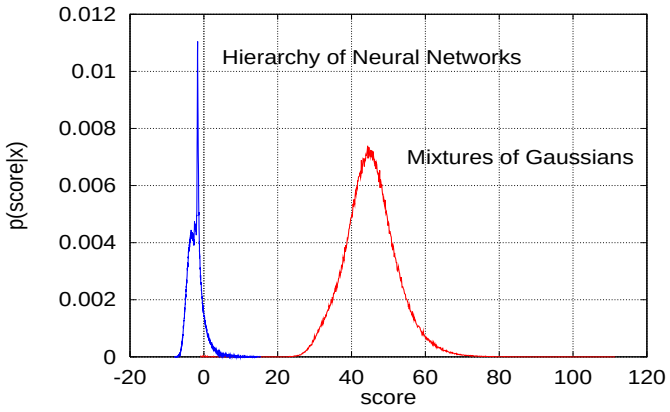

Figure 10.5: Distribution of likelihood scores for connectionist and Gaussian mixtures acoustic models

While the distribution of the Gaussian mixtures scores seems to smoothly follow a Gaussian, the distribution of the hierarchical connectionist model scores contains bumps and a very strong peak near zero. This peak is attributable to the very frequent silence model and the MAP training procedure which leads to good discrimination of silence and speech models. Note also, that part of the distribution for the connectionist model reaches into the area of negative scores due to the division by priors applied to obtain scaled likelihoods.
Let $Y_k = p_k(\mathbf{x}|s_i)$ be the random variable representing scores computed by the k-th acoustic model for all $\mathbf{x}$ and s_i. According to this notation, the histograms depicted in Fig. 10.5 represent $p(Y = Y_k)$, the probability distribution of scores. We now introduce a score normalization function q for each acoustic model that maps the original model-dependent scores into the range $[0, 1]$ by evaluating the empiric probability

mass function (pmf):

$$q(Y_k) := p(Y > Y_k)$$

Fig. 10.6 depicts the probability mass functions computed from the histograms in Fig. 10.5 for normalizing the corresponding two acoustic models.

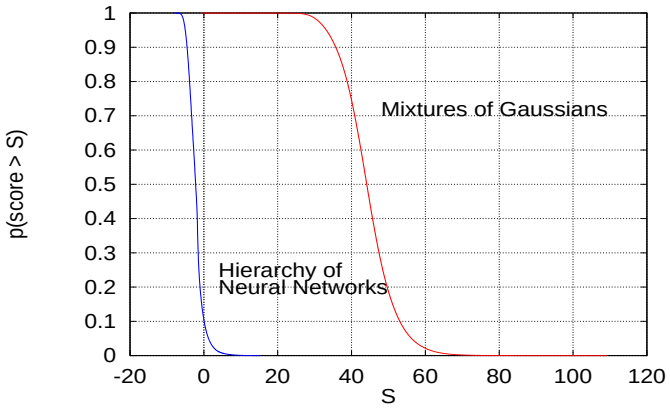

Figure 10.6: Normalization functions for connectionist and Gaussian mixtures acoustic models

These functions effectively and consistently normalize the scores of heterogeneous acoustic models. For instance, a normalized score of 0.5 represents in both models an original, model-dependent score that is located at the center of gravity in the distribution of model-dependent scores. The above normalization method corresponds to a monotonic mapping which preserves the order of scored models while maximizing the entropy of the resulting normalized scores by approximating a uniform distribution. Furthermore, the proposed method requires only a single table look-up to evaluate the empiric, histogram-based probability mass function and thus places virtually no additional computational burden on acoustic model combination.

10.3 Dynamic Combination

In dynamic acoustic model combination, we allow for combination weights that vary with each time frame, assuming that the quality of each contributing acoustic model

is not fixed a-priori but varies locally, maybe depending on acoustic conditions or broad class of phonetic sound being uttered. Some models might provide accurate estimates for vowel sounds and be uncertain for stop consonants while others behave vice versa (additionally depending on the acoustic front-ends being used). In the remainder, we will investigate dynamic acoustic model combination based on linear opinion pools, that is, we model the combined acoustic probabilities according to

$$p(\mathbf{x}|s_i) = \sum_{k=1}^{m} g_k(\mathbf{x})\, p_k(\mathbf{x}|s_i) \quad \text{with} \quad g_k(\mathbf{x}) \geq 0 \;\text{ and }\; \sum_{k=1}^{m} g_k(\mathbf{x}) = 1$$

where we have condensed potentially different feature spaces into a single feature space for simplicity. Furthermore, we assume that heterogeneous acoustic models have been adequately normalized, for instance using one of the techniques presented in the previous section, before being used in the above combination rule. The $g_k(\mathbf{x})$ are time-varying weighting functions that should reflect our relative confidence into each one of the acoustic models at each time step. There are several potential knowledge sources that might be used for deriving the above weighting functions. The corresponding approaches might be categorized into the following three classes

1. Using frame-level measures of confidence, e.g., the entropy of the a-posteriori distribution
2. Optimizing a frame-based training objective (MAP)
3. Using phone-, word- or sentence-level confidence scores

Since frame-level accuracy is not necessarily correlated with word-level recognition accuracy, the frame-level measures of confidence applied in the first case can not guarantee to improve recognition accuracy. The second and third approach appear more promising but require considerably more effort to compute the model weighting functions.

10.4 Gating Networks

The application of a gating network for computing weighting functions is motivated by the Meta-Pi [Hampshire & Waibel '89] and the mixtures of experts framework [Jacobs '95] and its extension to hierarchies of experts [Jordan & Jacobs '92, Jordan & Jacobs '94] for fusing the opinions of several experts, expressed as probability distributions, into a single probability distribution that can be used for decision making.

Assuming that each acoustic model (expert) computes a valid probability distribution, a mixture of experts computes a combined probability distribution through linear interpolation, using an additional estimator called a gating network that weights the contribution of each expert at each time frame based on the current feature vector (see Fig. 10.7). In order to assure that the weights produced by the gating network satisfy the constraints of a probability distribution, the output layer is typically parameterized using the softmax function.

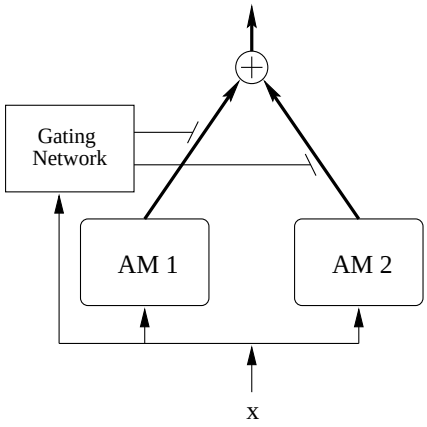

Figure 10.7: Mixtures of experts approach to dynamic acoustic model interpolation

Defining a global, differentiable training objective, e.g., maximum-likelihood or maximum a-posteriori, the parameters of the gating network can be learned by back-propagating errors from the output of the above architecture to the gating network and applying, for instance, gradient ascent based optimization. Assuming that the acoustic models estimate a-posteriori distributions, the combined system computes

$$p(s_i|\mathbf{x}) = \sum_{k=1}^{m} g_k(\mathbf{x})\; p_k(s_i|\mathbf{x})$$

In the case of Viterbi-based MAP training, we obtain the following log-likelihood based objective function for the combined system:

$$E = \sum_{t} \log \sum_{k=1}^{m} g_k(\mathbf{x})\; p_k(s_{c(t)}|\mathbf{x})$$

where $c(t)$ denotes the index of the HMM state that is assumed to have produced $\mathbf{x}$ at time t according to a Viterbi alignment. Furthermore assuming that the gating network is realized as a generalized linear model (single layer with softmax output non-linearity), we can iteratively optimize its weight matrix W using gradient ascent as follows:

$$W^{(i+1)} = W^{(i)} + \eta \sum_t \frac{dE(t)}{dW}$$

where η is the learning rate and the derivative of $E(t)$ with respect to the matrix of weights is given by

$$\frac{dE(t)}{dW} = (\mathbf{h}(\mathbf{x}) - \mathbf{g}(\mathbf{x}))\, \mathbf{x}^T$$

with the vector of posterior probabilities $\mathbf{h}(\mathbf{x})$ consisting of components

$$h_j(\mathbf{x}) = \frac{g_j(\mathbf{x})\, p_j(s_{c(t)}|\mathbf{x})}{\sum_{k=1}^{m} g_k(\mathbf{x})\, p_k(s_{c(t)}|\mathbf{x})}$$

Note that the above optimization rules can easily extended to more complex gating networks such as multi-layer perceptrons by applying the chain rule as in the original backpropagation algorithm.
We have been investigating the viability of this approach by training a multi-layer perceptron with a single hidden layer of 64 units as the gating network in a dynamic combination of the two heterogeneous Switchboard models used earlier for the experiments with static combination. Estimates of both systems were normalized using the empiric histogram mapping approach. The gating network was trained on 28 hours from the Switchboard corpus until log-likelihood converged. Fig. 10.2 gives recognition results of the individual models and the gated combination, again on a subset of the 1996 evaluation set.

Acoustic Model	**Word Error Rate**
Mixture of Gaussians (MOG)	31.5 %
Hierarchy of Neural Networks (HNN)	34.4 %
MOG+HNN, dynamically gated by 39-64-2 MLP	30.2 %

Table 10.2: Results of dynamic acoustic model combination using a gating network

In our experiment, dynamic combination reduces the word error rate by 4.1% relative to using the best model in the ensemble in isolation. However, the gains of using

a dynamic weighting scheme over simple static combination are comparably small (0.2% absolute), suggesting that the optimal weighting functions are too complex to be learned adequately by a single gating network.

10.5 Discussion

In this chapter, we have been presenting techniques for combining the estimates of several acoustic models, focusing on the combination of heterogeneous models, e.g., Gaussian mixtures and connectionist models. In contrast to combinations of homogeneous models, we had to address the issue of score normalization in order to be able to dynamically combine models that estimate different quantities. We have argued for the normalization to an a-posteriori probability distribution. However, as this kind of normalization can become computationally prohibitive in large continuous density HMM systems, we have presented an alternative normalization scheme based on the probability mass function of the distribution of model scores. We have demonstrated the viability of this more efficient approach by using it to dynamically combine a Gaussian mixtures model with a hierarchical connectionist model using a gating network to estimate the weighting functions. While this proved to be an effective method for dynamic model combination, it turned out that simple static combination was almost as effective. We assume that the optimal weighting functions in the dynamic case are too complex to be learned by a single hidden layer perceptron. A solution to this problem would be to assign a separate gating network to suitable clusters of HMM states, for instance to HMM states belonging to the same monophone.
In our approaches to acoustic model combination, we have assumed that the contributing models are pre-trained and that their parameters are fixed such that the combination algorithms can treat them as black boxes. It should be noted that approaches based on an integrated training of several acoustic models such as boosting or mixtures of experts typically yield larger gains in recognition accuracy since these approaches specifically push towards independent experts as part of the training objective. It has been shown [Jordan '95] that the degree of independence of the estimates of an ensemble of learners is directly correlated with the potential gain in accuracy obtained from a combination of these learners. However, given the large and complex architectures in acoustic modeling that often require several days of training, an approach such as the one presented in this thesis is considered more appropriate, especially when dealing with heterogeneous estimators.

Chapter 11

Conclusions

This thesis has presented a new, principled framework for connectionist acoustic modeling in large vocabulary statistical speech recognition. Based on the divide-and-conquer paradigm, it was possible to derive a hierarchical, tree-structured architecture that decomposes the task of estimating HMM state posteriors into thousands of smaller tasks, each of which being solved by a small neural network. In contrast to the conventional approach, this strategy allows (1) to construct *scalable* acoustic models which allow to adapt the specificity of context modeling to previously unseen domains and to arbitrarily downsize the model without retraining, and (2) to devise algorithms for effective speaker adaptation and accelerated model evaluation that *exploit* the inherent hierarchical structure.

Based on the proposed hierarchical architecture, it was for the first time possible to construct competitive connectionist acoustic models for a state-of-the-art large vocabulary conversational speech recognition system, modeling as much as 24000 unique HMM states using a tree structure consisting of over 4000 neural networks. Using unsupervised speaker adaptation, our system achieves a word error rate of 31.8% on the Switchboard conversational telephone speech corpus. While this result is comparable to what we have achieved with our best conventional, non-connectionist system, the hierarchical connectionist acoustic model requires only one forth the number of parameters and decodes more than 5 times faster at about 25 times real-time.

Furthermore, we have tackled the important problem of domain dependence of acoustic models that usually prohibits the application of a speech recognition system in domains that differ from its training domain because of an unacceptable increase in the word error rate. This problem can be attributed to domain-specific context-dependent modeling that on the one hand appears to be inevitable for state-of-the-art performance but on the other hand ties the acoustic model to the phonetic characteristics of the training corpus. The resulting trade-off prevents us from building

accurate and universal acoustic models with respect to different application domains. However, the hierarchical architecture presented in this thesis allows for *domain-adaptive* acoustic modeling which achieves domain-specific performance after only 45-60 minutes of acoustic adaptation data from an unseen target domain. We developed an algorithm called Structural Domain Adaptation (SDA), that takes advantage of the multi-scale representation of phonetic context in a hierarchical connectionist acoustic model for adapting the specificity of phonetic context modeling to a new domain. The effectiveness of the SDA approach was experimentally demonstrated by adapting a conversational telephone speech system to two significantly different, previously unseen application domains.
In the following, we summarize the main contributions of this thesis and give suggestions for future work.

11.1 Thesis Contributions

This thesis has presented both a novel hierarchical connectionist framework for acoustic modeling in large vocabulary statistical speech recognition and a selection of algorithms that exploit the unique properties of the resulting tree-structured architecture for purposes such as fast decoding, effective speaker adaptation and domain-adaptive speech recognition. Following is a list of the specific contributions presented in this thesis.

- **Hierarchy of Neural Networks (HNN):**
  Based on hierarchically factoring context-dependent HMM state posteriors, we have derived a tree-structured, locally-discriminative acoustic model that scales to an arbitrarily large number of HMM states. The model was termed a Hierarchy of Neural Networks since we apply small feed-forward neural networks to the task of estimating the required local conditional posterior probabilities at each internal node of the modeling tree. The most prominent advantages of the proposed architecture in contrast to existing acoustic models are the multi-scale representation of phonetic context and the hierarchical structure reflecting acoustic similarity at various levels in the modeling tree. We have evaluated Hierarchy of Neural Networks based acoustic models on the Switchboard large vocabulary conversational telephone speech corpus and demonstrated that the model (1) achieves state-of-the-art performance and (2) scales to as much as 24000 HMM states which was found to be beneficial in terms of performance.

- **Constructive Methods for Designing HNNs:**
  We have presented agglomerative (bottom-up) and divisive (top-down) clustering algorithms developed specifically for the construction of a hierarchically

structured acoustic model from second order statistics of HMM state emissions and compared both approaches extensively on data from the Switchboard corpus. Our algorithms have been designed to favor balanced trees and close-to-uniform prior distributions of child nodes at each tree node such that accurate training of local conditional a-posteriori probabilities is facilitated and the hierarchical structure can be exploited most effectively.

- **Investigation of Local Model Selection:**
  Given the exponential decrease in the amount of available training data from root to leaves in an HNN tree, the size and complexity of local estimators of conditional a-posteriori probabilities must be chosen carefully in order to maximize generalization performance. We have simplified the problem of model selection significantly by using single hidden layer MLPs in our HNNs. The single degree of freedom concerning model complexity - the number of hidden units - allowed us to determine optimal model size in a two step process: (1) preselection of rough model size based on available training data and (2) iteratively increasing or decreasing model size based on performance improvement on an independent validation set.

- **Efficient Distributed Training:**
  Typically, training of connectionist acoustic models is computationally very expensive, often requiring days of training on special hardware. In contrast, we have presented efficient distributed training methods for Hierarchies of Neural Networks that allow to train this model as fast and sometimes even faster than a conventional, mixture density based acoustic model using standard hardware. As each network in an HNN can be trained independently without any communication or synchronization overhead, training of the complete model can be easily distributed among several low-cost standard computers, allowing training times of less than 24 hours for the full Switchboard corpus.

- **Dynamic Tree Pruning for Fast Decoding:**
  Using simple dynamic tree pruning based on partially computed posteriors, it is possible to significantly reduce the amount of computation required to evaluate our hierarchical connectionist acoustic model during decoding. We have demonstrated that this technique allows to speed-up the evaluation of acoustic scores by a factor of up to 10 with almost no measurable decrease in performance. The overall decoding time on the Switchboard corpus could be improved by a factor of 6 from 140 times real-time to only 24 times real-time without an increase in word error rate.

- **Effective Speaker Adaptation:**
  It has been shown that our tree-structured acoustic model inherently realizes

parameter sharing at multiple scales according to acoustic similarity which can be exploited for effective speaker adaptation with limited amounts of adaptation data. We have developed an algorithm for speaker adaptation that benefits from this build-in structure and thus requires no additional tying structures such as regression trees that are typically needed for adapting conventional models. Using unsupervised adaptation on only up to 3 minutes of speech from each speaker, our method for speaker adaptation achieves a relative decrease in word error rate of 9.5% on a subset of 20 speakers from the 1996 Switchboard evaluation test set.

- **Structural Domain Adaptation (SDA):**
  We have presented an algorithm to efficiently and effectively downsize and adapt the structure of large vocabulary conversational speech recognition systems based on the proposed hierarchical connectionist acoustic model to previously unseen application domains. In contrast to conventional, domain-dependent models, the SDA approach allows to adapt the structure and thus the specificity of phonetic context modeling in an HNN based acoustic model for optimal modeling in new domains. Experimental validation of the SDA approach has been carried out by adapting size and structure of HNN based acoustic models trained on Switchboard to two quite different, unseen domains, Wall Street Journal newspaper articles and English spontaneous scheduling conversations. In both cases, our approach yields considerably downsized acoustic models with performance equal to domain-specific models and improvements of up to 18% over the unadapted baseline model.

- **Mixture Trees:**
  Hierarchies of Neural Networks are based on factoring posterior state probabilities which are not available in conventional mixture based modeling. However, we have demonstrated that it is still possible to derive a likelihood-based tree-structured acoustic model with properties similar to HNNs by hierarchically tying mixture density components. We have derived a variant of the EM algorithm for estimating the parameters of the resulting model, which we have called Mixture Tree.

- **Downsizing of Hierarchical Acoustic Models:**
  Another quite attractive property of the hierarchical acoustic models that we have presented in this thesis is the possibility of downsizing the trained models to accommodate limited processor speed and/or memory. Downsizing can be realized by simple tree trimming and allows to operate a hierarchical connectionist acoustic model in a variety of context resolutions, from fully context-dependent down to context-independent modeling. No parameter re-training is required.

We have demonstrated the effects of model downsizing for the case of mixture trees.

- **Combination of Heterogeneous Acoustic Models:**
  We have discussed methods for the combination of multiple, possibly heterogeneous acoustic models and presented a novel efficient normalization technique that allows for effective combination of a conventional, mixture densities based acoustic model with a hierarchical connectionist acoustic model. A combination approach based on dynamically weighting the models using an MLP gating network yielded a relative reduction in word error rate of 4%.

11.2 Future Work

The hierarchical acoustic modeling framework presented in this thesis can be extended in various ways. We give some brief suggestions for further work on tree-structured acoustic models in statistical speech recognition and other applications:

- **Merging Decision Trees and Hierarchical Acoustic Models:**
  Instead of having separate trees for classifying and scoring phonetic context models as proposed in this thesis, we could try to merge these structures into a single model. Of course, we would then have to find a compromise between the partially different requirements and constraints of each one of the models.

- **Application to Multilingual Speech Recognition:**
  While the experiments presented in this thesis have been restricted to a single language, hierarchical connectionist acoustic models may be constructed and trained on multiple languages, resulting in a multilingual acoustic model which may be beneficial for multilingual and crosslingual speech recognition [Cohen et.al. '97, Schultz & Waibel '98]. The SDA algorithm presented in chapter 8 may then be used to adapt the model to a specific target language.

- **Application to Extended HMM Frameworks:**
  Recently, several innovative and alternative modeling frameworks have been proposed for statistical speech recognition and the recognition of time series in general. Although still related to standard HMMs, approaches such as IOHMMs [Bengio & Frasconi '96], factorial HMMs [Ghahramani & Jordan '97] and the REMAP framework [Bourlard et al. '94] attempt to improve modeling accuracy by avoiding some of the false assumptions typically found in standard models. Both approaches might benefit from using the architecture proposed in this thesis for the purpose of probability estimation in large HMMs.

- **Boosting of Local Estimators:**
  Boosting has been around for quite some time now and has been shown to improve classifier performance considerably by combining the estimates of several learners trained subsequently on patterns re-weighted depending on the errors of the predecessors. This technique might readily be applied to the local estimators in an HNN tree. To avoid excessive computations leading to unfeasible long training times, boosting might be restricted to the estimators in the vicinity of the root node and still improve performance of the overall model.

- **Other Applications:**
  The hierarchical connectionist model presented in this thesis is unique in its ability to estimate posterior probabilities for a very large number of classes. In that respect, it might be interesting to apply our model to other tasks such as speaker identification or face recognition and furthermore benefit from the similarity groupings represented in the tree structure.

Appendix A

Connectionist Posterior Probability Estimation

We consider the N class classification problem. The following proof assumes that a classifier neural network of arbitrary architecture with N output neurons (one for each target class ω_i) is trained to minimize the squared error (MSE) between network outputs y_i and targets t_i. Target vectors $\mathbf{t}$ are encoded according to the 1-out-of-N scheme, meaning that the correct class is encoded using a 1, while all others are encoded using a 0. Furthermore, we assume continuous valued input vectors $\mathbf{x}$.
The network error under the MSE criterion can be expressed as follows:

$$E = \int p(\mathbf{x}) \sum_{i=1}^{N} \sum_{j=1}^{N} p(\omega_i|\mathbf{x}) \left[y_j(\mathbf{x}) - t_j(\mathbf{x})\right]^2 d\mathbf{x}$$

Here, $p(\omega_i|\mathbf{x})$ denotes the Bayesian a-posteriori probability of class ω_i given input vector $\mathbf{x}$. Since $p(\mathbf{x}) = \sum_{k=1}^{N} p(\omega_k, \mathbf{x})$, we have

$$E = \int \sum_{k=1}^{N} \left(\sum_{i=1}^{N} \sum_{j=1}^{N} p(\omega_i|\mathbf{x}) \left[y_j(\mathbf{x}) - t_j(\mathbf{x})\right]^2 \right) p(\omega_k, \mathbf{x}) \, d\mathbf{x}$$

Using the 1-out-of-N assumption that $t_j(\mathbf{x}) = \delta_{ij}$ if $\mathbf{x} \in \omega_i$ and adding and subtracting $p^2(\omega_j|\mathbf{x})$ leads to

$$\begin{aligned} E &= \int \sum_{k=1}^{N} \left(\sum_{j=1}^{N} (y_j^2(\mathbf{x}) - 2y_j(\mathbf{x})p(\omega_j|\mathbf{x}) + p^2(\omega_j|\mathbf{x})) \right) p(\omega_k, \mathbf{x}) \, d\mathbf{x} \\ &+ \int \sum_{k=1}^{N} \left(\sum_{j=1}^{N} (p(\omega_j|\mathbf{x}) - p^2(\omega_j|\mathbf{x})) \right) p(\omega_k, \mathbf{x}) \, d\mathbf{x} \end{aligned}$$

which can be further simplified to

$$\begin{aligned} E &= \int \sum_{k=1}^{N} \left(\sum_{j=1}^{N} (y_j(\mathbf{x}) - p(\omega_j|\mathbf{x}))^2 \right) p(\omega_k, \mathbf{x})\, d\mathbf{x} \\ &+ \int \sum_{k=1}^{N} \left(\sum_{j=1}^{N} (p(\omega_j|\mathbf{x}) - p^2(\omega_j|\mathbf{x})) \right) p(\omega_k, \mathbf{x})\, d\mathbf{x} \end{aligned}$$

The second term in the above expression can be neglected since it is independent of the network parameters. Minimization of the mean squared error criterion can thus be achieved by minimizing the first term in the above expression which is simply the mean squared error between the network outputs $y_j(\mathbf{x})$ and the Bayesian a-posteriori class probabilities $p(\omega_j|\mathbf{x})$. Therefore, training a network to minimize the MSE between outputs and 1-out-of-N targets results in the best approximation to the true a-posteriori distribution in the sense of that criterion. However, the given proof contains implicit assumptions:

- The network must be trained to the global minimum of error. Since training a feed-forward neural network is NP-complete, reaching the global minimum of error can not be guaranteed in practice. However, it was shown, that for real world problems, local minima often do not differ significantly from the global one.
- The network must contain enough free parameters (plasticity) to model the potentially complex posterior probability distribution. For instance, a single-layer network will not be able to model a non-Gaussian, multimodal posterior distribution.
- An infinite amount of training samples is available for training the network. This assumption is of course not realizable but in practice, a reasonably large training corpus is usually sufficient as long as adding more samples improves network performance only marginally.

It should be noted that the proof can be given for other continuous network optimization criteria such as relative entropy as well.
The presented proof was originally published in [Richard & Lippmann '91] for the relative entropy criterion and later in [Morgan & Bourlard '95] for the mean square error criterion. A similar proof can be found in [Bridle '90].

Appendix B

Allophonic Variation in 24000 State Switchboard Model

The following Table shows the distribution of the roughly 24000 allophonic variations (tied states), modeled by the Hierarchy of Neural Networks used in some of the experiments in this thesis, among the positions in the underlying 3-state left-right phone models (only the phones modeling speech sounds are shown). The 4 phones marked with an ampersand denote special phones for modeling interjections. Using the split likelihood gain criterion introduced in section 2.4.2, phonetic decision trees were constructed for each position of each phone model based on 170 hours of training data from the Switchboard LVCSR corpus. The number of allophonic variations shown for begin, middle and end positions in the Table below correspond to the number of leaf nodes in the corresponding decision trees. Note the large variation in the number of context-dependent states generated for each phone which reflects the highly non-uniform distribution of monophone priors in the training corpus.

phone name	position in 3-state HMM			total number of tied states
	begin	middle	end	
T	609	421	598	1628
N	454	357	501	1312
R	415	430	440	1285
IY	327	304	435	1066
L	381	282	372	1035
AX	326	292	347	965
D	350	204	321	875
K	304	208	298	810
IX	265	230	260	755
M	288	166	275	729

phone name	position in 3-state HMM			total number of tied states
	begin	middle	end	
S	232	199	294	725
IH	244	206	263	713
AE	240	198	252	690
OW	242	177	266	685
EH	220	193	265	678
AY	196	179	245	620
EY	176	182	243	601
W	196	177	222	595
AH	191	153	207	551
UW	158	184	188	530
DH	164	186	173	523
V	172	140	180	492
Z	164	132	189	485
B	189	117	155	461
HH	170	125	151	446
P	134	122	172	428
AXR	141	112	163	416
Y	121	134	140	395
AA	114	98	153	365
F	122	109	126	357
G	129	86	121	336
ER	83	86	103	272
AO	84	75	101	260
NG	67	65	67	199
AW	64	59	63	186
&AH	60	35	68	163
TH	60	48	51	159
JH	49	59	46	154
SH	52	46	53	151
DX	57	38	50	145
CH	38	55	39	132
&M	30	43	41	114
UH	32	36	38	106
EN	20	28	25	73
OY	13	20	16	49
&HH	9	10	9	28
&OW	10	11	7	28
ZH	6	5	7	18

Bibliography

ARPA (1994). *Proceedings of ARPA Spoken Language Systems Technology Workshop, Princeton, NJ*, 1994.

Bahl L. R., Souza de P. F., Gopalakrishnan P. S., Nahamoo D., A. Picheny M. (1991). Context Dependent Modeling of Phones in Continuous Speech using Decision Trees. In *Proceedings of the 1991 DARPA Speech and Natural Language Processing Workshop, Pacific Grove, CA*, pp. 264–270, February 1991.

Baum E. B., Haussler D. (1989). What Size Net Gives Valid Generalization. *Neural Computation*, 1:151–160.

Beattie V. L., Young S. J. (1992). Hidden Markov Model State based Cepstral Noise Compensation. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 92)*, pp. 519–522, 1992.

Bengio Y., Frasconi P. (1996). Input-Output HMMs for Sequence Processing. *IEEE Transactions on Neural Networks*, 7:1231–1249.

Bengio Y. (1996). Markovian Models for Sequential Data. Technical Report 1049, Dept. IRO, Université de Montréal, 1996.

Bishop C. M. (1995a). *Neural Networks for Pattern Recognition*. Oxford University Press.

Bishop C. M. (1995b). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7:108–116.

Bourlard H., Morgan N. (1994). *Connectionist Speech Recognition – A Hybrid Approach*. Kluwer Academic Press.

Bourlard H., Konig Y., Morgan N. (1994). REMAP: Recursive Estimation and Maximization of a Posteriori Probabilities – Application to Transition Based

Connectionist Speech Recognition. Technical Report TR-94-064, International Computer Science Institute (ICSI), 1994.

Breiman L., Friedman J. H., Olshen R. A., Stone C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, California.

Bridle J. (1990). Probabilistic Interpretation of Feed Forward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. *Neurocomputing: Algorithms, Architectures and Applications*.

Chow Y., Schwartz R., Roucos S., Kimball O., Price P., Kubala F., Dunham M., Krasner M., Makhoul J. (1986). The Role of Word-Dependent Coarticulatory Effects in a Phoneme-Based Speech Recognition System. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 86)*, April 1986.

Cohen M., Franco H., Morgan N., Rumelhart D., Abrash V. (1992). Context-Dependent Multiple Distribution Phonetic Modeling with MLPs. *Advances in Neural Information Processing*, 5:649–657.

Cohen J., Kamm T., Andreou A. G. (1995). Vocal Tract Normalization in Speech Recognition: Compensation for Systematic Speaker Variability. *Journal of the Acoustical Society of America*, 97(5).

Cohen P., Dharanipragada J., Gros J., Monkowski M., Neti C., Roukos S., Ward T. (1997). Towards a Universal Speech Recognizer for Multiple Languages. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 97), Santa Barbara, CA*, December 1997.

Cook G. D., Robinson A. J. (1996). Boosting the Performance of Connectionist Speech Recognition. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 96), Philadelphia, PA*, 1996.

Cook G. D., Robinson A. J. (1998). The 1997 ABBOT System for the Transcription of Broadcast News. In *Proceedings of Broadcast News Transcription and Understanding Workshop, 1998*, February 1998.

Cook G. D., Kershaw D. J., Christie J. D. M., Robinson A. J. (1997a). Transcription of Broadcast Television and Radio News: The 1996 ABBOT System. In *Proceedings of Broadcast News Transcription and Understanding Workshop, 1998*, February 1997.

Cook G. D., Waterhouse S. R., Robinson A. J. (1997b). Ensemble Methods for Connectionist Acoustic Modelling. In *Proceedings of Eurospeech 97, Rhodes, Greece*, September 1997.

Cook G. D., Christie J., Ellis D., Fosler-Lussier E., Gotoh Y., Kingsbury B., Morgan N., Renals S., Robinson A. J., Williams G. (1999). An Overview of the SPRACH System for the Transcription of Broadcast News. In *Proceedings of the 1999 DARPA Broadcast News Workshop, Herndon, VA*, February 1999.

DARPA (1998). *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne Conference Resort, Lansdowne, VA, 1998.

Davis S. B., Mermelstein P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Trans. Acoustics Speech and Signal Processing*, 28(4):357–366.

Dempster A. P., Laird N. M., Rubin D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, 39(1).

Duda R., Hart P. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Finke M., Waibel A. (1997a). Flexible Transcription Alignment. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 97), Santa Barbara, CA*, December 1997.

Finke M., Waibel A. (1997b). Speaking Mode Dependent Pronunciation Modeling in Large Vocabulary Conversational Speech Recognition. In *Proceedings of Eurospeech 97, Rhodes, Greece*, September 1997.

Finke M., Fritsch J., Geutner P., Ries K., Zeppenfeld T. (1997). The JanusRTk Switchboard/Callhome 1997 Evaluation System. In *Proceedings of LVCSR Hub-5E Workshop, Linthicum Heights, MD*, May 1997.

Finke M., Fritsch J., Koll D., Waibel A. (1999). Modeling and Efficient Decoding of Large Vocabulary Conversational Speech. In *Proceedings of Eurospeech 99, Budapest, Hungary*, September 1999.

Finke M. (1996). Mode Dependent Pronunciation Modeling in LVCSR. In *Proceedings of the CLSP Summer Workshop, Johns Hopkins University, Baltimore, MD*, 1996.

Fiscus J. G. (1997). A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER). In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 97), Santa Barbara, CA*, December 1997.

Franco H., Cohen M., Morgan N., Rumelhart D., Abrash V. (1994). Context-Dependent Connectionist Probability Estimation in a hybrid HMM – Neural Net Speech Recognition System. *Computer, Speech and Language*, 8.

Franco H., Weintraub M., Cohen M. (1997). Context Modeling in a Hybrid HMM – Neural Net Speech Recognition System. In *Proceedings of International Conference on Neural Networks (ICNN 97)*, 1997.

Franzini M., Lee K. F., Waibel A. (1990). Connectionist Viterbi Training: A New Hybrid Method for Continuous Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 90), Albuquerque, NM*, volume 1, pp. 425–428, 1990.

Fritsch J., Finke M. (1997). Improving Performance on Switchboard by Combining Hybrid HME/HMM and Mixture of Gaussians Acoustic Models. In *Proceedings of Eurospeech 97, Rhodes, Greece*, September 1997.

Fritsch J., Finke M. (1998a). ACID/HNN: Clustering Hierarchies of Neural Networks for Context-Dependent Connectionist Acoustic Modeling. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 98), Seattle, WA*, May 1998.

Fritsch J., Finke M. (1998b). Applying Divide and Conquer to Large Scale Pattern Recognition Tasks. *Neural Networks: Tricks of the Trade, Orr, G. B. and Müller, K. R. (eds), LNCS series, vol. 1524, Springer*, pp. 315–342.

Fritsch J., Rogina I. (1996). The Bucket Box Intersection (BBI) Algorithm for Fast Approximative Evaluation of Diagonal Mixture Gaussians. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 96), Atlanta, GA*, May 1996.

Fritsch J., Waibel A. (1998). Hierarchies of Neural Networks for Context-Dependent Connectionist Acoustic Modeling. In *Proceedings of European Symposium on Artificial Neural Networks (ESANN 98), Brugges, Belgium*, April 1998.

Fritsch J., Rogina I., Sloboda T. (1995). Speeding up the Score Computation of HMM Speech Recognizers with the Bucket Voronoi Intersection Algorithm. In *Proceedings of Eurospeech 95, Madrid, Spain*, September 1995.

FRITSCH J., FINKE M., WAIBEL A. (1996). Adaptively Growing Hierarchical Mixtures of Experts. *Advances in Neural Information Processing*, 9.

FRITSCH J., FINKE M., WAIBEL A. (1997). Context-Dependent Hybrid HME/HMM Speech Recognition using Polyphone Clustering Decision Trees. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 97), Munich, Germany*, April 1997.

FRITSCH J., FINKE M., WAIBEL A. (1998a). Effective Structural Adaptation of LVCSR Systems to Unseen Domains Using Hierarchical Connectionist Acoustic Models. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 98), Sydney, Australia*, December 1998.

FRITSCH J., FINKE M., WAIBEL A. (1998b). Structural Adaptation of Hierarchical Connectionist Acoustic Models to Unseen Domains. In *Proceedings of LVCSR Hub-5E Workshop, Linthicum Heights, MD*, September 1998.

FRITSCH J. (1996). Modular Neural Networks for Speech Recognition. Technical Report CMU-CS-96-203, Carnegie Mellon University, Pittsburgh, PA, August 1996.

FRITSCH J. (1997). ACID/HNN: A Framework for Hierarchical Connectionist Acoustic Modeling. In *Proceedings of IEEE Workshop on Speech Recognition and Understanding (ASRU 97), Santa Barbara, CA*, December 1997.

FRITSCH J. (1999a). Mixture Trees – Hierarchically Tied Mixture Densities for Modeling HMM Emission Probabilities. In *Proceedings of Eurospeech 99, Budapest, Hungary*, September 1999.

FRITSCH J. (1999b). Mixture Trees – Hierarchically Tied Mixture Densities for Scalable Acoustic Modeling. In *Proceedings of LVCSR Hub-5E Workshop, Linthicum Heights, MD*, June 1999.

FUKUNAGA K. (1990). *Introduction to Statistical Pattern Recognition.* Academic Press.

GEMAN S., BIENENSTOCK E., DOURSAT R. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4:1–52.

GERSHO A., GRAY R. (1992). *Vector Quantization and Signal Compression.* Kluwer Academic Publishers.

Geutner P., Finke M., Scheytt P. (1997). Hypothesis Driven Lexical Adaptation for Transcribing Multilingual Broadcast News. Technical Report CMU-LTI-97-155, Carnegie Mellon University, Pittsburgh, PA, December 1997.

Ghahramani Z., Jordan M. I. (1997). Factorial Hidden Markov Models. *Machine Learning*, 29:245–273.

Girosi F., Jones M., Poggio T. (1995). Regularization Theory and Neural Network Architectures. *Neural Computation*, 7:219–269.

Godfrey J. J., C. Holliman E., McDaniel J. (1992). SWITCHBOARD: Telephone Speech Corpus for Research and Development. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 92), San Francisco, CA*, 1992.

Haeb-Umbach R., Ney H. (1992). Linear Discriminat Analysis for Improved Large Vocabulary Continuous Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 92), San Francisco, CA*, 1992.

Hampshire J. B., Waibel A. H. (1989). The Meta-Pi Network: Building Distributed Knowledge Representations for Robust Pattern Recognition. Technical Report CMU-CS-89-166, Carnegie Mellon University, Pittsburgh, PA, August 1989.

Hild H., Waibel A. (1993). Speaker-Independent Connected Letter Recognition with a Multi-State Time Delay Neural Network. In *Proceedings of Eurospeech 93*, pp. 1481–1484, 1993.

Hochberg M. M., Cook G. D., Renals S. J., Robinson A. J., Schechtman R. S. (1995). The 1994 ABBOT Hybrid Connectionist-HMM Large-Vocabulary Recognition System. In *Spoken Language Systems Technology Workshop*, 1995.

Huang X. D., Ariki Y., Jack M. A. (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh University Press.

Hunt M. J., Lennig M., Mermelstein P. (1980). Experiments in Syllable-Based Recognition of Continuous Speech. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 80)*, pp. 880–883, 1980.

Hwang M., Huang X. (1998). Dynamically Configurable Acoustic Models for Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 98), Seattle, WA*, May 1998.

JACOBS R. A. (1988). Increased Rates of Convergence. *Neural Networks*, 1(4).

JACOBS R. A. (1995). Methods for Combining Experts' Probability Assessments. *Neural Computation*, 7:867–888.

JELINEK F., MERIALDO B., ROUKOS S., STRAUSS M. (1991). A Dynamic Language Model for Speech Recognition. In *Proceedings of the Speech and Natural Language DARPA Workshop*, pp. 293–295, February 1991.

JELINEK F. (1997). *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, MA.

JOLLIFFE I.T. (1986). *Principal Component Analysis.* Springer, New York.

JORDAN M. I., JACOBS R. A. (1992). Hierarchies of Adaptive Experts. *Advances in Neural Information Processing*, 4:985–993.

JORDAN M. I., JACOBS R. A. (1994). Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 6:181–214.

JORDAN M. I. (1995). Why the Logistic Function: A Tutorial Discussion on Probabilities and Neural Networks. Technical Report MIT Technical Report 9503, Massachusetts Institute of Technology, April 1995.

KERSHAW D. J., M. HOCHBERG M., ROBINSON A. J. (1995). Context-Dependent Classes in a Hybrid Recurrent Network HMM Speech Recognition System. Technical Report CUED/F-INFENG/TR-217, Cambridge University Engineering Department, Cambridge, England, 1995.

KERSHAW D. J. (1997). *Phonetic Context-Dependency in a Hybrid ANN/HMM Speech Recognition System.* PhD Thesis, St. John's College, Cambridge University Engineering Department, Cambridge, England, 1997.

KNESER R., NEY H. (1995). Improved Backing-Off for m-gram Language Modeling. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 95), Detroit, MI*, pp. 181–184, 1995.

KNILL K. M., GALES M. J. F., YOUNG S. J. (1996). Use of Gaussian Selection in Large Vocabulary Continuous Speech Recognition Systems using HMMs. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 96), Philadelphia, PA*, 1996.

KULLBACK S., LEIBLER R. A. (1951). On Information and Sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Lau R., Rosenfeld R., Roukos S. (1993). Trigger-based Language Models: A Maximum Entropy Approach. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 93), Minneapolis, MN*, 1993.

Lawrence S., Burns I., Back A., Tsoi A. C., Giles C. L. (1998). Neural Network Classification and Prior Class Probabilities. *Neural Networks: Tricks of the Trade, Orr, G. B. and Müller, K. R. (eds), LNCS series, vol. 1524, Springer*, pp. 299–313.

Lee K. F. (1988). *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.* PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, Pittsburgh, PA, 1988.

Leggetter C. J., Woodland P. C. (1994). Speaker Adaptation of HMMs using Linear Regression. Technical Report CUED/F-INFENG/TR-181, Cambridge University Engineering Department, Cambridge, England, 1994.

Lippmann R. P. (1989). Review of Neural Networks for Speech Recognition. *Neural Computation*, 1(1):1–38.

Lippmann R. P. (1997). Speech Recognition by Machines and Humans. *Speech Communication*, 22:1–33.

Martin A., Fiscus J., Fisher B., Pallett D., Przybocki M. (1997). 1997 LVCSR/Hub-5E Workshop: System Descriptions & Performance Summary. In *Proceedings of LVCSR Hub-5E Workshop, Linthicum Heights, MD*, May 1997.

McCullagh P., Nelder J. A. (1989). *Generalized Linear Models.* Chapman & Hall, London, England.

McLachlan G. J., Krishnan T. (1997). *The EM Algorithm and Extensions.* John Wiley & Sons, New York, NY.

Møller M. (1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks*, 6:525–533.

Moody J., Darken C. J. (1989). Fast Learning in Networks of Locally Tuned Processing Units. *Neural Computation*, 1(2):281–294.

Morgan N., Bourlard H. (1990). Continuous Speech Recognition Using Multilayer Perceptrons with Hidden Markov Models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 90), Albuquerque, NM*, pp. 413–416, 1990.

MORGAN N., BOURLARD H. (1992). Factoring Networks by a Statistical Method. *Neural Computation*, 4:835–838.

MORGAN N., BOURLARD H. (1995). An Introduction to Hybrid HMM / Connectionist Continuous Speech Recognition. *Signal Processing Magazine*, 15:25–42.

MORI Y., JOE K. (1989). A Large Scale Neural Network which Recognizes Handwritten Kanji Characters. *Advances in Neural Information Processing*, 2:415–420.

NETO J., ALMEIDA L., HOCHBERG M. M., MARTINS C., NUNES L., RENALS S. J., ROBINSON A. J. (1995). Speaker Adaptation for Hybrid HMM-ANN Continuous Speech Recognition Systems. In *Proceedings of Eurospeech 95, Madrid, Spain*, 1995.

NEY H. (1991). Speech Recognition In A Neural Network Framework: Discriminative Training Of Gaussian Models And Mixture Densities As Radial Basis Functions. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 91), Toronto, Canada*, 1991.

NIESLER T. R., WOODLAND P. C. (1995). Variable-length Category-based n-grams for Language Modeling. Technical Report CUED/F-INFENG/TR-215, Cambridge University Engineering Department, Cambridge, England, April 1995.

ODELL J. J. (1995). *The Use of Context in Large Vocabulary Speech Recognition.* PhD Thesis, Cambridge University Engineering Department, Cambridge, England, Cambridge, England, 1995.

PAUL D. B. (1992). An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 92), San Francisco, CA*, pp. 25–28, 1992.

POGGIO T., GIROSI F. (1990). Networks for Approximation and Learning. *Proceedings of the IEEE*, 78:1481–1497.

PYE D., WOODLAND P. C. (1997). Experiments in Speaker Normalisation and Adaptation for Large Vocabulary Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 97), Munich, Germany*, pp. 1047–1050, 1997.

QUINLAN J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1:81–106.

RABINER L. R., JUANG B. H. (1993). *Fundamentals of Speech Recognition.* Prentice-Hall, Englewood Cliffs, NJ.

RABINER L. R., SCHAFER R. W. (1978). *Digital Processing of Speech Signals.* Prentice-Hall, Englewood Cliffs, NJ.

RABINER L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77:257–285.

RAVISHANKAR M. K. (1996). *Efficient Algorithms for Speech Recognition.* PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, Pittsburgh, PA, 1996.

REDNER R. A., WALKER H. F. (1984). Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, 26.

RENALS S., HOCHBERG M. (1999). Start-Synchronous Search for Large Vocabulary Continuous Speech Recognition. *IEEE Transactions on Speech and Audio Processing*, 7.

RENALS S., MORGAN N., BOURLARD H., FRANCO H., COHEN M. (1991). Connectionist Optimization of Tied Mixture Hidden Markov Models. *Advances in Neural Information Processing*, 4:167–174.

RENALS S. (1989). Radial Basis Function Networks for Speech Pattern Classification. *Electronics Letters*, 25:437–439.

RENALS S. (1996). Phone Deactivation Pruning in Large Vocabulary Continuous Speech Recognition. *IEEE Signal Processing Letters*, 3.

RICHARD M. D., LIPPMANN R. P. (1991). Neural Network Classifiers Estimate Bayesian a posteriori Probabilities. *Neural Computation*, 3:461–483.

RIPLEY B. D. (1996). *Pattern Recognition and Neural Networks.* Cambridge University Press.

ROBINSON A. J., FALLSIDE F. (1991). A Recurrent Error Propagation Network Speech Recognition System. *Computer, Speech and Language*, 5:259–274.

ROBINSON A. J., HOCHBERG M. M., RENALS S. J. (1996). The Use of Recurrent Neural Networks in Continuous Speech Recognition. *In Automatic Speech and Speaker Recognition - Advanced Topics, Lee, C. H., Paliwal K. K. and Soong F. K. (eds).*

ROBINSON A. J. (1994). An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks*, 5.

ROGINA I., WAIBEL A. (1994). Learning State-Dependent Stream Weights for Multi-Codebook HMM Speech Recognition Systems. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 94), Adelaide, Australia*, 1994.

RUMELHART D. E., MCCLELLAND J. L. (EDS) (1986). *Parallel Distributed Processing, Vol. 1 and Vol. 2*. MIT Press, Cambridge, MA.

RUMELHART D. E., HINTON G. E., WILLIAMS R. J. (1986). Learning Internal Representations by Error Backpropagation. *In Rumelhart, D. E. and McClelland, J. L. (eds), Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, MA*, pp. 318–362.

SAFAVIAN S. R., LANDGREBE D. (1991). A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21:660–674.

SCHULTZ T., WAIBEL A. (1998). Multilingual and Crosslingual Speech Recognition. In *Proceedings of the DARPA Broadcast News Workshop 1998, Lansdowne, VA*, February 1998.

SCHÜRMANN J., DOSTER W. (1984). A Decision Theoretic Approach to Hierarchical Classifier Design. *Pattern Recognition*, 17.

SCHÜRMANN J. (1996). *Pattern Classification: A Unified View of Statistical and Neural Approaches*. John Wiley & Sons.

SCHUSTER M., PALIWAL K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45:2673–2681.

SCHWARTZ R., CHOW Y., KIMBALL O., ROUCOS S., KRASNER M., MAKHOUL J. (1985). Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 85)*, April 1985.

SIU M., JONAS M., GISH H. (1999). Using a Large Vocabulary Continuous Speech Recognizer for a Constrained Domain with Limited Training. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 99), Phoenix, AZ*, March 1999.

STROMBERG J., ZRIDA J., ISAKSSON A. (1991). Neural Trees – Using Neural Nets in a Tree Classifier Structure. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 91), Toronto, Canada*, 1991.

TEBELSKIS J. (1995). *Speech Recognition using Neural Networks.* PhD Thesis, Carnegie Mellon University, Pittsburgh, PA, Pittsburgh, PA, May 1995. CMU-CS-95-142.

THOMSON D. L. (1997). Ten Case Studies of the Effect of Field Conditions on Speech Recognition Errors. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 97), Santa Barbara, CA*, May 1997.

TIBSHIRANI R. (1996). Bias, Variance and Prediction Error for Classification Rules. Technical Report, Department of Statistics, University of Toronto, 1996.

TOU J. T., GANZALES R. C. (1974). *Pattern Recognition Principles.* John Wiley & Sons.

VALTCHEV V. (1995). *Discriminative Methods in HMM based Speech Recognition.* PhD Thesis, Cambridge University Engineering Department, 1995.

WAIBEL A., LEE K. F. (Eds.). *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, CA.

WAIBEL A., HANAZAWA T., HINTON G., SHIKANO K., LANG K. (1987). Phoneme Recognition Using Time-Delay Neural Networks. Technical Report TR-1-0006, ATR Interpreting Telephony Research Laboratories, October 1987.

WAIBEL A., SAWAI H., SHIKANO K. (1988). Modularity and Scaling in Large Phonemic Neural Networks. Technical Report TR-I-0034, ATR Interpreting Telephony Research Laboratories, July 1988.

WAIBEL A. (1988). Connectionist Glue: Modular Design of Neural Speech Systems. In TOURETZKY D. S., HINTON G. E., SEJNOWSKI T. J. (Eds.), *Proceedings of the 1988 Connectionist Models Summer School.* Morgan Kaufmann, 1988.

WAIBEL A. (1989). Modular Construction of Time-Delay Neural Networks for Speech Recognition. *Neural Computation, MIT-Press*, 1(1):39–46.

WAIBEL A. (1991). Neural Networks for Speech Recognition. In FURUI S., SONDHI M. (Eds.), *Advances in Speech Signal Processing.* Marcel Dekker Publishers, New York, NY, 1991.

WATANABE T., SHINODA K., TAKAGI K., YAMADA E. (1994). Speech Recognition using Tree-Structured Probability Density Functions. In *Proceedings of International Conference on Spoken Language Processing (ICSLP 94), Yokohama, Japan*, 1994.

WATERHOUSE S. R., COOK G. D. (1996). Esemble Methods for Phoneme Classification. *Advances in Neural Information Processing*, 9.

WATERHOUSE S. R., ROBINSON A. J. (1995). Constructive Algorithms for Hierarchical Mixtures of Experts. *Advances in Neural Information Processing*, 8:584–590.

WATERHOUSE S. R. (1997). *Classification and Regression using Mixtures of Experts*. PhD Thesis, Cambridge University Engineering Department, 1997.

WILLETT D., NEUKIRCHEN C., RIGOLL G. (1998). Efficient Search with Posterior Probability Estimates in HMM-based Speech Recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 98), Seattle, WA*, May 1998.

YEE P. (1992). Classification Experiments involving Back Propagation and Radial Basis Function Networks. Technical Report TR-249, Communications Research Laboratory, McMaster University, Ontario, Canada., 1992.

YOUNG S. J. (1994). The HTK Hidden Markov Model Toolkit: Design and Philosophy. Technical Report CUED/F-INFENG/TR-152, Cambridge University Engineering Department, Cambridge England, 1994.

YOUNG S. J. (1996). Large Vocabulary Continuous Speech Recognition: a Review. Technical Report, Cambridge University Engineering Department, Cambridge England, 1996.

ZEPPENFELD T., FINKE M., RIES K., WESTPHAL M., WAIBEL A. (1997). Recognition of Conversational Telephone Speech using the Janus Speech Engine. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP 97), Munich, Germany*, 1997.

Index