Hidden Markov Models and Neural Networks for Speech Recognition

Ph.D. Thesis

Søren Kamaric Riis

LYNGBY 1998

IMM-PHD-1998-??

IMM

IMM
DEPARTMENT OF MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Lyngby – Denmark

April 20, 1998
sr

Hidden Markov Models and Neural Networks for Speech Recognition

Ph.D. Thesis

Søren Kamaric Riis

LYNGBY 1998

IMM-PHD-1998-??

IMM

ISSN 0909–3192

Printed by IMM, Technical University of Denmark

Abstract

The Hidden Markov Model (HMMs) is one of the most successful modeling approaches for acoustic events in speech recognition, and more recently it has proven useful for several problems in biological sequence analysis. Although the HMM is good at capturing the temporal nature of processes such as speech, it has a very limited capacity for recognizing complex patterns involving more than first order dependencies in the observed data sequences. This is due to the first order state process and the assumption of state conditional independence between observations. Artificial Neural Networks (NNs) are almost the opposite: they cannot model dynamic, temporally extended phenomena very well, but are good at static classification and regression tasks. Combining the two frameworks in a sensible way can therefore lead to a more powerful model with better classification abilities.

The overall aim of this work has been to develop a probabilistic hybrid of hidden Markov models and neural networks and to evaluate this model on a number of standard speech recognition tasks. This has resulted in a hybrid called a *Hidden Neural Network* (HNN), in which the HMM emission and transition probabilities are replaced by the outputs of state-specific neural networks. The HNN framework is characterized by:

Discriminative training: HMMs are commonly trained by the Maximum Likelihood (ML) criterion to model within-class data distributions. As opposed to this, the HNN is trained by the Conditional Maximum Likelihood (CML) criterion to discriminate between different classes. CML training is in this work implemented by a gradient descent algorithm in which the neural networks are updated by *backpropagation* of errors calculated by a modified version of the forward-backward algorithm for HMMs.

Global normalization: A valid probabilistic interpretation of the HNN is ensured by normalizing the model globally at the sequence level during CML training. This is different from the local normalization of probabilities enforced at the state level in standard HMMs.

Flexibility: The global normalization makes the HNN architecture very flexible. Any combination of neural network estimated parameters and standard HMM parameters can be used. Furthermore, the global normalization of the HNN gives a large freedom in selecting the architecture and output functions of the neural networks.

Transition-based modeling: Contrary to HMMs, which model data as sequences of "steady-state" segments, the HNN can focus on "transitional" regions between "steady-state" segments by using neural networks to estimate conditional transition probabilities. Experimental evidence suggest that such "transitional" regions are important for human perception of speech.

The HNN has been evaluated on three standard speech recognition tasks, namely speaker independent recognition of 1) broad phoneme classes, 2) 39 phoneme classes and 3) isolated-word telephone-speech. The TIMIT database has been used for the phoneme experiments, whereas the recently released PhoneBook database has been used for the isolated word experiments. For these tasks, the HNN yields recognition accuracies that compare favorably to results for standard HMMs and other hybrids.

Resumé (Abstract in Danish)

Den skjulte Markov model (HMM) er en af de mest succesfulde metoder inden for talegenkendelse, og har for nyligt også fundet anvendelse inden for analyse af biologiske sekvenser. HMM'en er god til at modelere den tidslige natur i processer så som tale. Imidlertid kan HMM'en kun i begrænset omfang genkende komplekse mønstre med mere end første ordens afhængigheder i de observerede data sekvenser. Dette skyldes første ordens Markov antagelsen, samt antagelsen om tilstands betinget uafhængighed mellem observationer. Kunstige Neurale Netværk (NN) er nærmest modsatte af HMM'en: De kan ikke modelere dynamiske, tidsligt udstrakte fænomener særligt godt, men er gode til "statisk" klassifikation og regression. En fornuftig kombination af HMM'er og NN'er kan derfor lede til en mere effektiv model med bedre klassifikations egenskaber.

Hovedformålet med dette arbejde har været at udvikle en hybrid af skjulte Markov modeller og neurale netværk, samt at evaluere denne model på en række standard talegenkendelses problemer. Dette har resulteret i en hybrid kaldet *skjulte neurale netværk* (HNN), hvori emissions- og overgangs sandsynlighederne i en HMM er erstattet med output'ene fra tilstands specifikke neurale netværk. HNN'en er karakteriseret ved:

Diskriminativ træning: Standard HMM'er trænes sædvanligvis ud fra Maximum Likelihood (ML) kriteriet til at modellere sandsynlighedsfordelinger af data i bestemte klasser. I modsætning hertil trænes HNN'en ud fra Conditional Maximum Likelihood (CML) kriteriet til at skelne eller diskriminere mellem klasser. CML træning af HNN'en er i dette arbejde implementeret med en gradient nedstignings algoritme, hvori de neurale netværk opdateres ved *backpropagation* af fejl beregnet med en modificeret version af forward-backward algoritmen for standard HMM'er.

Global normalisering: En korrekt sandsynlighedsteoretisk fortolkning af HNN'en er sikret ved at normalisere modellen globalt på sekvens niveau. Dette er forskelligt fra den lokale normalisering af sandsynligheder på tilstands niveau, som er påkrævet i standard HMM'er.

Fleksibilitet: Den globale normalisering gør HNN meget fleksibel. Enhver kombination af parametre estimeret med neurale netværk og standard HMM parametre kan benyttes. Den globale normalisering giver endvidere en stor frihed i valget af arkitektur og ouput funktion for de neurale netværk.

Overgangs-baseret modellering: HNN'en kan, i modsætning til HMM'er, fokusere på "overgangs" regioner mellem "stationære" segmenter i datasekvensen ved brug af neurale netværk til estimering af betingede overgangssandsynligheder. Det er experimentelt bevist at sådanne "overgangs" regioner er vigtige for menneskelig perception af tale.

HNN'en er i dette arbejde blevet evalueret på tre standard talegenkendelses problemer, nemlig taler uafhængig genkendelse af 1) brede fonem klasser, 2) 39 fonem klasser og 3) isolerede ord udtalt over en telefon linie. TIMIT databasen er blevet benyttet til fonem eksperimenterne mens den for nyligt udgivne PhoneBook database er blevet benyttet til isoleret ord genkendelses eksperimenterne. HNN'en opnår genkendelses nøjagtigheder for disse tre problemer, som er gode sammenlignet med resultater for standard HMM'er og andre hybrider.

Preface

The present thesis has been submitted in partial fulfillment for the Ph.D. degree in electrical engineering. The work documented in this thesis has been carried out at the Department for Mathematical Modeling, Section for Digital Signal Processing (the former Electronics Institute) at the Technical University of Denmark and was supervised by associate professor Steffen Duus Hansen and research associate professor Anders Krogh. The work was commenced in November 1994 and completed in April 1998 with a six months parental leave in the autumn of 1997.

The reader is assumed to have a basic knowledge of general pattern recognition techniques and artificial neural network modeling. These subjects are not introduced in this thesis due to time and space considerations.

During the Ph.D. study a total of five conference papers and two journal papers have been written,

- Krogh, A. and **Riis, S.** Hidden Neural Networks. Submitted to *Journal of Neural Computation*, 1998.
- **Riis, S.**. Hidden Neural Networks: Application to Speech Recognition. To appear in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Seattle, May 1998.
- **Riis, S.** and Krogh, A. Hidden Neural Networks: A Framework for HMM/NN Hybrids. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Münich, April 1997.
- **Riis, S.** and Krogh, A. Joint estimation of parameters in Hidden Neural Networks. In *Proceedings of IEEE Nordic Signal Processing Symposium*, 431-434, Helsinki, 1996.
- **Riis, S.** and Krogh, A. Improving Protein Secondary Structure Prediction using Structured Neural Networks and Multiple Sequence Profiles. In *Journal of Computational Biology*, **3**:163-183, 1996
- Krogh, A. and **Riis, S**. Prediction of Beta Sheets in Proteins. In *Advances in Neural Information Processing Systems*8*, 1996
- **Riis, S.** Combining Neural Networks for Protein Secondary Structure. In *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, 1995

Postscript files of this thesis and all the above listed papers can be downloaded from the WWW-server at the Section for Digital Signal Processing.[1] The papers relevant to the work described in this thesis are furthermore included in appendix C-F of this thesis.

Acknowledgments

At this point I would like to thank Steffen Duus Hansen and Anders Krogh for their supervision of my Ph.D. project. Especially I wish to express my gratitude to Anders Krogh for the guidance, encouragement and friendship that he managed to extend to me during our almost five years of collaboration. Even during his stay at the Sanger Centre in Cambridge he managed to guide me through the project by always responding to my emails and telephone calls and by inviting me to visit him. Anders' scientific integrity, great intuition, ambition and pleasant company has earned him my respect. Without his encouragement and optimistic faith in this work it might never have come to an end.

The staff and Ph.D. students at the Section for Digital Signal Processing are thanked for creating a very pleasant research environment and for the many joyful moments at the office and during conference trips. Thanks also to Mogens Dyrdahl and everybody else involved in maintaining the excellent computing facilities which were crucial for carrying out my research. Center for Biological Sequence Analysis is also acknowledged for providing CPU-time which made some of the computationally intensive evaluations possible. Similarly, Peter Toft is thanked for learning me to master the force of Linux.

I sincerely wish to express my gratitude to Steve Renals for inviting me to work at the Department of Computer Science, University of Sheffield from February to July 1997. It was a very pleasant and rewarding stay. The Ph.D. students and staff at the Department of Computer Science are acknowledged for their great hospitality and for creating a pleasant research atmosphere. I'm especially grateful to Gethin Williams for the many discussions on hybrid speech recognizers and for proofreading large parts of this thesis. I'm indebted to Gethin for his many valuable comments and suggestions to improve this manuscript.

Morten With Pedersen and Kirsten Pedersen are also acknowledged for their comments and suggestions to this manuscript. Morten is furthermore thanked for the many fruitful discussions we've had and for his pleasant company at the office during the years.

The speech group and in particular Christophe Ris at the Circuit Theory and Signal Processing Lab (TCTS), Faculté Polytechnique de Mons is acknowledged for providing data necessary to carry out the experiments presented in chapter 9 of this thesis.

The Technical University of Denmark is acknowledged for allowing me the opportunity of doing this work. Otto Mønsteds foundation and Valdemar Selmer Trane og Hustru Elisa Trane's foundation is acknowledged for financial support to travel activities.

Last but not least I thank my family and friends for their support, love and care during the Ph.D. study. A special heart-felt thanks goes to my wife and little daughter who helped me maintain my sanity during the study, as I felt myself drowning in ambitions. Without their support this work would not have been possible.

Technical University of Denmark, April 1998

Søren Kamaric Riis

[1] http://eivind.imm.dtu.dk.
See also the authors personal home-page: http://eivind.imm.dtu.dk/dspstaff/riis.html.

Contents

Chapter 1

Introduction

This chapter provides a short introduction to speech recognition systems and presents the motivation for the work documented in this thesis. Section 1.1 reviews the difficulties encountered in speech recognition and section 1.2 presents the general structure of a speech recognition system. The objectives of this work are stated in Section 1.3 and finally section 1.4 provides an overview of the thesis by chapters.

1.1 Dimensions of Difficulty in Speech Recognition

Speech is the most natural mode of communication between human beings. It comes so natural to us that we do not even realize how complex a phenomenon it is. Speech can be characterized as a signal conveyed by an acoustic field, and is the end product of conscious, formalized and voluntary motions of the *respiratory* and *articulatory* apparatus. Thus, speech planning takes place in the mind of a speaker and the actual acoustic wave is generated by a controlled airflow modulating the vocal cords and passing through the vocal tract (oral cavity, nasal cavity *etc.*). The vocal cord vibrations cause a slowly varying periodicity in the speech (voiced sounds, vocals) and the vocal tract shape causes variations in the spectral envelope. Similarly, constrictions in the vocal tract can cause turbulent airflow (unvoiced sounds, consonants). Because the human articulatory system is biological, the generated acoustic wave will be highly dependent on the actual speaker. Hence, factors like age, gender, upbringing and emotional state will affect the generation of speech. The generated acoustic wave is furthermore influenced by the transmission medium through which it passes before reaching the listener. The transmission medium can introduce noise and non-linear distortions but the speech will remain fully intelligible to the listener even for large noise levels or distortions. The listener extracts the message uttered by the speaker using his/her knowledge of the language, topic, context and possibly also visual information like gestures and facial expressions. This process is called *decoding* the speech signal and requires only a minimum of effort from the listener despite the large variations between different speakers and different transmission media.

Because we are so comfortable with speech there is a large desire for being able to interact with machines by speech communication. The ability to automatically *transcribe* speech signals is interesting in a wide number of contexts *e.g.*, computer interaction (dictation, command), telephone home-banking, hands-free operation, information retrieval from databases and automatic language translation. The wide range of potential applications has motivated research in *automatic speech recognition* since the 1950's. Since then speech recognition by machines has improved significantly and there are already several commercial solutions available for operation by single-users in noise-free environments. However, there is still a lot of research to be done when it comes to recognizing natural conversational speech from several different speakers.

The complexity of a speech recognition system can be characterized by the following keywords,

Speaker dependence/independence: A *speaker dependent* system is by definition designed only for use by a single speaker. Conversely, a *speaker independent* system is intended for use by *any* speaker. In general, speaker dependent systems are considerably more accurate than speaker independent systems because the acoustic variation between different speakers is very difficult to describe and model. Intermediate to speaker dependent and speaker independent systems are *multi-speaker* systems aimed at a small but fixed group of people. For both multi-speaker and speaker independent recognizers a large improvement in accuracy can be obtained by *adapting* the recognizer to a specific speaker during operation. Such systems are known as *speaker-adaptive* and work by tuning themselves to a speaker during operation. The adaptation can be done in a supervised fashion from a set of *enrollment* utterances or in an unsupervised fashion by adapting to the user as he/she speaks.

Isolated/connected/continuous speech: In *isolated word* recognition the words must be uttered in isolation and each word is treated independently by the recognizer. *Connected word* recognition requires the words in a sentence to be uttered in isolation separated by artificial periods of silence. However, contrary to isolated word recognition it is here the sequence of words that is of interest and not just each word in the sequence. Connected speech can be viewed as an idealization of *continuous speech* in which sentences are uttered in a natural manner without artificial pauses between words. Continuous speech recognition is by far the most difficult case because the words are not separated by well defined pauses and because word pronunciations are corrupted by *coarticulation*. Coarticulation is a result of the human speech production system not being able to change instantaneously and of the symbol generation and planning process taking place in the speakers mind. Thus, the pronunciation of a word in a sentence is affected by forming the articulators in anticipation of the next word.

Vocabulary size: The number of words in the *vocabulary* is naturally an important factor when assessing the performance of a speech recognizer. Thus, small vocabulary systems (less than 100 words) are usually capable of obtaining close to 100% accuracy even for speaker independent recognition. However, the accuracy of the recognizer also depends on the actual words that are included in the vocabulary. If the words are highly *confusable* it can be quite difficult to obtain 100% accuracy even for very small vocabularies. A good example of a confusable vocabulary is the English "E-set": `''B, C, D, E ,G, P, T, V, Z''`. The E-set has been used for evaluating the ability of speech recognizers to *discriminate* between words with similar acoustic realizations.

Linguistic constraints: The linguistic constraints (possibly specific to a given task) can be imposed by an abstract model of the language. A *language model* of a natural language is composed of four components, namely symbolic, grammatical, semantic and pragmatic. The symbols of a language are defined to be the most natural units from which all messages can be composed. These symbols are sometimes denoted *symbolic message units* (SMUs) and typically represent words or sub-words like *syllables* or *phonemes*. The *grammar* of a language is composed of *lexical* and *syntactic* constraints that describe how words are formed from sub-words and how sentences are formed from words, respectively. *Semantics* is concerned with the way words are combined to form meaningful messages, *e.g.*, the sentence "The dog is talking" is syntactically correct but semantically incorrect. At the highest level of abstraction is the *pragmatics* of a language which describes how the utterances relate to the speakers and the environment. This aspect of a language is hard to formalize but it is illustrated quite well by the sentence "He saw that gas can burn". Depending on the context of the conversation the word "can" is either a verb or a noun. Semantic and pragmatic constraints are rarely used in speech recognition systems because of the difficulty in formalizing these constraints. On the other hand, grammatical constraints are used in almost every continuous speech recognition system as the lexical and syntactical constraints significantly reduce the number of sentences that the recognizer must be able to *hypothesize*.

Read/spontaneous speech: Until recently, most research in speech recognition has focused on *read* speech. However, speech uttered naturally in a *spontaneous* conversational manner is vastly more difficult to recognize by machines. Spontaneous speech is characterized by disfluencies like false starts, incomplete sentences, restarts, laughter and coughing and furthermore the vocabulary is practically unlimited.

Environment: The environment in which the recognizer is supposed to operate naturally also influences the recognition performance. Adverse conditions like environmental noise, acoustic distortions, microphone and transmission channel distortions may degrade performance significantly. In general, if a recognizer designed for clean, noise-free speech is applied without modification in adverse conditions the performance is likely to be very poor. Therefore, recognizers supposed to operate in such environments must take actions to reduce the effects of noise and other degradations of the speech signal. Typically, this is done by using various speech *enhancement* techniques and by letting the system adapt to the environmental conditions in a manner similar to speaker adaption.

The above list indicates that in order to compare speech recognizers it is mandatory to evaluate them under benign conditions. Therefore, a number of standardized databases have been created in order to allow comparisons under well-defined conditions. A few examples of such databases are the TIMIT database of read American English speech, the PhoneBook database of isolated words uttered over North American telephone lines, the DARPA Resource Management database containing read sentences from the field of naval resource management and the ARPA Wall Street Journal database containing speech read from the New York Wall Street Journal. Except for PhoneBook all of these databases contain speech recorded in noise-free environments.

While speaker dependent systems have found their way into commercial applications, large vocabulary speaker independent systems are in general only available as prototypes in a number of speech research labs. A few examples of speaker independent large vocabulary systems are BYBLOS from Bolt, Beranek and Newmann incorporated (BBN) [KCD88], SPINHX from Carnegie-Mellon University [Lee90], HTK from Entropic Research limited [You92b] and ABBOT from Cambridge University [RHR96]. Examples of commercial single-user systems include the recently released (March 1998) versions of ViaVoice from IBM corporation and Dragon NaturallySpeaking from Dragon systems incorporated. ViaVoice and Dragon NaturallySpeaking both aim at speech dictation tasks and runs on standard PCs.

1.2 Conventional Speech Recognition Systems

Speech recognition is basically a pattern recognition task. However, as opposed to conventional "static" pattern classification, speech recognition aims at assigning a *sequence* of class labels (words) to an observed acoustic signal. If the duration of words were fixed *a priori*, speech recognition could be done in much the same way as "static" pattern classification by assigning class labels independently to each fixed length speech segment. However, the time-boundaries between words are not known a priori during recognition and it is thus necessary somehow to *align* hypothesized word sequences to the acoustic signal, *i.e.*, to search for those segments of the speech signal that in some sense optimally represent the hypothesized words. This procedure is commonly referred to as *time-alignment and pattern matching.*

The structure of a general speech recognition system is illustrated in figure 1.1. A brief description of the different components shown in the figure is given below.

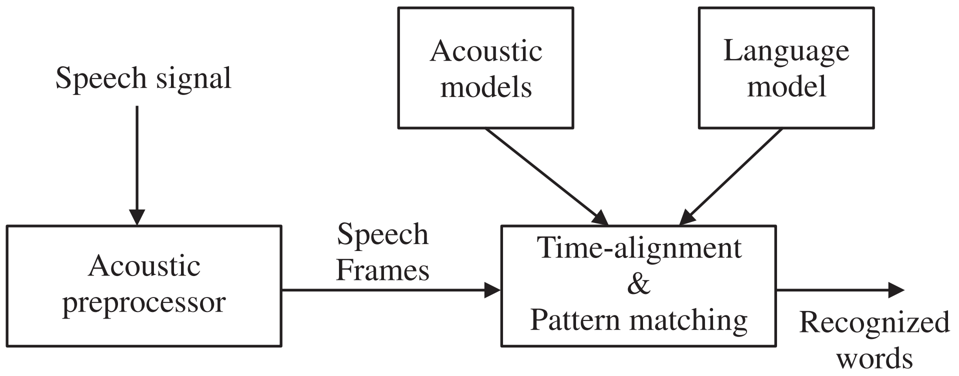

Figure 1.1: Structure of a speech recognition system.

1.2.1 The Preprocessor

The *preprocessor* transforms the raw acoustic waveform into an intermediate compressed representation that is used for subsequent processing. Typically, the preprocessor is capable of compressing the speech data by a factor of ten by extracting a set of *feature vectors* from the speech signal that preserves information about the uttered message. Although the preprocessor is separated from the time alignment and pattern recognition module in figure 1.1, it is in principle an integrated part of the overall classifier. Thus, the optimal preprocessor is nothing but the *Bayes' optimal classifier* [DH73]

In speech recognition the speech signal is assumed *piecewise stationary* and the preprocessor typically yields a feature vector every 10-20ms calculated from a 20-30ms window of speech. The result of preprocessing is thus a sequence of *speech frames* or *feature vectors* at 10ms intervals with 10-30 coefficients per frame. The feature vectors are often augmented by their first and sometimes second order derivatives calculated from linear regression on a number of consecutive speech frames. The *delta-features* provide explicit information about speech dynamics. Commonly used techniques for preprocessing are filterbank analysis, linear prediction analysis, perceptual linear prediction and cepstral analysis. The interested reader is referred to *e.g.*, [DPH93] for details on the different methods.

1.2.2 Time-Alignment and Pattern Matching

The time-alignment and pattern matching process uses information from both the *acoustic model* and the *language model* to assign a sequence of words to the sequence of speech frames. The acoustic model converts the speech frames into symbolic message units of a language like *e.g.*, words, syllables or phonemes that can be concatenated under the constraints imposed by the language model to form meaningful sentences.

Depending on the actual form of the acoustic model there are different ways of doing the temporal alignment. Two of the most popular approaches are *Dynamic Time Warping* and *Hidden Markov Modeling* (HMM). These methods will be described briefly below and an elaboration on the theory of HMMs will be given in chapter 2.

Dynamic Time Warping — A Template-Based Approach

Dynamic time warping is a so-called template-based approach in which the acoustic model is a collection of pre-recorded *word-templates*. The templates typically consist of a representative sequence of feature vectors for the corresponding words. The basic idea in dynamic time warping is to align the utterance to each of the template words and then select the word (or word sequence) that obtains the "best" alignment. For each frame in the utterance, the distance between the template and observed feature vectors are computed using some distance measure and these local distances are accumulated along each possible alignment path, see figure 1.2. The lowest scoring path[1] then identifies the optimal alignment for a word and the word-template obtaining the lowest overall score depicts the recognized word (or word sequence). The alignment can be done with a linear time and memory complexity by so-called *dynamic programming*.

Because of the nature of speech the alignment paths must obey certain local constraints expressing that we can only move forward in both the observed and stored reference templates. Similarly, there are a number of constraints imposed by the language model which dictate the allowed word sequences.

As shown in figure 1.2 the optimal alignment path induces a *segmentation* on the hypothesized word sequence. This segmentation indicates the time-boundaries between words relative to the acoustic signal.

Dynamic time warping is an elegant solution to the time alignment and pattern matching problem. However, there are three serious limitations of this method. Firstly, word-templates cannot model acoustic variability between speakers very well, except in a coarse manner by using multiple templates for each word in the vocabulary. Secondly, only templates representing whole words can be used because, in practice, it is almost impossible to record speech segments shorter than a word, *i.e.*, it is not possible to utter *e.g.*, phonemes in isolation. Thirdly, there is no automatic way of generating representative templates.

[1] Actually, it is often better to evaluate each hypothesis not only by its optimal alignment, but by the composite score of all of its possible alignments. The optimal alignment approach is, however, the most widely used.

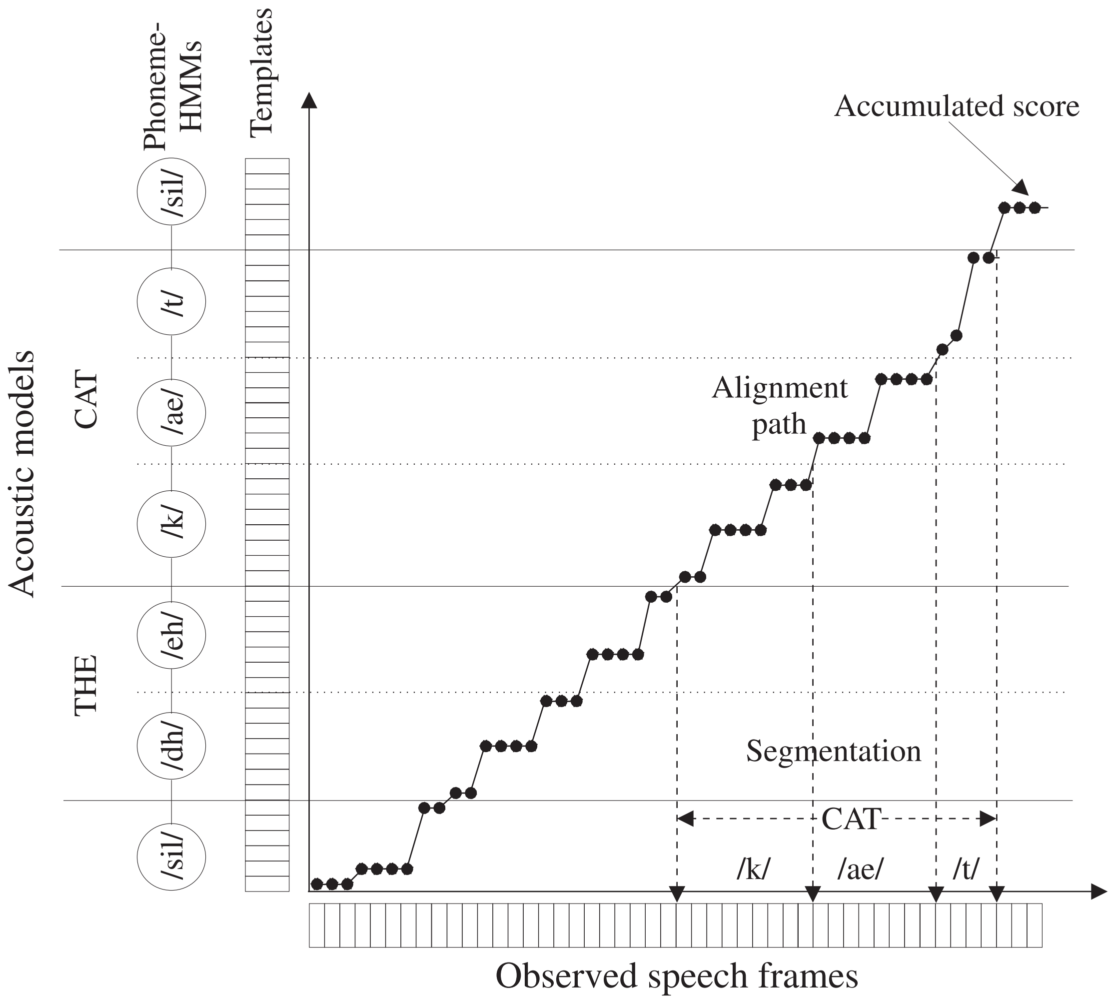

Figure 1.2: Dynamic time warping and HMM decoding by searching for the optimal alignment path. Note that when an HMM acoustic model is used the observed speech frames are aligned directly to the HMM states, *i.e.*, there is only one horizontal sequence of "dots" for each HMM state.

The simplest method is to use all the training utterances as templates but this is likely to result in very poor *generalization* to unseen data.

Hidden Markov Models — A Probabilistic Approach

The HMM removes the need for creating reference templates by using a probabilistic acoustic model. Thus, instead of templates for each SMU (word or sub-word) the HMM defines probability distributions for the assumed stationary speech segments within each SMU class. The advantage of using a probabilistic representation is that the data distributions within each class can be learned automatically from a set of training utterances. Furthermore, the probabilistic representation is far better capable of representing variations between speakers.

Similar to *finite state automatons* the HMM is characterized by a set of states connected by transition probabilities, see figure 1.3. However, in addition to the transition

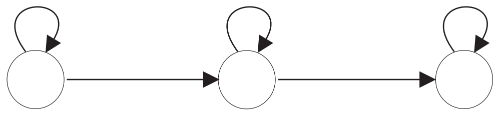

Figure 1.3: Schematic illustration of simple left-to-right connected HMM.

probabilities each state also has assigned[2] a probability density function describing the distribution of speech frames or *observations*. These state-local distributions are commonly denoted *emission* or *observation* distributions. Because of the "forward" nature of speech, the HMM states are usually connected in a left-to-right manner as illustrated in figure 1.3.

Time alignment and pattern matching with an HMM acoustic model is done in much the same way as for the template based approach described above. However, instead of the local distances between reference and observed feature vectors it is now the emission probabilities for the input speech frames in combination with the transition probabilities that are used in the search for the optimal alignment. The dynamic programming match is for HMMs called *Viterbi decoding* and can be viewed as a stochastic version of dynamic time warping because the stored reference "templates" are stochastic quantities. For an HMM acoustic model the segmentation corresponding to the optimal path identifies an association between HMM states and speech frames — a so-called state-segmentation, see figure 1.2.

Ideally, the probability parameters of the SMU-HMMs should be estimated such that the overall error rate is minimized. This can be obtained by maximizing the *a posteriori* probability of the observed word sequence given the observed acoustic signal. However, in many speech recognition systems the HMM for a given SMU is only trained to maximize the probability of the data corresponding to this SMU. This method is known as *Maximum Likelihood* (ML) training and will in practice often lead to HMMs that give large probabilities for other SMUs than the ones they were trained to model. In fact, ML estimation will only be optimal if the HMMs are *correct* models of speech, that is, capable of representing the true within-class data distributions. This is obviously not the case since speech is generated by a non-stationary and highly complex biological process.

As a final remark it should be noted that language models sometimes can be formulated in the probabilistic framework of HMMs. In such cases, large *composite* HMMs can be constructed using SMU-HMMs as intermediate building blocks. Despite the increased complexity of composite HMMs compared to simple acoustic HMMs the basic theory of parameter estimation and decoding remains the same. We will delve into the details of HMM theory and modeling in chapter 2.

1.3 Objectives

HMMs have been used for more than 20 years in the speech recognition area. Although ML estimation has been the main choice for many years due to its computational simplicity and easy implementation, several researches have recently acknowledged the need

[2]This corresponds to a so-called *Moore* type HMM where observations are emitted in the states. The transition-emitting *Mealy* type HMM is slightly more general but is not considered in this work.

for *discriminative* training in order to improve performance. Contrary to ML estimation, discriminative training aims at minimizing the error rate of the model and is not based on assumptions of model correctness. However, even with discriminative training the HMM still suffers from a number of serious limitations. For the past few years the so-called hybrids of HMMs and *artificial Neural Networks* (NN) have gained widespread interest among speech researchers as a way of mitigating some of the basic limitations of standard HMMs. HMM/NN hybrids combine the temporal alignment capabilities of the HMM with the powerful static classification and function approximating capabilities of neural networks. A particular neural network known as the *Multi-Layer Perceptron* (MLP) has for the past decade proven very useful for complex static pattern recognition and is probably the most widely used NN architecture in HMM/NN hybrids. This is commonly attributed to the following two key features of MLPs,

Function approximation: MLPs can in theory approximate any smooth non-linear function by using a hierarchical architecture where inputs are propagated through layers of simple processing units. Each unit or *neuron* typically calculates a weighted sum of its inputs and passes this sum through a non-linear function (typically a sigmoid). The MLP architecture furthermore allows for using any kind of input, *i.e.*, one could use several consecutive speech feature vectors as input or even information from other sources.

Training: MLPs can be trained efficiently by the so-called *Backpropagation* algorithm [RHW86] for either function approximation (regression) or static classification. The backpropagation algorithm works by changing the weights between units in the neural network.

The focus of this thesis, as suggested by its title, is on hybrids of HMMs and neural networks applied to speech recognition. However, contrary to many of the HMM/NN hybrids proposed in the literature, the aim of this work has been to develop a sound probabilistic model in which all parameters of both the overlying Markov model and the neural networks are trained jointly to minimize the same discriminative criterion.

The development is based on a "step-by-step" approach in which discriminative training of standard HMMs is first investigated in the framework of so-called *Class HMMs* (CHMMs). The CHMM was introduced in [Kro94] and can be viewed as a particular extension of the standard HMM architecture that allows for direct maximization of the a posteriori probability of the sequence of class-labels given the observed data. This is also known as discriminative *Conditional Maximum Likelihood* training and is, as mentioned above, more consistent with minimizing the error rate than ML training. One of the key features of the CML trained CHMM is that it can be normalized globally at the sequence level without extra computational burden. This feature makes it easy to extend the CHMM to an HMM/NN hybrid in a very intuitive manner. The hybrid, which is called Hidden Neural Networks (HNN), is based on replacing the standard transition and emission probabilities in the CHMM by the outputs of state-specific neural networks. By imposing a few constraints on the general HNN architecture, it is possible to let this model focus on "transitional" regions in the data sequence, that is, to do so-called *transition-based* modeling. This may be important in speech recognition, because several studies have indicated that the "transitional" regions of a speech signal containing the largest spectral changes over time are important for human perception of speech.

In this work three standard (speaker independent) speech recognition tasks have been selected for evaluating various HNN architectures,

1. Recognition of five broad phoneme classes in continuous read speech from the TIMIT database.
2. Recognition of 39 phonemes in continuous read speech from the TIMIT database.
3. Recognition of isolated-word telephone-speech from the PhoneBook database.

There are two main reasons for selecting these tasks. Firstly, numerous results reported in the literature are available for comparison. Secondly, the broad class task is sufficiently simple to allow a reasonable development and evaluation cycle, while the other two tasks are representative of typical situations encountered in more "real-world" speech recognition. By selecting these tasks, focus is furthermore limited to the acoustic modeling part of speech recognition.

1.4 Thesis Overview

This thesis presents a small step on the way towards improved models for tasks like acoustic modeling in speech recognition. It is divided into ten chapters and six appendices. The contents of the individual chapters and appendices are as follows:

Chapter 2 gives an introduction to the theory of hidden Markov modeling. Since the HMM theory has been extensively described in the literature, chapter 2 primarily discusses issues relevant to the work described in this thesis. The chapter is concluded by a description of practical issues in HMM modeling for speech recognition.

Chapter 3 discusses different strategies for training HMMs. The conventional *Maximum Likelihood* (ML) criterion is compared to different discriminative training criteria which all aim at minimizing the error rate of the HMM as a classifier. Based on this comparison, a particular discriminative criterion known as the *Conditional Maximum Likelihood* (CML) criterion is selected for evaluation in subsequent chapters.

Chapter 4 describes a particular HMM architecture called a *Class HMM* (CHMM) which allows several different classes to be modeled in each state. The CHMM is well suited for discriminative CML training and chapter 4 extends the original formulation given by Krogh in [Kro94] to accommodate the time alignment and pattern matching problem in speech recognition. Furthermore, it is shown how the CHMM can be normalized "globally" at the sequence level without extra computational cost during CML training.

Chapter 5 gives an evaluation of ML and gradient-based CML training of the CHMM on the highly simplified task of speaker independent recognition of five broad phoneme classes in continuous speech from TIMIT.

Chapter 6 describes the new HMM/NN hybrid called *Hidden Neural Networks* (HNN). The HNN is based on a simple extension of the CHMM and makes use of the global normalization property to ensure a valid probabilistic interpretation during CML training. The chapter furthermore describes how a few constraints on the general HNN architecture can lead to a transition-based model. Finally, the HNN is compared to main stream HMM/NN hybrids.

Chapter 7 reconsiders the TIMIT broad phoneme task but in the context of the HNN hybrid. Different HNN architectures including globally and locally normalized transition-based HNNs are evaluated on the broad phoneme task. Due to the limited complexity of the broad class task a thorough investigation of different strategies for initializing and training the HNN is given in this chapter.

Chapter 8 extends the simple broad class task to the "standard" task of recognizing 39 different phonemes in continuous speech from the TIMIT database.

Chapter 9 contains an evaluation of the HNN for speaker independent recognition of isolated words recorded over the existing North American telephone network. The utterances are taken from the PhoneBook database.

Chapter 10 wraps up the overall conclusion an gives directions for further work.

Appendix A describes the TIMIT and PhoneBook databases.

Appendix B elaborates on a particular approach for training CHMMs that allow several different labels in each state. This appendix serves as a reference for future research.

Appendix C-F contains reprints of selected papers which have been authored and co-authored during the Ph.D. study.

Chapter 2

Hidden Markov Models

2.1 Introduction

The root of the HMM theory can be traced back to the 1950s [DLR77, Lev85]. However, it was not until the independent work of Baker at Carnegie-Mellon University [Bak75b, Bak75a] and Jelinek and colleagues at IBM [JBM75, Jel76] in the 1970s that HMMs gradually entered the field of automatic speech recognition. The work by Baum and colleagues in the late 1960s and early 1970s [BP66, BE67, BPS70, Bau72] on efficient algorithms for HMM estimation made the application to speech recognition possible in practice. Nevertheless, template-based approaches like dynamic time warping still tended to dominate the field even in the late 1970s and it was not until the widespread publication of HMM theory and methods in the 1980s that the community experienced a shift in technology. Since then, HMMs have developed into the most successful approaches for acoustic modeling in speech and speaker recognition and are used today by virtually all speech research groups. More recently, HMMs have been successfully applied to other tasks including *e.g.*, protein and DNA modeling [DEKM98, Kro98, Kro97, Edd96, KBM$^+$94, KMH94, BCHM94], human face identification [SH94, Sam93], lip- and speech-reading [SH96, CH96], optical character recognition [LCBB97, BLNB95] and time series prediction [FD93].

To establish notation and set the stage for describing class HMMs and the hidden neural network hybrid, we give a brief introduction to standard hidden Markov models in this chapter. The chapter begins by reviewing the basic concepts of discrete HMM modeling and is concluded with a discussion of issues regarding the use of HMMs for speech recognition in practice. The HMM theory has been described extensively in the literature and the aim of the present chapter is not to give an in-depth coverage of the HMM theory and techniques but rather to describe elements considered relevant to this thesis. For a more comprehensive introduction, the reader is referred to the excellent tutorial-style reviews given in [LRS83, Rab89, Pic90, JR91, RJ93] which also contain numerous references.

2.2 HMM Architecture

The HMM is an extension of the well known Markov model (see *e.g.*, [LG89]) which defines a probability distribution over sequences of discrete or continuous valued random variables $\boldsymbol{\pi} = \boldsymbol{\pi}_1^L = \pi_1, \ldots, \pi_L$. For a k'th order Markov model the following conditional independence assumptions are made,

$$P(\boldsymbol{\pi}_1^L) = \prod_{l=1}^{L} P(\pi_l|\boldsymbol{\pi}_1^{l-1}) = \prod_{l=1}^{L} P(\pi_l|\boldsymbol{\pi}_{l-k}^{l-1}). \tag{2.1}$$

where $\boldsymbol{\pi}_{l-k}^{l-1} = \pi_{l-k}, \ldots, \pi_{l-1}$ (for $l \leq k$ we define $\boldsymbol{\pi}_{l-k}^{l-1} = \boldsymbol{\pi}_1^{l-1}$). The variables π_l are usually called *state variables* and for a k'th order Markov model $\boldsymbol{\pi}_{l-k}^{l-1}$ *summarizes* all relevant past information to derive the current state π_l. Markov models can be used to model sequential data like speech and biological sequences (see *e.g.*, [DEKM98]) and the k'th order model gives a good fit if the data satisfies the k'th order Markov assumption. However, the larger the model order k, the larger the datasets required in order to estimate the conditional probabilities $P(\pi_l|\pi_{l-k}^{l-1})$ reliably. Therefore, k is usually restricted to be quite small in practice which is clearly inappropriate for many applications.

In practice, it is often observed that, at least approximately, all relevant past data up to time l can be summarized in a set of state variables. This is exactly the idea in hidden Markov models, where the data sequence is modeled as a *piecewise stationary* process. A k'th order HMM is characterized by a set of N numbered discrete states,[1] $\pi = i, \ldots, N$, and two concurrent stochastic processes; (1) a k'th order Markov process modeling the temporal structure of the data and (2) an *emission* process assigned to each state modeling the assumed locally stationary part of the data, see figure 2.1. The Markov process of an HMM is hidden — all we can observe is the output of the emission processes. Hence, the name hidden Markov model. Any k'th order HMM can be emulated by a first order HMM simply by increasing the number of states. First order HMMs are therefore usually preferred because they are conceptually simple and because they can be trained using the very efficient *Baum-Welch reestimation* algorithm. Here we will only consider the first order HMM.

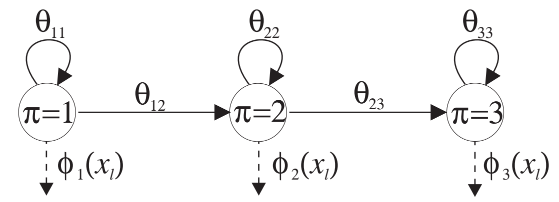

Figure 2.1: A simple first order discrete HMM.

The Markov process in first order HMMs is commonly described by *homogeneous transition probabilities*; the probability of a transition from state i to state j is given by,

$$\theta_{ij} = P(\pi_l = j|\pi_{l-1} = i), \tag{2.2}$$

which is independent of the time index l. Often prior knowledge is incorporated by forcing some transition probabilities to zero, *e.g.*, in speech recognition a left-to-right architecture is typically used in favor of an ergodic (fully connected) model.

The emission process in state i can take different forms depending upon whether the data sequence consists of discrete or continuously valued observations. For simplicity, we here assume that the observations are discrete symbols from a finite alphabet $\mathcal{A}$. The case of a continuously valued observation variable will be treated in section 2.7. When the

[1]The state variables of an HMM are discrete and the possible values are denoted *states*.

observations are discrete the model is often denoted a *discrete* HMM, and the probability of *emitting* symbol $x_l = a$ from the alphabet $\mathcal{A}$ at any time l in state i is given by,

$$\phi_i(a) = P(x_l = a|\pi_l = i). \tag{2.3}$$

Similar to the transition probabilities, the probability of emitting symbol a in state i is the same no matter where the symbol occurs in the data sequence.

The set of all transition and emission probabilities

$$\mathbf{\Theta} = \{i, j = 1, \ldots, N; a \in \mathcal{A}|\theta_{ij}, \phi_i(a)\} \tag{2.4}$$

completely specifies the discrete HMM.

Given a first order HMM, the probability of a data sequence of L discrete observations $\boldsymbol{x} = \boldsymbol{x}_1^L = x_1, \ldots, x_L$ can be written,

$$P(\boldsymbol{x}; \mathbf{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{\pi}; \mathbf{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}_1^L, \boldsymbol{\pi}_1^L; \mathbf{\Theta}). \tag{2.5}$$

Here $\boldsymbol{\pi} = \boldsymbol{\pi}_1^L = \pi_1, \ldots, \pi_L$ is a state sequence; π_l is simply the number of the l'th HMM state in the sequence. Such a state sequence is usually called a *path* through the model. Since the paths are mutually exclusive $P(\boldsymbol{x}; \mathbf{\Theta})$ is simply the sum of the joint probability $P(\boldsymbol{x}, \boldsymbol{\pi}; \mathbf{\Theta})$ of each path. Using Bayes rule the joint probability can be written (conditioning on $\mathbf{\Theta}$ is dropped for notational convenience),

$$\begin{aligned}
P(\boldsymbol{x}_1^L, \boldsymbol{\pi}_1^L) &= P(x_L, \pi_L|\boldsymbol{x}_1^{L-1}, \boldsymbol{\pi}_1^{L-1}) P(\boldsymbol{x}_1^{L-1}, \boldsymbol{\pi}_1^{L-1}) \\
&= P(x_L|\boldsymbol{x}_1^{L-1}, \boldsymbol{\pi}_1^{L}) P(\pi_L|\boldsymbol{x}_1^{L-1}, \boldsymbol{\pi}_1^{L-1}) P(\boldsymbol{x}_1^{L-1}, \boldsymbol{\pi}_1^{L-1}) \\
&\vdots \\
&= \prod_{l=1}^{L} P(x_l|\boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l}) P(\pi_l|\boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l-1}).
\end{aligned} \tag{2.6}$$

In the first order HMM it is assumed that (1) the Markov process is first order and independent of the observation sequence and (2) the observations are state conditionally independent, that is, all relevant past data is summarized in the current state variable π_l. Formally, this assumption is

$$P(\pi_l|\boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l-1}) = P(\pi_l|\pi_{l-1}) \tag{2.7}$$
$$P(x_l|\boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l}) = P(x_l|\pi_l), \tag{2.8}$$

and (2.6) simplifies to a product over l of two "state-local" probabilities[2]

$$P(\boldsymbol{x}, \boldsymbol{\pi}) \approx \prod_{l=1}^{L} P(x_l|\pi_l) P(\pi_l|\pi_{l-1})$$
$$\Downarrow$$
$$P(\boldsymbol{x}; \mathbf{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{\pi}; \mathbf{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l} \phi_{\pi_l}(x_l). \tag{2.9}$$

[2]The approximation in (2.8) defines a so-called *Moore* form HMM. If the observations are assumed conditionally independent on the state pair $\boldsymbol{\pi}_{l-1}^{l}$, then $P(x_l|\boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l}) \approx P(x_l|\pi_l, \pi_{l-1})$. This defines a so-called *Mealy* form HMM where the emission processes are associated with the transitions between states.

An auxiliary zeroth non-emitting state, $\pi_0 = 0$, termed the *begin state* has been introduced in (2.9) so that $\theta_{0j} = P(\pi_1 = j|\pi_0 = 0; \mathbf{\Theta})$ denotes the probability of initiating a path in state j.

Strictly speaking, $P(\boldsymbol{x}; \mathbf{\Theta})$ as given by (2.9) is only a valid probabilistic model if all observation sequences have the same length L. If this is not the case, it is straightforward to see that the sum, $\sum_{\boldsymbol{x} \in \mathcal{X}} P(\boldsymbol{x}; \mathbf{\Theta})$, over the space $\mathcal{X}$ of all possible sequences is infinite. To properly model sequences of different lengths such that $\sum_{\boldsymbol{x} \in \mathcal{X}} P(\boldsymbol{x}; \mathbf{\Theta}) = 1$, one of the states in the model must be an end state, *i.e.*, a state with no out-going transitions. Alternatively, a termination symbol can be added to the alphabet. In the following we will use an auxiliary $N+1$'th non-emitting state, $\pi_{L+1} = N+1$, called the *end state*, with no outgoing transitions in order to ensure proper length modeling.

It is often useful to think of an HMM as a generative model, where sequences are generated by random walks from state to state governed by the transition probabilities θ_{ij} and emission probabilities $\phi_i(a)$. Here, however, the HMM framework will be used for assigning a class or a sequence of classes to the observed data sequence, so instead we wish to interpret the emission probability in state i as the probability that the current observation *matches* this state. Throughout the rest of this thesis the term *match probability* will therefore be used for $\phi_i(a)$. The reason for this becomes more clear when we introduce the concept of global normalization and the hidden neural network hybrid.

Given the formal definition of HMMs, we will now examine the following three classic issues in HMM modeling:

Probability calculation. For a given HMM how do we compute the probability $P(\boldsymbol{x}; \mathbf{\Theta})$ of an observation sequence?

Decoding. When using HMMs for speech recognition or other classification tasks, how do we find the class or sequence of classes corresponding to an observation sequence?

Estimation. How do we estimate the parameters $\mathbf{\Theta}$ of an HMM given a set of training data sequences?

2.3 Probability Calculation

Equation (2.9) for computing the probability of an observation sequence is very intuitive; it states that we just have to multiply match and transition probabilities along each possible path and then add these products together. Unfortunately, this is intractable in practice even for sequences of moderate lengths because the number of possible paths grows as $\mathcal{O}(N^L)$. Each path requires approximately $2L$ floating point operations (flops) and a direct computation of the probability would thus require as much as $\mathcal{O}(2LN^L)$ flops. There is, however, a very efficient way of doing the probability calculation. The *forward algorithm* [BE67] is a dynamic programming algorithm which exploits the constrained HMM architecture to compute the probability of an observation sequence using only $\mathcal{O}(N^2L)$ flops. It is based on the *forward* variable $\alpha_i(l)$ defined as the joint probability of matching the partial observation sequence $\boldsymbol{x}_1^l = x_1, \ldots, x_l$ and being in state i at time l,

$$\alpha_i(l) = P(\boldsymbol{x}_1^l, \pi_l = i; \mathbf{\Theta}). \tag{2.10}$$

Suppose that we now unfold the HMM in time to form a lattice as illustrated in figure 2.2, and suppose that at time l we have arrived at state i and have computed $\alpha_i(l)$. From the lattice it is now very easy to compute $\alpha_j(l+1)$ by utilizing the standard HMM assumptions

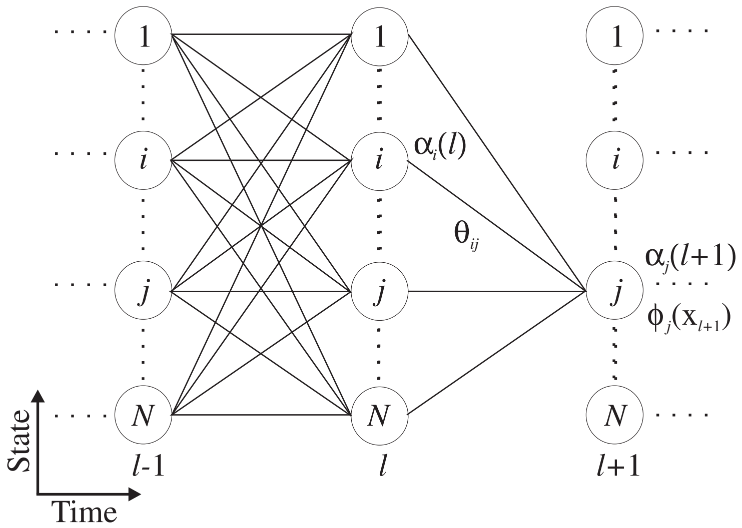

Figure 2.2: State lattice illustration of forward recursion.

$$
\begin{aligned}
\alpha_j(l+1) &= \sum_{i=1}^{N} P(\boldsymbol{x}_1^{l+1}, \pi_{l+1} = j, \pi_l = i; \boldsymbol{\Theta}) \\
&= \sum_{i=1}^{N} P(\boldsymbol{x}_1^{l}, \pi_l = i) P(\pi_{l+1} = j | \pi_l = i, \boldsymbol{x}_1^l) P(x_{l+1} | \pi_{l+1} = j, \pi_l = i, \boldsymbol{x}_1^l) \\
&= \phi_j(x_{l+1}) \sum_{i=1}^{N} \theta_{ij} \alpha_i(l). \qquad (2.11)
\end{aligned}
$$

The probability $P(\boldsymbol{x}; \boldsymbol{\Theta})$ is now simply computed as

$$
P(\boldsymbol{x}; \boldsymbol{\Theta}) = \sum_{i=1}^{N} \theta_{iN+1} \alpha_i(L), \qquad (2.12)
$$

where state $N+1$ is the non-matching end state. The forward algorithm is summarized in algorithm 2.1.

Algorithm 2.1 Forward algorithm $[P(\boldsymbol{x}; \boldsymbol{\Theta})]$

Definition:	$\alpha_j(l) = P(\boldsymbol{x}_1^l, \pi_l = i; \boldsymbol{\Theta})$	
Initialization:	$\alpha_j(1) = \phi_j(x_1)\theta_{0j}$,	$1 \le j \le N$
Recursion:	$\alpha_j(l) = \phi_j(x_l) \sum_i \theta_{ij} \alpha_i(l-1)$,	$1 \le j \le N, 1 < l \le L$
Termination:	$P(\boldsymbol{x}; \boldsymbol{\Theta}) = \sum_i \theta_{iN+1} \alpha_i(L)$	

When implementing the forward algorithm care must be taken to avoid numerical underflow due to the repeated multiplication of small probabilities. The underflow problem can be handled by scaling the forward variables at each time step as discussed in [Rab89].

The forward algorithm considers *all* paths through the model and the probability calculation can be viewed as a weighted average of probabilities given each path; $P(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}|\boldsymbol{\pi};\boldsymbol{\Theta})P(\boldsymbol{\pi};\boldsymbol{\Theta})$. If only one particular path gives a significant contribution to the weighted average it is in practice sufficient to consider this path only. The *Viterbi algorithm* [Vit67, For73] utilizes this fact by approximating $P(\boldsymbol{x};\boldsymbol{\Theta})$ with the probability corresponding to the optimal path $\hat{\boldsymbol{\pi}}$,

$$P(\boldsymbol{x};\boldsymbol{\Theta}) \approx P(\boldsymbol{x},\hat{\boldsymbol{\pi}};\boldsymbol{\Theta}), \tag{2.13}$$

where

$$\hat{\boldsymbol{\pi}} = \operatorname*{argmax}_{\boldsymbol{\pi}} P(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}) = \operatorname*{argmax}_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\Theta}). \tag{2.14}$$

The last equality holds because $P(\boldsymbol{x};\boldsymbol{\Theta})$ is independent of the path. The Viterbi algorithm is an efficient dynamic programming algorithm which is simply obtained by replacing the summations in the forward algorithm by maximization operations. Define $\alpha_j^*(l) = P(\boldsymbol{x}_1^l, \hat{\pi}_1,\ldots,\hat{\pi}_l = j;\boldsymbol{\Theta})$, where $\hat{\pi}_1,\ldots,\hat{\pi}_l = j$ is the the optimal path leading to state j at time l. Then the Viterbi algorithm can be expressed as follows:

Algorithm 2.2 Viterbi algorithm $[P(\boldsymbol{x},\hat{\boldsymbol{\pi}};\boldsymbol{\Theta})]$

Definition:	$\alpha_j^*(l) = P(\boldsymbol{x}_1^l, \hat{\pi}_1,\ldots,\hat{\pi}_l = j;\boldsymbol{\Theta})$	
Initialization:	$\alpha_j^*(1) = \phi_j(x_1)\theta_{0j}$,	$1 \le j \le N$
Recursion:	$\alpha_j^*(l) = \phi_j(x_l)\max_i \theta_{ij}\alpha_i^*(l-1)$,	$1 \le j \le N, 1 < l \le L$
Termination:	$P(\boldsymbol{x},\hat{\boldsymbol{\pi}};\boldsymbol{\Theta}) = \max_i \theta_{iN+1}\alpha_i^*(L)$	

Since no summations are involved in the Viterbi recursion, numerical underflow is easily avoided by representing the probabilities by their logarithms.

The Viterbi algorithm is approximately of the same computational complexity as the forward algorithm — it requires about NL fewer additions than the forward algorithm. It thus seems that the Viterbi algorithm can only be justified for calculating the probability of an observation sequence if the best path approximation, $P(\boldsymbol{x};\boldsymbol{\Theta}) \approx P(\boldsymbol{x},\hat{\boldsymbol{\pi}};\boldsymbol{\Theta})$, is very accurate. However, the Viterbi algorithm is widely used for estimating and decoding HMMs as discussed later. It is interesting to note that the Viterbi algorithm is actually similar to a "stochastic" version of the dynamic time warping approach discussed in chapter 1 — an allusion to the fact that the stored "reference templates" are stochastic quantities.

2.4 Decoding

Decoding with an HMM is the task of assigning a class or a sequence of classes to an observation sequence. An example of the former is isolated word recognition, where the aim is to classify a speech signal as one of the possible words from a given vocabulary. Continuous speech recognition constitutes a good example of how to use an HMM for assigning a sequence of words to an utterance. The case of one class per observation sequence will be described below. The case of assigning a sequence of class-labels to the data is treated in section 2.7.

The *Bayes optimal classifier* minimizes the probability of error by selecting the class y with *Maximum A Posteriori* (MAP) probability [DH73],

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{C}} P(y|\boldsymbol{x}), \tag{2.15}$$

where $\mathcal{C}$ is the set of possible classes. This is also known as *minimum error classification.* Using Bayes' rule and the fact that $P(\boldsymbol{x})$ is independent of y, we see that (2.15) is equivalent to,

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{C}} P(\boldsymbol{x}|y)P(y). \tag{2.16}$$

In (2.16) $P(\boldsymbol{x}|y)$ represents the class conditional probability and $P(y)$ the *a priori* probability for class y.

2.4.1 Forward Decoding

Assume that we have a set $\mathcal{C} = \{1, \ldots, C\}$ of C different classes and one HMM for each of these classes, $\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_C$. The collection of class specific HMMs can be gathered into one large HMM $\boldsymbol{\Theta} = \{c = 1, \ldots, C|\boldsymbol{\Theta}_c, \theta_{0c}\}$ as illustrated in figure 2.3.

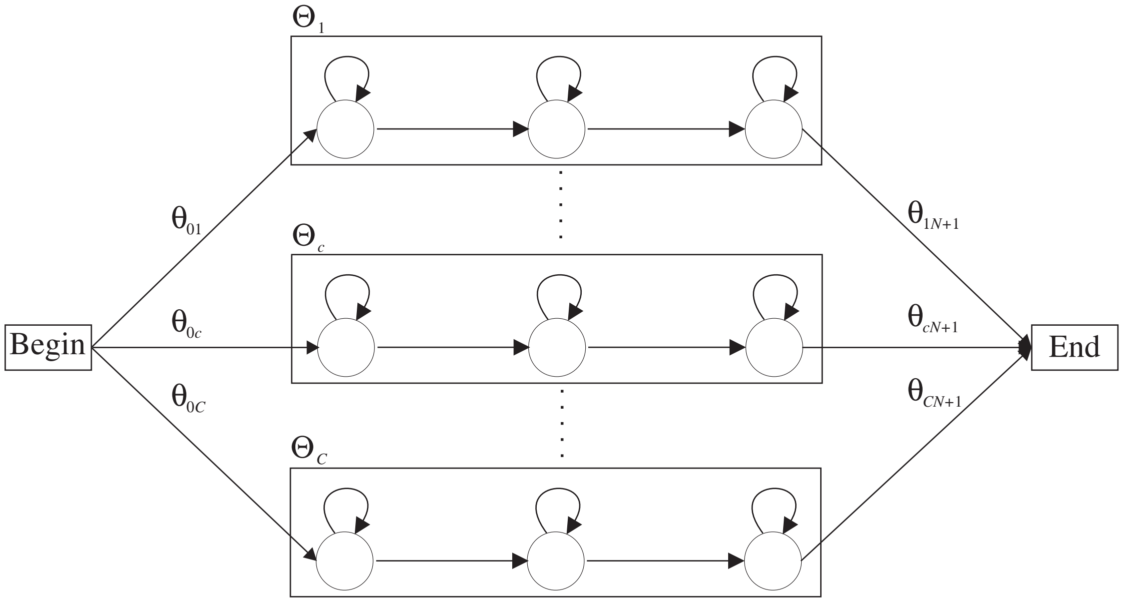

Figure 2.3: An HMM for sequence classification.

For the model in figure 2.3 the MAP decoder can be expressed,

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{C}} P(\boldsymbol{x}|y;\boldsymbol{\Theta})P(y;\boldsymbol{\Theta}) = \operatorname*{argmax}_{y \in \mathcal{C}} P(\boldsymbol{x};\boldsymbol{\Theta}_y)\theta_{0y}, \tag{2.17}$$

where $\boldsymbol{\Theta}_y$ is the sub-HMM representing class y, and θ_{0y} represents the a priori probability of this class. Now, the forward algorithm is used to compute the probability for each of the competing sub-HMMs and these are multiplied by the a priori probabilities of the corresponding classes. The largest of these products then identifies the optimal class.

2.4.2 Viterbi Decoding

The Viterbi algorithm can be used instead of the forward algorithm, but in this case the optimal class is selected according to,

$$\hat{y} = \underset{y \in \mathcal{C}}{\operatorname{argmax}}\, P(\boldsymbol{x}, \hat{\boldsymbol{\pi}} | y; \boldsymbol{\Theta}) P(y; \boldsymbol{\Theta}). \tag{2.18}$$

The advantage of the Viterbi algorithm compared to the forward algorithm is that the optimal path $\hat{\boldsymbol{\pi}}$ can be computed by *backtracking*. By saving the argument of the maximization operations in algorithm 2.2 in a *backtracking pointer* $\delta_i(l)$ the optimal path can be found as shown in figure 2.4 and algorithm 2.3. Knowledge of the optimal path makes it possible to align observations to HMM states, *i.e.*, to segment the data sequence into time intervals spent in each of the states. For applications such as continuous speech recognition this is extremely useful, as will become apparent in section 2.7, and it is probably the main reason for the widespread use of the Viterbi algorithm. Note that for the simple classification example given in figure 2.3 the optimal path will pass through only one of the class submodels and thereby identify the recognized class; $\hat{y} = y(\hat{\boldsymbol{\pi}})$.

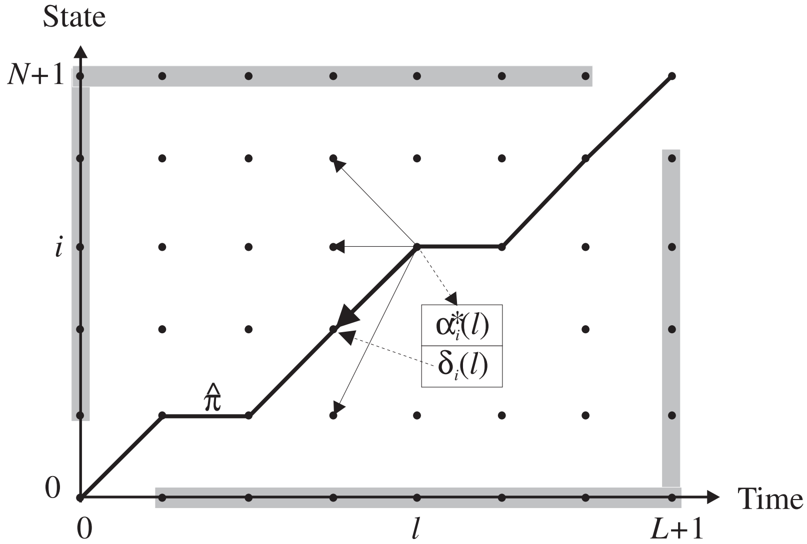

Figure 2.4: Viterbi backtracking.

Algorithm 2.3 Viterbi backtracking [$\hat{\boldsymbol{\pi}} = \hat{\pi}_1, \ldots, \hat{\pi}_L$]

Definition:	$\delta_j(l)$ = best predecessor state at time l	
Initialization:	$\delta_j(1) = 0,$	$1 \le j \le N$
Recursion:	$\delta_j(l) = \operatorname{argmax}_i \theta_{ij} \alpha_i^*(l-1),$	$1 \le j \le N, 1 < l \le L$
Termination:	$\hat{\pi}_L = \operatorname{argmax}_i \theta_{iN+1} \alpha_i^*(L)$	
Backtracking:	$\hat{\pi}_l = \delta_{\hat{\pi}_{l+1}}(l+1),$	$l = L-1, \ldots, 1$

An alternative way of associating a path with an observation sequence is by selecting the state with the maximum a posteriori probability at time l,

$$\hat{\pi}_l = \operatorname*{argmax}_i P(\pi_l = i|\boldsymbol{x};\boldsymbol{\Theta}) = \operatorname*{argmax}_i n_i(l). \tag{2.19}$$

The state posteriors $n_i(l)$ can be computed by the *forward-backward* algorithm which will be shown later. Note that the "optimal" path found using (2.19) not necessarily corresponds to a valid path through the model.

2.5 Maximum Likelihood Estimation

So far we have assumed that the parameters of the model are given. In this section we will show how the parameters can be estimated from data using a *Maximum Likelihood* (ML) based method. A discussion of theoretical and practical issues in ML estimation will be given in chapter 3.

Let the data set (training set) of K symbol sequences from the alphabet $\mathcal{A}$ be denoted,

$$\mathcal{D} = \{\boldsymbol{x}_1^{L_k}(k); k = 1, \ldots, K\}. \tag{2.20}$$

L_k is the number of observations in training sequence k. Given the model $\boldsymbol{\Theta}$, the sequences $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(K)$ are assumed to be independent, and the total likelihood is[3]

$$L(\boldsymbol{\Theta};\mathcal{D}) = \prod_{k=1}^{K} P(\boldsymbol{x}(k);\boldsymbol{\Theta}). \tag{2.21}$$

Because of the independence assumption we consider only one training sequence hereafter without loss of generality. Generalization to multiple sequences is straightforward by placing sums or products over the K sequences at appropriate places.

The objective of maximum likelihood estimation is to find the parameters $\hat{\boldsymbol{\Theta}}_{ML}$ that maximize (2.21),

$$\hat{\boldsymbol{\Theta}}_{ML} = \operatorname*{argmax}_{\boldsymbol{\Theta}} L(\boldsymbol{\Theta};\mathcal{D}). \tag{2.22}$$

To maximize the likelihood, one can apply several different methods. The most widely used is probably the Baum-Welch or forward-backward algorithm [BE67, BPS70, Bau72] which is a particular implementation of the Expectation-Maximization (EM) algorithm [DLR77] for HMMs. The Baum-Welch algorithm increases the likelihood monotonically by iteratively *reestimating* the transition and match probabilities using a set of reestimation formulas. A very pleasing property of the Baum-Welch algorithm is that it guarantees convergence of (2.21) to a local maximum. The derivation of the reestimation formulas and the proof of convergence given in the papers by Baum *et al.* is rather intricate. Based on the formalism of the EM algorithm described in the following section it is, however, simple to derive the reestimation formulas and prove convergence. The introduction to the EM algorithm given below is based on [YHS98, Yos96].

[3]Note that we use the same symbol, $P(\cdot;\cdot)$, for likelihoods and probabilities. It should, however, be clear from the context whether $P(\boldsymbol{x};\boldsymbol{\Theta})$ denotes the probability of $\boldsymbol{x}$ given by the model $\boldsymbol{\Theta}$ or the likelihood given the observation sequence.

2.5.1 The EM Algorithm

The EM algorithm is an iterative approach to maximum likelihood estimation proposed by Dempster *et al.* [DLR77] which is useful when the optimization problem can be simplified by introducing an additional *missing* or *hidden* variable. A very appealing property of the EM algorithm is that it guarantees a monotonic increase of the likelihood to a local maximum by performing two separate steps at each iteration; an Estimation (E) step and a Maximization (M) step. The algorithm converges at a linear rate [DLR77] which is slow compared to the ideal convergence rate of *e.g.*, second order methods such as the Newton-Raphson family of algorithms. The simplicity and guarantee of convergence is nevertheless very attractive to many applications.

Let the (*incomplete data*) likelihood be denoted $P(\boldsymbol{x};\boldsymbol{\Theta})$, where $\boldsymbol{\Theta}$ are the parameters we wish to estimate and $\boldsymbol{x}$ is the (incomplete) data. Furthermore, assume $\boldsymbol{z}$ is the missing data such that $P(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Theta})$ denotes the *complete data* likelihood. The EM procedure is attractive only when the maximization of the complete data likelihood is more tractable than the maximization of the incomplete data likelihood. Using Bayes' rule we find,

$$P(\boldsymbol{x};\boldsymbol{\Theta}) = \frac{P(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Theta})}{P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta})} \tag{2.23}$$

and taking the logarithm, multiplying by $P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta})$ and summing over $\boldsymbol{z}$ yields,

$$\begin{aligned}\log P(\boldsymbol{x};\boldsymbol{\Theta}) &= \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta}) \log P(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Theta}) - \sum_{\boldsymbol{z}} P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta}) \log P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta}) \\ &= E_{\boldsymbol{z}}[\log P(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Theta})|\boldsymbol{x},\boldsymbol{\Theta}] - E_{\boldsymbol{z}}[\log P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Theta})|\boldsymbol{x},\boldsymbol{\Theta}]. \qquad (2.24)\end{aligned}$$

Now define the $\mathcal{Q}$ and the $\mathcal{H}$ auxiliary functions by,

$$\begin{aligned}\mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta}) &= E_{\boldsymbol{z}}[\log P(\boldsymbol{x},\boldsymbol{z};\boldsymbol{\Phi})|\boldsymbol{x},\boldsymbol{\Theta}] \qquad (2.25)\\ \mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta}) &= E_{\boldsymbol{z}}[\log P(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\Phi})|\boldsymbol{x},\boldsymbol{\Theta}]. \qquad (2.26)\end{aligned}$$

Then, the (incomplete) data log likelihood is a difference between the $\mathcal{Q}$ and $\mathcal{H}$ functions,

$$\log P(\boldsymbol{x};\boldsymbol{\Theta}) = \mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) - \mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) \tag{2.27}$$

and for the two parameter sets $\boldsymbol{\Theta}$ and $\boldsymbol{\Phi}$ the log likelihood difference becomes,

$$\log P(\boldsymbol{x};\boldsymbol{\Theta}) - \log P(\boldsymbol{x};\boldsymbol{\Phi}) = \{\mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) - \mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta})\} - \{\mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) - \mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta})\}. \tag{2.28}$$

From Jensen's inequality,[4] the following inequality always holds,

$$-\{\mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}) - \mathcal{H}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta})\} \leq 0, \tag{2.29}$$

with equality if and only if $\boldsymbol{\Theta} = \boldsymbol{\Phi}$. Consequently, if the parameters $\boldsymbol{\Phi}$ are chosen such that,

$$\mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Phi}|\boldsymbol{\Theta}) \geq \mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}), \tag{2.30}$$

[4] Jensen's inequality states that $E[g(z)] \geq g(E[z])$ for any convex function $g(z)$.

then

$$\log P(\boldsymbol{x}; \boldsymbol{\Phi}) \geq \log P(\boldsymbol{x}; \boldsymbol{\Theta}). \tag{2.31}$$

Equations (2.30) and (2.31) suggest an algorithm that iteratively estimates the parameters $\boldsymbol{\Theta}$ such that,

$$P(\boldsymbol{x}; \boldsymbol{\Theta}^{(t+1)}) \geq P(\boldsymbol{x}; \boldsymbol{\Theta}^{(t)}) \tag{2.32}$$

is satisfied for a sequence $\boldsymbol{\Theta}^{(0)}, \boldsymbol{\Theta}^{(1)}, \ldots, \boldsymbol{\Theta}^{(t)}, \boldsymbol{\Theta}^{(t+1)}, \ldots$ of parameter sets. Such an approach implies that the likelihood increases monotonically until a (local) maximum is reached. The EM algorithm is summarized in algorithm 2.4.

Algorithm 2.4 EM algorithm

Initialization:	$\boldsymbol{\Theta}^{(0)} = \text{Initial guess}$	
E-step:	Given $\boldsymbol{\Theta}^{(t)}$, compute $\mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Theta}\vert\boldsymbol{\Theta}^{(t)})$,	$t = 1, 2, \ldots$
M-step:	Update parameters $\boldsymbol{\Theta}^{(t+1)} = \text{argmax}_{\boldsymbol{\Theta}}\, \mathcal{Q}_{\boldsymbol{z}}(\boldsymbol{\Theta}\vert\boldsymbol{\Theta}^{(t)})$,	$t = 1, 2, \ldots$

Sometimes it is not possible to analytically maximize the $\mathcal{Q}$-function as required by the M-step. However, if we can find a parameter estimate that *increases* the $\mathcal{Q}$-function at each iteration instead of maximizing it, the convergence property of the algorithm is still valid. In this case the EM algorithm is called a *Generalized* EM or GEM algorithm.

The standard (G)EM algorithm is a so-called *batch* or *offline* training algorithm, where the parameter update is done after collecting the statistics for all training sequences. That is, for a set of K training sequences the auxiliary function is defined over the entire training set by adding sums over k in algorithm 2.4, and the M-step gives a parameter estimate based on the entire training set. *Incremental* or *online* variants of the (G)EM algorithm, where the E- and M-steps are carried out for each training sequence, have recently been proposed in *e.g.*, [YHS98]. However, these online approaches have considerably larger memory requirements than the conventional EM algorithm.

2.5.2 Baum-Welch Reestimation

For an HMM the hidden variable is the state sequence $\boldsymbol{\pi}$. Indeed, if $\boldsymbol{\pi}$ is given, then maximizing (2.21) reduces to $2N$ static learning problems; we can estimate the transition and match probabilities independently for each state. If we denote the current model by $\boldsymbol{\Theta}^{(t)}$, then the auxiliary $\mathcal{Q}$-function can be expressed as the expectation of the complete data log likelihood $P(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\Theta})$ over the path distribution given by the current model[5] $\boldsymbol{\Theta}^{(t)}$,

[5] The Baum-Welch algorithm was actually proposed several years before the formal description of the EM algorithm. In the derivation by Baum and colleagues a slightly different auxiliary function was used,

$$\tilde{\mathcal{Q}}(\boldsymbol{\Theta}\vert\boldsymbol{\Theta}^{(t)}) = \mathcal{Q}(\boldsymbol{\Theta}\vert\boldsymbol{\Theta}^{(t)})/P(\boldsymbol{x}; \boldsymbol{\Theta}^{(t)}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}, \boldsymbol{x}; \boldsymbol{\Theta}^{(t)}) \log P(\boldsymbol{\pi}, \boldsymbol{x}; \boldsymbol{\Theta}).$$

This function essentially measures the difference between the distributions represented by $\boldsymbol{\Theta}$ and $\boldsymbol{\Theta}^{(t)}$, and since $P(\boldsymbol{x}; \boldsymbol{\Theta}^{(t)})$ is independent of $\boldsymbol{\Theta}$ maximization of $\mathcal{Q}$ and $\tilde{\mathcal{Q}}$ w.r.t. $\boldsymbol{\Theta}$ yields identical results.

$$\begin{aligned} \mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= E_{\boldsymbol{\pi}}[\log P(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta})|\boldsymbol{x},\boldsymbol{\Theta}^{(t)}] \\ &= \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\Theta}^{(t)}) \log P(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}). \end{aligned} \tag{2.33}$$

From (2.9) we find for the complete data likelihood,

$$P(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}) = \prod_l \theta_{\pi_{l-1}\pi_l}\phi_{\pi_l}(x_l) = \prod_{lij} \theta_{ij}^{\delta_{\pi_{l-1},i}\delta_{\pi_l,j}} \prod_{li} \phi_i(x_l)^{\delta_{\pi_l,i}}, \tag{2.34}$$

where $\delta_{\pi_l,i}$ is the Kronecker delta function which is one if and only if $\pi_l = i$ and otherwise zero. Taking the logarithm yields the complete data log likelihood,

$$\log P(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}) = \sum_{lij} \delta_{\pi_{l-1},i}\delta_{\pi_l,j} \log\theta_{ij} + \sum_{li} \delta_{\pi_l,i} \log\phi_i(x_l) \tag{2.35}$$

and substituting (2.35) into (2.33) gives,

$$\begin{aligned} \mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= \sum_{li} n_i^{(t)}(l) \log\phi_i(x_l) + \sum_{lij} n_{ij}^{(t)}(l) \log\theta_{ij} \\ &= \mathcal{Q}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\theta}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}), \end{aligned} \tag{2.36}$$

where we have defined,

$$\begin{aligned} n_{ij}(l) &= E_{\boldsymbol{\pi}}[\delta_{\pi_{l-1},i}\delta_{\pi_l,j}|\boldsymbol{x},\boldsymbol{\Theta}] \\ &= \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\Theta})\delta_{\pi_{l-1},i}\delta_{\pi_l,j} \\ &= P(\pi_{l-1}=i,\pi_l=j|\boldsymbol{x};\boldsymbol{\Theta}) \end{aligned} \tag{2.37}$$

and

$$n_i(l) = E_{\boldsymbol{\pi}}[\delta_{\pi_l,i}|\boldsymbol{x},\boldsymbol{\Theta}] = \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\Theta})\delta_{\pi_l,i} = P(\pi_l = i|\boldsymbol{x};\boldsymbol{\Theta}) \tag{2.38}$$

$n_{ij}(l)$ is the *expected* number of times the transition from state i to state j is used at time l and $n_i(l)$ is the expected number of times state i is visited at time l.

As shown in (2.36) the auxiliary function is composed of two separate terms; one corresponding to the transition probabilities, $\mathcal{Q}^{\theta}_{\boldsymbol{\pi}} = \sum_{lij} n_{ij}^{(t)}(l)\log\theta_{ij}$, and one corresponding to the match probabilities, $\mathcal{Q}^{\phi}_{\boldsymbol{\pi}} = \sum_{li} n_i^{(t)}(l)\log\phi_i(x_l)$. Thus, the M-step can be carried out separately for transition and match probabilities. The probabilistic constraints are easily incorporated by viewing the maximization step as a constrained optimization problem. For the match probabilities we define,

$$\mathcal{J}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) = \mathcal{Q}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \sum_i \lambda_i \left(1 - \sum_{a\in\mathcal{A}} \phi_i(a)\right), \tag{2.39}$$

where the λ_i's are Lagrange multipliers. The derivative of $\mathcal{J}^{\phi}_{\boldsymbol{\pi}}$ w.r.t. the probability of matching symbol a in state i reads,

$$\frac{\partial \mathcal{J}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)})}{\partial \phi_i(a)} = \frac{1}{\phi_i(a)} \sum_l n_i^{(t)}(l) \frac{\partial \phi_i(x_l)}{\partial \phi_i(a)} - \lambda_i = 0. \tag{2.40}$$

Observing that $\frac{\partial \phi_i(x_l)}{\partial \phi_i(a)} = \delta_{x_l,a}$ we find,

$$\frac{\partial \mathcal{J}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)})}{\partial \phi_i(a)} = \frac{1}{\phi_i(a)} \sum_l n_i^{(t)}(l) \delta_{x_l,a} - \lambda_i. \tag{2.41}$$

Setting the derivative to zero and solving for $\phi_i(a)$ gives,

$$\phi_i^{(t+1)}(a) = \frac{\sum_l n_i^{(t)}(l)\delta_{x_l,a}}{\lambda_i}. \tag{2.42}$$

The constraint $\sum_a \phi_i(a) = 1$ yields $\lambda_i = \sum_{la} n_i^{(t)}(l)\delta_{x_l,a}$ and therefore the reestimation formulas for the match probabilities are,

$$\phi_i^{(t+1)}(a) = \frac{\sum_l n_i^{(t)}(l)\delta_{x_l,a}}{\sum_{la'} n_i^{(t)}(l)\delta_{x_l,a'}} = \frac{\bar{n}_i^{(t)}(a)}{\sum_{a'} \bar{n}_i^{(t)}(a')}, \tag{2.43}$$

where $\bar{n}_i(a) = \sum_l n_i(l)\delta_{x_l,a}$ is defined as the expected number of times symbol a is matched in state i for observation sequence $\boldsymbol{x}$. The derivation for the transition probabilities is of exactly the same form and leads to the reestimation formulas

$$\theta_{ij}^{(t+1)} = \frac{\sum_l n_{ij}^{(t)}(l)}{\sum_{lj'} n_{ij'}^{(t)}(l)} = \frac{\bar{n}_{ij}^{(t)}}{\sum_{j'} \bar{n}_{ij'}^{(t)}}, \tag{2.44}$$

where $\bar{n}_{ij} = \sum_l n_{ij}(l)$ is the expected number of times a transition from state i to j is used for observation sequence $\boldsymbol{x}$. For a set of K training sequences the update formulas remain unchanged but the expected $\bar{n}$-counts must now be accumulated over the K training sequences.

The expected counts in the Baum-Welch reestimation equations above can be computed efficiently using the *forward-backward* algorithm. In addition to the recursion for the forward variable $\alpha_j(l)$ given in algorithm 2.1 a similar backward recursion is introduced for the backward variable $\beta_i(l)$. Let $\beta_i(l) = P(\boldsymbol{x}_{l+1}^L|\pi_l = i; \boldsymbol{\Theta})$, *i.e.*, the probability of matching the rest of the sequence $\boldsymbol{x}_{l+1}^L = \boldsymbol{x}_{l+1}, \ldots, \boldsymbol{x}_L$ given that we are in state i at time l. The backward recursion is easily derived and the procedure is shown in algorithm 2.5. Observe that the likelihood can be calculated as,

$$P(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{i=1}^{N} \alpha_i(l)\beta_i(l) \tag{2.45}$$

for any $l \in [1;L]$.

Now, by consulting figure 2.5 and exploiting the standard HMM assumptions, we see that the expected number of times, $n_{ij}(l)$, a transition from state i to state j is used at time l is given by,

$$n_{ij}(l) = \frac{P(\pi_{l-1} = i, \pi_l = j, \boldsymbol{x}; \boldsymbol{\Theta})}{P(\boldsymbol{x};\boldsymbol{\Theta})} = \frac{\alpha_i(l-1)\theta_{ij}\phi_j(x_l)\beta_j(l)}{\sum_{i'} \alpha_{i'}(l)\beta_{i'}(l)}. \tag{2.46}$$

Algorithm 2.5 Backward algorithm $[P(\boldsymbol{x}; \boldsymbol{\Theta})]$

Definition:	$\beta_i(l) = P(\boldsymbol{x}_{l+1}^L \mid \pi_l = i; \boldsymbol{\Theta})$	
Initialization:	$\beta_i(L) = \theta_{iN+1},$	$1 \leq i \leq N$
Recursion:	$\beta_i(l) = \sum_j \beta_j(l+1)\theta_{ij}\phi_j(x_{l+1}),$	$1 \leq i \leq N, 1 \leq l < L$
Termination:	$P(\boldsymbol{x}; \boldsymbol{\Theta}) = \sum_j \beta_j(1)\theta_{0j}\phi_j(x_1)$	

Similarly, the expected count $n_i(l)$ is obtained by summing (2.46) over j and using the backward recursion,

$$n_i(l) = \frac{P(\pi_l = i, \boldsymbol{x}; \boldsymbol{\Theta})}{P(\boldsymbol{x}; \boldsymbol{\Theta})} = \frac{\alpha_i(l)\beta_i(l)}{\sum_{i'} \alpha_{i'}(l)\beta_{i'}(l)}. \tag{2.47}$$

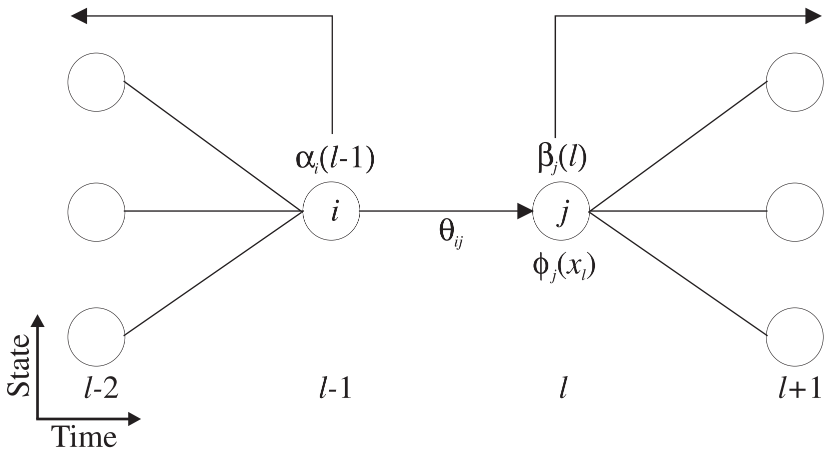

Figure 2.5: Calculation of expected counts.

2.5.3 Viterbi Reestimation

When calculating the counts by the forward-backward algorithm, all possible paths through the model are taken into account. Another approach is to use the Viterbi algorithm which only considers the optimal path $\hat{\boldsymbol{\pi}}$ through the model. In this case the parameters are selected according to,

$$\hat{\boldsymbol{\Theta}}_{ML} = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}[\max_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\Theta})]. \tag{2.48}$$

The counts can now be computed during Viterbi backtracking and they effectively become delta functions,

$$n_{ij}(l) = \delta_{\hat{\pi}_{l-1},i}\delta_{\hat{\pi}_l,j} \tag{2.49}$$
$$n_i(l) = \delta_{\hat{\pi}_l,i}. \tag{2.50}$$

Thus, $n_{ij}(l) = 1.0$ only if the transition probability θ_{ij} is used at time l in the optimal path and similarly $n_i(l) = 1.0$ only if state i is visited at time l. The reestimation equations remain the same as (2.43)-(2.44) and the method is often called *Segmental k-means* reestimation [RWJ86] or simply Viterbi reestimation.

Compared to the forward-backward approach, Viterbi reestimation is computationally cheaper as the backward pass is not needed. Furthermore, if the number of training examples is very large compared to the number of parameters it has been theoretically argued [ME91] that Viterbi reestimation will give parameter estimates similar to the Baum-Welch estimates. In most applications, however, the training data is limited and the estimates will differ for the two approaches. The forward-backward algorithm calculates the counts as a weighted average over all paths in the model, thereby allowing the HMM to "share" modeling resources between several states to handle difficult segments in the training data. On the other hand, Viterbi reestimation forces the HMM to treat each state independently. Based on these observations one may argue that Viterbi decoding is more suitable for Viterbi trained models whereas "all path" forward decoding should be used for forward-backward reestimated models.

2.6 Variations

Through time a large number of variations have been proposed to the basic HMM architecture described above. In this section we discuss some of the variations relevant to this work.

2.6.1 Match Distributions

Standard HMMs typically make use of two different kinds of match distributions, namely discrete and continuous, but parameter sharing between states can lead to mixtures of the two, known as semi-continuous distributions. Figure 2.6 illustrates the three types of distributions which are described below.

2.6.1.1 Discrete Match Distribution

For discrete observations the probability of matching a symbol a from a finite alphabet $\mathcal{A}$ in state i at any time is denoted $\phi_i(a)$. HMMs employing discrete match distributions are commonly termed discrete HMMs. Discrete match probabilities are computationally very cheap as they can be implemented by look-up tables.

The discrete HMM seems to be the right choice for applications like biological sequence modeling where the observations are inherently symbols from a finite alphabet [DEKM98]. Most early speech recognition systems were also based on discrete HMMs [Jel76, Rab89, Lee89]. Such an approach, however, requires that the speech feature vectors are first passed though a *vector quantizer* which assigns a symbol from an alphabet $\mathcal{A}$ to each of the feature vectors. The symbols represent prototype feature vectors from a *codebook* obtained by *e.g.*, the k-means or Linde-Buzo-Gray (LBG) algorithm (see *e.g.*, [DPH93]) and the assignment is typically based on the Euclidean or Itakura [Ita75] distance between the prototype and observed feature vector. An important quality of the discrete distribution is that it does not assume any specific functional form of the match distribution. However, the quality of the quantizer in terms of the distortion introduced due to finite size codebooks can have a crucial impact on the performance of the recognizer [Rab89].

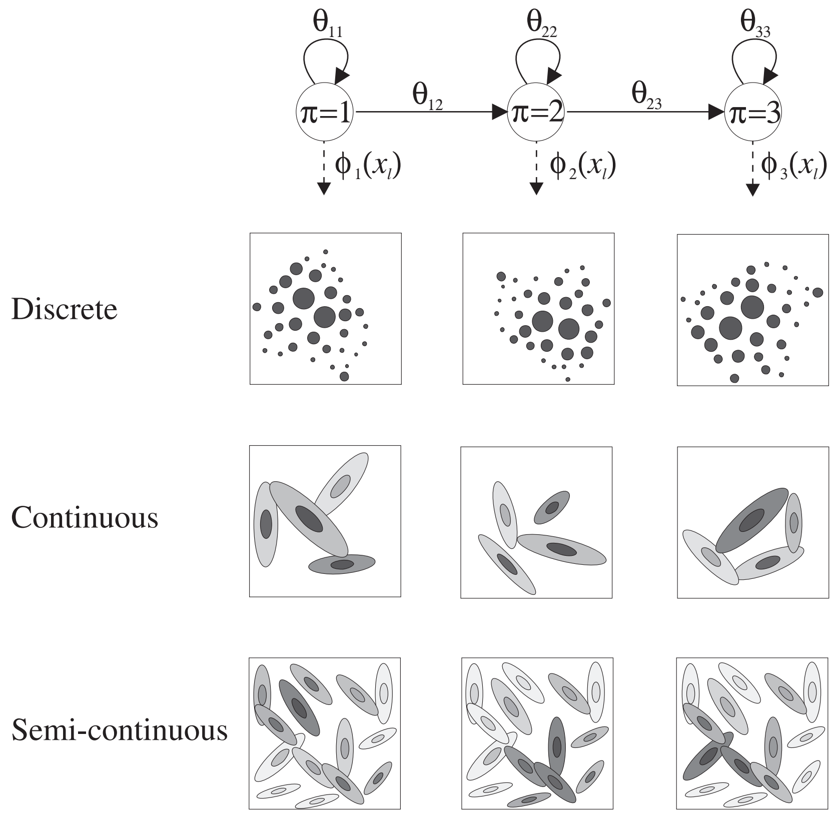

Figure 2.6: Illustration of match distributions in HMMs.

2.6.1.2 Continuous Match Distribution

The continuous HMM was introduced in the mid-1980s [LRS83] in order to account for the fact that the observations $\boldsymbol{x}_l \in \mathcal{R}^r$ are inherently continuous valued vectors in many applications. In the continuous HMM the match distribution $\phi_i(\boldsymbol{x}_l; \boldsymbol{w}^i)$ in state i is a parametric continuous density function. The most commonly used density is the Gaussian density function,

$$\phi_i(\boldsymbol{x}_l; \boldsymbol{w}^i) = \mathcal{N}(\boldsymbol{x}_l; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{r/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}_l - \boldsymbol{\mu}_i)^{\mathrm{T}} \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{x}_l - \boldsymbol{\mu}_i)\right), \quad (2.51)$$

where $\boldsymbol{\Sigma}_i \in \mathcal{R}^r \times \mathcal{R}^r$ and $\boldsymbol{\mu}_i \in \mathcal{R}^r$ identify the covariance matrix and mean vector, respectively. The vector $\boldsymbol{w}^i$ denotes the collection of parameters specifying the density associated with state i, that is, $\boldsymbol{w}^i = \{\boldsymbol{\Sigma}_i, \boldsymbol{\mu}_i\}$. The unimodal single-Gaussian density is very simple and usually inadequate for describing the complex data distributions encountered in practical applications like speech recognition. Therefore, it is common to use a *mixture* of Gaussians instead,

$$\phi_i(\boldsymbol{x}_l; \boldsymbol{w}^i) = \sum_{k=1}^{K} c_{ik} \mathcal{N}(\boldsymbol{x}_l; \boldsymbol{\Sigma}_{ik}, \boldsymbol{\mu}_{ik}). \quad (2.52)$$

Here there are K components in the mixture and the mixture weights c_{ik} satisfy $\sum_k c_{ik} = 1$ and $c_{ik} \geq 0$ for all states i. For the k'th component in the mixture, $\boldsymbol{\Sigma}_{ik}$ and $\boldsymbol{\mu}_{ik}$ identify the covariance matrix and mean vector, respectively. The Baum-Welch algorithm also applies to mixtures of Gaussians and the reestimation equations for the mixture weights, mean vectors and covariance matrices are readily expressed in terms of the expected count $n_i(l)$ introduced above (see *e.g.*, [Rab89]). They can be derived easily by a simple application of the chain rule when calculating the derivative of the auxiliary $\mathcal{Q}$-function.

Given a sufficient number of components the Gaussian mixture can, in theory, approximate any smooth continuous function to arbitrary accuracy. However, the available amount of data in most applications puts an upper limit on the number of mixture components that can be estimated reliably. To ensure robust training it is furthermore quite common to assume that the elements of the feature vector $\boldsymbol{x}_l$ are uncorrelated by ignoring off-diagonal elements in the covariance matrix. With a typical feature vector dimension of $r \approx 20 - 50$ this can lead to very large savings in the number of parameters but at the cost of reduced modeling capabilities.

2.6.1.3 Semi-Continuous Match Distribution

To alleviate the parameter problem in continuous mixtures a combination of discrete and continuous densities have been proposed in [BN88, HJ89, Hua92]. Models employing such densities are usually called semi-continuous HMMs and the idea is to *share* a set of common basis functions, typically Gaussians. Let the shared basis functions be denoted $\mathcal{N}(\boldsymbol{x}_l; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k)$. Then the semi-continuous match distribution $\phi_i(\boldsymbol{x}_l; \boldsymbol{w}^i)$ in state i is defined by,

$$\phi_i(\boldsymbol{x}_l; \boldsymbol{w}^i) = \sum_{k=1}^{K} c_{ik} \mathcal{N}(\boldsymbol{x}_l; \boldsymbol{\Sigma}_k, \boldsymbol{\mu}_k), \tag{2.53}$$

where the mixture coefficients satisfy positivity and sum-to-one constraints. The semi-continuous HMM can be viewed as a special kind of discrete HMM where the mixture coefficients replace the discrete match probabilities and the vector quantizer is represented by the set of basis functions. Contrary to the standard discrete HMM, the "vector-quantizer" in a semi-continuous HMM is trained along with all the other parameters in the model. Thus, the semi-continuous HMM can be considered a special case of a discrete HMM with an *adaptive input transformation*. The semi-continuous HMM also has interesting relations to the so-called *radial basis function* neural network, see *e.g.*, [RMCF92].

Reestimation equations for the mixture weights, mean vectors and covariance matrices can be derived in a way analogous to the continuous mixture case, see *e.g.*, [BN88, HJ89, Hua92].

2.6.2 Parameter Tying

The semi-continuous HMM constitutes a special case of a technique known as *parameter tying* [You92a] where some parameters or parameter sets are shared between one or more states in the HMM. The technique is very similar to *weight sharing* in neural networks (see *e.g.*, [HKP91, Bis95]) and may yield better generalization performance when the training data is sparse. The Baum-Welch algorithm applies without changes if the expected counts for the tied parameters are added up.

Parameter tying necessarily implies that some of the modeling capabilities are sacrificed to improve generalization and it is therefore very important to select the proper parameters to share. This is often done using prior knowledge about the problem at hand. For example, in speech recognition it is very common to share the same match distribution between several states in a phoneme model for duration modeling as discussed below. Similarly, for context dependent phoneme models the "core" or *nucleus* section can be shared between all context-models for a particular phoneme [LH89]. Alternatively, one can use unsupervised clustering techniques for tying different parameter sets automatically during model estimation, see *e.g.*, [Lee89, Lee90, You92a, KOR94].

2.6.3 Duration Modeling

If the self-loop transition for a state has the value $\theta_{ii} = q$, then the probability of staying in this state for d time-steps (observations) is given by,

$$P(d) = q^{d-1}(1-q) \tag{2.54}$$

and the average duration is $\bar{d} = (1-q)^{-1}$. For many applications this is a very poor duration model. A Poisson distribution, for example, is much more appropriate for phonemes, see figure 2.7. There are several ways to improve duration modeling in the HMM framework. One approach is to use explicit parametric or non-parametric (histogram) duration distributions instead of the self-loop transitions as proposed in *e.g.*, [RM85, Lev86]. In these approaches the forward-backward algorithm is altered by introducing sums over the allowed duration intervals and the histogram probabilities or the parameters of the parametric duration distributions can be reestimated along with the other parameters. Although appealing, the use of explicit duration distributions has not been widely used because of the added computational complexity — for a maximum duration of D observations the forward algorithm for models with explicit duration distributions requires approximately $D^2/2$ times the computation of the standard forward algorithm. Another approach is to use so-called "hardwired" duration modeling where state duplication is used to ensure minimum and/or maximum durations. When duplicating states the match distributions are tied between the duplicated states. An example of a model with hardwired minimum duration is shown in figure 2.8(a). If the two transitions associated with each of the duplicated states are tied as in figure 2.8(a) the model will have a Pascal or pseudo-Poisson duration distribution,

$$P(d) = \begin{pmatrix} l-1 \\ n-1 \end{pmatrix} q^{l-n}(1-q)^n, \quad l \geq n \tag{2.55}$$

with average duration $\bar{d} = n/q$, see figure 2.7. If the value of q is fixed a priori before training the number of duplicated states needed to obtain an observed average duration of D is simply $n = qD$. Note that if the model in figure 2.8 is trained or decoded using all possible paths the pseudo-Poisson distribution will be valid. On the other hand, if training or decoding is based only on the optimal Viterbi path then the duration distribution effectively becomes exponential because all paths of length l through the model in this case have equal probability. For Viterbi training or decoding this implies that the minimum duration model in figure 2.8(a) can be replaced by the model in figure 2.8(b).

The hardwired duration modeling approach has only little computational overhead and has been observed to give excellent results in many speech recognition applications, see *e.g.*, [Rii98, Rob94, BM94, Rob92]. Therefore, it tends to be the most widely used approach.

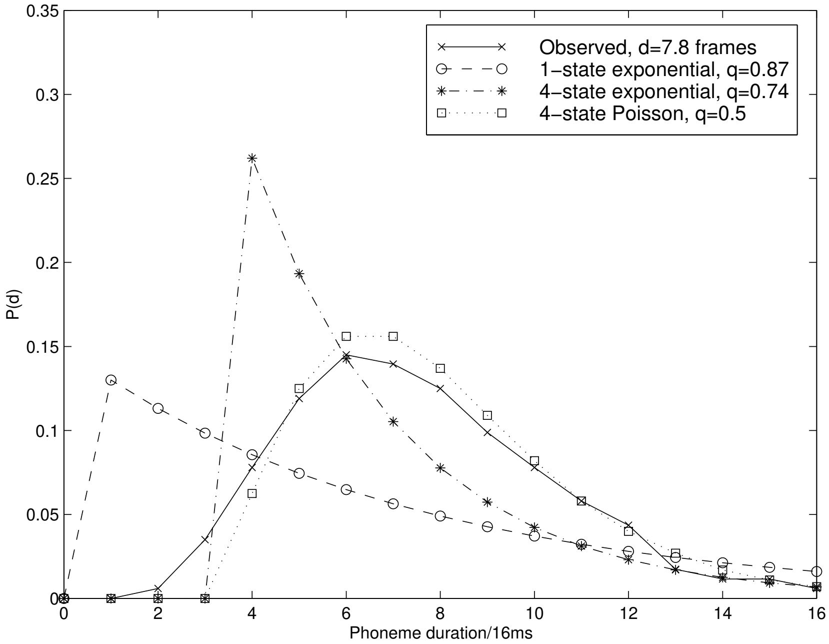

Figure 2.7: Example of duration distributions for the phoneme /eh/ in the TIMIT database. The plot shows the observed distribution and the distributions given by a one-state model and a four-state model. For the four-state model both the exponential and pseudo-Poisson distributions are shown. For all models, the self-loop probability q is selected to match the observed average duration of 124ms.

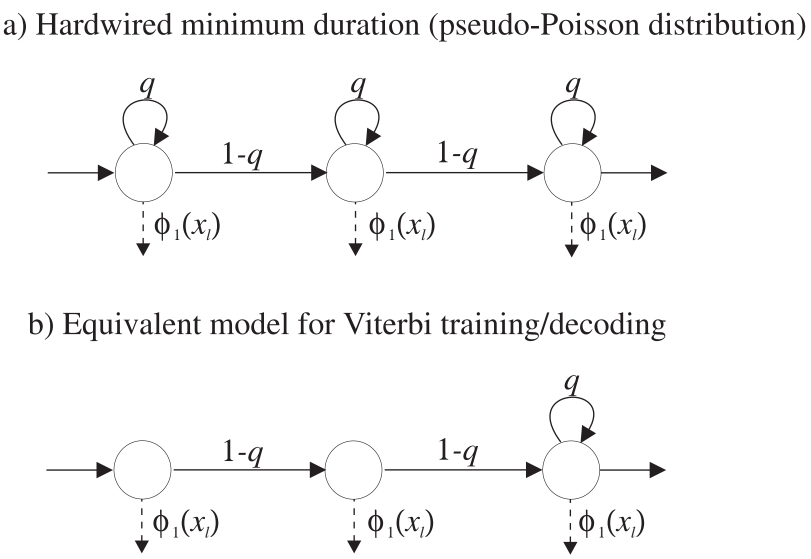

Figure 2.8: "Hardwired" duration modeling in HMMs by duplication states (all match and transition probabilities are tied between the duplicated states). (a) pseudo-Poisson minimum duration distribution, (b) equivalent model for Viterbi training or decoding.

2.7 HMMs for Speech Recognition

In the above discussion of the HMM framework it was assumed that a separate model could be built for each class to be recognized. In small vocabulary isolated word recognition it is indeed possible to construct an HMM for each word and to estimate its parameters using the training data available for that word. In principle large vocabulary isolated word recognition and continuous speech recognition can be done in exactly the same way by having models for each possible *hypothesis*, that is, each possible word or sentence, respectively. However, in practice there are a number of problems that have to be solved. These are defined below and discussed in the following sections.

Since training data (and the power of todays computers) is always limited in speech recognition it is difficult if not impossible to train a separate model for each word (or sentence) in a large vocabulary task. This problem is usually solved by splitting the recognition task into two separate parts as discussed in chapter 1: 1) Acoustic modeling and 2) Language modeling. The acoustic model is a collection of HMMs which translate the acoustic observations $\boldsymbol{x}$ into a sequence $\boldsymbol{y}$ of meaningful *symbolic message units* (SMUs) of a language. As discussed in chapter 1 the SMUs are typically words, syllables or phonemes. By concatenating SMUs, words or sentences can be built according to the constraints imposed by a language model.

Furthermore, the large number of possible hypotheses (sentences or words) can make an exhaustive search over all possible concatenations of SMUs during decoding impractical. To decode utterances in real time more efficient search strategies are needed.

2.7.1 Acoustic Modeling

Similar to the case of only one class for each sequence, the MAP decoder for finding the optimal sequence of SMU labels $\boldsymbol{y}$ corresponding to an utterance $\boldsymbol{x}$ is given by,

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} P(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\Theta})P(\boldsymbol{y};\boldsymbol{\Theta}), \tag{2.56}$$

where $\mathcal{Y}$ is the space of all possible SMU label sequences. In (2.56) $P(\boldsymbol{y};\boldsymbol{\Theta})$ is commonly denoted the language model and $P(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\Theta})$ the acoustic model. In many applications the language model parameters are assumed independent of the acoustic model parameters, that is, the language model is given by a separate model; $P(\boldsymbol{y};\boldsymbol{\Phi})$. In this way the modeling problem has been divided into two separate parts; acoustic modeling and language modeling.

The acoustic model described by $\boldsymbol{\Theta}$ consists of one sub-HMM for each SMU and it maps the input speech feature vectors into a sequence of SMUs. The choice of SMU is very important for the overall performance of the system. In Lee [Lee89] it was argued that good performance can be obtained provided that the SMUs satisfy two criteria:

Consistency. The SMUs must be acoustically consistent, that is, different acoustic realizations of the same SMU must be similar and significantly different from other SMUs. Consistency will improve the ability of the acoustic model to discriminate between classes and improve overall performance.

Trainability. The SMUs must be trainable from the available data, that is, there must be a sufficient number of examples in the training set for each SMU to guarantee robust estimates of the parameters. Trainability ensures that the acoustic model will generalize well to examples not found in the training set.

For small vocabulary isolated word recognition words are a natural choice for SMUs because they are the most natural unit of speech and because they are consistent and trainable. However, for large vocabularies it is not possible to train all word models due to limited training data and computer power. The trainability problem can be handled by using phoneme-like SMU models [LH89, Rob94, HRRC95]. In English there are about 40 distinct phonemes which constitute a set of building blocks for all words in the language. The small number of phonemes makes it possible to train phoneme-HMMs from relatively small training sets. Unfortunately, phonemes are abstract units of speech and it is very difficult to utter any phoneme in isolation. The actual realization of a particular phoneme, called a phone,[6] is highly influenced by coarticulation effects of the human speech production system, *i.e.*, the realization will depend on the phonetic context. The large variability in acoustic realizations can to some extent be handled using context dependent phone models *e.g.*, *biphones* or *triphones*. Biphones represent phonemes in the (right *or* left) context of another phoneme whereas triphones represent phonemes in the (left *and* right) context of two other phonemes, see [You96] for a discussion. Although this addresses the consistency problem, it re-introduces the trainability problem because for a set of 40 phonemes there are $40^3 = 64,000$ possible triphones (some of which do not actually occur in practice). In order to use biphone or triphone models it is therefore necessary to employ parameter tying and smoothing techniques as described in *e.g.*, [LH89, Lee89, Rab89, KOR94, YW94]. In the work described by this thesis only monophone models were used mainly for computational reasons.

For some definition of the SMUs let us now assume that the SMU y_l associated with each observation x_l is known. Then maximizing the likelihood of the acoustic model $P(\boldsymbol{x}|\boldsymbol{y};\boldsymbol{\Theta})$ is straightforward: The HMM modeling a particular SMU is just trained using those segments of the data which belong to that SMU class. Unfortunately, the exact time boundaries between SMUs relative to the acoustic signal are rarely known because they have to be manually assigned by human experts — a process which is both time consuming and expensive. At this point it is convenient to distinguish explicitly between two types of labeling common to HMM modeling:

Complete Labels. Each observation is associated with one label. In some applications such as the prediction of so-called secondary structures in proteins [RK96a] each observation is associated with a particular class label. Thus, the sequence of labels denoted $\boldsymbol{y} = y_1, \ldots, y_L$ is as long as the sequence of observations $\boldsymbol{x} = x_1, \ldots, x_L$, and typically the labels come in groups. In speech the complete labeling corresponds to knowing the SMU time-segmentation, that is, the exact time boundaries between SMUs relative to the acoustic signal.

Incomplete Labels. The whole sequence of observations is associated with a shorter sequence of labels $\boldsymbol{y} = y_1, \ldots, y_S$, $S < L$. The label of each individual observation is unknown—only the order of labels is available. In SMU based acoustic modeling the correct string of SMUs is usually known for the training utterances because

[6]Different phoneme sequences result in different words whereas the same word can be produced using many different phone sequences. Thus, phonemes are abstract sound classes and phones the particular realizations of these sounds. We will use the term phone and phoneme interchangeably throughout this thesis despite the slight difference in definition.

the spoken words are known but the time segmentation is unknown. Typically the sequence of observations is considerably longer than the label sequence. When $S = L$ the two types of labels are equivalent and $S = 1$ corresponds to only one label for each observation sequence.

Where needed we will distinguish between the two types of labeling by explicitly writing $\boldsymbol{y}_1^S$ for incomplete labeling and $\boldsymbol{y}_1^L$ for complete labeling.

If only the incomplete labeling is available the above approach for maximizing the likelihood cannot be used. There are two common solutions:

Viterbi resegmentation In this method the sequence of incomplete labels is iteratively changed into a sequence of complete labels by a so–called forced Viterbi alignment. The basic idea is to decode the current acoustic model using a Viterbi decoder where the sequence of SMU-HMMs is constrained to obey the incomplete label sequence associated with the training sequence. Since the Viterbi decoder gives a state segmentation this alignment can be translated into a sequence of complete labels. Using this complete labeling the models are trained independently as above (using either forward-backward or Viterbi training) until convergence and a new alignment can be generated based on these models.[7] The forced Viterbi resegmentation approach is widely used for HMM/NN hybrids as discussed later in chapter 6.

Embedded reestimation This method is sometimes more reasonable than Viterbi resegmentation since enforcing an explicit time segmentation can be inappropriate.[8] In the framework of maximum likelihood estimation the task is solved by concatenating SMU-HMMs according to the observed incomplete SMU labeling $\boldsymbol{y}_1^S$ [LH89, Lee89], see figure 2.9. Multiple pronunciations of a word can be included as parallel branches in the sequence of SMU-HMMs and the expected counts are calculated using the standard forward-backward algorithm on the concatenated HMM $\boldsymbol{\Theta}_{y_1^S}$. This is repeated for all training utterances and the accumulated counts are used for reestimating the parameters. Note that this approach relies heavily on good initial SMU-HMMs in order to attract the right portions of the training data during estimation. We will elaborate on this in chapter 5.

2.7.2 Language Modeling

The language model defines a set of rules describing how utterances are formed from sequences of SMUs. The language model imposes large constraints on the sequence of SMUs and a well designed language model aids considerably in guiding the search during decoding and consequently in reducing the decoding time. The degree of constraint that a language model imposes is usually measured in terms of the *perplexity* [DPH93] which is roughly the average branching factor for an SMU, *i.e.*, the average number of SMUs that can follow any other SMU. In a typical continuous speech recognition application, the language model contains a pronunciation lexicon defining phonetic transcriptions of words and a syntax defining how sentences are formed from words. If some of the words

[7]Typically, the initial complete labeling is a linear expansion of the incomplete labeling, possibly constrained by average durations of the SMUs.

[8]Explicit time boundaries between *e.g.*, phonemes do not exist because of coarticulation — the human articulatory system cannot change instantaneously.

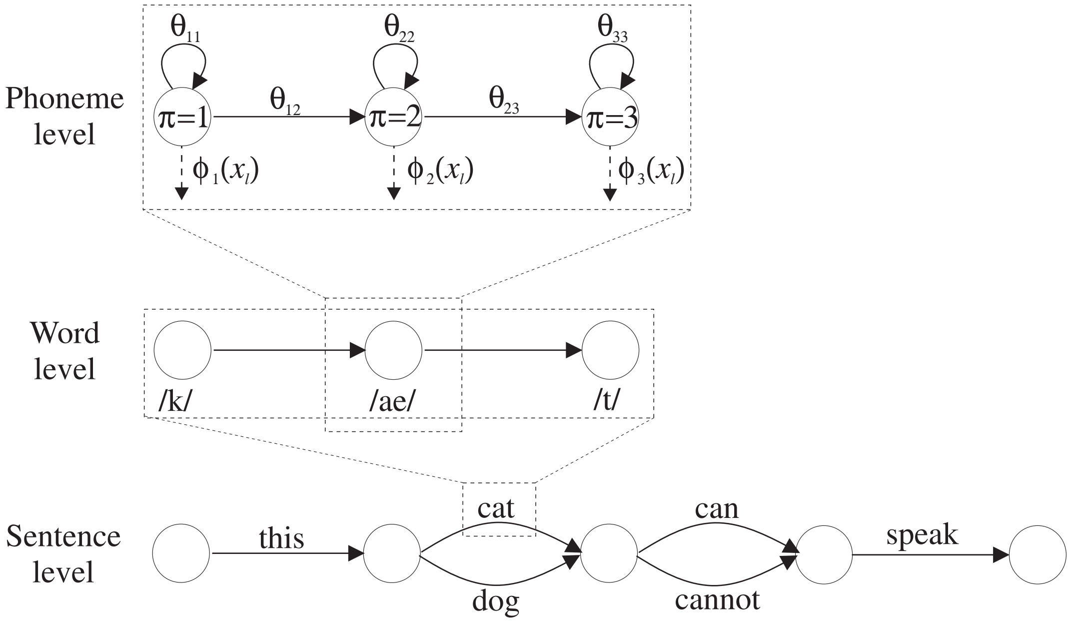

Figure 2.9: Illustration of concatenated phoneme-HMMs for the word "cat" with phonetic transcription /k/ /ae/ /t/ (TIMIT phoneme labels, see appendix A).

have multiple pronunciations the lexicon will have one entry for each pronunciation. The rules in the grammar are usually described by *N-gram* stochastic grammars giving the probability that a word (or sub-word) follows a particular sequence of $N-1$ words (or sub-words). Thus, an N-gram effectively approximates the probability of a word (or sub-word) sequence $\boldsymbol{y}$ as,

$$P(\boldsymbol{y}) \approx \prod_{s=1}^{S} P(y_s | \boldsymbol{y}_{s-N+1}^{s-1}). \tag{2.57}$$

For word sequences the N-gram probabilities can be estimated reliably from large text-only corpera provided N is not larger than three or four. The degenerate case of $N=1$ defines a uni-gram grammar where the a priori probability of the SMU sequence $\boldsymbol{y}$ is given by the product of a priori probabilities for each of the SMU classes in the sequence. Similarly, $N=0$ defines a zero- or null-gram where all label sequences are equiprobable. Note that the N-grams actually describe an $(N-1)$'th order Markov chain and for *regular* or *bigram* grammars ($N=2$), the language model can be included in the acoustic HMM during decoding as transition probabilities between SMUs.[9] Such a combined model is called a *composite* HMM, see figure 2.9. In chapter 4 we will describe a slight generalization of composite HMMs called *Class HMMs* (CHMMs), in which the acoustic and (bigram) language model parameters are jointly estimated from the same data.

[9]Because a first order HMM can emulate an HMM of any order by increasing the number of states, it is actually also possible to use the idea of composite HMMs for N-grams with $N>2$. For $N>2$ the resulting composite HMM will, however, be far too complex to handle in most speech recognition applications.

2.7.3 Viterbi Decoding

For composite HMMs Viterbi decoding is practically identical to the case of small vocabulary isolated word recognition discussed in section 2.4. The standard Viterbi decoder is simply applied to the composite model, and the optimal state path derived from the backtracking pass can be converted into a sentence hypothesis $\hat{\boldsymbol{y}} = \boldsymbol{y}(\hat{\boldsymbol{\pi}})$, since each state in the composite HMM is uniquely associated with a particular SMU. This also applies to the maximum posterior state path defined by (2.19) but the sequence of SMUs obtained in this way need not correspond to a valid path through the model. For large vocabularies and regular grammars the idea of composite HMMs can still be applied but it quickly becomes impractical to actually construct the composite HMM. Instead, one can implement the Viterbi decoder using the concept of *token passing* [YRT89, You96] where the recognition network can be built on-the-fly using the constraints imposed by the language model. The token passing approach also allows for *long-span* language models like trigrams. In the token passing algorithm the partial hypothesis at time l in state i along with its associated score is a token which is copied and passed on to all possible successor states j at time $l+1$. When passing the token on to the next state inside an SMU model the score is scaled by the transition probability θ_{ij}. If the token is passed on to another SMU model the score is scaled by the language model probability, and the partial hypothesis is extended with the new SMU. The best incoming token is selected by a maximization operation and before passing it on again it is scaled by the match probability $\phi_j(x_{l+1})$. Since the partial hypothesis at time l is stored in the token, the use of long-span language model probabilities and on-the-fly extension of the recognition lattice is possible. At the end of decoding the highest scoring hypothesis will be the token stored in the end state $N+1$. Thus, no backtracking is needed. The token passing algorithm is illustrated in figure 2.10.

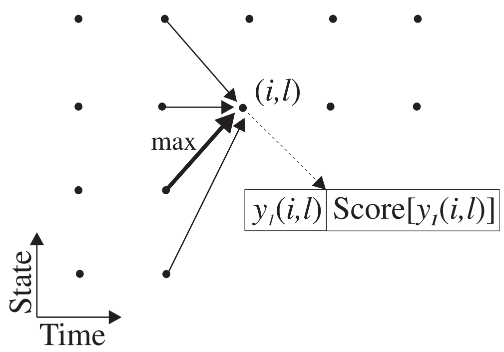

Figure 2.10: Viterbi decoding by token passing. $\boldsymbol{y}_1(i,l)$ is the partial hypothesis in the token associated with state i at time l. The score of this hypothesis is given by $Score[\boldsymbol{y}_1(i,l)] = \alpha_i^*(l)$.

To reduce the complexity of the Viterbi decoder it is very common to apply the technique of *beam search* to prune the search lattice. Only partial paths that have a log probability within a *beam width* threshold below the overall highest scoring partial path at time l is allowed to survive during the search of the lattice. More efficient pruning strategies exist but common to them all is the risk of pruning the optimal path.

2.7.4 Forward Decoding

Although the Viterbi decoder has been the main choice in speech recognition for many years, it is generally acknowledged that decoders which consider all possible paths through the model instead of just the optimal path, can yield considerably better results for large vocabulary and continuous speech recognition [SA91, RK96b, Joh96, Joh94]. Unfortunately, it is not straightforward to apply the standard forward decoder discussed in section 2.4.1 to continuous speech recognition. For example, applying the forward algorithm to a composite HMM will only give the probability of the observation sequence $P(\boldsymbol{x};\boldsymbol{\Theta})$ and not joint the probability of observations and labels $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ as needed by the MAP decoder.

A very simple "observation-local" decoder for composite HMMs which considers all paths, can be obtained by summing the state a posteriori probabilities for all states that belong to the same SMU submodel. Assume that the set of states in the sub-HMM for SMU $c \in \mathcal{C}$ carries the label c. Then the probability of label $y_l = c$ at time l can be calculated as,

$$P(y_l = c|\boldsymbol{x};\boldsymbol{\Theta}) = \sum_i P(\pi_l = i|\boldsymbol{x};\boldsymbol{\Theta})\delta_{c^i,c} \tag{2.58}$$

where c^i is the label associated with state i. The state posterior probabilities are calculated by running the standard forward-backward algorithm on the composite HMM. Note that the label distribution given by (2.58) can be interpreted as a *mixture of experts* distribution [JJNH91, JJ94b] where the mixture coefficients are the state a posteriori probabilities. From (2.58) it is now a simple matter to select the recognized label at time l,

$$\hat{y}_l = \operatorname*{argmax}_{c\in\mathcal{C}} P(y_l = c|\boldsymbol{x};\boldsymbol{\Theta}). \tag{2.59}$$

We will call this decoder the *forward-backward* decoder. If the aim of decoding is an incomplete label sequence, this can be obtained by folding identical successive labels into one label. Naturally, the forward-backward decoder does not ensure that the recognized labeling corresponds to a valid path, but it has been observed to yield considerably better performance than the Viterbi decoder for some speech recognition tasks, see chapter 5.

A different decoder known as the $N-best$ decoder [SC90] can find the hypothesis that maximizes the joint probability $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ without the need for constructing the composite HMM. The basic idea in N-best decoding is to implement the forward algorithm using the token passing paradigm where several hypotheses are allowed to survive in each state, see figure 2.11. The implementation is very similar to the one for the Viterbi decoder except that the maximization operation is replaced by an operation, where all identical partial incoming hypotheses in a state are merged and their associated costs added. The highest scoring hypothesis in the token associated with the exit state is then the one maximizing the joint probability of labels and observations.

Since new hypotheses are generated each time a new SMU is entered, the token associated with each state in the search lattice grows very quickly. In a practical implementation it is therefore necessary to sacrifice optimality by pruning hypotheses with low scores. In [SC90] two such pruning schemes were proposed; a local and a global thresholding. The local pruning works by removing all partial hypotheses in a token that have a log probability score lower than the highest scoring one minus a specified threshold. This is similar to the beam search idea and the result is a token containing only the highest

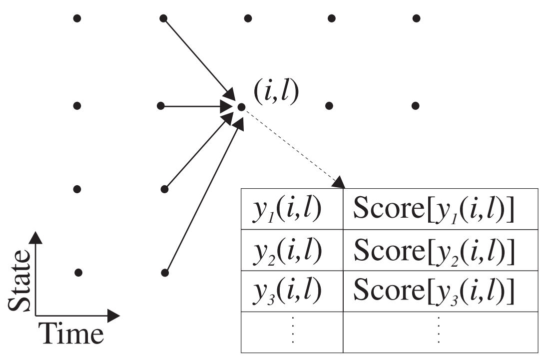

Figure 2.11: N-best decoding by token passing. $\boldsymbol{y}_p(i,l)$ is the p'th partial hypothesis in the token associated with state i at time l.

scoring hypotheses. The global pruning scheme further reduces the complexity by defining a maximum of N hypotheses allowed to survive in each state. These pruning schemes make the decoding complexity linear in the sequence length L and at the end of decoding the end state holds a list of the N best (highest scoring) hypotheses. Similar to the pruning mechanisms in the Viterbi decoder there is a risk of pruning the optimal hypothesis, but as discussed in [SC90] the risk is much smaller than for the Viterbi decoder if N is sufficiently large.

2.7.5 Asynchronous Search

Both the Viterbi and the N-best decoders are *time-synchronous* search techniques in the sense that all partial hypotheses at time l are available before processing at time $l+1$ starts. In the so-called *time-asynchronous* decoders the computational complexity of the search is reduced by pursuing the most probable partial hypothesis first. This best-first search is contrary to the exhaustive search in the Viterbi and N-best decoders. A widely used asynchronous decoder is the so-called A* or *stack decoder* used by Bahl, Jelinek and Mercer for the DRAGON speech recognizer [BJM83], and more recently in the HMM/NN hybrid ABBOT system [RH95, HRRC95].[10] The key to stack decoding is the score $f(\boldsymbol{y}_1^s;l)$ for a partial hypothesis $\boldsymbol{y}_1^s$ at time l defined by

$$f(\boldsymbol{y}_1^s;l) = a(\boldsymbol{y}_1^s;l) + b(\boldsymbol{y}_1^s;l). \tag{2.60}$$

$a(\boldsymbol{y}_1^s;l)$ is the log probability of the partial hypothesis $\boldsymbol{y}_1^s$ at time l obtained from the acoustic and language model, and $b(\boldsymbol{y}_1^s;l)$ is an *estimate* of the best possible score (minimum cost) in extending $\boldsymbol{y}_1^s$ to a valid complete hypothesis $\boldsymbol{y}_1^S$. It may be shown [BJM83] that as long as $b(\boldsymbol{y}_1^s;l)$ is an upper bound on the actual log probability of extending the partial hypothesis to a valid complete hypothesis then the search will be *admissible*, *i.e.*, it will not prune the optimal complete hypothesis.

In the stack decoder active hypotheses are placed in an ordered stack in descending order of scores. The top scoring hypothesis is then popped off the stack and extended by one word and the new cost is computed according to (2.60). Then the extended hypothesis is pushed back onto the stack and the stack is sorted according to the scores of the partial hypotheses. This procedure is continued until the top hypothesis is a complete hypothesis which is then the optimal hypothesis because extending any partial hypothesis below it

[10] A demo version of ABBOT is available at http://svr-www.eng.cam.ac.uk.

in the stack cannot yield a higher score. The stack decoder can use long-span language models and since the Viterbi criterion is not embedded in the search the decoder can consider all possible paths.

2.8 Summary

The standard HMM framework is based on a number of assumptions which limit their modeling capabilities:

First order Markov assumption. This assumption implies that time dependencies between successive observations are first-order linear correlations. In speech this is obviously a poor assumption since coarticulation effects can range much longer than just one frame.

State conditional observation independence. Implies that there are no correlations between adjacent observations other than the first-order time dependencies. Since the human speech production system cannot change instantaneously this severely hampers the ability to model coarticulation. To benefit from contextual information it is common in speech applications to augment the feature vector with its corresponding *delta* features [LH89] computed by linear regression from neighboring frames. A further improvement is to use multiple *streams* [LH89, NCM94] where features and delta features are modeled independently in each state by separate match distributions. The match probability used in the forward-backward calculations is then simply the product of the probability from each of the streams.

Match density models. The discrete match probability distribution suffers from quantization errors whereas the continuous Gaussian mixture suffers from model mismatch, because usually only small mixtures can be estimated reliably.

Discrimination. Maximum likelihood estimation is an unsupervised training algorithm in that it only uses the data belonging to a certain class (positive examples) to train the model representing this class. As discussed in the next chapter this is optimal for classification provided that the HMM can represent the true likelihood, but due to the assumptions discussed above this will never be the case for speech applications.

The remainder of this thesis will describe how all of these assumptions except the first order assumption can be mitigated. Firstly, the discrimination problem can be solved by using discriminative training criteria. This is discussed in the following three chapters describing different discriminative training algorithms and a particular HMM, called a Class HMM, designed for classification. Secondly, the observation context independence and the problems with model mismatch due to poor match distributions can be handled efficiently by using neural networks to estimate probabilities in the HMM framework. This is the topic of chapters 6-9.

Chapter 3

Discriminative Training

The ultimate goal of HMM estimation is to obtain a model that results in the lowest possible error rate for the decoder used. In the previous chapter a particular approach called Maximum Likelihood (ML) estimation was discussed. The benefits of ML estimation are many, but the most important is probably the guaranteed convergence and computational efficiency of the associated Baum-Welch algorithm. A serious problem of ML estimation, however, is that it is not directly aimed at minimizing the error rate of the HMM.

In this chapter we start by discussing the problems encountered in ML estimation and then turn our attention towards so-called *discriminative training* techniques. The aim of discriminative training is to "push" the HMM toward modeling the *decision boundaries* between classes rather than the often much more complex within-class distributions. If the model approaches the optimal decision boundaries during training the error rate will approach that of the optimal Bayes classifier. Intuitively this sounds right, but unfortunately it is not possible to prove that discriminative training will always be at least as good as likelihood-based approaches in terms of minimizing the error rate. Nevertheless, a vast body of experimental work supports this hypothesis for a variety of modeling frameworks and real-world applications.

The most commonly used discriminative training algorithms can roughly be divided into 1) those that directly minimize a smooth representation of the error rate and 2) those that indirectly minimize the error rate by training the HMM to model the class posterior distribution. Typical indirect methods include *Conditional Maximum Likelihood* (CML) estimation, which has *Maximum Mutual Information* (MMI) estimation as a special case, and the *Mean Squared Error* criterion. *Minimum Classification Error* and *corrective training* constitutes two examples of direct methods.

One of the main disadvantages of most discriminative methods is that computationally they are more expensive than ML training. In addition, efficient algorithms like EM generally do not apply and optimization must be based on more general iterative techniques like gradient descent. This is probably the main reason why these methods have not gained much attention until recently when powerful computers have made them feasible. Although discriminative training is costly, it should be noted that the complexity of decoding is the same for both ML and discriminatively trained models.

3.1 Maximum Likelihood

ML training of acoustic models can be interpreted as estimating the model parameters so as to minimize the *Kullback-Leibler distance* or *discrimination information* [ER90] between the true data distribution $P = P(\boldsymbol{x}|\boldsymbol{y})$ and the likelihood given by the model $P_\Theta = P(\boldsymbol{x}; \boldsymbol{\Theta}_y)$. This distance can be expressed as[1]

$$D(P||P_\Theta) = \sum_{\boldsymbol{y}\in\mathcal{Y}} \sum_{\boldsymbol{x}\in\mathcal{X}} P(\boldsymbol{x}|\boldsymbol{y}) \log \frac{P(\boldsymbol{x}|\boldsymbol{y})}{P(\boldsymbol{x}; \boldsymbol{\Theta}_y)} \tag{3.1}$$

and is always larger than or equal to zero. $D(P||P_\Theta) = 0$ means that the two distributions are identical, whereas non-overlapping distributions will have infinite discrimination information. Thus, minimizing $D(P||P_\Theta)$ will give a set of parameters that ensures the best possible match between the true distribution and the model distribution. The discrimination information as given by (3.1) is mainly of theoretical interest since the true distribution is unknown. Replacing the true distribution by the training set empirical distribution gives [ER90]

$$D(P||P_\Theta) = -\frac{1}{K} \sum_{k=1}^{K} \log P(\boldsymbol{x}(k); \boldsymbol{\Theta}_{y(k)}) + const., \tag{3.2}$$

where $\boldsymbol{y}(k)$ is the labeling associated with the k'th training sequence $\boldsymbol{x}(k)$. Equation (3.2) is seen to be proportional to the negative logarithm of the training set likelihood and *Minimum Discrimination Information* is thus equivalent to ML.

An alternative to using the training set distribution is to replace $P(\boldsymbol{x}|\boldsymbol{y})$ by a distribution, $Q(\boldsymbol{x}|\boldsymbol{\Phi}_y)$, which is "less constrained" than the HMM distribution [ER89, ER90], and then minimize $D(Q_\Phi||P_\Theta)$ in a two step iterative procedure. In the first step, $\boldsymbol{\Phi}$ is chosen such that $D(Q_\Phi||P_\Theta)$ is minimized for the current estimate of the HMM distribution. In the second step $\boldsymbol{\Theta}$ is chosen to minimize the same D but now given the current $\boldsymbol{\Phi}$. This procedure will converge to a solution where the HMM distribution has minimum discrimination information w.r.t. the final Q [ER89]. To our knowledge, this approach has never been implemented in practice and it is thus unclear whether it will lead to better classifiers than ML estimation.

The ML criterion states that the model representing a particular class should only be trained using examples from that class. In theory, this is optimal because the ML estimate is *asymptotically efficient* (see *e.g.*, [DH73]), *i.e.*, it is a minimum variance, unbiased estimate of the optimal parameters if the functional form $P(\boldsymbol{x}; \boldsymbol{\Theta}_y)$ is sufficiently general to "mirror" the true distribution and if the training set is sufficiently large. Thus, provided the model is *correct* the model distribution will approach the true distribution as the model parameters approach the optimal parameters. In theory, ML estimation can therefore lead to the optimal Bayes classifier with minimum error rate if the associated MAP decoder is used and if the true a priori distribution $P(\boldsymbol{y})$ is known [Nád83].

In speech recognition it is well known that the true acoustic model is not contained in the HMM model space since the human speech system is complex and non-stationary. Similarly, parametric language models are generally not capable of capturing all of the complexity and variation in any natural language and will therefore be poor representations of the true a priori distributions. Furthermore, the assumption in the likelihood criterion that training sequences are independent implies that all examples are given equal weight. The effect of this is that typical examples from a given class will tend to dominate the ML estimate for that class. The result may be a suboptimal placement of decision boundaries.

[1] For continuously valued observation vectors, the sum in (3.1) is replaced by an integration over the space of continuous observation vectors.

These restrictions and the fact that training data is always limited means that the optimal Bayes classifier is never obtained in practice.

3.2 Discriminative Training by Indirect Methods

The indirect methods for discriminative training are similar in spirit to ML estimation, in the sense that they rely on matching a model distribution to a true distribution. The main difference is that here the model is trained to optimally match the *a posteriori probability* distribution of classes $P(\boldsymbol{y}|\boldsymbol{x})$ rather than the within-class data distributions. Typically, data distributions are much more complex to describe than posterior distributions, because the data distributions must describe all variations of the data within a class while the posterior distribution is only concerned with the boundaries between the classes. For example, to describe the two data distributions in the upper panel of figure 3.1 fairly complex class conditional density functions are necessary. On the other hand, the posterior distribution illustrated in figure 3.1 is far simpler than the data distributions because the boundary between the two classes is clear regardless of the fine structure in the data. This means that a simpler model can be expected to obtain better performance when trained discriminatively instead of using the ML criterion. Thus, discriminative training can lead to more *robust* models with improved generalization to new examples. Experimental evidence of this has been given for speech recognition in *e.g.*, [Joh96, KVY93, NCM94].

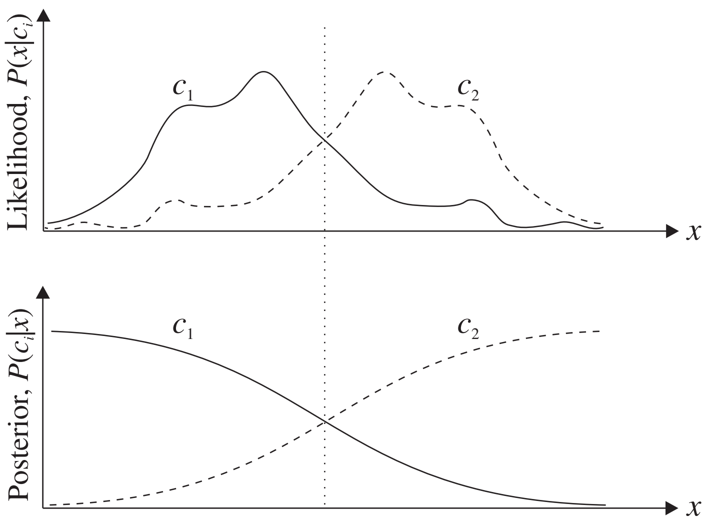

Figure 3.1: Illustration of within-class and posterior distributions for a two class problem with continuous observations.

3.2.1 Conditional Maximum Likelihood

The discrimination information between the posterior distribution $P_\Theta = P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$ given by the model and the true distribution $P = P(\boldsymbol{y}|\boldsymbol{x})$ is

$$D(P||P_\Theta) = \sum_{\boldsymbol{y}\in\mathcal{Y}}\sum_{\boldsymbol{x}\in\mathcal{X}} P(\boldsymbol{y}|\boldsymbol{x}) \log \frac{P(\boldsymbol{y}|\boldsymbol{x})}{P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})}. \tag{3.3}$$

Minimizing this quantity gives an optimal match between the true and the model distribution but as for the ML case the true distribution is unknown. Replacing it with the training set empirical distribution yields [ER90]

$$D(P||P_\Theta) = -\frac{1}{K}\sum_{k=1}^{K} \log P(\boldsymbol{y}(k)|\boldsymbol{x}(k);\boldsymbol{\Theta}). \tag{3.4}$$

The parameter set minimizing (3.4) is known as the *Conditional Maximum Likelihood* (CML) estimate,

$$\begin{aligned}
\hat{\boldsymbol{\Theta}}_{CML} &= \operatorname*{argmax}_{\boldsymbol{\Theta}} \prod_{k=1}^{K} P(\boldsymbol{y}(k)|\boldsymbol{x}(k);\boldsymbol{\Theta}) & (3.5)\\
&= \operatorname*{argmax}_{\boldsymbol{\Theta}} \prod_{k=1}^{K} \frac{P(\boldsymbol{x}(k),\boldsymbol{y}(k);\boldsymbol{\Theta})}{P(\boldsymbol{x}(k);\boldsymbol{\Theta})}, & (3.6)
\end{aligned}$$

and was first proposed for speech recognition by Nádas in [Nád83]. The CML criterion is equivalent to the maximum likelihood criterion used for classification training of neural networks when the networks use a softmax output function [HKP91, RL91]. This is easy to see by noting that CML corresponds to ML training of a model with a softmax output transformation,

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) = \frac{\exp(\log P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}))}{\sum_{\boldsymbol{\nu}} \exp(\log P(\boldsymbol{x},\boldsymbol{\nu};\boldsymbol{\Theta}))} = \frac{\exp(h_y)}{\sum_{\boldsymbol{\nu}} \exp(h_\nu)}, \tag{3.7}$$

where $h_y = \log P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ and the sum in the denominator is extended over all possible labelings $\boldsymbol{\nu} \in \mathcal{Y}$. Richard and Lippmann [RL91] proved that for a sufficiently complex neural network with a softmax output function the posterior distribution given by the network will approach the true distribution as the likelihood approaches a global maximum in the limit of an infinite size training set. Naturally, this also holds for the CML criterion provided that the HMM is sufficiently complex. In [Kon96, BKM94] CML estimation was called Maximum A Posteriori (MAP) training and used for HMM/NN hybrid optimization.

From (3.6) we see that the CML criterion can be expressed as a rational function between two likelihoods; the *clamped phase* likelihood $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ and the *free-running* or *recognition phase* likelihood $P(\boldsymbol{x};\boldsymbol{\Theta})$. The term free-running means that the labels are not taken into account, so this phase is similar to the decoding phase, where we wish to find the labels for an observation sequence. The constraint imposed by the labels during training gives rise to the name *clamped phase*; this terminology is borrowed from the Boltzmann machine literature [AHS85]. Thus, CML estimation adjusts the model parameters so as to make the free-running 'recognition model' as close as possible to the clamped model.

The free-running term constitutes the difference to standard ML estimation. For small vocabulary isolated word recognition the free-running likelihood is simple to compute;

it is just a sum over all words of the product of acoustic likelihood and word prior, $P(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_\nu P(\boldsymbol{x};\boldsymbol{\Theta}_\nu)P(\nu)$. For large vocabulary isolated word recognition or continuous speech recognition the free-running term can be computationally inhibitive due to the large number of possible labelings. However, if the language model is a bigram it can be combined directly with the acoustic model into a composite HMM, and then the free-running likelihood can be computed by applying the standard forward algorithm to this composite HMM. In cases where a reasonable size recognition phase model cannot be designed one can use an N-best approximation, where only the N largest likelihoods obtained in an N-best decoding pass are included in the sum [Cho90]. Furthermore, pruning schemes similar to beam search for Viterbi decoding can also reduce the complexity of the forward and backward recursions considerably [Val95]. For composite models the clamped phase likelihood can be computed by a slightly modified forward algorithm which only considers those paths through the composite HMM that conform with the observed labeling $\boldsymbol{y}$, see chapter 4. Alternatively, one can use the standard forward algorithm on the concatenation $(\boldsymbol{\Theta}_y)$ of symbolic message unit HMMs corresponding to the observed labeling and then scale this likelihood with the language model probability.

Unfortunately, the rational function form of the conditional likelihood makes it impossible to define an auxiliary Q-function as for the ML approach and it is thus not possible to apply the EM algorithm. Instead CML estimation must be based on general iterative optimization techniques like *e.g.*, gradient-based methods. An alternative method known as *extended Baum-Welch* reestimation was proposed by Gopalakrishnan *et al.* [GKNN91] for discrete HMMs and later for continuous and semi-continuous HMMs [NM91, NCM94]. The method was developed in the context of Maximum Mutual Information estimation (see below) but applies equally well to CML estimation. We will discuss this method in further detail in chapter 4.

The discrimination between classes can easily be explained for the CML criterion by expressing the conditional likelihood of the labeling as follows

$$\begin{aligned}
\hat{\boldsymbol{\Theta}}_{CML} &= \operatorname*{argmax}_{\boldsymbol{\Theta}} P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) \\
&= \operatorname*{argmax}_{\boldsymbol{\Theta}} \frac{P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{P(\boldsymbol{x};\boldsymbol{\Theta})} \\
&= \operatorname*{argmax}_{\boldsymbol{\Theta}} \frac{P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}) + \sum_{\boldsymbol{\nu}\neq\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{\nu};\boldsymbol{\Theta})} \\
&= \operatorname*{argmax}_{\boldsymbol{\Theta}} \frac{P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{\sum_{\boldsymbol{\nu}\neq\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{\nu};\boldsymbol{\Theta})}.
\end{aligned} \tag{3.8}$$

From (3.8) we see that increasing the conditional likelihood $P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta})$ corresponds to increasing the likelihood of the correct model $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ and at the same time decreasing the likelihoods of all competing models $\sum_{\boldsymbol{\nu}\neq\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{\nu};\boldsymbol{\Theta})$. This forces the model to discriminate between the different labelings by learning the decision boundaries. Although this explanation is intuitive, it is not always the result of training. The effect of increasing the conditional likelihood can also be obtained if *e.g.*, the likelihood for *all* hypotheses are decreased (increased), but in such a way that the likelihood for the competing hypotheses decrease (increase) at a faster rate than that of the correct hypothesis. This scenario is often encountered in practice as we shall see in chapter 5. Note that the discriminative aspect of training means that modeling resources are distributed among the individual submodels for the different classes.

Contrary to ML, the training set samples contribute differently when using the CML criterion according to whether they are well classified or not. The CML criterion is a *geometrical* average of training sample label a posteriori probabilities and samples with large posteriors, $P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) \approx 1$, will not contribute much whereas samples with small posteriors will tend to dominate the CML criterion. If the model gives a large conditional probability for the observed labeling the sequence will be classified correctly with high probability, whereas smaller values increase the risk of misclassification. CML estimation is therefore effectively dominated by examples that are incorrectly classified and far away from a decision boundary ($P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) \approx 0$) [NSB90]. The large contribution from clearly misclassified examples has the undesirable effect that CML estimation is very sensitive to *outliers* and mislabelings in the training set. Unfortunately, this can result in oscillatory training scenarios and suboptimal placement of decision boundaries. For small training sets and well fitting (correct) models better performance can therefore be expected by using ML estimation [Nád83, NNP88, NSB90]. However, for incorrect models CML estimation can result in better recognizers as discussed and experimentally verified in [NNP88, NSB90].

3.2.2 Maximum Mutual Information

Let X, Y be random variables with realizations x, y. Then the entropy of X is defined by

$$H(X) = -\sum_{x} P(x) \log P(x) = -E[\log P(x)]. \tag{3.9}$$

$H(X)$ measures the amount of information[2] needed to specify the outcome of X when using an optimal encoding scheme. Similarly, the conditional entropy

$$H(Y|X) = -\sum_{x,y} P(x,y) \log P(y|x) = -E[\log P(y|x)], \tag{3.10}$$

measures the average amount of information needed to specify the outcome of Y given knowledge of X. By replacing the expectation with the training set empirical average and the conditional distribution by the one given by the model we find

$$H_{\Theta}(Y|X) = -\frac{1}{K}\sum_{k=1}^{K} \log P(\boldsymbol{y}(k)|\boldsymbol{x}(k);\boldsymbol{\Theta}), \tag{3.11}$$

from which we see that CML estimation corresponds to minimizing the conditional entropy. A conditional entropy of zero means that the observed acoustic signal gives sufficient information for perfect decoding. A conditional entropy equal to the entropy of the language model,

$$H_{\Theta}(Y) = -\frac{1}{K}\sum_{k=1}^{K} \log P(\boldsymbol{y}(k);\boldsymbol{\Theta}), \tag{3.12}$$

means that no information about the labeling is extracted from the acoustic signal.

Based on this formulation the relationship between CML and the Maximum Mutual Information (MMI) criterion is straightforward to see. The average mutual information

[2] If $\log_2$ is used the amount of information is measured in bits.

between Y and X measures the average amount of information in X as to the identity of Y, and for the HMM distributions it is given by

$$\begin{aligned} I_\Theta(Y;X) &= H_\Theta(Y) + H_\Theta(X) - H_\Theta(X,Y) \\ &= H_\Theta(Y) - H_\Theta(Y|X) \\ &= H_\Theta(X) - H_\Theta(X|Y) \\ &= -\frac{1}{K}\sum_{k=1}^{K} \log \frac{P(\boldsymbol{x}(k);\boldsymbol{\Theta}_{y(k)})}{P(\boldsymbol{x}(k);\boldsymbol{\Theta})}. \end{aligned} \tag{3.13}$$

In terms of HMM acoustic and language models, $I_\Theta(Y;X) = 0$ means that the acoustic model does not extract any information from the acoustic signal about the labeling, that is, $H_\Theta(X|Y) = H_\Theta(X)$. On the other hand, a value equal to the entropy of the language model, $I_\Theta(Y;X) = H_\Theta(Y)$, implies that the acoustic model removes all uncertainty about the labeling. Thus, maximizing the mutual information can lead to a perfect decoder.

If the language model and consequently $H_\Theta(Y)$ is independent of the parameter set $\boldsymbol{\Theta}$ then it follows from (3.13) that CML estimation is equivalent to MMI estimation. The assumption of an independent language model during acoustic model training is reasonable if it can be estimated more reliably from a representative text corpus. In this thesis we will use the term MMI estimation for models with a fixed language model, and CML estimation when acoustic and language model parameters are jointly estimated from the acoustic data.

MMI, CML and ML estimation can all be considered special cases of the so-called H-criterion proposed in [GKN+98, GKNN91],

$$H_{\alpha,\beta,\gamma} = \alpha H(Y,X) + \beta H(Y) + \gamma H(X). \tag{3.14}$$

For $(\alpha,\beta,\gamma) = (1,0,0)$, $(\alpha,\beta,\gamma) = (1,-1,-1)$ and $(\alpha,\beta,\gamma) = (1,0,-1)$ (3.14) is equivalent to ML, MMI and CML estimation respectively. If $\alpha = 1$ and $\beta = 0$ then $\gamma \neq 0$ corresponds to a weighting exponent applied to the free-running likelihood in the denominator of the CML criterion [GKNN91].

One of the first speech recognition applications of MMI was probably the work by Bahl and colleagues [BBdSM86] at IBM on isolated word recognition. Since then a number of applications have been reported in the literature including connected digit recognition [CNM91, NM91, NCM94], continuous phoneme recognition [Mer88, KVY93] and large vocabulary continuous speech recognition [Cho90, YNC94]. Recently, CML estimation has been applied to continuous phoneme recognition in [KR98, RK97, Joh96].

3.2.3 Mean Squared Error

Neural network classifiers are commonly trained using the so-called *Mean Squared Error* criterion [HKP91, Bis95], which measures the average squared difference between the model output and the desired output. The mean squared error criterion for HMM training can be formulated as [NSB90]

$$E_{MSE} = \frac{1}{K}\sum_{k=1}^{K}\sum_{\boldsymbol{\nu}\in\mathcal{Y}} [P(\boldsymbol{\nu}|\boldsymbol{x}(k);\boldsymbol{\Theta}) - \delta_{\boldsymbol{\nu},\boldsymbol{y}(k)}]^2, \tag{3.15}$$

where $\delta_{\boldsymbol{\nu},\boldsymbol{y}(k)}$ is one if and only if $\boldsymbol{\nu}$ is identical to the observed labeling $\boldsymbol{y}(k)$ for training sequence k and otherwise zero. In the limit of an infinite size training set the model will estimate the true a posteriori distribution if it is possible to reach the global minimum of (3.15) during training as shown in [RL91]. Thus, analogous to CML estimation the optimal classifier can in theory be obtained with the mean squared error criterion.

In the mean squared error criterion training samples contribute differently according to their distance from a decision boundary [NSB90]. Clearly correctly classified samples with $P(\boldsymbol{y}(k)|\boldsymbol{x}(k);\boldsymbol{\Theta}) \approx 1$ and clearly incorrectly classified samples with $P(\boldsymbol{\nu}|\boldsymbol{x}(k);\boldsymbol{\Theta}) \approx 1$, for some $\boldsymbol{\nu} \neq \boldsymbol{y}(k)$ do not contribute significantly to the error criterion. The samples that provide the largest contributions are those which lie close to decision boundaries because for these samples $P(\boldsymbol{\nu}|\boldsymbol{x}(k);\boldsymbol{\Theta})$ is non-zero for several different labelings. Mean squared error estimation is therefore robust to outliers, however, at the cost of utilizing even fewer training examples than CML. A slower convergence compared to CML training is therefore anticipated in general.

Mean squared error estimation is straightforward to implement for HMM based small vocabulary isolated word recognition where the sum over all possible words in (3.15) is tractable. For large vocabularies or continuous speech recognition it is, however, not possible to compute the sum unless an N-best approximation is used. In combination with the slow convergence this is probably the main reason why mean squared error estimation has not found widespread use in discriminative HMM modeling for speech recognition.

3.3 Discriminative Training by Direct Methods

The aim of discriminative training by direct methods is to directly minimize the expected error rate of the classifier by adjusting the model parameters. Thus, these methods do not aim at finding the "correct" model in the sense of optimally matching the true distribution, but rather to find the placement of decision boundaries that minimizes the error rate on the training set.

3.3.1 Corrective Training

From classical linear discriminant theory the direct approaches are known as *relaxation procedures*. A well known example is the *perceptron learning rule*, which was designed for finding the optimal placement of a separating plane in linearly separable problems. Although convergence of the perceptron learning rule can only be proven for linearly separable problems experimental evidence suggests that it often applies equally well to problems that are not linearly separable.

A simple heuristic algorithm called *corrective training*, which is similar in spirit to the perceptron learning rule, was proposed for isolated word recognition using discrete HMMs by Bahl and colleagues [BBdSM93]. In corrective training there is no assumption of model correctness and for any set of models the algorithm attempts to adjust the parameters such that the number of recognition errors is decreased. Instead of only correcting parameters for misrecognized words Bahl and colleagues observed a faster convergence by doing parameter corrections whenever the probability of an incorrect word was "close" to the probability of the correct word. On an isolated word recognition task the corrective training algorithm obtained significantly better results than both ML and MMI estimation [BBdSM93].

In [NM91, CNM91, NCM94] a related approach denoted *corrective MMI* estimation was proposed for connected digit recognition. In corrective MMI an initially ML estimated model is retrained by standard MMI estimation but only using the utterances that are incorrectly recognized. Since correctly recognized utterances do not contribute much to MMI training, estimation based on incorrectly classified examples gives practically the same parameters as estimation based on the full training set.

Corrective training only makes sense when it is possible to recognize entire utterances with fairly high accuracy. Although the word error rate can be quite small in a continuous speech recognition task, this does not apply to the sentence error rate, which is often larger than 50%. The corrective methods are thus not directly applicable to continuous speech recognition tasks.

3.3.2 Minimum Empirical Error Rate

In [LER90] an estimate of the expected error rate of a classifier associated with the MAP decoder was defined by

$$\hat{P}(error; \mathbf{\Theta}) = 1 - \frac{1}{K}\sum_{k=1}^{K} P(\boldsymbol{y}(k)|\boldsymbol{x}(k); \mathbf{\Theta}). \tag{3.16}$$

Minimizing $\hat{P}(error; \mathbf{\Theta})$ is called *Minimum Empirical Error* estimation [LER90, ER90]. Comparing to CML estimation we see that both of these approaches use the same statistics but in a rather different way. Whereas the CML criterion is a geometric average over the training set posteriors, the empirical error rate estimate is an arithmetic average. Thus, all training samples contribute equally to the minimum empirical error criterion and the sensitivity to outliers and mislabelings is thereby reduced compared to the CML criterion. Note that if the training set is considered as one long sequence then the CML and minimum empirical error rate criteria are identical.

3.3.3 Minimum Classification Error

The *Minimum Classification Error* training for HMMs proposed in [KLJ91, JK92] is similar to the empirical error rate method in the sense that it maximizes a smooth estimate of the expected error rate. It is based on defining a so-called *misclassification measure*, which gives a scalar value corresponding to how "well" the model discriminates between classes. In terms of the joint probability of observations and labels, the misclassification measure can be written [KLJ91, JK92]

$$\widehat{d_y}(\boldsymbol{x}) = -\log P(\boldsymbol{x}, \boldsymbol{y}; \mathbf{\Theta}) + \log\left[\frac{1}{M-1}\sum_{\boldsymbol{\nu}\neq\boldsymbol{y}} P(\boldsymbol{x}, \boldsymbol{\nu}; \mathbf{\Theta})^{\psi}\right]^{\frac{1}{\psi}}, \tag{3.17}$$

where M is the total number of possible hypotheses and the sum is extended over all competing hypotheses. In line with the discussion for the mean squared error criterion, this sum generally cannot be computed for continuous speech recognition unless only the N best competing hypotheses are included. For large values of ψ only the top scoring competing hypotheses will contribute significantly to the sum and for $\psi \to \infty$ the misclassification measure equals the discriminant distance between the correct and highest scoring competing hypothesis,

$$d_y(\boldsymbol{x}) = -\log P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) + \max_{\boldsymbol{\nu} \neq \boldsymbol{y}} \log P(\boldsymbol{x}, \boldsymbol{\nu}; \boldsymbol{\Theta}). \tag{3.18}$$

The value of $\widehat{d}_y(\boldsymbol{x})$ for a given utterance indicates how well it is recognized by the model. Large positive (negative) values correspond to clearly incorrect (clearly correct) recognition, whereas small positive (negative) values indicate a near-miss (almost-correct).

Based on the misclassification measure approximation to the discriminant distance it is possible to define the average training set classification error. Ideally, the average number of errors can be calculated by passing the discriminant distance through a binary step function. However, since the step function is discontinuous and consequently not differentiable, a sigmoid approximation is normally used such that

$$E_{MCE} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{1 + \exp[-\alpha \widehat{d}_{y(k)}(\boldsymbol{x}(k))]}, \tag{3.19}$$

where α is a positive constant controlling the steepness of the sigmoid. For large α's the sigmoid approaches the binary step function, and incorrectly classified ($d_y \geq 0$) training samples give an equal contribution ($\approx 1/K$) to the error criterion, whereas correctly classified samples do not contribute at all. A small α implies that all samples contribute significantly, which is similar to the minimum empirical error rate case. Note that for $\psi \rightarrow \infty$ and $\alpha = 1$ the minimum classification error and minimum empirical error rate criteria are equivalent.

3.4 Summary

The aim of all the discriminative training algorithms discussed in this chapter is to give parameter estimates that result in better recognition performance than when ML estimation is used. Contrary to ML, the discriminative methods adjust model parameters in a supervised fashion to optimally place the decision boundaries implemented by the model. In the CML/MMI and mean squared error approaches this is obtained by matching the true a posteriori probability of the classes, whereas minimum empirical error rate and minimum classification error estimation directly minimize a smooth representation of the expected error rate on the training set. The main difference between the different optimization algorithms lies in the way they use the examples for training. CML/MMI rely mainly on misclassified examples far away from decision boundaries. The mean squared error estimator effectively only utilizes samples close to decision boundaries, whereby the outlier problem in the CML estimator is circumvented but at the cost of slower convergence. Finally, in minimum empirical error rate estimation all samples contribute significantly.

In this thesis we will use CML estimation. The arguments in favor of CML are the straightforward extension from ML estimation and that it contains no "control" parameters to adjust. Furthermore, all competing hypotheses can be used for discrimination if a composite HMM architecture is employed. This is contrary to mean squared error and minimum classification error training, where the likelihoods for <u>each</u> of the competing hypotheses and not just the sum of them must be computed. For continuous speech recognition this is computationally prohibitive unless an N-best approximation is used [CS94, Cho90].

With this choice in mind, one must be aware of the outlier sensitivity of the CML criterion. However, a possible way of handling the CML outlier problem is to use CML for initial training and then switch to *e.g.*, mean squared error training to fine-tune the decision boundary placement. This is reasonable if the model is not capable of overfitting the training set. Unfortunately, a common observation in HMM modeling for speech recognition is that the more general the model, the better performance can be obtained on unseen data. A likely explanation for this is that the HMM architecture is too constrained to implement the true distributions unless a large number of parameters are used. Such models will overfit the training data if trained for a sufficient number of iterations. Training is therefore usually stopped when the performance on an independent representative validation set is at a maximum. Thus, training will usually be terminated before a local minimum of the CML criterion is reached, and so fine-tuning by mean squared error estimation seems meaningless. Consequently, we have only used pure CML estimation in the work documented by this thesis.

Chapter 4

Class HMMs

In this chapter we describe the so-called Class HMM (CHMM), which can be viewed as a generalization of the composite HMM discussed in chapter 2. We will introduce the architecture as a simple extension of the standard HMM framework where each state in addition to the match distribution also has a distribution over labels. As will be discussed in section 4.2 and section 4.3 this formulation allows the CHMM to be trained by Baum-Welch like reestimation formulas to maximize the joint likelihood $P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta})$ for observation and label sequences. Based on the derived reestimation formulas for ML training it is a simple matter to find the gradients w.r.t. various parameters in the CHMM needed for discriminative conditional maximum likelihood estimation. This will be discussed in section 4.4. We conclude the chapter by describing how the CML estimated CHMM can be *normalized globally* at the sequence level with no extra computational burden. This normalization is different from the *local normalization* enforced in standard HMMs, where the match and transition distributions are required to normalize to one for each state. With global normalization the model can still be given a probabilistic interpretation even though the parameters associated with the individual states no longer normalize. This can be very attractive in some applications and it will be necessary when we turn the CHMM into the hidden neural network hybrid in chapter 6.

4.1 The Model

To accommodate modeling of the joint probability of observations and labels we use one "global" HMM and assign a distribution over possible labels to each state. Let the set of possible labels be denoted $\mathcal{C}$. Then the probability of matching label $c \in \mathcal{C}$ in state i is defined by

$$\psi_i(c) = P(y_l = c | \pi = i). \tag{4.1}$$

Analogous to the match distributions for discrete HMMs, the label distribution is time-invariant so the probability of matching label $c \in \mathcal{C}$ is the same no matter where this label occurs in the label sequence. In some applications it makes sense to restrict some states to model only a subset of all the possible labels and sometimes even to fix the label probabilities a priori. A particularly simple case is obtained if we only allow one label to be modeled in each state, *i.e.*, if state i is supposed to model label c^i then $\psi_i(c) = \delta_{c^i,c}$. States modeling a single label will be denoted *single-label-states* and states modeling several or all labels will be denoted *multiple-label-states*.

Using this setup the mental picture of class submodels as used in speech recognition can be discarded; the model can have any structure, that is, any state can be connected to any other state in the big model. Because each state has a label or a distribution over labels this model was called a Class HMM (CHMM) in [Kro94].

In the original formulation of the CHMM by Krogh [Kro94] the labeling was assumed complete. This provides a much simpler case than incomplete labeling because the label distribution for complete labeling can be treated in a similar way as the match distribution, see below. Thus, for complete labeling one can consider the CHMM as a model capable of matching two sequences in parallel; the label and the observation sequence. For incomplete labeling there is no longer a one-to-one correspondence between the label and the observation sequence, *i.e.*, the s'th label in the label sequence is not associated with a particular observation. To handle this case it is necessary somehow to allow the states not to match any label and to keep track of the alignment between the observation and label sequence.

For simplicity we will limit the discussion below to discrete observations, but keeping in mind that the framework applies equally well to continuous observations. For discrete observations the CHMM is completely specified by the parameter set

$$\boldsymbol{\Theta} = \{i, j = 1, \ldots, N; a \in \mathcal{A}; c \in \mathcal{C} | \theta_{ij}, \phi_i(a), \psi_i(c)\}. \tag{4.2}$$

Since only the case of single-label-states will be evaluated experimentally in this thesis, the presentation below will focus mainly on this case. The multiple-label-state case is detailed in appendix B as a reference for future research.

4.2 Complete Label Maximum Likelihood Estimation

Assume that the observation sequence $\boldsymbol{x}_1^L = x_1, \ldots, x_L$ is associated with a complete label sequence $\boldsymbol{y}_1^L = y_1, \ldots, y_L$. For single-label-states only one particular label is allowed in each state, and the label distribution is a delta-function; $\psi_i(y_l) = \delta_{c^i, y_l}$, where c^i is the label assigned to state i. Consequently, the joint likelihood for labels and observations can be computed by considering only allowed paths Π_y through the model, in which the labels of the states agree with the labels of the observations,[1]

$$P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\pi}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi} \in \Pi_y} P(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\Theta}). \tag{4.3}$$

If the labels are assumed *state conditionally independent* then the joint likelihood can be expressed as (see appendix B)

$$\begin{aligned} P(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{\Theta}) &= \sum_{\boldsymbol{\pi}} \prod_l \phi_{\pi_l}(x_l) \delta_{c^{\pi_l}, y_l} \theta_{\pi_{l-1}\pi_l} \\ &= \sum_{\boldsymbol{\pi} \in \Pi_y} \prod_l \phi_{\pi_l}(x_l) \theta_{\pi_{l-1}\pi_l}, \end{aligned} \tag{4.4}$$

which shows that the labels are treated in the same way as the observations. The likelihood (4.4) can therefore be computed by a very simple modification of the standard forward-backward algorithm, where all multiplications by the match probability are replaced by the

[1] As for the HMM it is assumed that we only have one training sequence in the following. The generalization to multiple training sequences is straightforward.

product $\phi_i(x_l)\delta_{c^i,y_l}$. The modified forward-backward procedure is shown in algorithm 4.1 and algorithm 4.2.

Algorithm 4.1 Forward algorithm $[P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^L; \boldsymbol{\Theta})]$

Definition:	$\tilde{\alpha}_j(l) = P(\boldsymbol{x}_1^l, \boldsymbol{y}_1^l, \pi_l = j; \boldsymbol{\Theta})$	
Initialization:	$\tilde{\alpha}_j(1) = \phi_j(x_1)\delta_{c^j,y_1}\theta_{0j}$,	$1 \le j \le N$
Recursion:	$\tilde{\alpha}_j(l) = \phi_j(x_l)\delta_{c^j,y_l}\sum_{i=1}^N \theta_{ij}\tilde{\alpha}_i(l-1)$,	$1 \le j \le N, 1 < l \le L$
Termination:	$P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{i=1}^N \theta_{iN+1}\tilde{\alpha}_i(L)$	

Algorithm 4.2 Backward algorithm $[P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^L; \boldsymbol{\Theta})]$

Definition:	$\tilde{\beta}_i(l) = P(\boldsymbol{x}_{l+1}^L, \boldsymbol{y}_{l+1}^L \vert \pi_l = i; \boldsymbol{\Theta})$	
Initialization:	$\tilde{\beta}_i(L) = \theta_{iN+1}$,	$1 \le i \le N$
Recursion:	$\tilde{\beta}_i(l) = \sum_{j=1}^N \tilde{\beta}_j(l+1)\theta_{ij}\delta_{c^j,y_{l+1}}\phi_j(x_{l+1})$,	$1 \le i \le N, 1 \le l < L$
Termination:	$P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{j=1}^N \tilde{\beta}_j(1)\theta_{0j}\delta_{c^j,y_1}\phi_j(x_1)$	

Note that a computationally more efficient implementation of the backward algorithm can be obtained by moving the delta-functions outside the summation such that, $\tilde{\beta}_i(l) = \delta_{c^i,y_l}\sum_j \tilde{\beta}_j(l+1)\theta_{ij}\phi_j(x_{l+1})$. In this case the backward recursion is initialized by $\tilde{\beta}_i(L) = \delta_{c^i,y_L}\theta_{iN+1}$ for all i.

If we think of $\alpha_i(l)$ (or $\beta_i(l)$) as a matrix, the new algorithm corresponds to masking this matrix such that only allowed regions are calculated, see figure 4.1. Therefore the calculation is faster than the standard forward (or backward) calculation of the whole matrix. In fact, if we assume that the same number N_c of states are used for modeling *each* class then the computational requirement of the new forward-backward algorithm for an ergodic model scales as $\mathcal{O}(N_c^2 L)$, see figure 4.1. This should be compared to $\mathcal{O}(N^2 L)$ for the standard forward-backward algorithm. In most cases $N^2 \gg N_c^2$ so the computational complexity is reduced by a factor $(N/N_c)^2 \gg 1$ by only considering allowed paths.

The Baum-Welch reestimation equations given for the standard HMM in chapter 2 also applies for complete label ML training of single-label-state CHMMs. The only difference is that the expected counts should now only be computed along those paths Π_y that are allowed. Define

$$
\begin{aligned}
m_i(l) &= E_{\boldsymbol{\pi}}[\delta_{\pi_l,i} \vert \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Theta}] \\
&= P(\pi_l = i \vert \boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) \\
&= \frac{P(\pi_l = i, \boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta})}{P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta})}
\end{aligned}
\tag{4.5}
$$

and

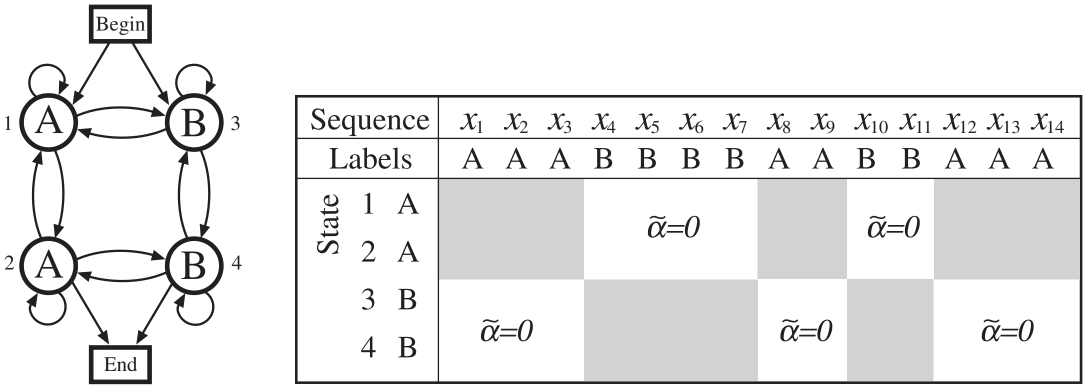

Figure 4.1: ***Left panel:*** A very simple model with four states, two with label 'A' and two with label 'B'. ***Right panel:*** The $\tilde{\alpha}$ matrix for an example observation sequence $\boldsymbol{x} = x_1, \ldots, x_{14}$ with complete labels. The grey areas of the matrix are calculated as in the standard forward algorithm whereas $\tilde{\alpha}$ is set to zero in the white areas. The $\tilde{\beta}$ matrix is calculated in the same way, but from right to left (only applies to the computational efficient implementation, see main text).

$$\begin{aligned} m_{ij}(l) &= E_{\boldsymbol{\pi}}[\delta_{\pi_{l-1},i}\delta_{\pi_l,j}|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}] \\ &= P(\pi_{l-1} = i, \pi_l = j|\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}) \\ &= \frac{P(\pi_{l-1} = i, \pi_l = j, \boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}. \end{aligned} \tag{4.6}$$

$m_i(l)$ is the expected number of times we are in state i at time l computed only along allowed paths Π_y. Similarly, $m_{ij}(l)$ denotes the expected number of times we use the transition from state i to state j at time l in the allowed paths. These counts are computed in exactly the same way as the n's for the standard HMM, but using the new forward and backward variables in (2.46)-(2.47) and replacing $\phi_i(x_l)$ with $\phi_i(x_l)\delta_{c^i,y_l}$,

$$m_{ij}(l) = \frac{\tilde{\alpha}_i(l-1)\theta_{ij}\phi_j(x_l)\delta_{c^j,y_l}\tilde{\beta}_j(l)}{\sum_{i'}\tilde{\alpha}_{i'}(l)\tilde{\beta}_{i'}(l)} \tag{4.7}$$

and

$$m_i(l) = \frac{\tilde{\alpha}_i(l)\tilde{\beta}_i(l)}{\sum_{i'}\tilde{\alpha}_{i'}(l)\tilde{\beta}_{i'}(l)}. \tag{4.8}$$

The reestimation equations for match and transition probabilities are now obtained by replacing the n's in (2.43)-(2.44) by the m's given above.

The single-label-state case is a generalization of the composite HMM discussed in section 2.7.2, in which acoustic symbolic message unit HMMs are connected in a big model with bigram language model transition probabilities. Indeed, if states with identical labels are arranged in submodels and if we only allow one transition between such submodels, then the CHMM is equivalent to a composite HMM. That is, the joint probability for observations and complete labels equals the product of acoustic model probabilities and separately (ML) estimated language model bigram probabilities. However, the CHMM formulation is more general because it is possible to have transition probabilities from several states modeling a particular class to any other state in the model. Such "language model" transitions are not straightforward to estimate separately. Furthermore, when training the CHMM by conditional maximum likelihood as discussed in section 4.4, these transition probabilities are estimated in a discriminative fashion.

4.2.1 Multiple-Label-States

The extension of the above framework to states with several labels is simple. As shown in appendix B we just have to replace the delta-function in the forward-backward algorithm described above with the full distribution $\psi_i(y_l)$. All other equations remain the same, but in addition to the reestimation equations for match and transition probabilities there will now also be one for estimating the parameters of the label distribution, see appendix B. The multiple-label-state CHMM is somewhat similar to the so-called *Input/Output HMM* (IOHMM) recently proposed by Bengio and Frasconi [BF96]. This will be discussed further in chapter 6.

Whereas decoding of the single-label-state CHMM is identical to that of the standard HMM, this is not the case for states with multiple labels. Consider *e.g.*, the optimal path found by the standard Viterbi decoder. For multiple-label-states this path cannot be translated uniquely into a label sequence, but as discussed in the appendix it is possible to modify the standard decoding algorithms so that they also apply for multiple-label-states.

4.3 Incomplete Label Maximum Likelihood Estimation

For incomplete labeling the joint likelihood $P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta})$ for the observation and label sequence can also be expressed as in (4.3), but Π_y now has a different, less restrictive interpretation: it is the set of paths in which the incomplete label sequence corresponds to the sequence of *groups of states with the same label*. The easiest way to ensure that we only allow paths from Π_y is by rearranging the (big) model temporarily for each observation sequence. This is similar to the model building technique discussed in chapter 2, where symbolic message unit submodels are concatenated with *unit transitions* according to the incomplete labeling of the spoken utterance. In the CHMM all the states with a label matching the first label are copied to a temporary model including all transitions between them as well as the θ_{0i} corresponding to these states. Then all the states matching the second label are copied including all internal transitions *and* transitions from the first set of states to the second set of states (*not vice versa*). This is done for the whole incomplete label sequence. All the transition probabilities retain their *original value* in the temporary model. The joint likelihood for observations and incomplete labels is now computed by applying the *standard* forward algorithm to this temporary model, see figure 4.2. Similarly, the expected counts $m_i(l)$ and $m_{ij}(l)$ can be computed by the expressions for the standard HMM but using the standard forward-backward algorithm on the temporary HMM. This is done for all K observation sequences and the accumulated m values are used to update the model by the Baum-Welch reestimation equations (2.43)-(2.44).

Strictly speaking, the temporary model is not a proper HMM because the transition probabilities in some states will no longer sum to one. Fortunately, the forward-backward algorithm as well as the reestimation equations are still valid as the above approach corresponds to putting restrictions on which paths are allowed.

The computational complexity of running the forward-backward algorithm on the temporary model can be slightly higher than for the originating model. Again, if N_c states

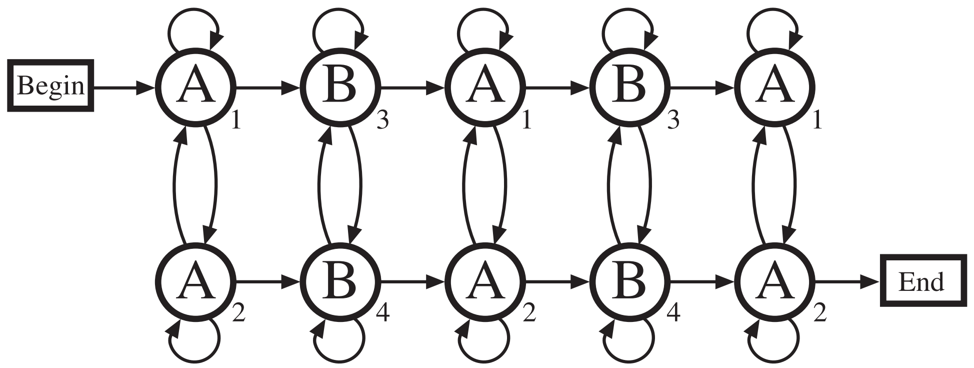

Figure 4.2: For the same model as in figure 4.1, this example shows how the model is temporarily rearranged for gathering statistics (*i.e.* calculate expected m-counts) for a sequence with incomplete labels $ABABA$.

are used for each class then the computational complexity for the temporary model is proportional to the length S of the incomplete label sequence and it scales as $\mathcal{O}(N_c^2 SL)$. This is a factor S more than for the complete label case.

4.3.1 Multiple-Label-States

When we allow several labels in each state the simple approach of rearranging the model no longer works. A possible way of handling this case is by adding an extra *empty* label, $c = \epsilon$, to the set of labels allowed in each state. The empty label does not "extend" the label sequence and the label distribution in all states now satisfy

$$\psi_i(\epsilon) + \sum_{c \in \mathcal{C}} \psi_i(c) = 1. \tag{4.9}$$

An analogous approach was recently proposed by Bengio and Bengio [BB96] for training so-called asynchronous Input/Output HMMs (IOHMMs), but they used an additional "emit-or-not" distribution to take care of empty labels.

To keep track of the alignment between the observation sequence $\boldsymbol{x} = \boldsymbol{x}_1^L$ and incomplete label sequence $\boldsymbol{y} = \boldsymbol{y}_1^S$ an additional hidden variable τ_l can be used; τ_l is equal to s if the s'th label y_s is matched at time l. Then $\boldsymbol{\tau} = \boldsymbol{\tau}_1^L = \tau_1, \ldots, \tau_L$ describes the alignment between the two sequences and the joint likelihood can be expressed as

$$P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}, \boldsymbol{\tau}} P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S, \boldsymbol{\pi}_1^L, \boldsymbol{\tau}_1^L; \boldsymbol{\Theta}), \tag{4.10}$$

where we have used explicit time super- and subscripts to indicate that the two sequences are of different length. Based on this formulation it is possible to derive Baum-Welch like reestimation formulas similar to those for the complete label case, but with a different set of expected counts. As shown in appendix B the basic idea in the derivation is to observe that at any time l we can either match a "true" label from $\mathcal{C}$ or the empty label ϵ. This leads to forward-backward recursions and expressions for the expected m-counts, which are composed of two terms; one for matching a label $y_s \in \mathcal{C}$ at time l in state i and one for matching the null label ϵ.

Incomplete label ML estimation for the single-label-state model can in principle also be done by the empty label method. The parameter estimates obtained by the forward-backward algorithm shown in appendix B will, however, generally be different from those obtained by the temporary model approach described above. This is so because the possibility of matching empty labels in all states implies that we can follow paths in the CHMM which are not allowed in the temporary model. Consider for example the observation sequence $\boldsymbol{x} = x_1, \ldots, x_5$ with incomplete labels ABC and assume that we wish to train a three-state fully connected CHMM where state 1 can model label A, state 2 label B and state 3 label C. In the temporary model corresponding to this label sequence we can never model observation x_2 in state 3, but this is indeed possible in the approach where all states can also match the empty label. Furthermore, the computational complexity of the empty label approach scales as $\mathcal{O}(N^2SL)$ (see appendix B) compared to $\mathcal{O}(N_c^2SL)$ for the temporary model method (if each class is modeled by $N_c < N$ states). For these reasons we will only use the temporary model approach in this thesis.

4.4 Conditional Maximum Likelihood Estimation

The Baum-Welch EM algorithm is very elegant and efficient, but unfortunately it does not generalize to CML estimation as discussed in chapter 3. Another possibility is to use more general gradient-based iterative optimization methods, where the new estimate of a generic parameter $\omega \in \boldsymbol{\Theta}$ at time $t+1$ can be expressed by

$$\omega^{(t+1)} = \omega^{(t)} + \Delta\omega^{(t)}. \tag{4.11}$$

For purely gradient-based methods, the parameter change $\Delta\omega^{(t)}$ at iteration t is expressed entirely in terms of the gradients of the CML criterion w.r.t. the ω's. Note that the parameter update can be done either online (update after each sequence) or offline (update after all sequences). Contrary to the EM algorithm, online and offline gradient-based training require the same amount of memory. For online training the $\Delta\omega$'s are computed using the gradients for one sequence, whereas the accumulated gradients over all sequences are used in batch or offline training. We will elaborate on this and various approaches for computing the $\Delta\omega$'s in chapter 5.

In this section we derive the gradients of the CML criterion w.r.t. to the parameters in the CHMM necessary for gradient-based optimization. Furthermore, we will discuss different ways of ensuring that the parameters normalize correctly during training and show that one of these methods can be used to "derive" the extended Baum-Welch reestimation formulas in a heuristic manner.

The expressions derived below are all stated in terms of the expected n- and m-counts. By using the appropriate m-counts defined above and in appendix B the gradients apply to complete as well as incomplete label training for both single- and multiple-label-state CHMMs.

4.4.1 Gradient Derivation

To make the gradient derivation easier we switch to log likelihoods and define the *negative log conditional likelihood*

$$\mathcal{L}(\boldsymbol{\Theta}) = -\log P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) = \mathcal{L}_c(\boldsymbol{\Theta}) - \mathcal{L}_f(\boldsymbol{\Theta}), \tag{4.12}$$

where the negative log likelihood $\mathcal{L}_c(\boldsymbol{\Theta})$ in the clamped phase and $\mathcal{L}_f(\boldsymbol{\Theta})$ in the recognition phase are defined by

$$\begin{aligned}\mathcal{L}_c(\mathbf{\Theta}) &= -\log P(\boldsymbol{x},\boldsymbol{y};\mathbf{\Theta}) &(4.13)\\ \mathcal{L}_f(\mathbf{\Theta}) &= -\log P(\boldsymbol{x};\mathbf{\Theta}). &(4.14)\end{aligned}$$

The derivative of the free-running log likelihood $\mathcal{L}_f(\mathbf{\Theta})$ w.r.t. a generic parameter $\omega \in \mathbf{\Theta}$ can be written

$$\begin{aligned}\frac{\partial \mathcal{L}_f(\mathbf{\Theta})}{\partial \omega} &= \frac{1}{P(\boldsymbol{x};\mathbf{\Theta})}\frac{\partial P(\boldsymbol{x};\mathbf{\Theta})}{\partial \omega}\\ &= \frac{1}{P(\boldsymbol{x};\mathbf{\Theta})}\sum_{\boldsymbol{\pi}}\frac{\partial P(\boldsymbol{x},\boldsymbol{\pi};\mathbf{\Theta})}{\partial \omega}\\ &= \frac{1}{P(\boldsymbol{x};\mathbf{\Theta})}\sum_{\boldsymbol{\pi}}P(\boldsymbol{x},\boldsymbol{\pi};\mathbf{\Theta})\frac{\partial \log P(\boldsymbol{x},\boldsymbol{\pi};\mathbf{\Theta})}{\partial \omega}\\ &= \sum_{\boldsymbol{\pi}}P(\boldsymbol{\pi}|\boldsymbol{x};\mathbf{\Theta})\frac{\partial \log P(\boldsymbol{x},\boldsymbol{\pi};\mathbf{\Theta})}{\partial \omega}, &(4.15)\end{aligned}$$

which is very similar to the gradient of the auxiliary function (2.36) defined for standard HMMs in chapter 2,

$$\frac{\partial \mathcal{Q}_{\boldsymbol{\pi}}(\mathbf{\Theta}|\mathbf{\Theta}^{(t)})}{\partial \omega} = \sum_{\boldsymbol{\pi}}P(\boldsymbol{\pi}|\boldsymbol{x};\mathbf{\Theta}^{(t)})\frac{\partial \log P(\boldsymbol{x},\boldsymbol{\pi};\mathbf{\Theta})}{\partial \omega}. \tag{4.16}$$

The only difference is that in the EM algorithm the path distribution given at iteration t is assumed to be constant during the maximization of the auxiliary function. If the $\mathcal{Q}$-function cannot be maximized exactly one can resort to iterative gradient-based techniques as discussed in chapter 2 and the approach is then a GEM algorithm. We see that maximizing $\mathcal{Q}$ by a gradient method is very similar to maximizing the log likelihood by a gradient method. Therefore, the two approaches can be expected to have similar convergence rates.

Using (2.36) the gradient of $\mathcal{L}_f(\mathbf{\Theta})$ is now easily found,

$$\frac{\partial \mathcal{L}_f(\mathbf{\Theta})}{\partial \omega} = -\sum_{li}\frac{n_i(l)}{\phi_i(x_l)}\frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{lij}\frac{n_{ij}(l)}{\theta_{ij}}\frac{\partial \theta_{ij}}{\partial \omega}. \tag{4.17}$$

For a model without parameter tying the gradient w.r.t. the transition probability θ_{ij} is

$$\frac{\partial \mathcal{L}_f(\mathbf{\Theta})}{\partial \theta_{ij}} = -\frac{\sum_l n_{ij}(l)}{\theta_{ij}} = -\frac{\bar{n}_{ij}}{\theta_{ij}} \tag{4.18}$$

and the gradient w.r.t. the probability of matching symbol a in state i is

$$\frac{\partial \mathcal{L}_f(\mathbf{\Theta})}{\partial \phi_i(a)} = -\frac{\sum_l n_i(l)\delta_{x_l,a}}{\phi_i(a)} = -\frac{\bar{n}_i(a)}{\phi_i(a)}. \tag{4.19}$$

Note that if $\frac{\partial \mathcal{L}_f(\mathbf{\Theta})}{\partial \omega} = 0$ then solving for ω under the positivity and sum-to-one constraints yields the Baum-Welch reestimation formulas. This shows that the reestimation formulas are exactly correct at critical points of $\mathcal{L}_f(\mathbf{\Theta})$.

The gradient of the negative log likelihood $\mathcal{L}_c(\mathbf{\Theta})$ in the clamped phase is computed similarly, but the expectation in (4.15) is taken only over allowed paths $\boldsymbol{\pi} \in \Pi_y$. This leads to

$$\frac{\partial \mathcal{L}_c(\mathbf{\Theta})}{\partial \omega} = -\sum_{li} \frac{m_i(l)}{\phi_i(x_l)} \frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{lij} \frac{m_{ij}(l)}{\theta_{ij}} \frac{\partial \theta_{ij}}{\partial \omega}, \tag{4.20}$$

where the m's are the expected counts given the labeling. If multiple labels are allowed in each state then (4.20) will contain an additional term for the parameters of the label distribution, see appendix B. Since $\mathcal{L}_f(\mathbf{\Theta})$ does not depend on the parameters of the label distribution, these parameters can be updated using the reestimation equations derived in appendix B or alternatively by a gradient method as for the other parameters.

For a model without parameter tying we can now express the derivative of $\mathcal{L}(\mathbf{\Theta})$ w.r.t. the transition probability θ_{ij} as

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \theta_{ij}} = -\frac{\sum_l m_{ij}(l) - n_{ij}(l)}{\theta_{ij}} = -\frac{\bar{m}_{ij} - \bar{n}_{ij}}{\theta_{ij}} \tag{4.21}$$

and w.r.t. the probability of matching symbol $a \in \mathcal{A}$ in state i as

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \phi_i(a)} = -\frac{\sum_l [m_i(l) - n_i(l)]\delta_{a,x_l}}{\phi_i(a)} = -\frac{\bar{m}_i(a) - \bar{n}_i(a)}{\phi_i(a)}. \tag{4.22}$$

From (4.21) and (4.22) we can easily see how the discrimination between classes enters training. Compared to the gradients in the free running phase the m-terms can be viewed as observed "values" or supervision information. In neural network terminology a positive difference between the m's and the n's represents positive training examples and a negative difference represents negative examples. The gradient expressions also illustrate why it is not possible to use Baum-Welch like reestimation equations directly. Thus, replacing *e.g.*, $n_i(l)$ in the Baum-Welch reestimation equation for $\phi_i(a)$ with $m_i(l) - n_i(l)$ can very well lead to a negative reestimate.

4.4.2 Normalization Constraints

Updating the parameters according to (4.11) will not ensure that the parameters normalize correctly and there is a risk of obtaining negative estimates. To ensure positive and normalizing parameters one can do an explicit normalization after each update,

$$\omega^{(t+1)} = \text{Normalize}\left(\omega^{(t)} + \Delta\omega^{(t)}\right). \tag{4.23}$$

The "Normalize" operator ensures that the required parameters sum to one. It is easy to prevent negative parameter estimates by *e.g.*, setting such estimates to zero before normalizing.

A more elegant approach is to use a parameter transformation that ensures positive and normalizing parameters. Here we can use the same method as [BC94] and do gradient-based optimization in another set of unconstrained variables. For the transition probabilities we define

$$\theta_{ij} = \frac{e^{z_{ij}}}{\sum_{j'} e^{z_{ij'}}}, \tag{4.24}$$

where $z_{ij} \in \mathcal{R}$ are the new unconstrained auxiliary variables. This transformation is known as the *softmax* function in the neural network literature. By construction, θ_{ij} will always be positive and properly normalized. Now gradient-based optimization in the auxiliary variables yields a new estimate of z_{ij},

$$z_{ij}^{(t+1)} = z_{ij}^{(t)} + \Delta z_{ij}^{(t)}, \tag{4.25}$$

where the parameter change $\Delta z_{ij}^{(t)}$ is computed using the gradients of $\mathcal{L}(\mathbf{\Theta})$ w.r.t. z_{ij}. The new estimate for the auxiliary variables corresponds to an estimate for θ_{ij} given by

$$\begin{aligned} \theta_{ij}^{(t+1)} &= \frac{\exp\left(z_{ij}^{(t)} + \Delta z_{ij}^{(t)}\right)}{\sum_{j'} \exp\left(z_{ij'}^{(t)} + \Delta z_{ij'}^{(t)}\right)} \\ &= \frac{\theta_{ij}^{(t)} \exp\left(\Delta z_{ij}^{(t)}\right)}{\sum_{j'} \theta_{ij'}^{(t)} \exp\left(\Delta z_{ij'}^{(t)}\right)}. \end{aligned} \tag{4.26}$$

The gradients of $\mathcal{L}(\mathbf{\Theta})$ with respect to z_{ij} can be expressed entirely in terms of θ_{ij} and $m_{ij} - n_{ij}$,

$$\frac{\partial \mathcal{L}}{\partial z_{ij}} = - \left[m_{ij} - n_{ij} - \theta_{ij} \sum_{j'} (m_{ij'} - n_{ij'}) \right]. \tag{4.27}$$

For purely gradient-based optimization the change in the auxiliary variables $\Delta z_{ij}^{(t)}$ can therefore be expressed entirely in terms of the expected counts and θ_{ij}. Thus, the update formula (4.26) for the transition probabilities does *not* depend explicitly on the auxiliary variables, but can be expressed entirely by the expected counts and the model parameters at the previous iteration. Equations for discrete match probabilities can be obtained in exactly the same way. This approach is slightly more straightforward than the one proposed in [Bri90, BC94], where the auxiliary variables were retained and the parameters of the model calculated explicitly from (4.24) after updating the auxiliary variables. This type of gradient optimization is also very similar to the exponentiated gradient descent proposed and investigated in [KW97, HSSW97].

In a similar way to which the Baum-Welch reestimation formula was "derived" for the standard HMM, we can set equation (4.27) equal to zero and solve for θ_{ij} (or one of the other parameters of the model). Unfortunately, the solution makes no sense as it will often be negative. However, the derivative does not change if an arbitrary positive number D is added and subtracted, so we might write

$$\frac{\partial \mathcal{L}}{\partial z_{ij}} = - \left(m_{ij} - n_{ij} + D\theta_{ij}\right) + \theta_{ij} \left(D + \sum_{j'} (m_{ij'} - n_{ij'}) \right). \tag{4.28}$$

Now "solving" for θ_{ij} yields

$$\theta_{ij}^{(t+1)} = \frac{\theta_{ij}^{(t)} \left(\frac{m_{ij}^{(t)} - n_{ij}^{(t)}}{\theta_{ij}^{(t)}} + D \right)}{\sum_{j'} \theta_{ij'}^{(t)} \left(\frac{m_{ij'}^{(t)} - n_{ij'}^{(t)}}{\theta_{ij'}^{(t)}} + D \right)}, \tag{4.29}$$

or more generally for a generic parameter $\omega_i \in \mathbf{\Theta}$,

$$\omega_i^{(t+1)} = \frac{\omega_i^{(t)} \left(-\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \omega_i^{(t)}} + D \right)}{\sum_{j'} \omega_{j'}^{(t)} \left(-\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \omega_{j'}^{(t)}} + D \right)}, \tag{4.30}$$

where the sum in the denominator is extended over the set of parameters that include ω_i and must normalize to one. This is essentially the updating formula known as *extended Baum-Welch reestimation* suggested in [GKNN91] for discrete HMMs and later for continuous and semi-continuous HMMs [NM91, NCM94]. Similar equations can be derived for the match probabilities. Extended Baum-Welch reestimation has been shown to compare favorably to gradient descent MMI estimation in several applications [KVY93, Mer88, NM91, NCM94, GKNN91]. In [GKNN91] it was shown that (4.30) gives new estimates of the parameters which guarantee convergence of the MMI (and CML) criterion to a local maximum provided that the constant D is sufficiently large. For practical purposes reasonably fast convergence is obtained if D is chosen such that the numerator of (4.30) is always larger than or equal to some small positive constant ϵ [NCM94],

$$D = \max_{\omega_i \in \mathbf{\Theta}} \left[\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \omega_i}, 0 \right] + \epsilon. \tag{4.31}$$

4.5 Global Normalization

In many applications there is a long and successful tradition of breaking the sacred principle that the probability parameters of the individual states must sum to one. These ad hoc methods include *e.g.*, language model biases, multiple independent streams and stream exponents [Rob94, NCM94, KVY93]. The rationale in using language model biases is to reduce what seems to be a 'mismatch' between acoustic model probabilities and language model probabilities. In [Rob94] better performance was obtained in a phoneme recognition task when scaling bigram language model probabilities with a certain "global" factor. Similarly, Kapadia *et al.* [KVY93] reported improved performance by squaring the bigram language model probabilities for phoneme recognition. In [HBAH93] the language model bias approach was succesfully extended to the so-called *Unified Stochastic Engine*, in which separate weights for different language model probabilities were used. The language weights and the parameters of the acoustic models were all jointly estimated by gradient-based minimization of a "global" discriminative training criterion similar to the minimum classification error criterion. The effect of language model biases can also be obtained by using stream exponents where the log match probabilities are scaled by small positive constants. The stream exponent approach can be further refined by using separate weights for each stream in multiple independent stream models [NCM94] (see section 2.8).

To retain a probabilistic model, even when the parameters for a state do not sum to one, one can normalize globally. For the CML estimated CHMM global normalization is particularly convenient because the normalizing constant need not be calculated. Furthermore, when introducing observation context dependence in the hidden neural network hybrid, described in chapter 6, it is impossible to properly normalize the probabilities for each state.

Similar to the probability (2.9) for a standard HMM we define

$$R(\boldsymbol{x};\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} R(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_l \phi_{\pi_l}(x_l)\theta_{\pi_{l-1}\pi_l}. \tag{4.32}$$

In general, *this is not a probability* because we will lift the normalization requirement for the parameters in the match and transition distributions, which is the reason for changing the notation from P to R. When the parameters do not normalize we will use the term *score* instead of probability. The R-function can be turned into a probability by explicit normalization,[2]

$$P(\boldsymbol{x};\boldsymbol{\Theta}) = \frac{R(\boldsymbol{x};\boldsymbol{\Theta})}{\sum_{\boldsymbol{x}'\in\mathcal{X}} R(\boldsymbol{x}';\boldsymbol{\Theta})}, \tag{4.33}$$

where $\mathcal{X}$ is the space of all possible discrete observation sequences of any length. With this normalization the model can still be given a probabilistic interpretation even though the parameters no longer normalize to one. Note that due to the normalization term, training of the globally normalized HMM cannot be done by the EM algorithm. However, gradient-based methods can be used instead.

With discrete match "distributions" (and many continuous "distributions") it is in theory possible to compute the normalization factor to arbitrary precision provided that $R(\boldsymbol{x};\boldsymbol{\Theta})$ approaches zero fast enough as a function of sequence length, or that a maximum sequence length can be enforced which is usually the case. To see this we express the normalization factor as a sum over all sequence lengths,

$$\sum_{\boldsymbol{x}'\in\mathcal{X}} R(\boldsymbol{x}';\boldsymbol{\Theta}) = \sum_{L=1}^{L_{max}} \sum_{\boldsymbol{x}'\in\mathcal{X}^L} R(\boldsymbol{x}';\boldsymbol{\Theta}), \tag{4.34}$$

where L_{max} is the assumed maximum sequence length and $\mathcal{X}^L$ is the space of observation sequences of length L. For a given length L the sum over sequences in $\mathcal{X}^L$ can be computed by the standard forward algorithm where multiplications by $\phi_i(x_l)$ are replaced with $\sum_{a\in\mathcal{A}} \phi_i(a)$ (or $\int_{\boldsymbol{x}} \phi_i(\boldsymbol{x})d\boldsymbol{x}$ for continuous observations).

For the CML estimated CHMM it is never necessary to actually calculate the normalization factor. Similar to the likelihood for the clamped phase $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ we define for the single-label-state CHMM

$$R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}\in\Pi_y} R(\boldsymbol{x},\boldsymbol{\pi};\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}\in\Pi_y} \prod_l \phi_{\pi_l}(x_l)\theta_{\pi_{l-1}\pi_l}. \tag{4.35}$$

A similar expression exists for CHMMs with multiple-label-states. Provided the label distributions normalize to one (this is trivially ensured for single-label-states) we have

$$\sum_{\boldsymbol{y}\in\mathcal{Y}} R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}) = R(\boldsymbol{x};\boldsymbol{\Theta}). \tag{4.36}$$

Therefore,

$$P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}) = \frac{R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{\sum_{\boldsymbol{x}'\in\mathcal{X}} R(\boldsymbol{x}';\boldsymbol{\Theta})}, \tag{4.37}$$

which leads to the following expression for the a posteriori probability of the labeling

[2] For continuous HMMs the summation is replaced by an integration over the space of continuous observation sequences.

$$P(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\Theta}) = \frac{R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})}{R(\boldsymbol{x};\boldsymbol{\Theta})}, \tag{4.38}$$

where the normalization constant has cancelled out. Thus, when training the CHMM by CML the global normalization is automatically ensured. This is true even if we do not use an auxiliary end state or if we allow states with outgoing transitions to be end states. In many applications it is nevertheless convenient to restrict only one or a few states to be end states. For example, in speech recognition the last few labels for most utterances usually correspond to the silence class. This constraint can easily be ensured by only allowing states modeling the silence class to be end states.

The calculation of $R(\boldsymbol{x};\boldsymbol{\Theta})$ and $R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ can be done exactly as described for $P(\boldsymbol{x};\boldsymbol{\Theta})$ and $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$, because the forward and backward algorithms are not dependent on the normalization of parameters. It is important to note, though, that the forward and backward variables in the case of non-normalizing parameters cannot be given a probabilistic interpretation. They should rather be considered as scores. Even though the forward-backward variables are not probabilities, the expected m and n counts are still probabilistic quantities because they are expressed as rational functions of the forward and backward variables, see (2.46)-(2.47). Similarly, the standard MAP decoder can be based on $R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ instead of $P(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ because the factor $R(\boldsymbol{x};\boldsymbol{\Theta})$, which turns $R(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta})$ into the posterior probability of the labeling, is the same for all possible labelings.

Non-normalizing parameters can lead to a much richer model than when the parameters are required to normalize. In principle the parameters need even not be positive. However, to ensure that the normalization terms in (4.33) and (4.37) are different from zero we propose to keep the parameters strictly positive by using *e.g.*, an exponential parameter transformation; for a generic parameter $\omega_i \in \boldsymbol{\Theta}$ we introduce an auxiliary variable $z_i \in \mathcal{R}$ and define

$$\omega_i = \exp(z_i). \tag{4.39}$$

This positivity constraint also implies that the standard forward-backward and decoding algorithms can be used directly without modification (*i.e.*, the "tricks" for handling numerical problems in the forward-backward and Viterbi algorithms only work for positive parameters). For the exponential transformation the update equations for the parameters become particularly simple,

$$\omega_i^{(t+1)} = \omega_i^{(t)} \exp\left(\Delta z_i^{(t)}\right), \tag{4.40}$$

and since

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_i} = \omega_i \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \omega_i} \tag{4.41}$$

the update equations (4.40) for the parameters will not depend explicitly on the auxiliary variables.

A potential problem with the exponential parameter transformation is that a large change in the auxiliary variable implies that the current parameter estimate $\omega_i^{(t)}$ is scaled by a huge positive value. Since a large $\Delta z_i^{(t)}$ corresponds to a large gradient of $\mathcal{L}(\boldsymbol{\Theta})$ w.r.t. ω_i this is very likely to happen. It can be cured by using a *sigmoid* parameter transformation instead,

$$\omega_i = \frac{1}{1 + \exp(-z_i)} \tag{4.42}$$

which yields non-normalizing parameters in the range $]0;1[$. Also for this transformation the new estimate $\omega_i^{(t+1)}$ can be expressed entirely in terms of $\omega_i^{(t)}$ and the expected counts.

4.6 Summary

In this chapter we have introduced the Class HMM as a particular extension of standard HMMs, where each state in addition to the usual match distribution also has been assigned a distribution over possible labels. This allows for modeling of the joint probability of observation and label sequences and equations for Baum-Welch-like ML training and gradient-based CML training were derived for both complete and incomplete labeling. Furthermore, it was shown how CML estimation of the CHMM automatically ensures that the model is normalized globally such that the parameters of the individual states need not normalize to one.

Each state in the CHMM architecture can in principle model all possible labels. This can be attractive to some applications, but for speech recognition it is believed that modeling all labels in all states is not beneficial. One of the main conclusions of many years of research in speech recognition is that the HMM architecture, although possibly still the best known model, is not a very good representation of speech. Good performance has mainly been obtained by submodels with a highly constrained topology, for instance, left-to-right phoneme submodels. By allowing all phoneme labels in all states, the states in the CHMM can no longer be given a physical interpretation, *i.e.*, one can no longer tell which states model which phonemes. Thus, allowing all labels implies that the prior knowledge obtained through years of research into speech recognition is discarded. On the other hand, we still believe that it can be beneficial to allow some states to model a subset of the possible labels in some situations, for example in order to make the model sensitive to phonetic context.

It should also to be noted that the "optimal" HMM topologies have been developed in the context of non-discriminative ML estimation. As such, there is no guarantee that these topologies will also be "optimal" for discriminative training. In principle, it is therefore necessary to reevaluate a large number of different topologies in the framework of discriminative training. This is naturally beyond the scope of this thesis and we will here make extensively use of left-to-right phoneme submodel topologies, keeping in mind that better performance might be possible with a completely different, but yet unknown, topology. The following chapter gives an evaluation of the ML and CML estimated CHMM on a simple continuous speech recognition task.

Chapter 5

Evaluation of Class HMMs

This chapter gives an evaluation of the discrete class HMM for the highly simplified task of recognizing five broad phoneme classes in continuous speech from the TIMIT database. Even though this task is speaker independent it has a number of limitations. Firstly, focus is put only on acoustic modeling, *i.e.*, it is only attempted to recognize broad class phoneme sequences and not words or sentences. Secondly, TIMIT is a database of *read* speech recorded in a *quiet* environment and it therefore does not encompass effects of noise and spontaneous speech disfluencies like hesitations, restarts and incomplete sentences. Such effects are important in practice but generally considered difficult to handle in a speech recognition system. By selecting a simple task, however, it is possible to evaluate a number of the ideas presented in the previous chapter in a reasonable time.

The chapter starts by defining the broad phoneme recognition task and by describing the datasets, preprocessor and chosen model topology. In section 5.2 a comparison between complete and incomplete label Maximum Likelihood (ML) training is given. Such a direct comparison is possible because the utterances in TIMIT are segmented manually into a set of phone labels. Complete and incomplete label Conditional Maximum Likelihood (CML) training is the topic of section 5.3. Since CML estimation cannot be done by the efficient Baum-Welch algorithm this section starts by evaluating a number of gradient-based training algorithms and compare these to the extended Baum-Welch algorithm. In section 5.4 and section 5.5 the issue of non-normalizing parameters is discussed briefly. Section 5.6 concludes the chapter by a comparison to results reported in the literature.

5.1 Task Definition

The TIMIT broad phoneme class task was originally intended as a demonstration of the Hidden Markov Model Toolkit (HTK) developed at Cambridge University [You92b], but it has recently been adopted for evaluation of discriminative MMI and CML estimation of continuous density HMMs [Joh96, JJ94a, Joh94]. The five broad phoneme classes are, vowels (V), consonants (C), nasals (N), liquids (L) and silence (S). They are defined in table 5.1 using the original 61 TIMIT phone labels shown in appendix A. Note that the glottal stop /q/ is deleted as in [Joh96, LH89]. This leaves undefined segments in the phonetic transcriptions (incomplete labelings). The broad classes constitute a major simplification compared to the full TIMIT phone inventory, but the classes are still potentially confusable and cover all phonetic variations in American English. The five broad class task is therefore a good test bed for the preliminary evaluation of speaker independent continuous speech recognizers.

Broad class	Label	TIMIT phone labels
Vowel	V	iy ih eh ae ix ax ah ax-h uw uh ao aa ey ay oy aw ow ux
Consonant	C	ch jh dh b d dx g p t k z zh v f th s sh hh hv
Nasal	N	m n en ng em nx eng
Liquid	L	l el r y w er axr
Silence	S	h# pau
	-	q

Table 5.1: Definition of five broad phoneme classes in terms of the 61 TIMIT phone labels.

5.1.1 Datasets

Only a subset of the TIMIT database was used in the broad class experiments, see table 5.2. The training set contained 462 sentences uttered by 462 different speakers. This corresponds to about 1/8th of the total recommended TIMIT training set, see appendix A. The sentences in the training set were all different and phonetically diverse ("SI" sentences). For evaluation of the models the recommended TIMIT core test set was used. The core test set contains a total of 192 sentences uttered by 24 different speakers. All sentences in the core test set are different and phonetically compact ("SX" sentences), *i.e.*, specifically designed to cover all possible biphones that can be generated from the TIMIT phones. For monitoring performance and model selection a separate validation set of 144 sentences uttered by 144 different speakers was used. As for the training set these sentences were textually different and phonetically diverse. Note that there are *no* overlapping sentences or speakers between any of the three datasets. The datasets are identical to those used in [Joh96].

5.1.2 Preprocessing and Scoring

The raw speech signal was preprocessed using a *mel-frequency cepstral* preprocessor, see *e.g.*, [DPH93]. Cepstral features can be computed indirectly by applying a recursion to the *linear prediction* coefficients or in a more direct fashion from the short term absolute spectrum of the speech signal. The basic idea in the latter approach, which was used in this work, is to take the logarithm of the absolute short term spectrum and then transform this new "signal" back to the time domain. By taking the logarithm in the frequency domain it is possible to transform the non-linear time-domain convolution of two signals into a sum. Hereby, the slowly varying vocal tract impulse response and the quickly varying excitation sequence for a speech signal can be separated.[1] Ideally, the low-order cepstral coefficients identify the vocal tract envelope spectrum and the higher order coefficients identify the excitation sequence. Typically, only the low-order cepstral coefficients are used in speaker independent HMM-based speech recognition. These coefficients tend to be less speaker dependent than the higher order ones [DPH93] and express the "configuration" of the vocal tract. Furthermore, HMMs model speech as a sequence of steady-state segments. This is

[1] In principle this only applies to voiced speech which is ideally generated by a periodic excitation of the vocal tract system.

Datasets	
General characteristics	
Speech type	Continuous read American English
Environment	Quiet studio, high quality microphone
Phoneme segmentation	Assigned by human expert phoneticians
Dialects	8 major dialect regions in USA
Speakers	630
Sentences	2342
Utterances	6300 (10 per speaker)
Training set	
Speakers	462
Sentences	462 (phonetically diverse)
10ms frames	158159 ($\approx$ 26 minutes)
Broad class labels	20430
Testing set	
Speakers	24
Sentences	192 (phonetically compact)
10ms frames	57438 ($\approx$ 10 minutes)
Broad class labels	7215
Validation set	
Speakers	144
Sentences	144 (phonetically diverse)
10ms frames	49142 ($\approx$ 8 minutes)
Broad class labels	6145

Table 5.2: Summary of TIMIT datasets used for broad class experiments.

consistent with modeling speech as a sequence of steady-state vocal tract configurations. Thus, the low-order cepstral coefficients are a reasonable choice for HMM-based speech recognition. Improved recognition performance has often been observed by using psycho-acoustic knowledge in the cepstral analysis. Typically, the log of the spectrum is passed through a set of perceptual filters that integrate the signal energy within a number of *critical bands*. The bandwidth of the critical bands increase logarithmically above 1kHz to simulate the sensitivity of the human ear to different frequencies. Another typically used method is to warp the frequency scale onto *e.g.*, the mel-scale. The mel-scale expresses the *perceived frequency* as a function of the actual frequency as obtained through a set of listening experiments. It is approximately linear below 1kHz and logarithmic above 1kHz. In this work, triangular shaped critical band filters and mel-scale warping was used.

The actual preprocessor implementation was based on the public domain C-function library accompanying the `OGI Speech Tools` package provided by Center for Spoken Language Understanding at the Oregon Graduate Institute.[2] For each 10 ms frame of speech the preprocessor yielded 12 mel-scaled cepstral coefficients and the signal log energy computed over a 20ms Hamming weighted and preemphasized window. This vector of 13 features was augmented by the corresponding first-order differential features (Δ-features)

[2]The OGI Speech Tools has recently been renamed to the CSLU Toolkit and is available at http://www.cse.ogi.edu/CSLU/.

computed by linear regression on a window of five frames centered on the current speech frame. Thus, the preprocessor yielded a 26 dimensional feature vector for every 10ms frame of speech. All feature vectors in a given utterance were normalized such that each element in the vector had zero mean and unit variance over that utterance.

As discrete CHMMs were used in all experiments reported in this chapter, the 26 dimensional feature vectors were quantized prior to training. A codebook of 256 prototype vectors was generated from the training set feature vectors by the Linde-Buzo-Gray algorithm, see *e.g.*, [DPH93]. The details of the preprocessor and vector quantizer are summarized in table 5.3.

For complete label training the manually assigned time-boundaries between phones in the TIMIT database were used for assigning broad class labels to each 10ms frame of speech. At positions where the glottal stop /q/ had been deleted the label of the previous class was simply used. Similarly, frames overlapping a phone time-boundary were assigned the broad class label corresponding to the phone that covered the largest part of the frame. This implies that some broad class boundaries were "artificially" shifted by up to ±5ms. However, the TIMIT phone boundaries are assigned by human phoneticians and are therefore subject to variations between these experts. Furthermore, one can argue that exact boundaries between phonemes do not exist as the human articulators cannot change "state" instantaneously. The artificially introduced shifts are therefore believed to have little consequence on the final recognition accuracy.

Mel-frequency preprocessor	
Amplitude compression	None
Sampling frequency	16 kHz
Preemphasis coefficient	0.97
Window length	20 ms
Weighting function	Hamming
Frame length	10 ms
Spectral analysis	FFT
Frequency warping	Mel-scale
Mel filterbanks	24
Mel filterbank shape	Triangular
Cepstral liftering window size	22
Cepstral liftering type	Raised sine
Cepstral features	26 (12 + log energy + Δ's)
Feature normalization	Zero mean, unit variance
Delta regression window	5 frames
Vector quantizer	
Codebook size	256
Codebook generation	Linde-Buzo-Gray

Table 5.3: Summary of mel-frequency cepstral preprocessor and vector quantizer for the broad phoneme class task.

Recognition results will be reported in terms of the string accuracy ($\%Acc$) defined by

$$\%Acc = 100\% - \frac{Ins + Del + Sub}{N_s} \cdot 100\% \tag{5.1}$$

where *Ins*, *Del* and *Sub* are the number of textual *insertions*, *deletions* and *substitutions* respectively obtained by *aligning* the recognized broad class label strings (*i.e.* the recognized incomplete labelings) against the reference label strings. N_s denotes the total number of incomplete labels in the reference symbol strings. In this work we used the public domain standard scoring program `sclite` (ver.1.4) provided by the American National Institute of Standards (NIST). [3] Contrary to the UNIX utility `diff` for assessing differences between text files, the NIST scoring program uses different penalty weights for insertions, deletions and substitutions, see *e.g.*, [PGMDF86] and the documentation accompanying the `sclite` distribution. This is equivalent to the sequence alignment techniques often used in biological sequence analysis.

5.1.3 General Model Setup

Although it is not necessary to use separate class submodels in the CHMM it was decided to use the simple topology shown in figure 5.1 for each of the five broad phoneme classes. The main reason for this is that initial experiments did not indicate any improvement by using *e.g.*, fully connected models where all states in the CHMM are connected regardless of class labels. This is agrees with several years of research in speech recognition indicating that left-to-right topologies yield performance superior to other topologies.

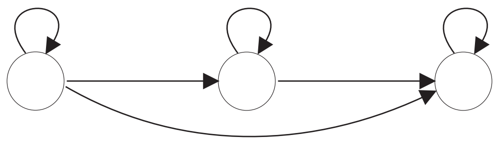

Figure 5.1: Submodel used for each broad class.

The submodel for each broad class contains three left-to-right connected states and a "skip" transition between the first and last state. The skip transition implies that the shortest broad class segment which can be modeled is two frames (or 20ms) long. The match distributions associated with each of the states are discrete distributions over the 256 codebook symbols. Since we only consider single-label-state CHMMs here, the label "distributions" in each state are simply delta-functions indicating the class label of the submodel.

Figure 5.2 illustrates the overall CHMM model which is similar to so-called *looped* models commonly used for phoneme recognition experiments. Note, however, that the "language model" transitions between the class submodels are considered part of all the other parameters of the model. This is contrary to common approaches in continuous phoneme recognition in which the bigram probabilities are estimated beforehand by the frequency of phoneme pairs in the training set.[4] Because all sentences in the three datasets start and end in the silence class we only connect the auxiliary non-matching begin (end) state to the first (last) state of the submodel for the silence class. The total number of parameters in the discrete CHMM is 3897. Note that some of these parameters should not be considered as trainable when sum-to-one constraints are enforced.

[3] The scoring program is available by anonymous ftp from ftp://svr-ftp.eng.cam.ac.uk.

[4] ML training of the CHMM yields bigram transitions identical to those found by simple counting of phoneme pairs in the training set. This is, however, not the case for CML estimation where the bigram is trained discriminatively.

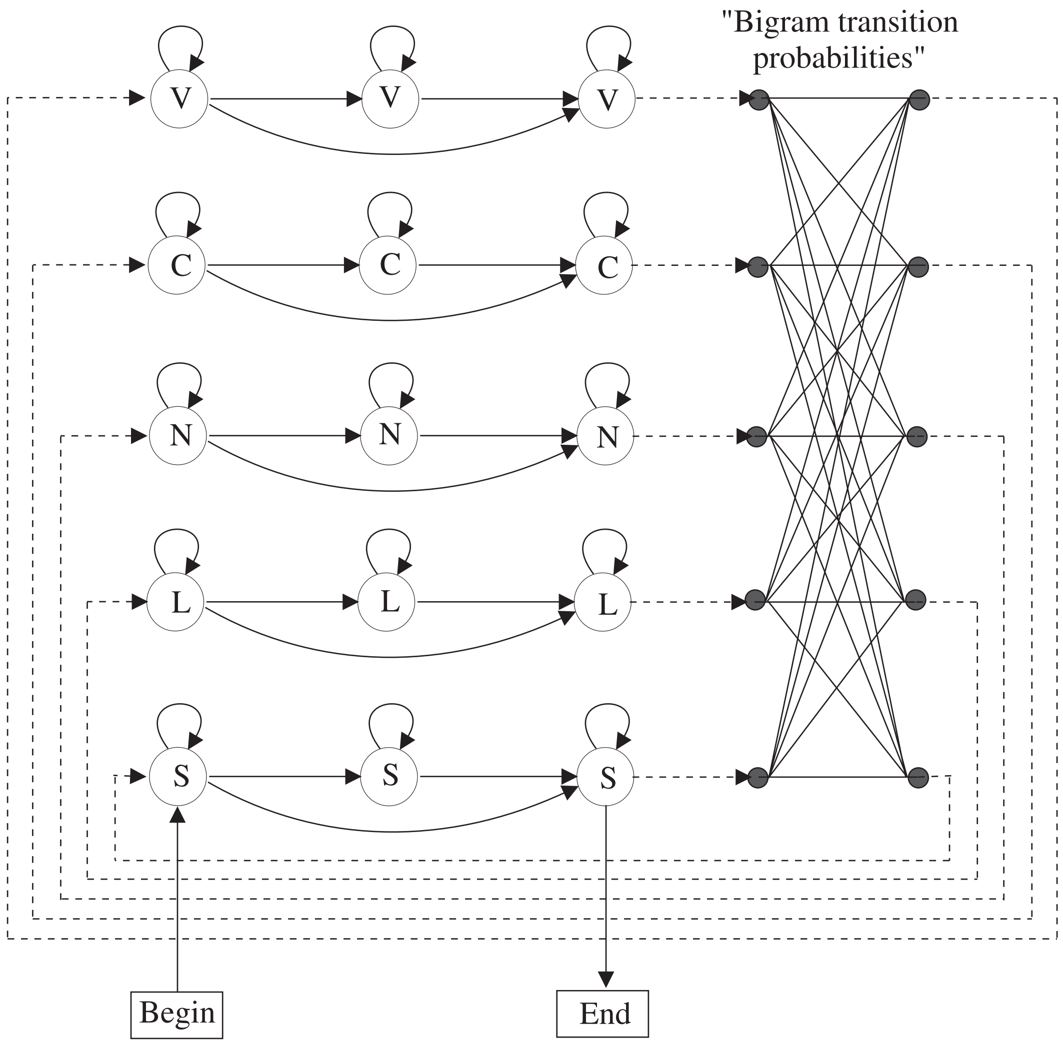

Figure 5.2: Overall CHMM topology for the broad class task. Note that the filled circles are *not* CHMM states, but used only to clarify the illustration, *i.e.*, the last state of *e.g.*, the consonant submodel is fully connected to the first state of all submodels.

Even for this simple model setup and fairly small datasets the computational complexity is fairly large. On a fast unix workstation,[5] training times varied from about 15 CPU minutes for complete label ML estimation to about three hours for incomplete label CML estimation.

5.2 Maximum Likelihood Training

As a first experiment the discrete locally normalized CHMM was trained to maximize the likelihood of the data within each class. Because the "true" phoneme segmentation is known, a direct comparison between complete and incomplete label training has been possible.

[5]A Silicon Graphics machine with a MIPS R10000 CPU.

5.2.1 Complete Label Training

In this experiment the model was trained from scratch using Baum-Welch reestimation and the complete labeling. Before training the model was initialized with uniform probabilities. Only two or three iterations of Baum-Welch reestimation were needed for convergence, that is, additional iterations provided only insignificant changes to the average training set likelihood. The recognition accuracies obtained by the forward-backward, Viterbi and N-best decoders are shown in table 5.4. Whereas no pruning was used for the forward-backward and Viterbi decoders, it was necessary to use both local and global pruning for the N-best decoder to reduce computational requirements. Thus, for all experiments with the N-best decoder we used a local pruning threshold of $\gamma_l = 10000$ and a maximum number of $M = 10$ partial hypotheses allowed to survive in each state. With $\gamma_l = 10000$, partial hypotheses that have a probability 10000 times smaller than the best hypothesis in each state are discarded. Only the top-scoring hypothesis in the end state was used for calculating recognition accuracies.

Train			Test		
F-B	N-best	Vit	F-B	N-best	Vit
75.7	***76.5***	76.2	75.6	***76.1***	75.9

Table 5.4: Recognition accuracies (%Acc) for a complete label ML trained CHMM.

From the table we observe that N-best decoding gives the best accuracies on both training and test sets, but also that the difference to Viterbi decoding is small. No overfitting is observed.

5.2.2 Incomplete Label Training

One can argue that complete label training is inconsistent with using the model to recognize the broad class incomplete labeling. Furthermore, the complete labeling as given in the TIMIT database is not always available. Therefore, in a second set of experiments we trained the model using incomplete label ML estimation. To illustrate the importance of a good initial model for incomplete label training three slightly different methods for initializing the model were tested. The first model was initialized by uniform probabilities and incomplete label ML training was thus done from “scratch”. The other two models were initialized by two iterations of complete label ML estimation using 1) 50 randomly selected sentences from the training set and 2) all 462 training sentences.

	Train			Test		
N_{cl}	F-B	N-best	Vit	F-B	N-best	Vit
0	74.3	75.2	75.4	73.0	73.5	73.6
50	75.4	76.0	76.1	74.4	75.0	75.1
462	75.5	76.1	***76.4***	75.2	75.6	***75.8***

Table 5.5: Recognition accuracies (%Acc) for an incomplete label ML trained CHMM. N_{cl} is the number of sequences used for initial complete label training before switching to incomplete label training on the full training set.

From table 5.5 we see that a good initial model significantly affects the performance of the incomplete label trained model. The reason for this is that a well initialized class model can "attract" the right portions of the data during incomplete label training. It is, however, interesting that the uniformly initialized model obtains a reasonably high recognition accuracy. The averaging effect resulting from repeated training on temporary models for each incomplete label sequence is thus sufficient for the submodels to learn some of the characteristics of the data distributions within each class. The best result was obtained by initial complete label training on the entire training set. However, the gain compared to using only 50 randomly selected training sequences is not very large. At least for the task considered here this indicates that we just need a few sentences with complete labeling to achieve good results.

A surprising observation is that the incomplete and complete label ML estimated models yield a comparable recognition accuracy. This is contrary to what was expected because maximizing the likelihood of the incomplete labeling is more consistent with using the model for generating broad class label strings during decoding. A possible explanation is that the initial complete label training gives a set of submodels which are capable of attracting the right portions of the speech signal in a maximum likelihood sense. As such, incomplete label ML training will not improve the recognition accuracy any further.

From table 5.5 it is seen that the Viterbi decoder gives slightly higher accuracies than the other two decoders. Consequently, the inconsistency between total likelihood or "all path" training and best path decoding does not seem to affect performance of the Viterbi decoder for ML trained models. This indicates that a single path through the model dominates the overall likelihood. For an example sentence this fact is clearly illustrated in figure 5.3, showing the state a posteriori probabilities computed by the forward-backward algorithm for an incomplete label ML estimated model. Note that most of the state posteriors are either very close to zero or very close to one. Furthermore, the most probable path (in terms of state posteriors) is "connected" from the start to the end of the sentence even though this is not enforced as in the Viterbi decoder. For the example sentence in the figure the ML estimated model yields an N-best accuracy of 62.5%.

All of the above results for the ML estimated models were obtained through only a single training session. One should therefore be careful when comparing the results for the various models because the same model might yield slightly different performance when trained from slightly different initial conditions. An indication of confidence intervals for the above results can be obtained by training the same model several times with slightly different initial conditions. For the five broad class task it is possible to make such an evaluation due to the fairly low computational complexity of the models considered here. In a first experiment 10 models with random probabilities were trained by incomplete label ML reestimation until convergence. For this experiment all models converged to practically the same likelihood level and the test set recognition accuracy for any of the 10 models was well within $\pm 0.1\%$ of the average accuracy for the 10 models. Thus, the Baum-Welch reestimation algorithm tends to end up in the same minimum for randomly initialized models. In a second experiment it was investigated whether the algorithm tends to get stuck in a poor local minimum. In the spirit of *simulated annealing* [KGJV83] one way to do this is by adding uniformly distributed random noise to the (accumulated) expected counts and then decrease the added noise amplitude gradually as training progresses. For this experiment we used an initial random noise amplitude of 10.0 and decreased this amplitude by a factor 0.85 for each epoch. Similar to the random probability initialization described above a series of ten training sessions resulted in practically the same final likelihood and the deviation in test set recognition accuracies was again within $\pm 0.1\%$ of the average accuracy for the ten runs. Based on these results we conclude that the Baum-

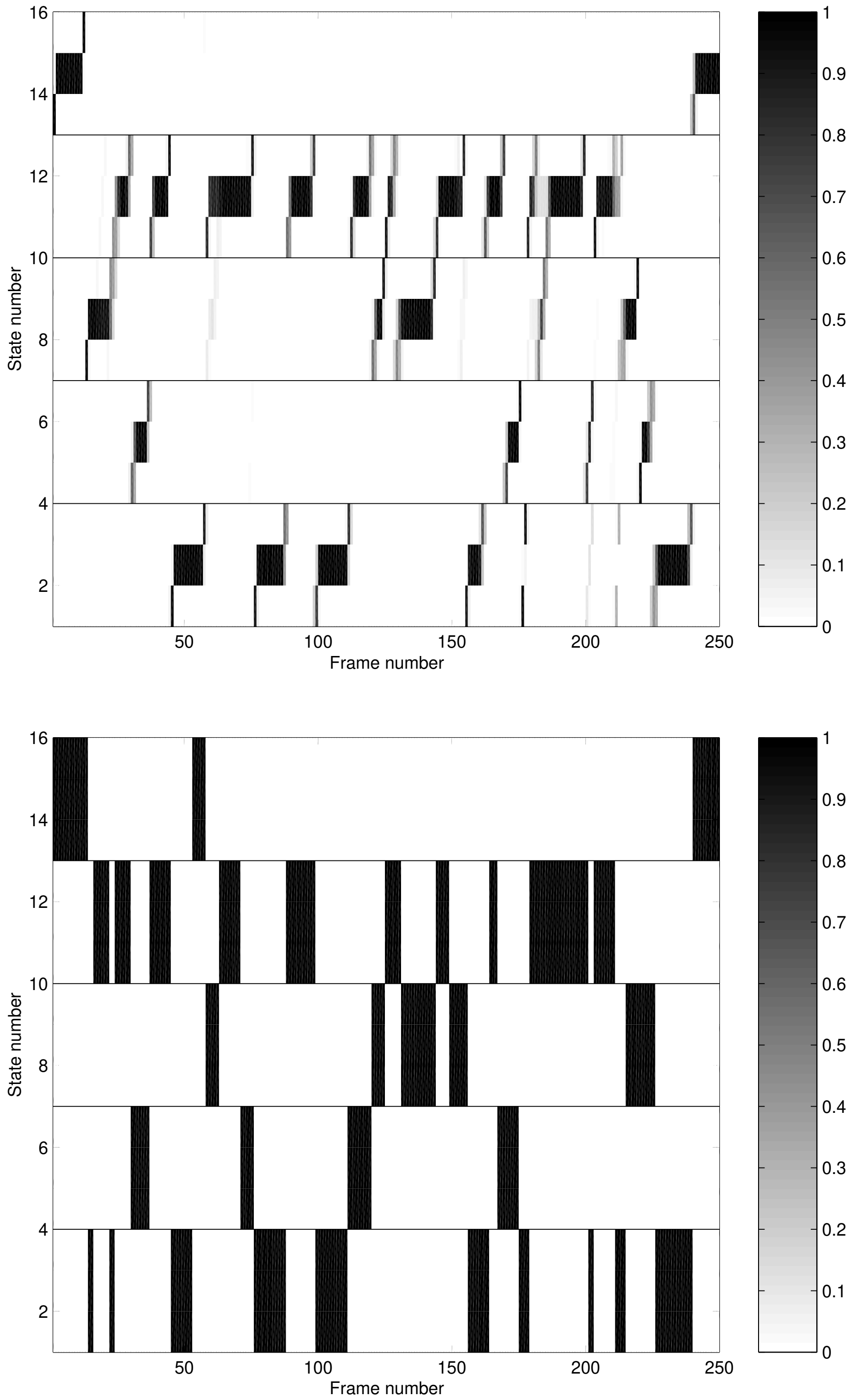

Figure 5.3: Graytone plot of state posterior probabilities $n_i(l) = P(\pi_l = i|\boldsymbol{x}, \boldsymbol{\Theta})$ for the utterance "But in this one section we welcomed auditors" (TIMIT id: si1361). Values of 1.0 are indicated by black and values of 0.0 by white. States 1–3 belong to the consonant model, 4–6 to the nasal model, 7–9 to the liquid model, 10–12 to the vowel model and 13–15 to the silence model. ***Upper panel:*** Incomplete label ML trained CHMM (N-best sentence accuracy:$\%Acc = 62.5$). ***Lower panel:*** Observed complete labeling.

Welch reestimation training for the models considered here is very robust and that the above presented results differ by no more than $\pm 0.1\%$ between different training sessions.

5.3 Conditional Maximum Likelihood Estimation

The ML criterion used above is not optimal for classification if the submodels are not capable of representing the true distribution of the data within each class. A better choice is to train the model to discriminate between the classes. Below we give an evaluation of the conditional maximum likelihood trained CHMM. As for the case of ML estimation it is possible to directly compare complete and incomplete label training. However, it is no longer possible to train the model by Baum-Welch like reestimation. Therefore, we start by briefly discussing different gradient-based approaches to training and compare these to the extended Baum-Welch reestimation algorithm. For a more detailed discussion of gradient-based optimization methods the reader is referred to [Bis95, HKP91] and the references therein.

5.3.1 Training Algorithms

For iterative minimization of the negative log conditional likelihood $\mathcal{L}(\mathbf{\Theta})$ given by (4.12), the parameter update can be written in the general form,

$$\mathbf{\Theta}^{(t+1)} = \mathbf{\Theta}^{(t)} + \eta \Delta \mathbf{\Theta}^{(t)}, \tag{5.2}$$

where $\mathbf{\Theta}^{(t)}$ is the parameter vector for the model at the t'th iteration and η is the *stepsize* or *learning rate*. The *search direction* $\Delta\mathbf{\Theta}^{(t)}$ must be chosen such that the the update leads "downhill" for the *cost function*, *i.e.*, $\mathcal{L}(\mathbf{\Theta}^{(t+1)}) < \mathcal{L}(\mathbf{\Theta}^{(t)})$ for a sufficiently small stepsize. A natural choice is the direction in which the negative log conditional likelihood decreases most rapidly, that is, the opposite direction of the gradient,

$$\Delta\mathbf{\Theta}^{(t)} = -\nabla\mathcal{L}(\mathbf{\Theta}^{(t)}) = -\left.\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \mathbf{\Theta}}\right|_{\mathbf{\Theta}=\mathbf{\Theta}^{(t)}}. \tag{5.3}$$

This method is known as *steepest descent* or simply gradient descent and belongs to the class of *first-order* methods as it is based only on first-order derivative information. The parameter update can be done using the gradient of $\mathcal{L}$ accumulated over either the entire training set or for a single training sequence. The former method is commonly denoted *batch* or *offline* gradient descent and the latter *sequence online* gradient descent. Note that the online update for the CHMM is not done for each *frame*, but rather for each *training utterance*. The reason for this is that the derivative of $\mathcal{L}$ is expressed in terms of the expected m and n counts which are not available before the forward-backward algorithm has been completed for the entire sentence. Thus, updating for each frame would require application of the forward-backward algorithm L times for a sequence of L observations and thereby increase the computational complexity by a factor of L.

For both online and batch training the size of the learning rate is very important for the convergence towards a local minimum of the cost function. If η is chosen sufficiently small a decrease of the cost can always be ensured, but a learning rate too small will result in very slow convergence. On the other hand, if η is too large the algorithm may diverge.

Provided we are sufficiently close to a local minimum an upper bound for the stepsize, which guarantees convergence to a local minimum of batch mode gradient descent, can be derived, see *e.g.*, [Bis95]. In practice, however, faster convergence is usually observed for stepsizes larger than this upper bound. Typically, the stepsize is found by trial-and-error. An alternative method is to invoke a so-called *line search* in each iteration. An exact line search finds the learning rate $\eta^{(t)}$ that minimizes the cost function in the search direction given at the current iteration,

$$\eta^{(t)} = \underset{\eta^{(t)}>0}{\operatorname{argmin}} \mathcal{L}(\mathbf{\Theta}^{(t)} + \eta^{(t)} \Delta\mathbf{\Theta}^{(t)}). \tag{5.4}$$

Usually it is not possible to perform an exact line search, but a simple approach that has been observed to work fairly well in practice (see *e.g.*, [Ped97]) is to start from some initial stepsize and then keep halving it until a decrease of the cost is observed. It should be noted that the line search leads to an extra computational burden compared to using a fixed learning rate because the cost function needs to be evaluated several times during the search for the best stepsize. Because of the high computational complexity associated with evaluating the likelihood over the entire training set in most large scale speech recognition systems, line search has not been widely applied in HMM modeling for speech recognition. One exception is the phoneme recognition experiment reported in [KVY93] where a line search was performed on a subset of the training data.

Line search as defined in (5.4) is inherently a batch method and will therefore not work well for online training. Contrary to batch training, online gradient descent does not guarantee convergence to a local minimum for some fixed small stepsize. However, convergence can be theoretically proven if η decreases inversely proportional to the iteration number *e.g.*, as [Joh96],

$$\eta^{(t)} = \frac{\eta^{(0)}}{\frac{t-1}{N_\eta} + 1}, \tag{5.5}$$

where $\eta^{(0)}$ is the initial stepsize and $N_\eta < \infty$ a factor that determines the rate of decrease. For $N_\eta = 1$ this learning rate schedule is simply $\eta^{(t)} = \eta^{(0)}/t$. For $N_\eta \to \infty$ the convergence property breaks down and the stepsize equals $\eta^{(0)}$ for all $t > 0$. Often faster convergence has been observed by using either a fixed stepsize or some heuristic schedule for selecting $\eta^{(t)}$. A multitude of such heuristic schedules have been proposed in the literature like *e.g.*, geometrically increasing/decreasing the stepsize by different factors if an iteration leads to a decrease/increase of the cost. Another popular method is to use separate learning rates for each parameter in the model. The reader is referred to [Bis95, Hay94, HKP91] for a discussion of various approaches.

An important advantage of the online approach over the batch methods arises if there is a high degree of redundant information in the data. As a simple example, suppose that a training set of K sequences is obtained by replicating the same sequence K times. For this training set the gradient calculation will take K times longer than for just one sequence. But the information provided by the accumulated gradient is the same as that provided by the gradient for each sequence and batch training will therefore converge considerably slower than online training if the same (sufficiently small) stepsize is used.

Another advantage of online gradient descent is that it is a stochastic algorithm because we do the update for each sequence independently. Hereby the algorithm has the ability to escape relatively flat local minima. A further refinement, which can aid further in escaping local minima, is to select the sequences in random order from the training set.

For both batch and online training, convergence will be very slow if we move along plateaus of the cost function because in these regions the gradient will be very small. A popular way of accelerating the search in such regions is to add a so-called momentum term to the parameter update,

$$\Delta\boldsymbol{\Theta}^{(t)} = -\eta\nabla\mathcal{L}(\boldsymbol{\Theta}^{(t)}) + \mu\Delta\boldsymbol{\Theta}^{(t-1)}, \tag{5.6}$$

where $\Delta\boldsymbol{\Theta}^{(t)} = \boldsymbol{\Theta}^{(t+1)} - \boldsymbol{\Theta}^{(t)}$ and $\mu \in [0;1[$ is the *momentum* parameter. The momentum term effectively adds inertia to the motion through parameter space. At places where the gradient is almost unchanging the effective stepsize can easily be shown to be $\eta_{\text{eff}} = \frac{\eta}{1-\mu}$.

In figure 5.4 a comparison of different gradient descent methods to the extended Baum-Welch algorithm is given for conditional maximum likelihood estimation of the CHMM for the simple broad class task. During gradient-based estimation, sum-to-one constraints for the parameters in the CHMM were enforced by the softmax parameter transformation (4.24). The batch mode gradient descent used either a fixed stepsize or the simple line search algorithm described above. Due to the relatively small size of the broad class task it was possible to do the line search over the entire training set in reasonable time. The fixed stepsize was $\eta = 0.01$ and the initial stepsize used for the line search was $\eta = 0.05$. The sequence online method also used a fixed stepsize of $\eta = 0.01$ and the training sequences were presented in random order. All three gradient-based approaches were accelerated by a momentum term with momentum parameter $\mu = 0.6$. In the extended Baum-Welch algorithm (4.30) the constant D was selected according to (4.31) with $\epsilon = 10^{-5}$ to ensure that the parameter update was positive and well defined. All models were initialized by two iterations of complete label ML reestimation and then trained for 28 epochs[6] of complete and 70 epochs of incomplete label CML estimation. Note that the figure illustrates the evolution of $\mathcal{L}(\boldsymbol{\Theta})$ calculated for the *incomplete* labeling observed for the training set.

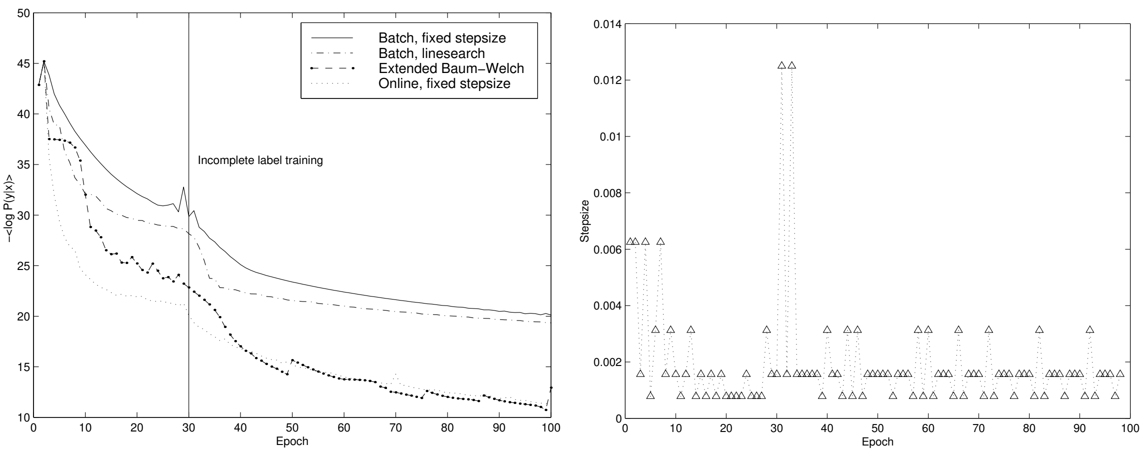

Figure 5.4: Comparison between extended Baum-Welch reestimation and batch and online gradient descent. ***Left panel:*** Evolution of $\mathcal{L}(\boldsymbol{\Theta})$ for the training set. ***Right panel:*** Stepsizes chosen by the simple linesearch algorithm during batch mode gradient descent.

From figure 5.4 we first observe that batch mode gradient descent decreases the cost very smoothly. The more rapid decrease in cost just after epoch 30 is due to the shift

[6] An epoch denotes a *full* pass through the training set.

in training strategy from complete to incomplete label estimation. We also note that the simple line search algorithm succeeds in finding a sequence of stepsizes that leads to a somewhat faster convergence at least during the initial stages of training. However, it seems that both the fixed stepsize and line search approach eventually ends up in the same minimum. In comparison to stochastic online gradient descent with fixed stepsize there is no doubt that this algorithm achieves a significantly faster convergence than either of the gradient batch methods. Thus, within the first 30 epochs it has reached a level of the cost that is below the final level obtained by batch mode gradient descent. This indicates that there are large redundancies in the training data which is very typical to speech recognition applications. Extended Baum-Welch reestimation is seen to perform slightly worse than online gradient descent in the initial stages of training, but ends up with roughly the same conditional likelihood after about 50 iterations.

Figure 5.5 illustrates the training scenario for online gradient descent with stepsizes decaying inversely proportional to the epoch number for different values of N_η in (5.5). Interestingly, the fastest convergence is obtained for $N_\eta \to \infty$ corresponding to a fixed stepsize of $\eta = 0.01$.

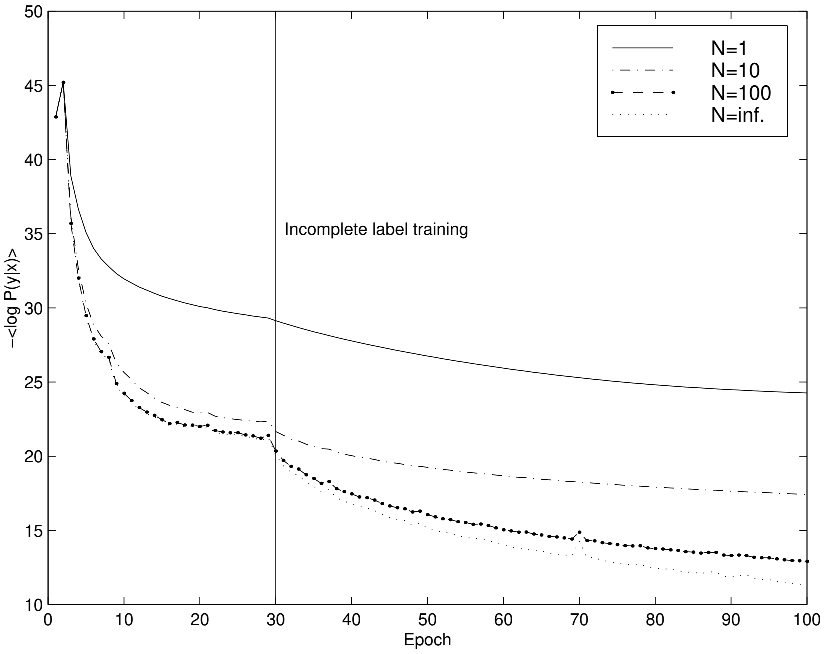

Figure 5.5: Online gradient descent training with decaying stepsize. The initial stepsize is $\eta^{(0)} = 0.01$.

In [Val95, KVY93] a comparison between different batch mode gradient-based training strategies for MMI estimation of continuous density HMMs was given for the recognition of the British English E-set[7] and for a TIMIT phoneme recognition task. The conclusion of these experiments was that batch gradient descent with a fixed or heuristically adapted stepsize performed worse than when an approximative line search was invoked. Furthermore, they observed that faster convergence could be obtained by an approximative second order method called *QuickProp* [Fah88] or by so-called *conjugate* gradient descent, see *e.g.*,

[7]The E-set is the set of acoustically confusable letters {“B”, “C”, “D”, “E”, “G”, “P”, “T” & “V”}.

[HKP91, Bis95]. However, all of these gradient-based batch approaches could not outperform the extended Baum-Welch reestimation algorithm. Because we here have observed a similar performance of online gradient descent and extended Baum-Welch reestimation for the broad class task it was decided not to try *e.g.*, conjugate gradient descent or approximative second order methods for the discrete CHMMs. The online gradient descent method has been used for all experiments reported below because it also applies to models with parameters that do not normalize locally and because it generalizes to the hidden neural network hybrid presented in the next chapter.

For this choice of algorithm it is very interesting to observe the behavior of the clamped ($\mathcal{L}_c$) and free running phase ($\mathcal{L}_f$) negative log likelihoods during training. For a uniformly initialized model and a model initialized by two iterations of complete label ML reestimation the scenarios are illustrated in figure 5.6. For both situations we see that the clamped and free running phase likelihoods behave similarly, but that the distance between them is decreased such that $\mathcal{L} = \mathcal{L}_c - \mathcal{L}_f$ for the training set is decreased. Interestingly, $\mathcal{L}_c$ and $\mathcal{L}_f$ starts by decreasing for the uniform model, but then suddenly starts to increase. Similarly, we see that both $\mathcal{L}_c$ and $\mathcal{L}_f$ for the model initialized by ML training increase during CML estimation.

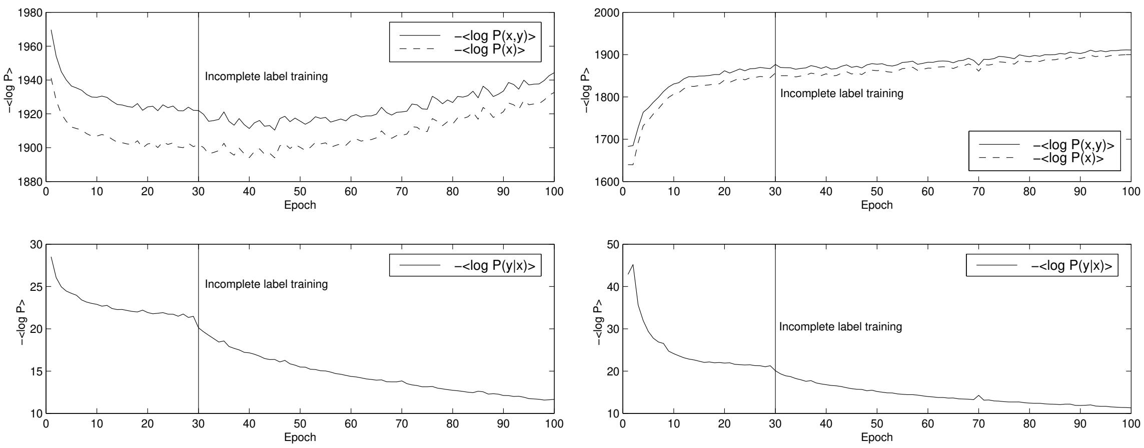

Figure 5.6: Evolution of average clamped phase ($< \log P(\boldsymbol{x},\boldsymbol{y}) >$), free running phase ($< \log P(\boldsymbol{x}) >$) and conditional ($< \log P(\boldsymbol{y}|\boldsymbol{x}) >$) log likelihoods for the training set. ***Left panel:*** Model initialized by uniform probabilities. ***Right panel:*** Model initialized by two iterations of complete label ML reestimation.

A set of initial experiments with CML estimation indicated a slight tendency to overfit the training data even though the models are sparse on parameters. It was therefore decided to adopt the method used by *e.g.*, Renals *et al.* [RMB+94] for adapting the stepsize according to the performance on a validation set. The schedule is as follows: if the forward-backward decoder recognition *accuracy* on the validation set drops by more than 0.5% over five epochs then the stepsize is multiplied by a factor of 0.8 and the momentum term is reset for the following epoch. After a few initial experiments this schedule was found to improve the final recognition accuracy slightly compared to using a fixed stepsize. Furthermore, it makes the selection of the initial stepsize less critical. To limit the computational requirement an upper bound of 100 training epochs was enforced in all experiments reported below.

To reduce the effect of overfitting *regularization* of the cost function was also attempted. The regularized cost function $C(\mathbf{\Theta})$ is defined by

$$C(\mathbf{\Theta}) = \mathcal{L}(\mathbf{\Theta}) + R(\mathbf{\Theta}||\tilde{\mathbf{\Theta}}), \tag{5.7}$$

where $R(\mathbf{\Theta}||\tilde{\mathbf{\Theta}})$ is the regularization term. Similar to the *weight decay* regularizer [HKP91, Bis95] often used for neural network training one can use a regularizer of the form,

$$R(\mathbf{\Theta}||\tilde{\mathbf{\Theta}}) = -\alpha \sum_i \tilde{\omega}_i \log \frac{\omega_i}{\tilde{\omega}_i}, \tag{5.8}$$

where $\omega_i \in \mathbf{\Theta}$ are generic parameters of the model and $\tilde{\omega}_i \in \tilde{\mathbf{\Theta}}$ are *normalized* regularization parameters which can be interpreted as the parameters of a constant *regularization* model $\tilde{\mathbf{\Theta}}$ with the same topology as the model $\mathbf{\Theta}$. We see that $R(\mathbf{\Theta}||\tilde{\mathbf{\Theta}})$ is the Kullback-Leibler divergence between the distributions given by $\mathbf{\Theta}$ and the regularization model $\tilde{\mathbf{\Theta}}$ and that it leads to a new set of gradients where the $\tilde{\omega}_i$'s are simply added to the difference between the expected counts;[8] $\partial C/\partial\omega = -(m - n + \alpha\tilde{\omega})/\omega$. Such a regularizer will make parameters ω_i that are not "reinforced" decay towards the associated $\tilde{\omega}_i$'s. This implicitly corresponds to reducing the effective number of trainable parameters and will hopefully lead to better recognition performance. A possible choice for $\tilde{\mathbf{\Theta}}$ is a model with uniform probabilities which is similar to the standard weight decay for neural networks. Another choice is to use the ML trained model for $\tilde{\mathbf{\Theta}}$. Unfortunately, none of these approaches seemed to improve recognition performance in a set of initial experiments and consequently the idea was not pursued any further.

5.3.2 Complete Label Training

Because of the overfitting effects and the stochastic nature of the online gradient algorithm, the CML training scenario is likely to be more sensitive to the initial model parameters than for Baum-Welch reestimation. Figure 5.7 shows the errorbars on $\mathcal{L}$ for ten independent complete label CML training sessions of random initial models with and without random noise added to the expected count difference in the gradient expressions (4.21)-(4.22). Whereas practically the same likelihood level was obtained by incomplete label ML reestimation in a similar experiment (see section 5.2.2), figure 5.7 shows a large variation in the training set conditional likelihood between the ten runs. However, the variation in the recognition accuracies between the ten runs was only of order $\pm 0.2\%$ for both experiments. A similar finding was done for incomplete label CML training by gradient descent. Thus, as a rule of thumb the results presented below may be considered as significantly different if they deviate by more than 0.4%.

Table 5.6 shows the recognition accuracies obtained by complete label CML training of a CHMM initialized by uniform probabilities or by two iterations of complete label ML reestimation. The ML bootstrapped model performs slightly better than the uniformly

[8] The Kullback-Leibler regularization is very closely related to imposing a discrete Dirichlet prior on the parameters of a discrete HMM. Such a prior distribution turns the maximum likelihood approach into a Bayesian maximum a posteriori approach, where the Baum-Welch reestimation algorithm estimates the parameters that maximize the posterior $P(\mathbf{\Theta}|\boldsymbol{x})$ instead of the likelihood $P(\boldsymbol{x}|\mathbf{\Theta})$, [DLR77]. These methods are commonly used in so-called speaker adaptive training where speaker independent models are adapted for a single speaker based on very limited training data from that speaker, see *e.g.*, [HCL95, GL94]. Another application is biological sequence analysis [DEKM98, KBM+94].

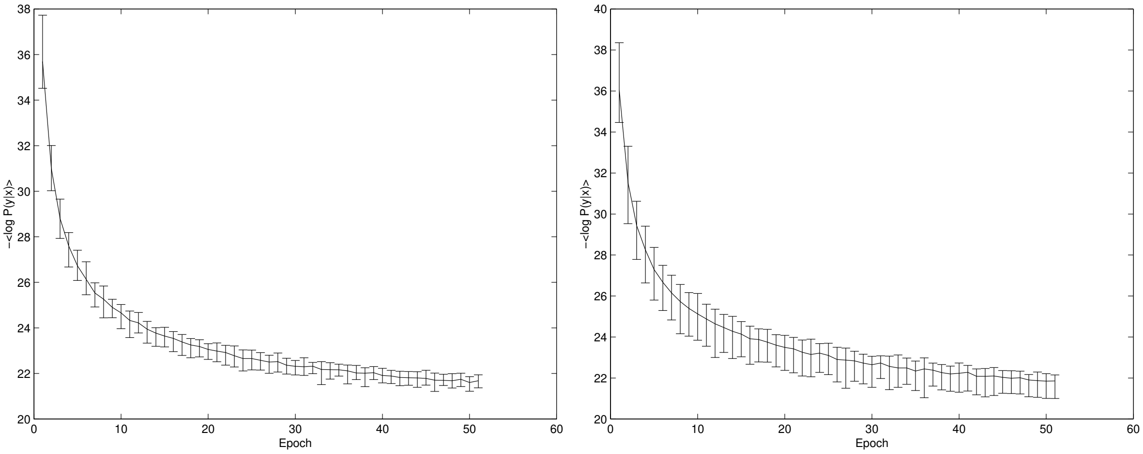

Figure 5.7: Errorbars on the training set negative log conditional likelihood for complete label CML training by stochastic online gradient descent with fixed stepsize. ***Left panel:*** Ten models initialized by random probabilities. ***Right panel:*** Ten models trained with random noise added to the difference between the m and n counts. The initial noise amplitude is 0.1 and the noise is decreased by a factor of 0.85 at each epoch.

	Train			Test		
Initial ML	F-B	N-best	Vit	F-B	N-best	Vit
0	80.3	81.2	76.9	78.6	78.8	76.3
2	80.3	***81.4***	77.1	78.9	***79.0***	76.6

Table 5.6: Recognition accuracies (%Acc) for complete label CML trained CHMM. The model is initialized by 0 or 2 iterations of complete label ML reestimation.

initialized model, but the difference is not significant. The table also shows that the N-best and forward-backward decoders give approximately the same recognition accuracies and that they significantly outperform the Viterbi decoder. Thus, for both initialization methods the "all path" decoders obtain more than 2% higher recognition accuracy than the "best path" Viterbi decoder. Compared to the ML estimated models we see from table 5.6 that CML estimation gives considerably higher recognition accuracies. Thus, the best test set result by complete label CML training is an accuracy of 79.0% which should be compared to 76.1% for ML training.

In the left panel of figure 5.8 the accuracies obtained by the three decoders are shown as complete label training progresses. For the forward-backward and N-best decoders we see that there is an almost constant difference in accuracy between the training and test set after about 10-20 epochs. A slight tendency to overfit the training data is responsible for part of this difference, but the intrinsic difference between the test and training sets also contributes. In fact, the training and validation set accuracies are almost identical throughout training, because both of these sets contain sentences of the same phonetically diverse type. On the other hand, the phonetically compact sentences in the test set are specifically designed to cover all biphone variations possible in the TIMIT phone inventory, and they are therefore somewhat harder to recognize. After just two to three complete label CML epochs the accuracy obtained by the Viterbi decoder decreases rapidly for *all* three datasets, see the left panel of figure 5.8. This is *not* an overfitting effect because the accuracy also drops for the training set. It merely illustrates that there is a mismatch between complete label CML training based on all paths and single path decoding. The mismatch is further emphasized by the relatively large oscillations in the curve for the Viterbi decoder.

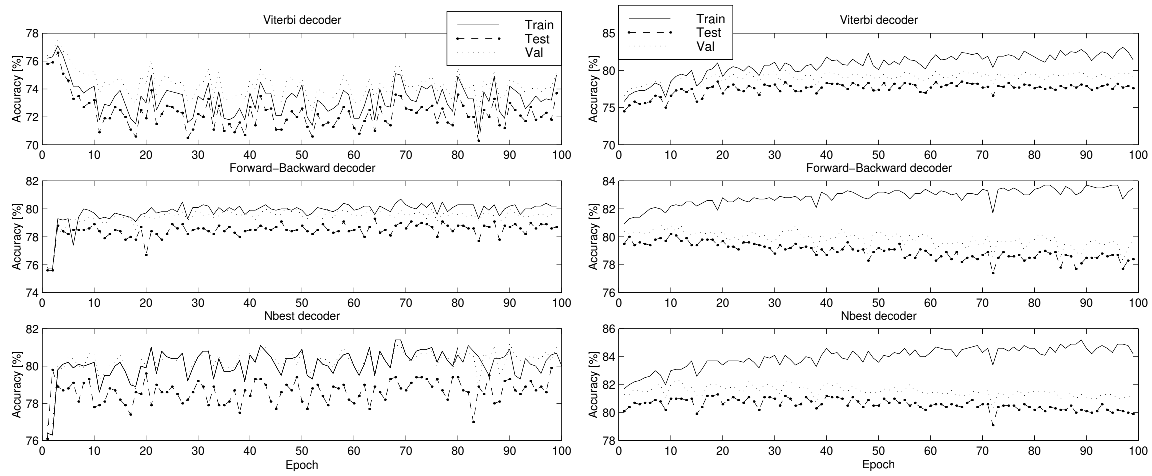

Figure 5.8: Evolution of recognition accuracies during CML training for a (from top to bottom) Viterbi, forward-backward or N-best decoder. ***Left panel:*** Complete label training. ***Right panel:*** Incomplete label training.

5.3.3 Incomplete Label Training

When switching to incomplete label training after five epochs of complete label CML training the accuracy for the training set increases fairly smooth for all three decoders as shown in the right panel of figure 5.8. However, the increasing size of the gap in accuracy between the training and test set indicates that the model now overfits the training data. This is also seen in table 5.7 showing the training and test set recognition accuracies for the incomplete label CML trained CHMM. By comparing table 5.7 and table 5.6 we furthermore see a significant gain in accuracy for both the training and test set by switching to incomplete label CML training. This is different from the ML trained models where practically no gain was observed by switching from complete to incomplete label training. One reason for this is that the CML criterion is highly sensitive to outliers and mislabelings as discussed in chapter 3. Since the TIMIT phone segmentation is not perfect the complete labeling for the broad class task may very well contain errors. By switching from complete to incomplete label CML training the effect of such errors is mitigated and the N-best accuracy is increased from 79.0% to 81.3%.

As for the complete label training experiments, the "all-path" N-best and forward-backward decoders obtain significantly higher accuracies than the Viterbi decoder, see table 5.7. For the same example sentence as in figure 5.3 for the ML estimated model, figure 5.9 clearly shows that several paths contribute to the probability of the sentence in the incomplete label CML trained model. Note that the N-best accuracy for the sentence in the figure is 78.1% compared to only 62.5% for the ML trained model.

Train			Test		
F-B	N-best	Vit	F-B	N-best	Vit
82.0	***83.7***	81.1	79.9	***81.3***	78.4

Table 5.7: Recognition accuracies (%Acc) for an incomplete label CML trained CHMM. The model is initialized by two iterations of complete label ML and five epochs of complete label CML training.

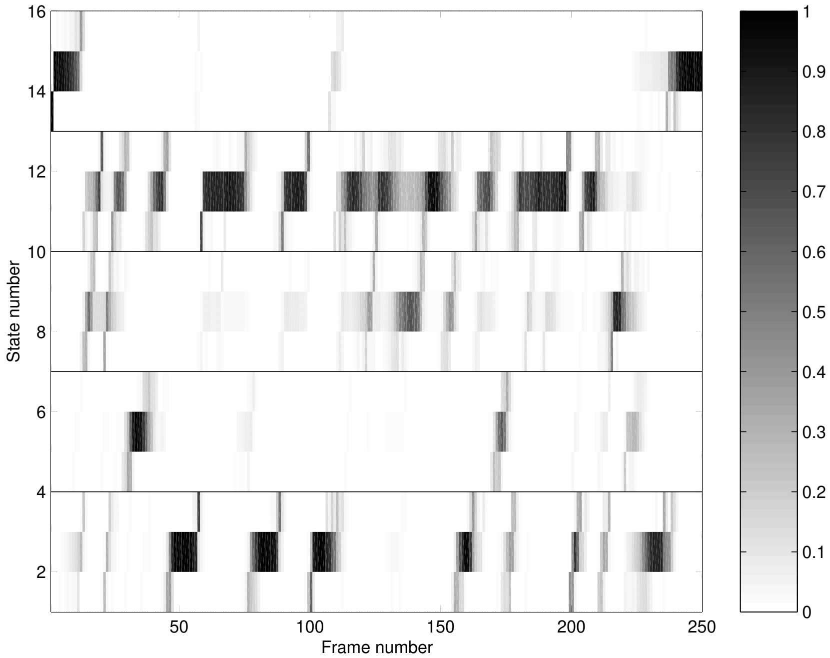

Figure 5.9: Same as figure 5.3, but for incomplete label CML trained model. Sentence N-best accuracy: $\%Acc = 78.1$. Compare to figure 5.3 for ML trained model.

5.4 The Language Model Probabilities

In the CML estimated CHMM all parameters including the "bigram" transition probabilities between different submodels were trained discriminatively from the acoustic data. Table 5.8 compares the test set accuracy obtained by CML estimation to that obtained by MMI estimation of a model with a fixed uniform or bigram language model. The uniform or *zero-gram* language model assigns the same probability to all pairs of broad classes whereas the fixed bigram probabilities are given by the relative frequencies of broad class pairs observed in the training set. Using a fixed bigram instead of a zero-gram improves the recognition accuracy considerably. The gain in accuracy by discriminatively training the bigram as opposed to using a fixed bigram is also significant.

In continuous speech recognition it is very common to scale the language model probabilities by some factor before decoding. This has been observed to increase accuracy significantly for some tasks. Figure 5.10 shows the effect of language model scaling on the recognition accuracy. For the complete label ML trained model a fairly large gain in accuracy can be obtained by appropriately scaling the bigram probabilities. Interestingly,

Language model	$\%Acc$
Fixed zero-gram (MMI)	80.1
Fixed bigram (MMI)	80.9
Trainable bigram (CML)	81.3

Table 5.8: Effect of language model transitions on test set accuracy. Only results for the N-best decoder are shown.

squaring the bigram probabilities for this model as proposed in [KVY93] increases the accuracy by more than 1% for the ML trained model. The advantage of scaled bigrams is still present for the MMI trained model with a bigram language model, but it is far smaller than for the ML trained model. Conversely, there is no gain in accuracy by scaling the discriminatively trained bigram in the CML estimated model. Thus, the CML criterion is able to "balance" the language model and acoustic model probabilities during training.

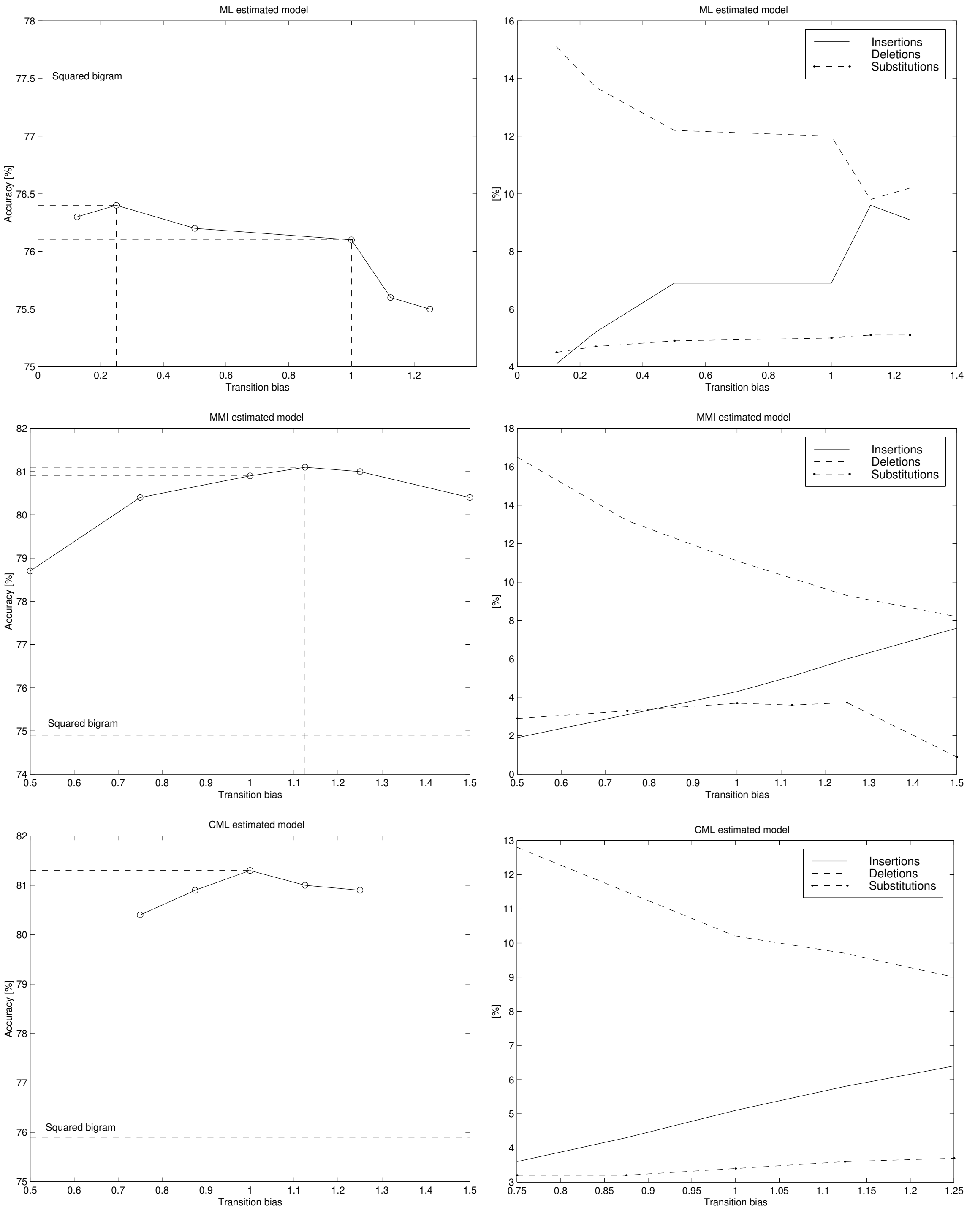

Figure 5.10: Effect of language model scaling for ML, MMI and CML trained model. ***Left panels:*** Accuracy for different biases. ***Right panels:*** Effect of scaling on the percentage of insertions, deletions and substitutions (the lower curves show substitutions).

5.5 Non-Normalizing Parameters

Because of the poor effect of language model scaling for the CML estimated model it cannot be expected that relaxing the sum-to-one constraint for all parameters will improve performance significantly. Indeed, if only a positivity constraint is enforced on the parameters by the exponential parameter transformation (4.39) the test set accuracy drops from 81.3% to 80.5% and the training set accuracy drops from 83.7% to 82.8%. A probable reason is the large fluctuations in the conditional likelihood illustrated in figure 5.11. As discussed in chapter 4 the exponential parameter transformation leads to a parameter update in which the current parameter value is multiplied by $\exp(\eta(m-n))$ where m and n are the expected counts defined in chapter 2 and chapter 4. This can lead to very large parameter changes resulting in a fluctuating conditional likelihood. The only way in which this effect can be avoided is by using a very small stepsize to keep the argument of the exponential function small even for large gradients. However, this will unfortunately also imply very slow convergence. Instead of the exponential transformation one can use the sigmoid transformation. With this transformation we were able to obtain an accuracy of 81.4% and as shown in figure 5.11 the sigmoid transformation is much more "well behaved" than the exponential transformation.

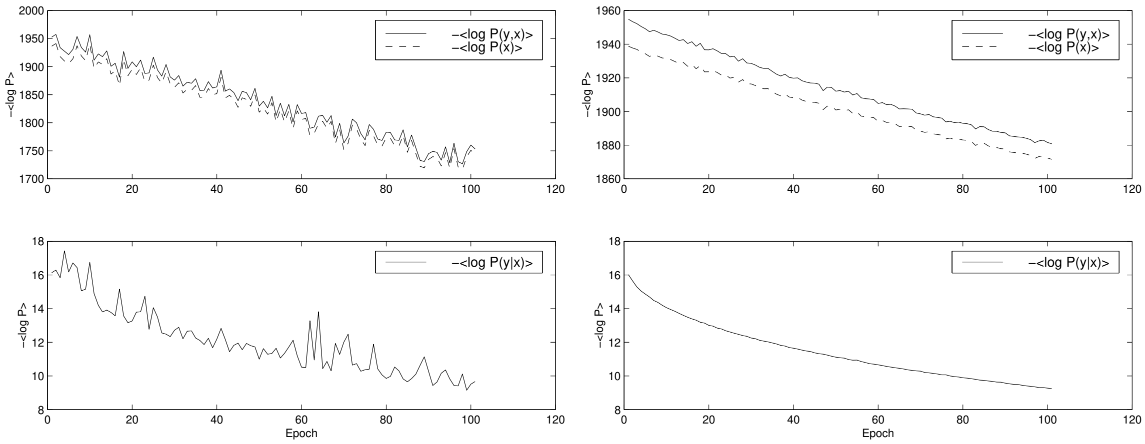

Figure 5.11: Training of a model with positivity constraint. ***Left panel:*** Exponential parameter transformation. ***Right panel:*** Sigmoid parameter transformation.

5.6 Reported Results

For the five broad phoneme class task Johansen *et al.* [Joh96, JJ94a, Joh94] have reported a number of results using continuous HMMs with diagonal covariance Gaussian match distributions. As we have selected an almost identical model setup and identical datasets, a direct comparison is possible. For various numbers of components in the Gaussian match distributions the best results for ML, MMI and CML trained models obtained by Johansen *et al.* are shown in table 5.9. Note that they also used an N-best decoder very similar to the one used here.

Interestingly, the accuracies for the continuous Gaussian density HMMs used by Johansen *et al.* are significantly lower than those obtained by discrete CHMMs. Thus, for

Criterion	K	Parms.	$\%Acc$
ML	1	824	60.0
	5	4004	67.9
	16	12749	73.3
MMI	1	824	77.5
CML	1	824	78.2

Table 5.9: Test set recognition accuracies ($\%Acc$) for ML, MMI and CML trained continuous HMMs reported by Johansen *et al.*. K is the number of diagonal covariance Gaussians used in each mixture.

approximately the same number of parameters as in the discrete CHMM they report an accuracy of only 67.9% for an ML trained model compared to 76.1% for the ML trained CHMM. Similarly, their best result for CML estimation is 78.5% accuracy compared to 81.3% for the discrete CHMM. The reasons for these large differences are probably different training strategies and the fact that they use diagonal covariance Gaussians. The diagonal approximation corresponds to assuming that the elements of the feature vectors are independent which is obviously untrue. This assumption is not used in the discrete distributions. Similarly, the large number of codebook vectors relative to the size and difficulty of the task implies that most of the class specific information in the feature vectors can be retained in the quantized observation symbols.

5.7 Summary

In this chapter a discrete CHMM was evaluated on the simple task of recognizing five broad phoneme classes in the TIMIT database. Through the experiments several topics concerning both discriminative training and decoding were illustrated. It was shown that for ML estimated models only one path tends to contribute to the probability of a sentence whereas several paths tend to contribute for the CML estimated models. Thus, for discriminatively trained models, a far better recognition accuracy can be obtained by a decoder that considers all paths in the search for the best hypothesis. Two such decoders, namely the simple forward-backward and the N-best decoder, were shown to outperform the Viterbi decoder. The large gain obtained by forward-backward instead of Viterbi decoding is rather surprising as the former is not guaranteed to yield a hypothesis that corresponds to a valid path through the model. This effectively means that the minimum duration enforced by the left-to-right topology of the class-models is not used during forward-backward decoding. A nice property of the forward-backward decoder is that it only requires approximately twice the computation of the Viterbi decoder. Contrary to this, the N-best decoder is computationally very expensive, and even for the simple broad class task local and global pruning was a necessity.

The importance of good initial models for incomplete label ML and CML training was also illustrated. For the ML estimated model we found that complete and incomplete label training yielded approximately the same accuracy ($\approx$76%). In contrast, CML training based on the incomplete labeling gave a far better recognition accuracy than training based on the complete labeling. Part of this difference is attributed to the sensitivity of the CML criterion to outliers. Compared to incomplete label ML estimation a gain of more than 5% in accuracy was observed for the incomplete label CML trained model.

Finally, it was shown how a simple language model bias can improve the accuracy for ML and MMI estimated models. A similar gain was not observed for the CML trained model. This indicates that a better "balance" between the different parameter sets in the CHMM is obtained through training all parameters to minimize the same discriminative criterion.

Chapter 6

Hidden Neural Networks

Over the past decade neural networks[1] have proven very useful for complex pattern recognition tasks. As speech recognition is basically a pattern recognition task, a number of researchers have recently tried to apply a variety of neural network architectures to the speech recognition problem. The first attempts used neural networks to classify small speech segments into *e.g.*, voiced/unvoiced classes. The success in these experiments encouraged researchers to try larger tasks like static phoneme and word classification. It quickly became evident, however, that neural networks are not very good at handling temporal integration over long time spans which is crucial in continuous speech recognition. Thus, with standard neural networks it is difficult to handle the temporal variation in *e.g.*, words uttered by different speakers or at different speaking rates. Therefore, several researchers started combining elements of HMMs and neural networks. Excellent reviews of early neural network approaches for speech recognition can be found in [Teb95, MS91, Lip89].

An obvious way to combine HMMs and neural networks is to try to implement various components of the HMM using neural networks. For example, in [LG87] a special recurrent neural network called a *Viterbi Net* was introduced as a way of implementing the Viterbi algorithm for an HMM. In a similar spirit the recurrent *AlphaNet* was introduced by Bridle [Bri90] as a way of implementing the forward recursions of an HMM. Another very interesting neural network implementation of a discrete HMM is the so-called *Boltzmann Chain* introduced in [SJ95]. The Boltzmann Chain is a particular variant of the *Boltzmann machine* [AHS85], which is very similar to the globally normalized discrete HMM[2] [Ped97, Mac96]. It can be trained by a gradient-based algorithm which has many similarities to gradient-based forward-backward training for globally normalized HMMs, see [Ped97]. Compared to the globally normalized HMM the Boltzmann Chain has the advantage that several chains can be interconnected to form a so-called *Boltzmann Zipper* [SJ95, Ped97]. The Boltzmann Zipper can model parallel sequences at disparate time-scales which is very attractive in *e.g.*, *speech reading* applications where the recognition of an utterance is based on both (rapidly varying) acoustic information and (slowly varying) video recordings of the speakers face. A tutorial style review of the Boltzmann Chain and Zipper can be found in [Ped97].

Rather than simply reimplementing the HMM with neural networks most current re-

[1] For an introduction to neural network modeling the reader is referred to the excellent text books [HKP91, Bis95, Hay94].

[2] If the exponential parameter transformation (4.39) is used for the parameters of the globally normalized HMM, then $P(\boldsymbol{x}|\boldsymbol{\Theta})$ in (4.33) is expressed by a Boltzmann distribution similar to the one for the Boltzmann Chain.

search in HMM/NN hybrids focus on capitalizing the strengths of each of the two frameworks: the temporal modeling capabilities of the HMM and the static classification or function approximation capabilities of neural networks. This theme is carried throughout the rest of this chapter where a new HMM/NN hybrid called *Hidden Neural Networks* (HNN) will be introduced.

The chapter starts out by elaborating on a particular approach for alleviating the assumption of observation context independent match and transition probabilities in the standard HMM. It is shown that this approach quickly becomes intractable in practice and the HNN is consequently introduced as a more flexible architecture for observation context dependent modeling. Besides mitigating the observation context assumption it will be shown that the HNN also has a number of other advantages compared to standard HMMs. Then conditional maximum likelihood estimation and computational complexity issues for the HNN is discussed and finally it is shown how a number of simplifying assumptions in the HNN architecture can lead to a so-called *transition-based* model. The chapter is concluded by a comparison of HNNs to related mainstream HMM/NN hybrids.

6.1 Observation Context

Let us consider the expression for the HMM probability $P(\boldsymbol{x}|\boldsymbol{\Theta})$ given in section 2.2 and repeated below,

$$\begin{aligned} P(\boldsymbol{x}|\boldsymbol{\Theta}) &= \sum_{\boldsymbol{\pi}} \prod_{l} P(\boldsymbol{x}_l|\boldsymbol{\pi}_1^l, \boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}) P(\pi_l|\boldsymbol{\pi}_1^{l-1}, \boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}) \\ &\approx \sum_{\boldsymbol{\pi}} \prod_{l} P(\boldsymbol{x}_l|\pi_l, \boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}) P(\pi_l|\pi_{l-1}, \boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}). \end{aligned} \tag{6.1}$$

In the above approximation we have applied the first order Markov assumption which facilitates the computationally efficient forward-backward and decoding algorithms associated with the HMM. For standard HMMs it is assumed that the transition and match probabilities are independent to the previous or *causal* observation sequence $\boldsymbol{x}_1^{l-1}$. However, this assumption is not really necessary as indicated by (6.1) and discussed below.

For simplicity, assume that the observations are discrete and that we condition the transition probabilities on one previous observation x_{l-1}, that is, we use "standard" match probabilities $\phi_i(a)$. In this case there is one set of transition probabilities for each possible observation symbol $a \in \mathcal{A}$, $\theta_{ij}(a) = P(\pi_l = j|\pi_{l-1} = i, x = a)$, and the probability of the observation sequence for this model is computed using the standard forward (or backward) algorithm[3] where θ_{ij} is replaced by $\theta_{ij}(x_{l-1})$. Similarly, the context dependent transition probabilities $\theta_{ij}(a)$ can be reestimated using the expressions for the standard transitions θ_{ij} given in chapter 2, but with the expected count $n_{ij}(l)$ replaced by $n_{ij}(l)\delta_{x_{l-1},a}$. This approach naturally also works for match probabilities conditioned on one previous observation and it is easy to generalize to larger observation contexts.

A generalization of the above approach called *Partially Hidden Markov Models* (PHMMs) was introduced in [FR95]. Instead of conditioning directly on the causal observation sequence, the PHMM conditions the match and transition probabilities on discrete *context symbols* defined by *deterministic discrete mapping functions* of the causal observation

[3]It is straightforward to see that these algorithms also apply for observation context dependent probabilities, see *e.g.*, (2.11) in section 2.3.

sequence or a part thereof.[4] In the PHMM there is one mapping function for the transition probabilities yielding a context symbol v_l and one mapping function for the match probabilities yielding a context symbol w_l such that

$$P(x_l|\pi_l, \boldsymbol{x}_1^{l-1}) = P(x_l|\pi_l, w_l) \tag{6.2}$$

and

$$P(\pi_l|\pi_{l-1}, \boldsymbol{x}_1^{l-1}) = P(\pi_l|\pi_{l-1}, v_{l-1}). \tag{6.3}$$

Both mapping functions have to be chosen a priori and the variables v_l and w_l can be one of N_v or N_w discrete context symbols, respectively. In the simplest form, the mapping functions yield a unique context symbol (v_l or w_l) for each possible realization of the causal observation sequence (or part thereof). This is equivalent to the direct approach described above. The mapping functions can, however, group two or more realizations such that the number of possible context symbols is reduced. For some a priori chosen mapping functions, the parameters of the PHMM can be reestimated in a manner similar to the parameters of the standard HMM. The reestimation formulas for the PHMM are identical to those for the standard HMM, except that the expected counts ($n_{ij}(l)$ or $n_i(l)$) must be multiplied by delta-functions indicating the observed context symbol (v_l or w_l) [FR95].

The PHMM has been successfully applied to coding of bi-level (black/white) images in [FR95]. Because the observations in this task can only be one of two distinct symbols (black/white) it is possible to use large observation contexts without limiting N_v and N_w (*i.e.* for one causal observation the maximum number of context symbols is $N_v, N_w = 2$, for two causal observations it is $N_v, N_w = 4$ *etc.*). For speech recognition with discrete HMMs, however, there will often be a large number of different observation symbols in the alphabet $\mathcal{A}$ so as to reduce the distortion introduced by the vector quantizer. Typically, $\mathcal{A}$ contains 256 distinct symbols and therefore we need to estimate 256 times as many probabilities compared to the standard HMM if we condition on just one previous observation. Naturally the number of parameters can be reduced by defining deterministic mappings with $N_v, N_w < 256$. But defining such functions a priori can be quite difficult, especially if the observations are continuously valued vectors. Furthermore, good mapping functions for one task will not necessarily be good for other tasks.

6.2 HNN Architecture

Instead of using deterministic discrete mapping functions as in the PHMM, a more general approach is to use continuously valued parameterized functions to estimate *scores* directly related to the context dependent probabilities. This is the basic idea in the Hidden Neural Network model in which the standard probability parameters of a CHMM are *replaced* by the outputs of neural networks assigned to each state. Thus, in the HNN it is possible to assign up to two[5] neural networks to each state: 1) a *match network* estimating a *score* for how well the current observation matches the state given the observation context and 2) a *transition network* that outputs transition *scores* dependent upon the observations.

[4] The partially HMM was originally proposed for discrete observations, but it applies equally well to continuous valued observation vectors if the discrete mapping functions are defined on the space of such vectors.

[5] Here only single-label-states are considered. For multiple-label-states it is possible to assign a third network for estimating label probabilities.

We have used the term 'score' instead of 'probability' because the HNN will be trained using the CML criterion which, as was discussed in chapter 4, automatically ensures *global* normalization. Thus, the outputs of these networks need not sum to one.

More formally, the CHMM match probability $\phi_i(\boldsymbol{x}_l)$ of observation $\boldsymbol{x}_l$ in state i is replaced by the output of a match network, $\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i)$, assigned to state i. The match network in state i has only *one* output, is parameterized by the weight vector $\boldsymbol{v}^i$ and takes the observation context vector $\boldsymbol{s}_l$ (defined below) as input, see figure 6.1. Using only a single output allows us to treat continuously valued observation vectors and discrete observations in practically the same way.[6] Similarly, the CHMM probability θ_{ij} of a transition from state i to j is replaced by the output of a transition network $\theta_{ij}(\boldsymbol{s}_l; \boldsymbol{w}^i)$ which is parameterized by weights $\boldsymbol{w}^i$. The transition network assigned to state i has as many outputs as there are non-zero transitions *from* state i, see figure 6.1.

According to (6.1) the network input $\boldsymbol{s}_l$ at time l should in theory only include causal observations. However, in the following chapters we will see that for the considered speech recognition problems the "orientation" of the observation context is not as important as the *size* of it. That is, a symmetric context of $2K+1$ observations $\boldsymbol{s}_l = \boldsymbol{x}_{l-K}^{l+K} = \boldsymbol{x}_{l-K}, \ldots, \boldsymbol{x}_l, \ldots, \boldsymbol{x}_{l+K}$ turns out to work just as well as a left or right context of the same size $\boldsymbol{s}_l = \boldsymbol{x}_{l-2K}^{l}$. In cases where the context $\boldsymbol{s}_l$ extends beyond the boundaries of the observation sequence zero padding can be used to ensure a well defined input to the networks.

We note in passing that $\boldsymbol{s}_l$ can in principle be any sort of information related to $\boldsymbol{x}_l$ or even the observation sequence in general. In speech for instance, one could imagine that other information about the signal was appropriate, such as the type of speaker (male/female; British/American). We call $\boldsymbol{s}_l$ the *context* of observation $\boldsymbol{x}_l$, but the reader should bear in mind that it can contain all sorts of information and that it can even differ from state to state. The only limitation is that it cannot depend on the path through the model, because then the state process is no longer first order Markovian.

The neural networks in the HNN can be feed-forward or recurrent networks. In fact, they need not even be neural networks — they can be any parameterized continuously valued approximating function. In this work the neural networks in the HNN are chosen to be *Multi-Layer Perceptrons* (MLPs) because 1) MLPs are parameter efficient and can be trained by the efficient *backpropagation* algorithm [RHW86] and 2) MLPs can approximate any continuously valued function on a compact set of points to arbitrary accuracy provided the MLPs are sufficiently complex and "well" trained, see *e.g.*, [Cyb89] or the neural network text books [HKP91, Bis95]. Contrary to the deterministic mapping functions in the PHMM, the MLPs in the HNN are adapted during training to suit the task at hand. Typical examples of match and transition networks used in this work are illustrated in figure 6.1

6.2.1 Global Normalization

In an HNN specified by the parameter vector $\boldsymbol{\Theta}$ it is generally not possible to make $\sum_{\boldsymbol{x} \in \mathcal{X}} P(\boldsymbol{x}|\boldsymbol{\Theta}) = 1$ by normalizing locally at the state level. Even though we can normalize the transition network outputs locally by *e.g.*, a softmax output function it is in general

[6] If the observations are discrete (as is inherently the case in *e.g.*, biological sequence modeling) they can be encoded in orthogonal binary vectors and used as input to the neural network in the same way as continuously valued vectors. The binary orthogonal encoding can result in a large number of weights in the neural networks and more optimal encoding schemes is sometimes preferable, see *e.g.*, [RK96a, Rii95]

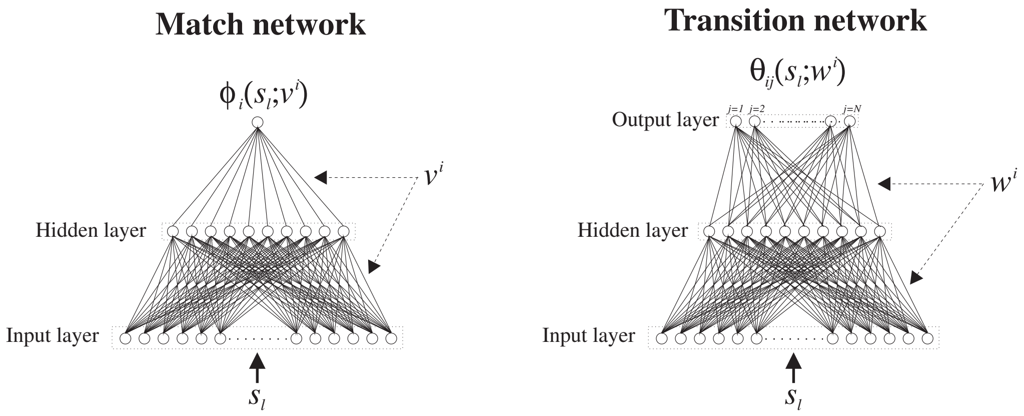

Figure 6.1: Examples of match and transition networks typically used in this work.

impossible to do this for the match networks because their outputs have to normalize over the space of possible inputs. A probabilistic interpretation of the HNN is ensured instead by normalizing globally. As discussed in chapter 4 such global normalization is automatically ensured when training the models by CML estimation. Similar to equations (4.32) and (4.35) for the CHMM we therefore define

$$R(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} R(\boldsymbol{x},\boldsymbol{\pi}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_{l} \theta_{\pi_{l-1}\pi_l}(\boldsymbol{s}_{l-1};\boldsymbol{w}^{\pi_{l-1}})\phi_{\pi_l}(\boldsymbol{s}_l;\boldsymbol{v}^{\pi_l}), \tag{6.4}$$

and

$$R(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}\in\Pi_y} R(\boldsymbol{x},\boldsymbol{\pi}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}\in\Pi_y} \prod_{l} \theta_{\pi_{l-1}\pi_l}(\boldsymbol{s}_{l-1};\boldsymbol{w}^{\pi_{l-1}})\phi_{\pi_l}(\boldsymbol{s}_l;\boldsymbol{v}^{\pi_l}). \tag{6.5}$$

The probability of the labeling is then computed as for the CHMM,

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta}) = \frac{R(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{\Theta})}{R(\boldsymbol{x}|\boldsymbol{\Theta})}. \tag{6.6}$$

The R-functions for the HNN are as usual computed by replacing $\phi_i(x_l)$ and θ_{ij} with $\phi_i(\boldsymbol{s}_l;\boldsymbol{v}^i)$ and $\theta_{ij}(\boldsymbol{s}_{l-1};\boldsymbol{w}^i)$, respectively, in the forward (or backward) algorithms defined in chapter 2 and 4.

The global normalization implies that the states in the HNN can be based on any combination of standard CHMM parameters and neural network estimated scores, see figure 6.2. Furthermore, we have a large degree of freedom in the selection of neural network output functions. A natural choice is a standard asymmetric sigmoid function (equation (4.42)). Another possible choice is an exponential output activation function, $g(h) = \exp(h)$ where h is the input to the output unit in question. In this case the HNN is equivalent to an MLP with a softmax output function provided that the HNN has one state per class and that these states are fully connected by uniform transition scores ($\theta_{ij}(\boldsymbol{s}_l;\boldsymbol{w}^i) = 1$ for all i,l), see figure 6.3. For this particular HNN only one path is allowed in the clamped phase for an observation sequence $\boldsymbol{x} = \boldsymbol{x}_1,\ldots,\boldsymbol{x}_L$ with an associated complete labeling $\boldsymbol{y} = y_1,\ldots,y_L$. Assuming that state i models label $c^i = i$ we

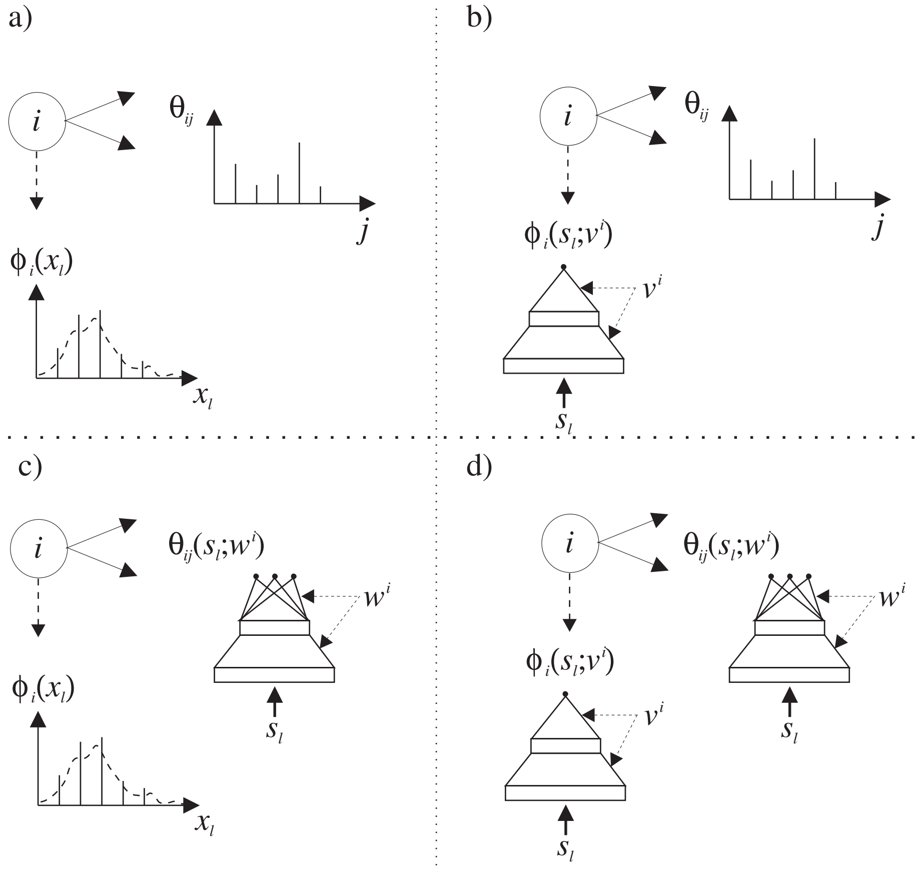

Figure 6.2: Possible states in an HNN: a) Standard CHMM state, b) state with match network and standard transition probabilities, c) state with standard match probabilities and transition network and d) state with both match and transition networks.

find

$$
\begin{aligned}
P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta}) &= \frac{R(\boldsymbol{x},\boldsymbol{y}|\boldsymbol{\Theta})}{R(\boldsymbol{x}|\boldsymbol{\Theta})} \\
&= \frac{\sum_{\boldsymbol{\pi}\in\Pi_y}\prod_l \phi_{\pi_l}(\boldsymbol{s}_l;\boldsymbol{v}^{\pi_l})}{\sum_{\boldsymbol{\pi}}\prod_l \phi_{\pi_l}(\boldsymbol{s}_l;\boldsymbol{v}^{\pi_l})} \\
&= \frac{\prod_l \phi_{y_l}(\boldsymbol{s}_l;\boldsymbol{w}^{y_l})}{\sum_{\boldsymbol{\pi}}\prod_l \phi_{\pi_l}(\boldsymbol{s}_l;\boldsymbol{w}^{\pi_l})} \\
&= \prod_l \frac{\exp[h_{y_l}(\boldsymbol{s}_l)]}{\sum_i \exp[h_i(\boldsymbol{s}_l)]}
\end{aligned}
\tag{6.7}
$$

where $h_i(\boldsymbol{s}_l)$ is the weighted input to the output unit of the match network assigned to state i. Equation (6.7) shows that the probability of the labeling is expressed as a product of "local" label a posteriori probabilities; $P(\boldsymbol{y}|\boldsymbol{x}) = \prod_l P(y_l|\boldsymbol{s}_l)$. These local probabilities are computed in the same manner as the output of the softmax normalized MLP shown in figure 6.3.

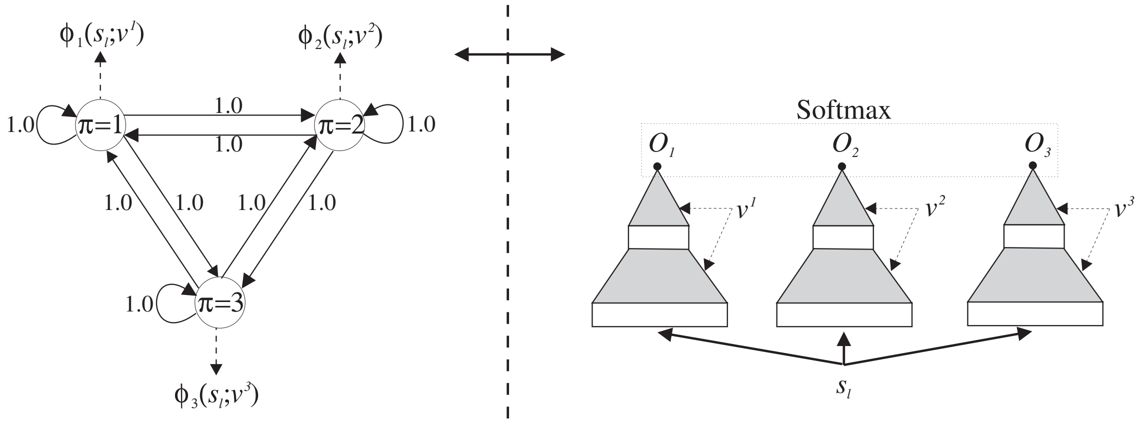

Figure 6.3: HNN equivalent to a softmax normalized MLP. Illustrated for the case of three classes, $\mathcal{C} = \{1, 2, 3\}$.

6.2.2 Advantages

Even though the HNN is a very intuitive and simple extension of the standard CHMM it is a more powerful model. Firstly, MLPs can often implement complex functions using fewer parameters than *e.g.*, a mixture of Gaussians. Furthermore, the neural networks in the HNN can directly use observation context as input and thereby exploit higher order correlations between neighboring observation vectors. Similarly, there is generally no assumption of independence between elements of continuous observation vectors. This is contrary to Gaussian mixture HMMs that use diagonal instead of full covariance matrices in order to reduce the number of parameters, see chapter 2. Finally, in HNNs the transition probabilities can depend on the context $\boldsymbol{s}_l$ at time l and the underlying Markov model can therefore have a "time-varying" dynamics. As discussed in [BF96, BF95] Markov models with "time-varying" dynamics can reduce the *diffusion of credit through time* whereby learning and representing *long-term dependencies* in the data becomes less difficult compared to standard HMMs. The ability to learn or represent the dependency between the observation at time l and time $l + l_0$ can essentially be measured in terms of the dependency between the state distribution at time l and time $l + l_0$ given by the model, $P(\pi_{l+l_0}|\pi_l, \boldsymbol{x}_l^{l+l_0}, \boldsymbol{\Theta})$. As discussed in [BF95] this quantity can be calculated as a product of transition matrices[7] between time l and $l + l_0$;

$$P(\pi_{l+l_0} = p|\pi_l = q, \boldsymbol{x}_l^{l+l_0}, \boldsymbol{\Theta}) = \{\boldsymbol{\theta}^l \boldsymbol{\theta}^{l+1} \cdots \boldsymbol{\theta}^{l+l_0}\}_{pq}. \tag{6.8}$$

For softmax normalized networks the transition matrix is given by $\boldsymbol{\theta}^l = \{\theta_{ij}(\boldsymbol{s}_l; \boldsymbol{w}^i)\}$ whereas it is given by $\boldsymbol{\theta}^l = \{\theta_{ij}\}$ for standard HMMs. Since the basic assumption in standard HMMs is that all relevant past data can be summarized in the current state variable (the state conditional observation independence assumption), we see that a dependency between state distributions at time l and $l+l_0$ also implies a dependency between observations at time l and $l + l_0$. However, the dependency between state distributions decreases exponentially as l_0 increases unless the transition probabilities are almost deterministic, *i.e.*, have values close to 1 or 0. Therefore it is difficult to learn and represent long-term dependencies in the data with standard HMMs unless the model has one/zero transitions

[7] Use the relation $P(a) = \sum_b P(a|b)P(b)$

(which is not very useful in practice!). On the other hand, "deterministic" transitions make a lot of sense in the HNN because they can vary along the observation sequence, *i.e.*, a transition probability close to one at time l can very well be close to zero at time $l' \neq l$ due to the observation context dependency. Therefore, long-term dependencies can in theory better be learned and represented by HNNs than by standard HMMs. The problem of learning and representing long-term dependencies in the data with HMMs and HNNs is similar to that encountered in recurrent neural network modeling of time-series [Ped97, BF94]. An additional benefit from context dependent transitions is that the model can learn to focus more on the non-steady state or "transitional" regions of the observation sequence. This is contrary to standard HMMs which model the sequence as a piecewise stationary process, *i.e.*, as a sequence of steady-state segments. Thus, the HNN with context dependent transitions can in theory better model "non-stationarities" in the observed sequence. We will elaborate on this topic in section 6.4.

6.3 Conditional Maximum Likelihood Estimation

As for the CHMM it is not possible to do CML estimation for the HNN using an EM or GEM algorithm and instead we suggest to train the model by a gradient method. Here we derive expressions for the HNN gradients and give a discussion of complexity issues for the HNN.

By using the chain rule we find the following gradients of $\mathcal{L}(\mathbf{\Theta}) = -\log P(\boldsymbol{y}|\boldsymbol{x}, \mathbf{\Theta})$ w.r.t. a generic weight ω^i in the match or transition network assigned to state i,

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \omega^i} = \sum_{kl} \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \phi_k(\boldsymbol{s}_l; \boldsymbol{v}^k)} \frac{\partial \phi_k(\boldsymbol{s}_l; \boldsymbol{v}^k)}{\partial \omega^i} + \sum_{kjl} \frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \theta_{kj}(\boldsymbol{s}_l; \boldsymbol{v}^k)} \frac{\partial \theta_{kj}(\boldsymbol{s}_l; \boldsymbol{v}^k)}{\partial \omega^i}. \tag{6.9}$$

If we assume that no weights are shared between networks we find by using the gradient expression for the CHMM (4.15) given in chapter 4,

$$\frac{\partial \mathcal{L}(\mathbf{\Theta})}{\partial \omega^i} = -\sum_{l} \frac{m_i(l) - n_i(l)}{\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i)} \frac{\partial \phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i)}{\partial \omega^i} - \sum_{lj} \frac{m_{ij}(l) - n_{ij}(l)}{\theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)} \frac{\partial \theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)}{\partial \omega^i}. \tag{6.10}$$

In the backpropagation algorithm for neural networks [RHW86] the derivative of a cost function w.r.t. a weight ω in the network can be computed efficiently by backpropagating *errors* at the output of the network. For an output activation function g and cost function $\mathcal{L}$ the derivative can be expressed as

$$\frac{\partial \mathcal{L}}{\partial \omega} = \mathcal{E} \times \frac{\partial g}{\partial \omega}, \tag{6.11}$$

where $\mathcal{E}$ is the error to backpropagate. We therefore see from (6.10) that the gradients can be computed by using the standard backpropagation algorithm on the neural networks in the HNN, where the error to backpropagate for *each* input $\boldsymbol{s}_l$ is,

$$\mathcal{E}_i^{\phi}(\boldsymbol{s}_l) = \frac{m_i(l) - n_i(l)}{\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i)} \tag{6.12}$$

for the match network assigned to state i and

$$\mathcal{E}^{\theta}_{ij}(\boldsymbol{s}_{l-1}) = \frac{m_{ij}(l) - n_{ij}(l)}{\theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)} \tag{6.13}$$

for output j of the transition network assigned to state i. The expected counts can be calculated by the forward-backward algorithms discussed previously by replacing $\phi_i(\boldsymbol{x}_l)$ and θ_{ij} with $\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i)$ and $\theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)$, respectively, for those parameters that are estimated by neural networks. Since we need both the m and n counts we have to run two forward-backward passes for each training sequence to find the errors to backpropagate; once in the free-running phase (the n's) and once in the clamped phase (the m's).

6.3.1 Complexity Issues

When the two forward-backward passes have been performed there is only little computational overhead in calculating the gradients for the CHMM. For the HNN the situation is different. From (6.10) we see that a backpropagation pass for each context input $\boldsymbol{s}_l$ to the neural networks is needed in order to compute the (accumulated) gradients for a particular training sequence. Since the expected counts are not available before we have finished the two forward-backward passes we have to store copies of the counts for each time-step l before we run the backpropagation algorithm. This is different from the CHMM where we only need to store the accumulated counts. In a direct implementation storage of the expected counts thus scales as $\mathcal{O}(L(N+N^2))$ for an ergodic HNN compared to $\mathcal{O}(N+N^2)$ for the CHMM. It is, however, possible to reduce this requirement to $\mathcal{O}(LN)$ for HNNs and $\mathcal{O}(N)$ for CHMMs by storing the two forward matrices instead of the expected counts and then do backpropagation training during the backward passes. In this work the direct approach has been used.

If all networks are assumed to have the same number of weights W then the backpropagation of errors results in an added computational complexity scaling as $\mathcal{O}(LNW)$ because the backpropagation algorithm has a complexity of $\mathcal{O}(W)$ for a network having W weights. Since the expected counts cannot be calculated before the two forward passes are completed, it is in principle necessary to store all activations of all units in the neural networks for each possible input $\boldsymbol{s}_l$ in order to run the backpropagation algorithm. For many applications this is unrealistic as L and N can be quite large and therefore lead to huge memory requirements even for small networks. Alternatively, one can evaluate the neural networks to determine the hidden and output unit activations just before each backpropagation pass (which was done in this work). This adds $\mathcal{O}(LNW)$ flops to the computational complexity.

In summary we see that the gradient calculation for the HNN requires an additional computation scaling as $\mathcal{O}(LNW)$ compared to the CHMM. Furthermore, for ergodic HNNs the memory complexity of storing expected counts is at least $\mathcal{O}(LN)$ compared to $\mathcal{O}(N)$ for the ergodic CHMM. In addition, we have to evaluate the neural networks for each sequence and store their outputs prior to running the forward-backward passes which effectively adds $\mathcal{O}(LNW)$ to the computational and $\mathcal{O}(L(N+N^2))$ to the memory complexity of the gradient calculation.

The additional computational complexity of CML estimation for the HNN is mostly due to computations related to the neural networks. Thus, the added complexity of doing two forward-backward passes instead of just one as for ML estimation is negligible compared to the neural network related computations. This is contrary to the CHMM for which CML estimation generally requires at least two times the computation of ML estimation.

For the HNN, a highly optimized implementation of the neural network computations is therefore important to fast training and decoding. In this work several "tricks" were used to speed up training. The most important ones include a lower threshold on the absolute value of the error to backpropagate and look-up-tables for the neural network activation functions.[8] The lower threshold gives large computational savings because only errors of a certain (absolute) size are backpropagated.

6.4 HNNs for Transition-Based Modeling

Standard HMM modeling is based on the assumption that the sequence we wish to describe is piecewise stationary, *i.e.*, that it consists of a sequence of steady-state segments. For the representation of speech signals this is consistent with a model of human speech production where the utterance is organized as a succession of *vocal-tract states.* The vocal-tract states represent different configurations of the human speech *articulators* and it is generally assumed that the gestures invoked to actualize these states are relatively slow. In the real human speech production system, however, the articulators merge spatially and temporally into a continuous process which can only be approximated by the vocal-tract states of the above model. In line with this, several perceptual and physiological studies have indicated that human perception of speech to some extent is based on "transitional" regions of the speech signal with fast spectral changes over time, see *e.g.*, [DPH93] and references therein. As such, the HNN with transition networks is a much better model than the standard HMM because it can (at least in theory) model speech signals as a succession of steady-state segments connected by "non-stationary" transitional regions.

More recently it has been shown [MBGH96, MBGH94, Kon96, Fur86a, Fur86b] that regions of the speech signal containing significant spectral change are critical to the recognition of speech. In [Fur86a, Fur86b] it was argued that sufficient information for identifying consonants and vowels in syllables is contained in those regions of an utterance with the largest spectral changes. Experiments reported by Morgan *et al.* in [MBGH96] indicated that good speech recognition performance can be obtained in noisy conditions if the model is able to focus on so-called auditory-event or *avent* regions of the speech signal. Morgan *et al.* used a model called a *Stochastic Perceptual Avent Model* in which all steady state-regions of the speech signal are modeled by the same submodel whereas "transitional" regions are modeled by separate avent models. The avent models were defined as transitional regions between phonemes, *i.e.*, local time-windows around phoneme boundaries.

In this section we will discuss how a few constraints on the general HNN architecture can lead to a purely transition-based model. If this model is normalized locally it is actually very similar to the IOHMM proposed by Bengio and Frasconi in [BF96].

6.4.1 The Transition-Based HNN Model

Based on the general formulation of the HNN in section 6.2 a purely transition-based model is easily obtained by simply setting all match "networks" to the trivial mapping: $\phi_i(\boldsymbol{s}_l;\boldsymbol{v}^i) = 1.0$. For this architecture we have the following R-functions,

[8] The algorithms in this thesis have been implemented in the C programming language. In C (as well as many other high-level programming languages) the evaluation of the exponential function is computationally very expensive. Look-up-tables can speed up the calculations significantly, but at the cost of a loss in accuracy.

$$R(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} R(\boldsymbol{x}, \boldsymbol{\pi}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_{l} \theta_{\pi_{l-1}\pi_l}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^{\pi_{l-1}}) \tag{6.14}$$

and (for single-label-states)

$$R(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi} \in \Pi_y} R(\boldsymbol{x}, \boldsymbol{\pi}|\boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi} \in \Pi_y} \prod_{l} \theta_{\pi_{l-1}\pi_l}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^{\pi_{l-1}}). \tag{6.15}$$

Because a probabilistic interpretation of the HNN is ensured by global normalization it is not required that the outputs of the transition networks sum to one. However, if we force the transition network outputs to sum to one by *e.g.*, a softmax output function the transition-based HNN has a very interesting probabilistic interpretation. Let us assume that the transition networks are normalized. The conditional probability of the labeling can then be expressed as a product of "local" observation context dependent probabilities. To see this, we write the conditional probability of the complete labeling as follows,

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{y}, \boldsymbol{\pi}|\boldsymbol{x}, \boldsymbol{\Theta}) \tag{6.16}$$

For $P(\boldsymbol{y}, \boldsymbol{\pi}|\boldsymbol{x}, \boldsymbol{\Theta})$ we find

$$P(\boldsymbol{y}, \boldsymbol{\pi}|\boldsymbol{x}) = P(y_L|\boldsymbol{\pi}_1^L, \boldsymbol{x}, \boldsymbol{y}_1^{L-1}) P(\pi_L|\boldsymbol{\pi}_1^{L-1}, \boldsymbol{x}, \boldsymbol{y}_1^{L-1}) P(\boldsymbol{y}_1^{L-1}, \boldsymbol{\pi}_1^{L-1}|\boldsymbol{x}) \tag{6.17}$$

and iterating this we end up with

$$P(\boldsymbol{y}, \boldsymbol{\pi}|\boldsymbol{x}) = \prod_{l} P(y_l|\boldsymbol{\pi}_1^l, \boldsymbol{x}, \boldsymbol{y}_1^{l-1}) P(\pi_l|\boldsymbol{\pi}_1^{l-1}, \boldsymbol{x}, \boldsymbol{y}_1^{l-1}). \tag{6.18}$$

If we assume that the model is first order Markovian and that the conditional dependency on the causal label sequence $\boldsymbol{y}_1^{l-1}$ can be ignored, then $P(\boldsymbol{y}, \boldsymbol{\pi}|\boldsymbol{x}, \boldsymbol{\Theta})$ can be expressed as a product of "state-local" observation conditional probabilities,

$$\begin{aligned}
P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}) &= \sum_{\boldsymbol{\pi}} \prod_{l} P(y_l|\pi_l, \boldsymbol{x}, \boldsymbol{\Theta}) P(\pi_l|\pi_{l-1}, \boldsymbol{x}, \boldsymbol{\Theta}) \\
&\approx \sum_{\boldsymbol{\pi}} \prod_{l} P(y_l|\pi_l, \boldsymbol{s}_l, \boldsymbol{\Theta}) P(\pi_l|\pi_{l-1}, \boldsymbol{s}_l, \boldsymbol{\Theta}) \\
&\approx \sum_{\boldsymbol{\pi}} \prod_{l} P(y_l|\pi_l, \boldsymbol{x}_l, \boldsymbol{\Theta}) P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}).
\end{aligned} \tag{6.19}$$

The last approximation in (6.19) defines the *synchronous* IOHMM[9] for complete label sequences introduced by Bengio and Frasconi [BF96]. In the IOHMM, the conditional probability of the labeling is thus expressed in terms of "state-local" label and transition probabilities which are conditioned only on the current observation $\boldsymbol{x}_l$. If the labeling is incomplete, a similar derivation can be done based on the discussion in appendix B, and the result defines the so-called *asynchronous* IOHMM [BB96].

[9]Note that the sequence $\boldsymbol{y}$ can in principle consist of continuously valued scalars or vectors. This constitutes a slightly more general formulation than the one considered here [BF96]. Also note that Bengio and Frasconi chose to condition the transition probabilitie on $\boldsymbol{x}_l$ (or $\boldsymbol{s}_l$) whereas the HNN conditions on $\boldsymbol{x}_{l-1}$ (or $\boldsymbol{s}_{l-1}$).

Since $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta})$ is an a posteriori probability it must sum to one for any observation sequence. This implies that

$$
\begin{aligned}
&\sum_{\boldsymbol{y}\in\mathcal{Y}} P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}) = 1 \\
\Updownarrow \\
&\sum_{\boldsymbol{y}\in\mathcal{Y}} \sum_{\boldsymbol{\pi}} \prod_{l} P(y_l|\pi_l, \boldsymbol{x}_l, \boldsymbol{\Theta}) P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}) = 1 \\
\Updownarrow \\
&\sum_{\boldsymbol{\pi}} \left(\prod_{l} P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}) \right) \left(\sum_{\boldsymbol{y}\in\mathcal{Y}} \prod_{l'} P(y_{l'}|\pi_{l'}, \boldsymbol{x}_{l'}, \boldsymbol{\Theta}) \right) = 1 \\
\Updownarrow \\
&\sum_{\boldsymbol{\pi}} \prod_{l} P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}) = 1 \qquad (6.20)
\end{aligned}
$$

because $\sum_{\boldsymbol{y}\in\mathcal{Y}} \prod_l P(y_l|\pi_l, \boldsymbol{x}_l, \boldsymbol{\Theta}) = 1$. This result also holds for incomplete labeling and when the label and transition probabilities are conditioned on a context $\boldsymbol{s}_l$ around $\boldsymbol{x}_l$. The normalization requirement stated in (6.20) can only be ensured if *all* states in the IOHMM are end-states, *i.e.*, if the observation sequence can end in any state even if it has outgoing transitions.

If we now assume that only one label is allowed in each state in the IOHMM the context dependent label probabilities simplify to delta-functions and we find

$$
P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_{l} \delta_{c^{\pi_l}, y_l} P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}\in\Pi_y} \prod_{l} P(\pi_l|\pi_{l-1}, \boldsymbol{x}_l, \boldsymbol{\Theta}). \qquad (6.21)
$$

Comparing (6.21) and (6.15) we see that this IOHMM is very similar to a locally normalized transition-based HNN when the transition networks only take the current observation as input. We also note that the more general IOHMM architecture with observation context dependent label probabilities corresponds to a locally normalized HNN with both label and transition networks. In the paper [KR98] included in appendix F a comparison between CHMMs, HNNs and IOHMMs are given in terms of so-called *graphical models of probabilistic independence networks*. Graphical models have recently gained widespread interest in many research communities for illustrating conditional dependencies between variables in globally and locally normalized probabilistic models. Furthermore, a set of very general probability inference algorithms have been developed for graphical models of which the forward-backward algorithm is a special case. The interested reader is referred to the excellent review of probabilistic independence networks given in the recent paper by Smyth *et al.* [SHJ97].

6.4.2 Notes on Locally Normalized Transition-Based HNNs

The locally normalized transition-based HNN with single-label-states can in principle be trained and decoded in exactly the same way as the more general globally normalized HNN using both match and transition networks. However, because of the local normalization it turns out that the computations are simplified somewhat as described below.

Probability Computation

For the locally normalized transition-based HNN where all states are allowed to be end states we see by comparing (6.14) and (6.20) that,

$$R(\boldsymbol{x}|\boldsymbol{\Theta}) = 1.0 \tag{6.22}$$

for all observation sequences. This implies that the conditional probability of the labeling can be computed in just one forward pass using algorithm 4.1 for complete labeling and algorithm 2.1 on the temporary model for incomplete labeling (see section 4.3),

$$P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}) = R(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\Theta}). \tag{6.23}$$

Thus, for locally normalized transitions we are enforcing discrimination between classes locally at the state level. This is different from the sequence level discrimination enforced in globally normalized HNNs.

Contrary to the case of globally normalized models the forward and backward variables can be given probabilistic interpretations for locally normalized transition based HNNs. For complete labeling we have [BF96]

$$\tilde{\alpha}_i(l) = P(\boldsymbol{y}_1^l, \pi_l = i|\boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}) \tag{6.24}$$

and

$$\tilde{\beta}_i(l) = P(\boldsymbol{y}_{l+1}^L|\pi_l = i, \boldsymbol{x}_l^L, \boldsymbol{\Theta}). \tag{6.25}$$

Thus, $\tilde{\alpha}_i(l)$ is the probability of matching the partial complete label sequence $\boldsymbol{y}_1^l$ and being in state i at time l given the partial observation sequence $\boldsymbol{x}_1^{l-1}$. Similarly, $\tilde{\beta}_i(l)$ is the probability of matching the rest of the label sequence $\boldsymbol{y}_{l+1}^L$ given $\boldsymbol{x}_l^L$ and that we are in state i at time l.

The fact that $R(\boldsymbol{x}|\boldsymbol{\Theta}) = 1$ is a result of interpreting the forward variable in algorithm 2.1 as the probability of being in state i at time l given the observations up to time $l-1$ [BF96],

$$\alpha_i(l) = P(\pi_l = i|\boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}). \tag{6.26}$$

Hereby, $R(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_i \alpha_i(L) = 1$ because all states are end states. The backward variable corresponding to $\alpha_i(l)$ is trivial; $\beta_i(l) = 1$ for all l and i. This is easily seen by applying backward algorithm 2.5 to the locally normalized transition-based HNN.

Decoding

Equation (6.26) implies that the "all-path" forward-backward decoder discussed in section 2.7.4 now effectively computes the a posteriori probability of label $y_l = c$ at time l as,

$$P(y_l = c|\boldsymbol{x}_1^{l-1}, \boldsymbol{\Theta}) = \sum_i \delta_{c^i,c}\alpha_i(l) \tag{6.27}$$

because the expected number of times state i is visited at time l is

$$n_i(l) = \frac{\alpha_i(l)\beta_i(l)}{\sum_{i'} \alpha_{i'}(l)\beta_{i'}(l)} = \frac{\alpha_i(l)}{\sum_{i'} \alpha_{i'}(l)} = \alpha_i(l). \tag{6.28}$$

Thus, the forward-backward decoder is now effectively a "forward" decoder where the most probable label at time l is selected based *only* on the observations up to time l. This is equivalent to the decoder proposed for the IOHMM in [BF96] for the special case of single-label-states.[10] For applications such as financial time-series prediction it is a very reasonable decoder because we do not know the value of future observations. However, in speech recognition or biological sequence analysis we usually know the entire observation sequence a priori, and the "forward" decoder thus seems to be a poor choice because it does not make use of the information contained in the rest of the sequence. For such applications a decoder like the Viterbi or N-best decoder might seem more appropriate. Unfortunately, it turns out the these decoders also do not use "future" observations. Thus, for the Viterbi decoder the state at time l in the optimal path can be computed based only on the observations up to time l; $\hat{\pi}_l = \operatorname{argmax}_i \alpha_i^*(l)$. That is, the information contained in future observations is "summed out" because the locally normalized transition-based HNN models a posteriori distributions by products of locally normalized probabilities.

The fact that all states are allowed to be end states implies that the last observation in any sequence can be modeled in any state. Thus, the last label y_L corresponding to $\boldsymbol{x}_L$ is allowed to be *any* of the possible classes. Whilst this might be appropriate for some applications it is not always reasonable in speech recognition (or biological sequence modeling) because the utterance usually ends in the same sound class; silence. This can have a negative impact on the performance of the IOHMM as a speech recognizer because the ability of HMMs, and IOHMMs in particular, to represent long-term dependencies implies that the entire label sequence may be affected by not constraining y_L to be *e.g.*, the silence class. In practice, however, it does not seem to be a serious problem, see the next chapter.

CML Estimation

If the transition networks are normalized, CML estimation is simplified somewhat as we can now express the conditional likelihood by a non-rational function of "state-local" probabilities. It is not difficult to see that the locally normalized transition-based HNN can be trained by an (G)EM algorithm with an auxiliary function defined by (compare to (B.4) in appendix B) [BF96],

$$\begin{aligned} \mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{\Theta};\boldsymbol{\Theta}^{(t)}) &= E_{\boldsymbol{\pi}}[\log P(\boldsymbol{y},\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta})|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}^{(t)}] \\ &= \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}^{(t)}) \log P(\boldsymbol{y},\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta}) \\ &= \sum_{lij} m_{ij}^{(t)}(l) \log \theta_{ij}(\boldsymbol{s}_{l-1};\boldsymbol{w}^i), \end{aligned} \tag{6.29}$$

For discrete observations and transition "networks" implemented by look-up-tables the model can be trained by a "pure" EM algorithm where the context dependent transitions are updated by reestimation formulas similar to (2.44). In the more general case of non-linear transition networks we have to rely on a GEM algorithm, in which the auxiliary function is not maximized exactly, but only increased by an iterative optimization method.

In order to do GEM training of a model with non-linear transition networks, we have to store the expected $m_{ij}(l)$-counts for *all* training sequences before iterative maximization

[10] In the general case of multiple-label-states, the delta-function $\delta_{c^i,c}$ is just replaced by the probability $P(y_l = c|\pi_l = i, \boldsymbol{s}_l)$ of label $y_l = c$.

of the auxiliary function can be initiated. As discussed in section 6.3.1 the memory complexity associated with storing the counts for just one sequence of length L scales as $\mathcal{O}(LN^2)$ for an ergodic transition-based HNN with N^2 transitions. Therefore, storing the counts for all sequences would require enormous memory resources. For this reason, the GEM approach has not been used in this work. Instead, the negative log conditional likelihood is minimized directly using a gradient method as usual. Since $R(\boldsymbol{x}|\boldsymbol{\Theta}) = 1$ the gradient of $\mathcal{L}(\boldsymbol{\Theta}) = -\log P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta})$ w.r.t. a weight w_k^i in the transition network assigned to state i is simply

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_k^i} = -\frac{\partial \log R(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\Theta})}{\partial w_k^i} = -\sum_l \frac{m_{ij}(l)}{\theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)} \frac{\partial \theta_{ij}(\boldsymbol{s}_{l-1}; \boldsymbol{w}^i)}{\partial w_k^i}. \tag{6.30}$$

Hardwired Duration Modeling

A serious problem of local normalization is related to cases where we wish to apply a "hardwired" duration model. Consider for example a task where the class submodels have a minimum duration of three observations as shown in figure 2.8. Also assume that at time l some path has just entered a particular class submodel with label c. Then, for softmax normalized transitions, the label at time $l+1$ and $l+2$ will be c with probability one because the path through this submodel cannot be terminated due to a low transition probability. That is, we will either remain in the same state or go to the next state in the submodel because of the softmax normalization. On the other hand, if the transitions are not required to normalize to one it will indeed be possible to terminate a path through a particular submodel because all outputs of the transition networks are allowed to be close to zero. Thus, hardwired duration modeling is not suitable for locally normalized transition-based models. Even for submodels based on a single state the normalized transitions can lead to poor performance because any path is allowed to pass through any submodel at any time.

6.5 Comparison to other HMM/NN Hybrids

In this section we briefly review a number of recently proposed HMM/NN hybrids. Since the literature in this field is very large no attempt is made to cover all proposed variations. Instead focus is put on mainstream HMM/NN hybrids that have some similarity to the HNN.

There are in general two ways of doing hybrid modeling. One approach is to decouple the neural network(s) from the HMM during training, *i.e.*, to train the HMM and neural network(s) separately and only combine them for recognition. Alternatively, one can jointly estimate all parameters of both the HMM and the neural network(s) (as done in the HNN) by optimizing a "global" training criterion. The approaches described in the first two subsections below belong to the former category.

6.5.1 Neural Networks for Scaled Likelihood Estimation

Some of the first attempts for combining separately trained neural networks and HMMs were done simultaneously by researchers at the International Computer Science Institute, University of California at Berkeley in USA and at the Department of Engineering, University of Cambridge in England. A summary of the early work done at Berkeley University can be found in the book by Hervè Bourlard and Nelson Morgan [BM94]. The most comprehensive summary of the Cambridge system is given in the book chapter [RHR96]. In these approaches an MLP (Berkeley) or recurrent network (Cambridge) is trained separately to classify frames of speech into a set of phonemes. After training, the MLP estimates the posterior probability of the different phonemes $P(c|\boldsymbol{s}_l)$, $c \in \mathcal{C}$, given a symmetric context input $\boldsymbol{s}_l = \boldsymbol{x}_{l-K}^{l+K}$. The recurrent network also estimates phoneme posteriors after training, but because of the feed-back connections in this network the context is now effectively equal to the causal observation sequence, $\boldsymbol{s}_l = \boldsymbol{x}_1^l$.

The estimate of posterior probabilities can be turned into "scaled likelihoods" by dividing by the prior probabilities of the phonemes $P(c)$, $c \in \mathcal{C}$,

$$\frac{P(c|\boldsymbol{s}_l)}{P(c)} = \frac{P(\boldsymbol{s}_l|c)}{P(\boldsymbol{s}_l)}. \tag{6.31}$$

The a priori probabilities are estimated by the relative frequencies of the phonemes as observed in the training set and the scaled likelihoods *replace* the standard match probabilities in an HMM during decoding. Typically, "hardwired" minimum duration phone models (see figure 2.8) are used such that the (single) match distribution in each phone model is replaced by the associated scaled network output. Compared to standard HMMs based on minimum duration phone submodels, the scaled likelihood hybrids have yielded better performance on a variety of speech recognition tasks [RHR96, Rob94, BM94, RMB$^+$94]. For example, in [RMB$^+$94, RMCF92] a hybrid with an MLP containing more than one million weights yielded roughly half the error rate of an equally complex HMM on a continuous speech recognition task.

Many parts of the recurrent network hybrid developed at Cambridge University have been optimized through several years of intense research for the recognition of phonemes in the TIMIT database [RF88, THPF90, TF90, Rob91, Rob94, HRRC95]. It is today believed to be one of the best performing recognizers on the TIMIT phoneme recognition task [Rob94]. The Cambridge hybrid has been named ABBOT and a demo version can be obtained from the group's web-site.[11]

Several variations of the scaled likelihood framework is of course possible. For example, in [SL92] a different kind of network known as a *radial basis function network* was used for estimating the scaled likelihoods. However, they found that the radial basis function networks did not perform as well as MLPs which was also observed in [RBFC92]. Franzini *et al.* [FLW90] employed an Elman type recurrent network (see *e.g.*, [HKP91]) which used a symmetric context $\boldsymbol{s}_l$ and the previous 10 hidden unit activations as input. The network was trained to estimate the posterior probability of *state pairs* given by a forced Viterbi alignment in a Mealy form HMM. Since the match distributions are associated with state pairs in the Mealy HMM the outputs of this network can be used for replacing the match distributions in a Mealy HMM. However, instead of scaling the neural network outputs by the corresponding prior probabilities of state pairs, Franzini *et al.* used the unscaled outputs directly during Viterbi decoding.

In [MHJ96] the so-called *Stochastic Observation HMM* (SOHMM) was proposed as a way of generalizing the scaled likelihood approach. The basic idea in the SOHMM is to assign a discrete probability distribution over the different classes to each state and view this distribution as a vector. This is similar to the match distribution in standard discrete HMMs, but the SOHMM differs by using *stochastic observation* input vectors instead of

[11] http://svr-www.eng.cam.ac.uk

discrete observations. The elements of the stochastic observation vector are typically estimates of class posterior probabilities and the match probability at time l in state i is given by the dot-product of the stochastic observation input vector and the observation probability vector associated with state i. Given the stochastic observation vectors the SOHMM can be trained by an EM algorithm with reestimation equations similar to those for the standard HMM [MHJ96]. The SOHMM can be combined with a separately trained neural network by simply using the network output vector as observations. Thus, in [MHJ96] the Cambridge recurrent network output vector of phoneme posterior probabilities was used as stochastic observations in an SOHMM during training. This hybrid achieved slightly higher recognition accuracy on a continuous speech recognition task compared to the Cambridge recurrent network hybrid. Note that the SOHMM also has interesting applications in biological sequence modeling where probability *profiles* for each sequence position, calculated by aligning familiar sequences, are known to contain far more information than the discrete observations corresponding to a single sequence, see *e.g.*, [RK96a]. In this case the stochastic observation vectors would be the probability profile for each position and the match probability vector in each state of the SOHMM would then represent a "template" profile.

One drawback of the above approaches is that the complete labeling (*i.e.* the phoneme segmentation) for the training set is required in order to train the neural networks. If only the incomplete labeling (*i.e.* the phonetic transcription) is available, the hybrids must be trained in a two step iterative procedure: First a complete labeling is obtained by performing a forced Viterbi alignment for each training sequence based on the current network and HMM. Then this complete labeling is used for training the networks. This procedure is repeated until the labeling does not change anymore and is analogous to the method described for standard HMMs in chapter 2. Another drawback is that the HMM and neural network are trained independently to solve different tasks: the HMM is trained to recognize phoneme *transcriptions* whereas the neural network is trained to do phoneme classification on a frame-by-frame basis. This is different from the HNN where all parameters including the neural network weights are trained by minimizing a "global" training criterion.

6.5.2 Neural Networks for Vector Quantization

Instead of using neural networks to estimate scaled likelihoods, some researchers have proposed to replace the vector quantizer front-end in a discrete HMM by a neural network. The idea is to train a neural network separately to assign discrete "VQ-labels" to the speech frames. After training, the VQ-label stream from the neural network is used as observations in a discrete HMM.

Le Cerf *et al.* [LCMC94] trained an MLP to classify speech feature vectors into a set of phoneme labels. After training, this network was used for assigning phoneme labels to the input speech frames. The phoneme "VQ-label" sequence was then used as an observation sequence in a standard discrete HMM. Instead of using a single MLP, Le Cerf *et al.* proposed to use different MLPs for different features (cepstral, delta-cepstral *etc.*). By assuming independence[12] between the different MLPs, the VQ-label for any speech frame was selected by the largest product of MLP outputs. On a speaker independent Flemish digit database the multiple MLP approach was shown to outperform both the

[12] The independence assumption is equivalent to using an HMM with multiple independent streams, see section 2.8.

model with only one MLP and an HMM with multiple standard Euclidean codebooks.

An alternative to MLPs is the so-called *Learning Vector Quantizer* network [KKLT92, KBC88] which is a particular neural network architecture trained by an *unsupervised winner-take-all* algorithm to group input vectors into a predefined number of clusters. For speech recognition the clusters are typically related to different phoneme classes and learning vector quantizers can produce VQ-labels in much the same way as the MLPs above. There are several successful applications of learning vector quantizers as front-ends for standard HMMs, see *e.g.*, [Tor94, MK91].

A direct comparison between MLPs for vector quantization and MLPs for estimating scaled likelihoods was given by Fontaine *et al.* [FRL$^+$96]. They concluded that the latter approach is superior in terms of recognizing isolated German words.

6.5.3 Adaptive Input Transformations

Instead of training the HMM and neural networks separately, several authors have proposed architectures where all parameters are estimated simultaneously as in the HNN. One way to achieve this is by using the neural networks for *adaptive input transformations* where the continuous observation vectors are passed through one or more networks before entering the Gaussian mixture match distributions. Using the chain rule it is easy to see that the weights of these neural networks can be trained by backpropagating errors calculated by the HMM just as in the HNN. The result is a non-linear input transformation of the feature vectors which is adapted during training.

When training an HMM with an adaptive input transformation by ML estimation one must be aware of the ability of such models to converge to trivial solutions. Since likelihoods are generally unbounded functions, an unconstrained maximization can lead to an infinite likelihood [DH73, BDMFK92] if the outputs of the neural network converge to constant values. In such cases the HMM training will lead to diverging Gaussian mixture distributions because the mean vectors converge to the constant network output and because the variances converge to zero. The cure is to constrain the covariance matrices or alternatively to use discriminative training like MMI or CML.

In [BDMFK92] a cascade of MLPs was used to perform a "global" adaptive input transformation for an HMM. Although this system was trained by the ML criterion the above problem of infinite likelihood was not observed for this application. For a phoneme recognition task this hybrid obtained significantly better performance than a standard HMM. It was also shown that the jointly trained system performed better than when training the neural networks independently of the HMM to perform specialized mappings of the feature vectors. For the recognition of broad phoneme classes good results were obtained in [Joh94] by jointly training an MLP input transformation and a Gaussian mixture HMM to maximize the MMI/CML criterion. Valchev *et al.* [VKY93] also used MMI for joint training, but a recurrent network was used instead of an MLP. The recurrent network approach was later extended to use separate networks for different match distributions in the HMM *e.g.*, one network for each phoneme model [Val95].

The HNN approach without transition networks is similar to the idea of adaptive input transformations, but instead of retaining the computationally expensive mixture densities we *replace* these by the output of match networks. This is possible because the model is normalized globally. A similar idea was recently used in the CML estimated "LeRec" hybrid proposed by Bengio *et al.* [BLNB95]. LeRec is very similar to the HNN without transition networks, but instead of using state local match networks LeRec uses one "global" match network with as many outputs as there are states in the model. LeRec has been successfully applied to on-line handwriting recognition and is today part of a commercial system for reading bank checks [LCBB97].

6.5.4 The Discriminant HMM/NN Hybrid

The *discriminant HMM/NN hybrid* introduced in [BKM94] is a transition-based model similar to the single-label-state transition-based HNN and IOHMM. As for the HNN this hybrid is trained by the CML criterion[13] to model the a posteriori probability of the labeling.

In the discriminant HMM the probability of the labeling is expressed as a product of an acoustic model contribution and a language model contribution,

$$\begin{aligned} P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\Theta}) &= \sum_{\boldsymbol{\pi}} P(\boldsymbol{y},\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta}) \\ &= \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta})P(\boldsymbol{y}|\boldsymbol{\pi},\boldsymbol{x},\boldsymbol{\Theta}) \\ &\approx \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta})P(\boldsymbol{y}|\boldsymbol{\pi},\boldsymbol{\Theta}) \end{aligned} \tag{6.32}$$

The last approximation is based on the assumption that given the sequence of states, the probability of the labeling is independent of the observation sequence. The acoustic model is defined by the first term ($P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta})$) and the language model by the second term ($P(\boldsymbol{y}|\boldsymbol{\pi},\boldsymbol{\Theta})$). As for the HNN, the acoustic model probability in the discriminant HMM can be written as a product of observation context dependent transition probabilities; $P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{\Theta}) = \prod_l P(\pi_l|\pi_{l-1},\boldsymbol{s}_l,\boldsymbol{\Theta})$. For the discriminant HMM/NN hybrid, Bourlard *et al.* proposed to use a "global" MLP with a softmax output function instead of state local networks as in the HNN to estimate these conditional transition probabilities. This big MLP takes the observation context $\boldsymbol{s}_l$ and the previous state distribution as inputs, and outputs estimates of the observation context dependent transition probabilities.

In most work reported for this hybrid, see *e.g.*, [Kon96, KBM96], the language model contribution is assumed only to constrain the sequence of acoustic models according to the observed phonetic labeling. That is, for a training sequence with an associated phonetic labeling the corresponding concatenation of acoustic models is trained and the language model contribution is ignored. This implies that (6.32) can be written in the same form as for the locally normalized transition-based HNN, and the two approaches are therefore very similar except that state local transition networks are used in this work.

The first speech recognition experiments with the discriminant HMM/NN hybrid were based on Viterbi training where 0/1 targets for the neural network were found by doing a forced Viterbi alignment. These targets were then used to train the network to minimize *e.g.*, the mean squared error between targets and network outputs. Unfortunately, this approach gave very poor results [BKM94, Kon96] mainly because several possible combinations of previous and next state were not investigated by doing a forced Viterbi alignment. That is, several possible previous and next state combinations were not presented to the neural network during training, thereby leading to an extremely poor generalization ability to utterances not present in the training set. These findings initiated development of the

[13] CML estimation was called MAP estimation by Bourlard *et al.*.

so-called *Recursive Estimation and Maximization of A Posteriori probabilities* (REMAP) algorithm which, instead of 0/1 targets, iteratively reestimates a set of "soft" targets for each possible combination of previous and next state. As proven in [BKM94, Kon96] the REMAP algorithm is a GEM algorithm which is guaranteed to converge to a local maximum of the CML training criterion, even though the networks are trained to minimize a different cost function.

The REMAP trained discriminant hybrid was applied to the recognition of isolated digits and naturally spoken numbers over a telephone line in [Kon96, KBM96]. For both of these tasks a discriminant hybrid with one state per phoneme obtained better performance than a baseline scaled likelihood hybrid. However, the comparison was based on ignoring the language model contribution for both hybrids. If language model information and minimum duration modeling was used in the baseline it clearly outperformed the discriminant HMM/NN hybrid [KBM96].

In [HRB+97] it was argued that the discriminant HMM/NN hybrid can be viewed as a generalization of the scaled likelihood approach discussed above. Thus, instead of training the neural network separately to estimate phoneme posterior probabilities it is possible to train the network on "soft" scaled likelihood targets iteratively estimated by the current HMM and MLP. The idea is to apply a "scaled" forward-backward algorithm [HRB+97] to the current HMM/NN hybrid in order to calculate estimates of the state posteriors $P(\pi_l|\boldsymbol{x}, \boldsymbol{\Theta})$. The priors corresponding to the different HMM states are given by the average of the state posteriors over all training sequences. In each iteration new "soft" targets for the neural network are then simply given by the state posteriors divided by the appropriate priors. As for the REMAP algorithm it can be shown that this approach is also a GEM algorithm. In [SR96] a similar approach was proposed, but the state priors where instead obtained by an initial forced Viterbi alignment.

6.6 Summary

In this chapter we have introduced a very general framework called a Hidden Neural Network (HNN) for hybrids of HMMs and neural networks. The major advantages of the HNN include the ability to use observation context as input, the probabilistic interpretation ensured by global normalization and the flexibility of the architecture. We have also described a number of similarities of particular HNN architectures to both standard HMMs and other popular HMM/NN hybrids. In particular, it was found that the HNN can be used as a purely transition-based model and that this particular architecture contains many of the ideas in the the recently proposed IOHMM and in the discriminant HMM/NN hybrid.

A potential problem in using the HNN (as well as any other hybrid) is the computational complexity. Contrary to standard HMMs, it is not the added complexity of discriminative training compared to ML estimation that is the main concern for the HNN. Rather, it is the added complexity due to neural network related computations. However, neural networks and multi-layer perceptrons in particular are highly parallel architectures and the HNN is therefore well suited for implementation on parallel computers.

In continuation of the CHMM evaluation on the broad phoneme class task presented in chapter 5 the next chapter will give an evaluation of the HNN on this task. Because of the limited size of this task it has been possible to evaluate a number of the ideas presented in this chapter within reasonable time.

Chapter 7

HNNs for Broad Phoneme Recognition

This chapter gives an evaluation of three different HNN architectures on the TIMIT broad phoneme class task introduced in chapter 5. In the first configuration, which is considered in section 7.2, each state in the HNN used a match network and standard transition probabilities. This architecture is somewhat similar to a scaled-likelihood hybrid. Recall, however, that the HNN uses separate networks in each state and that all parameters are trained jointly to maximize the conditional likelihood. The other two HNN architectures were motivated by the recent literature on purely transition-based systems (*e.g.*, [BKM94, MBGH96]). To asses the importance of focusing on the transitional regions of the speech signal two different HNN architectures using transition networks will be discussed in section 7.3; a purely transition-based HNN and a "mixed" HNN. In the transition-based HNN all match networks were set to the trivial mapping $\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i) = 1$ for all i, l and transition networks were used for estimating observation context dependent transitions. The "mixed" HNN contained both match and transition networks.

The discrete CHMM results presented in chapter 5 serve as a baseline for the HNN experiments.[1]

7.1 Experimental Setup

For the HNN experiments reported in this chapter the same datasets and the same preprocessor as described in chapter 5 were used. Note, however, that no vector quantization was needed since the HNN can directly use the continuously valued feature vectors as input. Similarly, the three-state submodel topology used in the CHMM experiments was also adopted for the HNNs using match networks and standard transitions. For the transition-based HNNs a simpler topology with only one state per class was also evaluated.

Because of the neural network related computations in the HNN the computational complexity is considerably larger for this model than for the CHMM. Thus, training times for HNNs containing simple match or transition networks without hidden units were of the order of 3-4 CPU hours whereas more complex networks lead to training times typically around 1-2 CPU days. This computational demand made it very difficult to evaluate a large number of models or to select stepsizes or pruning thresholds by trial-and-error. Therefore, the training strategy developed for the CHMM was adopted for the HNN even

[1] The HNN use the continuously valued feature vectors directly as input. Since continuous density HMMs usually perform better than discrete HMMs for speech recognition one can argue that comparing the HNN to a discrete CHMM is unfair. However, since the discrete CHMM evaluated in chapter 5 was shown to outperform a continuous density HMM it serves as a good baseline for the comparison.

though there is a risk that it is not optimal for the HNN. Thus, CML training of the HNN passed through an appropriate initialization of the networks, initial complete label training and finally incomplete label training.

Although the online gradient descent method worked well for the CHMM other methods might be better for the HNN. However, in a set of initial experiments it was found that online gradient descent consistently resulted in faster convergence for HNN training than both batch mode gradient descent and approximative second order methods like QuickProp [Fah88] and Gauss-Newton[2] training. Consequently, HNN training was done by sentence online gradient descent using an initial stepsize of $\eta = 0.01$ for *all* parameters. The stepsize was adapted according to the performance on the validation set as described in chapter 5. The momentum term was $\mu = 0.6$ for all parameters and a maximum of 100 epochs was enforced.

The large number of parameters in some of the HNNs evaluated below lead to considerable overfitting. As for the CHMM we therefore report training and test set accuracies for the model that obtained the best performance on the validation set. Unless otherwise stated all accuracies were computed using an N-best decoder with the same local and global pruning thresholds as for the CHMM experiments ($\gamma_l = 10000$ and $M = 10$).

7.2 Match Networks and Standard Transitions

We will start by considering the HNN with match networks and standard transitions ($\theta_{ij}(\boldsymbol{s}_l; \boldsymbol{w}^i) = \theta_{ij}$), see figure 7.1. The match networks in this HNN were standard multilayer perceptrons with a single output. The output activation functions were asymmetric sigmoid functions yielding outputs in the range $]0;1[$. The input to the networks was the context vector $\boldsymbol{s}_l$ and for a context size of K feature vectors the networks had $26 \times K$ real valued inputs, as the dimension of the cepstral feature vectors was 26. For all hidden units a symmetric sigmoid (tanh) activation function was used. All match networks shared the same input and had the same architecture. The standard transition probabilities were initialized by those from an ML estimated CHMM in all experiments.

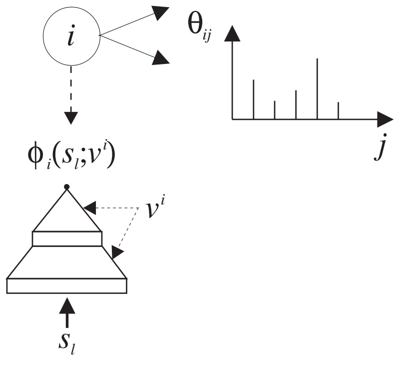

Figure 7.1: An HNN state with a match network and standard transitions.

For the HNN it is even harder than for the CHMM to asses the statistical significance of the results because of the added computational complexity. However, to obtain a feeling

[2] Because of the large number of parameters a diagonal approximation to the Hessian matrix of second order derivatives was necessary in these experiments.

of the variation due to different initial conditions for training, an experiment similar to the one carried out for the CHMM was also done for the HNN. Figure 7.2 illustrates the errorbars on the average training set log conditional likelihood obtained by training an HNN ten times from different random initial parameters. The match networks in this HNN had no hidden units and used only the current observation as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$). Comparing to figure 5.7 we see that the errorbars for the HNN are somewhat larger than for the CHMM. However, the model selection method based on the validation set lead to a deviation in the test set recognition accuracy of no more than $\pm 0.1\%$ from the average accuracy over the ten runs. Thus, for the simple HNN considered in this experiment we can expect the results to be significantly different if they deviate by more than 0.2% for runs with different initial conditions.

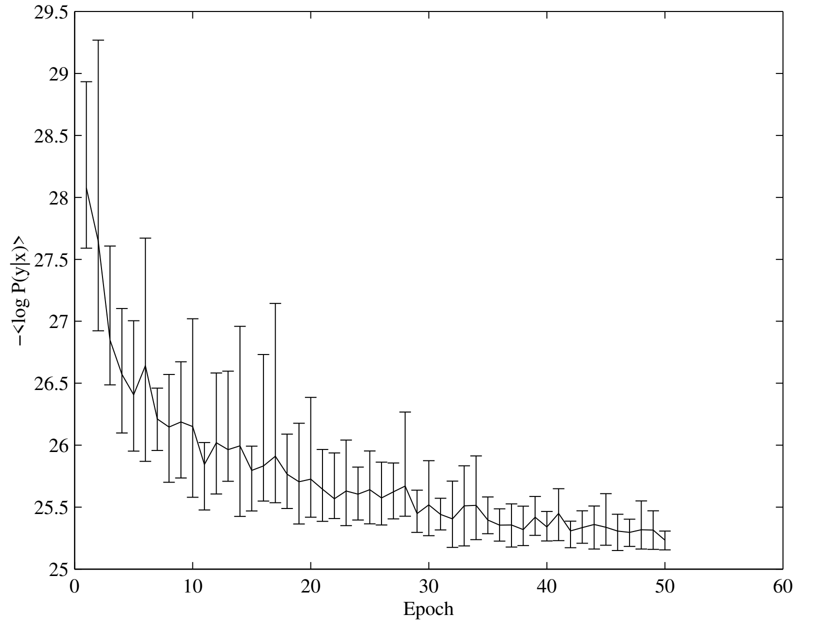

Figure 7.2: Complete label online gradient CML training of an HNN where the match networks are initialized by random weights. All match networks have no hidden units and only use the current observation as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$). The model contains a total of 455 parameters.

7.2.1 Match Network Initialization

Contrary to the match distributions in the CHMM it is not possible to initialize the match network weights in the HNN by the efficient Baum-Welch reestimation algorithm. Instead of just using a set of match networks initialized by random weights, two other initialization methods were tried.

The first initialization method is based on interpreting the match network outputs as a score for how well the current input matches the state to which the network is assigned. Since only single-label-states are considered this corresponds to a score for how well the current input matches the label of that state. A reasonable initialization is therefore to train each match network separately to classify the speech frames into each of the five classes. That is, a match network for *e.g.*, the consonant class is trained to classify the speech frames into two categories: consonant and non-consonant. Since this match network has only one output this can be achieved by backpropagation training where the mean squared error between the network outputs and the observed targets is minimized. The observed targets are 1.0 for speech frames belonging to the consonant class and 0.0 for all other classes, and the network is initialized using small weights in the range $[-1;1]$. After training the same copy of the consonant match network is used in all three states of the consonant model. This procedure is repeated for all five classes. In practice, the easiest way to do the initial classification training is to collect the five networks into one big network for training as illustrated in figure 7.3. Note that if hidden units are used (as shown in the figure) they should not be "shared" between the five outputs.

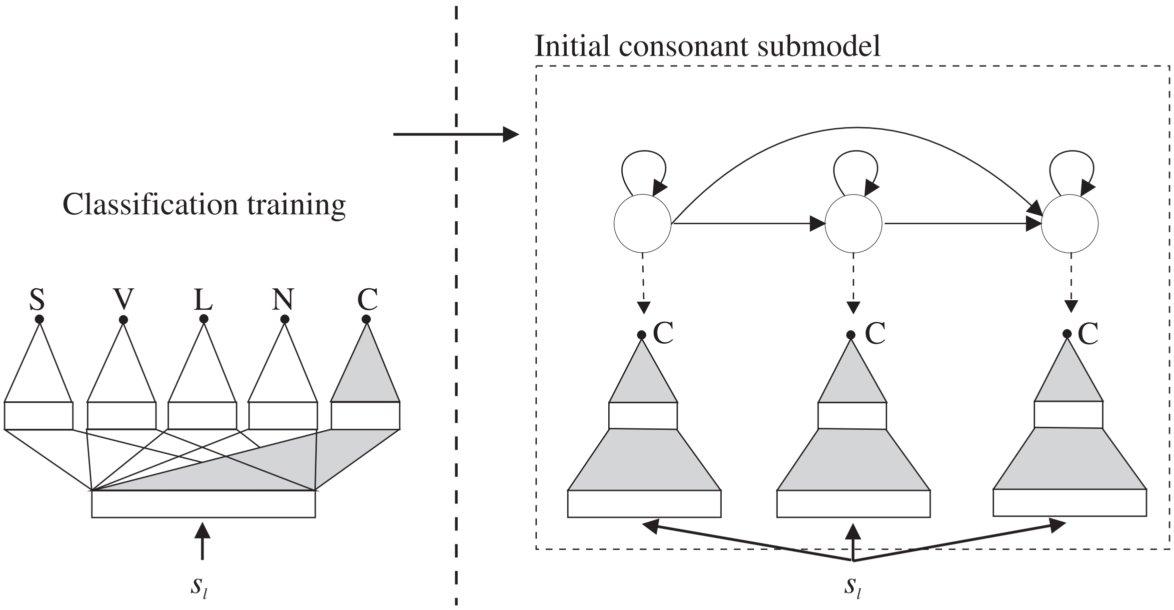

Figure 7.3: Initial classification training of the match networks can be facilitated by collecting the match networks into a large network. Note that all output units use an asymmetric sigmoid activation function, *i.e.*, they are not forced to normalize to one.

The second method is slightly more direct. It is based on observing that the expected m and n counts are delta-functions if a Viterbi approximation is used for both the free-running and clamped phase likelihoods. That is, the difference $m_i(l) - n_i(l)$ is either -1, 0 or +1 depending on whether the optimal path in the free-running and clamped phase visits state i at time l or not. As discussed in the previous chapter, maximizing the CML criterion for the HNN corresponds to backpropagation training of the match networks where the error to backpropagate is $\mathcal{E} \propto (m_i(l) - n_i(l))/\phi_i(\boldsymbol{s}_l; \boldsymbol{u}^i)$. Viterbi CML training is therefore similar to the classification training discussed above. The standard Viterbi algorithm can easily be constrained to search for the optimal path only among the allowed ones. For complete labeling this is ensured simply by replacing all multiplications by $\phi_i(\boldsymbol{s}_l; \boldsymbol{u}^i)$ with $\phi_i(\boldsymbol{s}_l; \boldsymbol{u}^i)\delta_{c^i, y_l}$ in the standard Viterbi algorithm.[3] As for the classification approach the match networks are initialized by small random weights prior to training.

[3]For incomplete label CML training the Viterbi approximation to the clamped phase likelihood is obtained by using the standard Viterbi algorithm on the temporary model constructed according to the observed incomplete labeling.

For both the classification and the Viterbi initialization approach a maximum number of 20 training epochs was used. The set of match networks giving the largest *frame classification rate* on the validation set within the 20 epochs was selected for initializing the HNN.

Initialization	Train	Test
Random weights	77.5	77.1 (49.5)
Viterbi training	78.1	77.3 (59.4)
Classification training	***78.3***	***77.9 (69.9)***

Table 7.1: Recognition accuracies (%Acc) for complete label CML trained HNN for three different ways of initializing the match networks. The accuracies in parentheses are for an HNN using the initialized match networks, but before CML training. All match networks have no hidden units and use only the current observation as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$). The model contains a total of 455 parameters.

Table 7.1 compares the accuracies obtained by complete label CML training of an HNN where the match networks are initialized by three different methods. Surprisingly, the Viterbi initialization gives only an insignificant gain compared to random initialization of the network weights. However, the accuracy obtained after Viterbi initialization but before (all path) CML training is much better than for a random model. This indicates that the Viterbi initialization is actually useful. From the table we see that initializing the networks to classify the frames into the five broad classes yields a considerably higher accuracy than any of the other two methods. Similarly, the accuracy obtained by the classification initialization, but before CML training of the HNN is about 20% higher than for the corresponding random model. The difference between the three initialization methods was observed to be even more pronounced for more complex match networks. Therefore, the classification approach was used for initializing the match networks in all subsequent experiments. The classification training is of course only possible when the complete labeling is available. For the more common case where only the incomplete labeling is known one can use another recognizer to give a complete labeling. Alternatively, one may use an iterative approach which alternates between classification training and forced Viterbi alignment as described for the scaled likelihood hybrid in chapter 6.

7.2.2 Initial Complete Label Training

As for the discrete CHMM we found it beneficial to do complete label CML training before switching to incomplete label training. Table 7.2 compares the result of incomplete label CML training with and without initial complete label training. The initial complete label training was done using all training sequences and was limited to a maximum of 20 epochs. The model that achieved the highest accuracy within the 20 epochs was used for subsequent incomplete label training.

As seen in table 7.2 there is a gain of more than 2% in accuracy by initially training the model using the complete labeling. This indicates that the initial classification training of the networks is not sufficient for the class submodels to attract the right portions of the data during incomplete label training. Initial complete label training was consequently used in all subsequent experiments.

The very simple HNN considered in table 7.2 contains only 455 parameters. Nevertheless it yields an accuracy of 80.8%. This compares well to the best result of 81.3% obtained by the discrete CHMM having approximately nine times more parameters.

Initialization	Train	Test
Scratch	79.4	78.5
Complete label	***81.8***	***80.8***

Table 7.2: Effect on recognition accuracy ($\%Acc$) of initial complete label CML training before switching to incomplete label CML training. All match networks have no hidden units and use only the current observation as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$). The model contains a total of 455 parameters.

7.2.3 Architecture of Match Networks

Having chosen the strategy for initializing and training the HNN with simple match networks we now turn towards finding the "optimal" architecture of the match networks. Below the effect on the accuracy of both the input context size and the number of hidden units in fully connected match networks are investigated.

The Context Input

Below in table 7.3 are shown the recognition accuracies obtained by HNNs using match networks with different sizes and orientations of the context input. The match networks have no hidden units.

Orientation	K	Param.	Train	Test
-	0	455	81.8	80.8
Symmetric	1	1235	82.5	***81.7***
$\boldsymbol{s}_l = \boldsymbol{x}_{l-K}, \ldots, \boldsymbol{x}_{l+K}$	2	2015	82.9	80.9
Left	1	845	82.3	81.4
$\boldsymbol{s}_l = \boldsymbol{x}_{l-K}, \ldots, \boldsymbol{x}_l$	2	1235	***82.9***	81.2
Right	1	845	82.2	81.2
$\boldsymbol{s}_l = \boldsymbol{x}_l, \ldots, \boldsymbol{x}_{l+K}$	2	1235	82.7	81.1

Table 7.3: Recognition accuracies (%Acc) for HNN with different sizes and orientations of the input context to the match networks. No hidden units are used.

Although there is little difference in accuracy for the different input contexts shown in table 7.3, the symmetric input context of one (left and right) frame seems to be slightly better than any of the other context sizes and orientations. It is interesting to note that a symmetric context larger than one frame actually decreases performance on the test set. This indicates that the information contained in frames more than one time step away from the central frame ($\boldsymbol{x}_l$) is "noisy" and therefore leads to slight overfitting of the training data if used as input. All models in the following therefore use a symmetric input context of one frame, *i.e.*, $\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$.

Even with a symmetric context of one frame the HNN contains about one third the parameters of the discrete CHMM. Nevertheless, it obtains an accuracy of 81.7% compared to 81.3% for the CHMM. Note that the match networks without hidden units actually just implement linear weighted sums of the feature vector elements (passed through a sigmoid function).

The Number of Hidden Units

The effect of using hidden units in the match networks is shown in table 7.4. It is seen that a few hidden units improve the accuracy on the test set significantly, but that they also introduce overfitting. Thus, for more than ten hidden units the test set accuracy drops below the best result of 83.8% obtained by the model with ten hidden units in all match networks. For ten hidden units the model contains 12065 parameters.

Hidden	Param.	Train	Test
0	1235	82.3	81.7
5	6065	85.8	82.5
10	12065	89.7	***83.8***
15	18065	***95.9***	83.0

Table 7.4: Effect on recognition accuracy ($\%Acc$) of using hidden units in the match networks. All match networks use a symmetric input context of one frame ($\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$).

For the model with ten hidden units the Viterbi decoder accuracy on the test set is only 81.9% or roughly 2% lower than for the N-best decoder, see table 7.4. Thus, several paths contribute to the optimal labeling in the HNN as illustrated by the state a posteriori plot in figure 7.4. The HNN obtains an N-best accuracy of 93.5% for the utterance in the figure which is approximately 15% higher than that of the CML trained discrete CHMM and 31% higher than that of the ML trained CHMM. Part of this gain is due to a better ability of the HNN to locate and recognize nasals and liquids, see figure 7.4.

For a test set utterance figure 7.5 shows the output from each of the five classification trained match networks used for initializing the HNN. Even though these networks are trained independently they are capable of discriminating fairly well between the five classes. After CML training the output of the match network in the center state of each submodel is quite different from that of the initial networks as seen from figure 7.5. Thus, the networks in the center states have learned a task that is different from classifying speech frames into the five classes. Interestingly, the output from the match network in the center state of both the consonant and nasal model is very close to 1.0 at the end of the utterance even though the associated frames clearly belong to the silence class. Similarly, the output of the network in the center state of the silence model drops below 0.5 towards the end of the utterance. However, in the HNN it does not really matter what the output of the match networks in the center states are at the utterance boundaries because we have constrained the HNN to begin and end any utterance in the silence submodel. That is, there are no possible paths starting or ending in the consonant submodel (or any other non-silence submodel). In fact, because the networks in the HNN are jointly trained to maximize the conditional likelihood it is no longer possible to interpret the output of each of the match networks independently. This fact is more clearly illustrated in figure 7.6 showing the output of the three match networks assigned to the first, middle and last state, respectively, of the consonant and nasal submodel. From this figure we see that the model has learned to distribute the recognition task among the different match networks. Consider for example the plot for the consonant submodel in figure 7.6. The network in the first state of this submodel only gives a large output (close to 1.0) for frames in a close region around the consonant onset boundaries. Thus, it acts so as to "filter" the possible paths that can pass through the consonant submodel. As long as the "filter" network gives a low output it is not possible to enter the consonant submodel and the output of

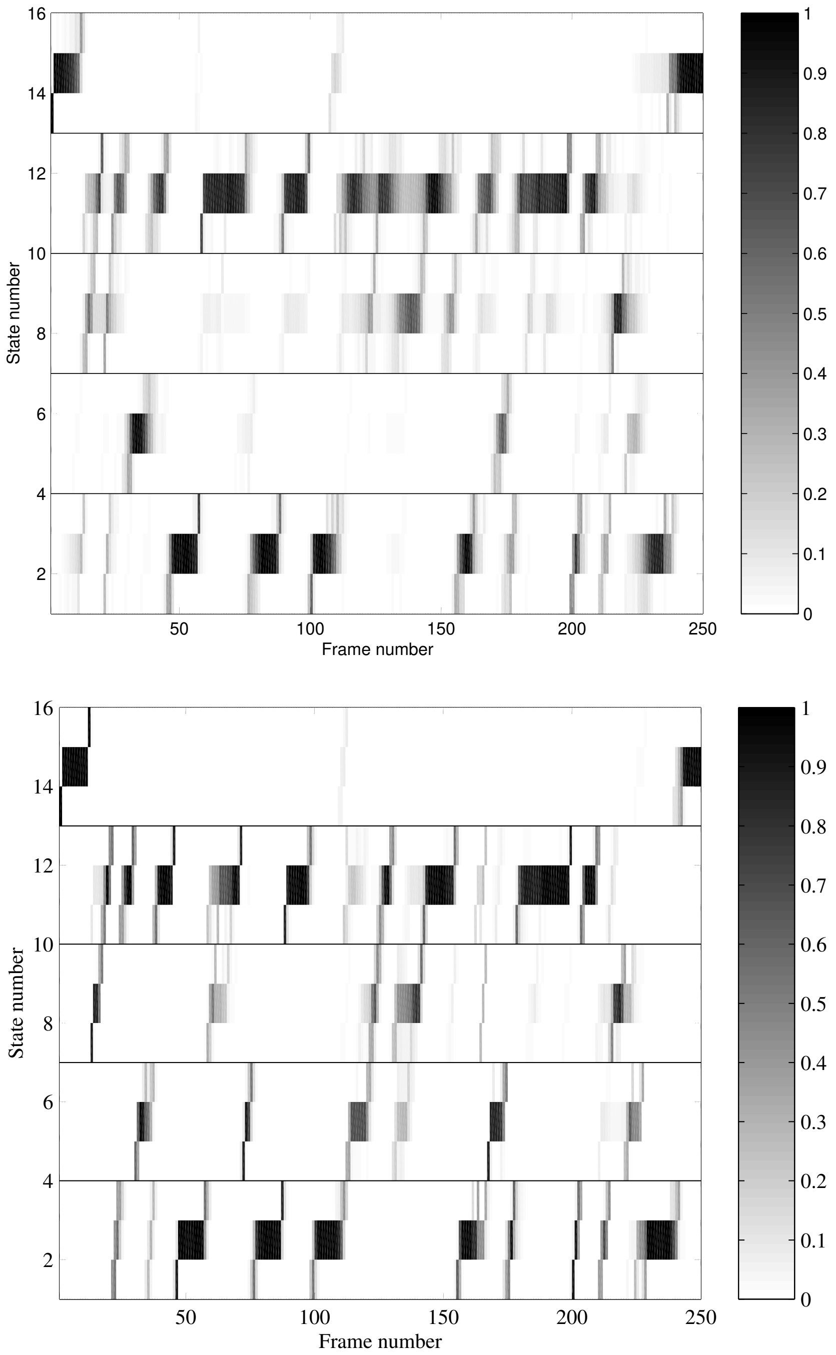

Figure 7.4: Graytone plot of state posteriors $n_i(l) = P(\pi_l = i|\boldsymbol{x}, \boldsymbol{\Theta})$ for the utterance "But in this one section we welcomed auditors" (TIMIT id: si1361). States 1–3 belong to the consonant model, 4–6 to the nasal model, 7–9 to the liquid model, 10–12 to the vowel model and 13–15 to the silence model. ***Upper panel:*** The CML trained discrete CHMM (same as figure 5.9). Sentence accuracy: $Acc = 78.1\%$. ***Lower panel:*** The CML trained HNN. Sentence accuracy: $Acc = 93.5\%$.

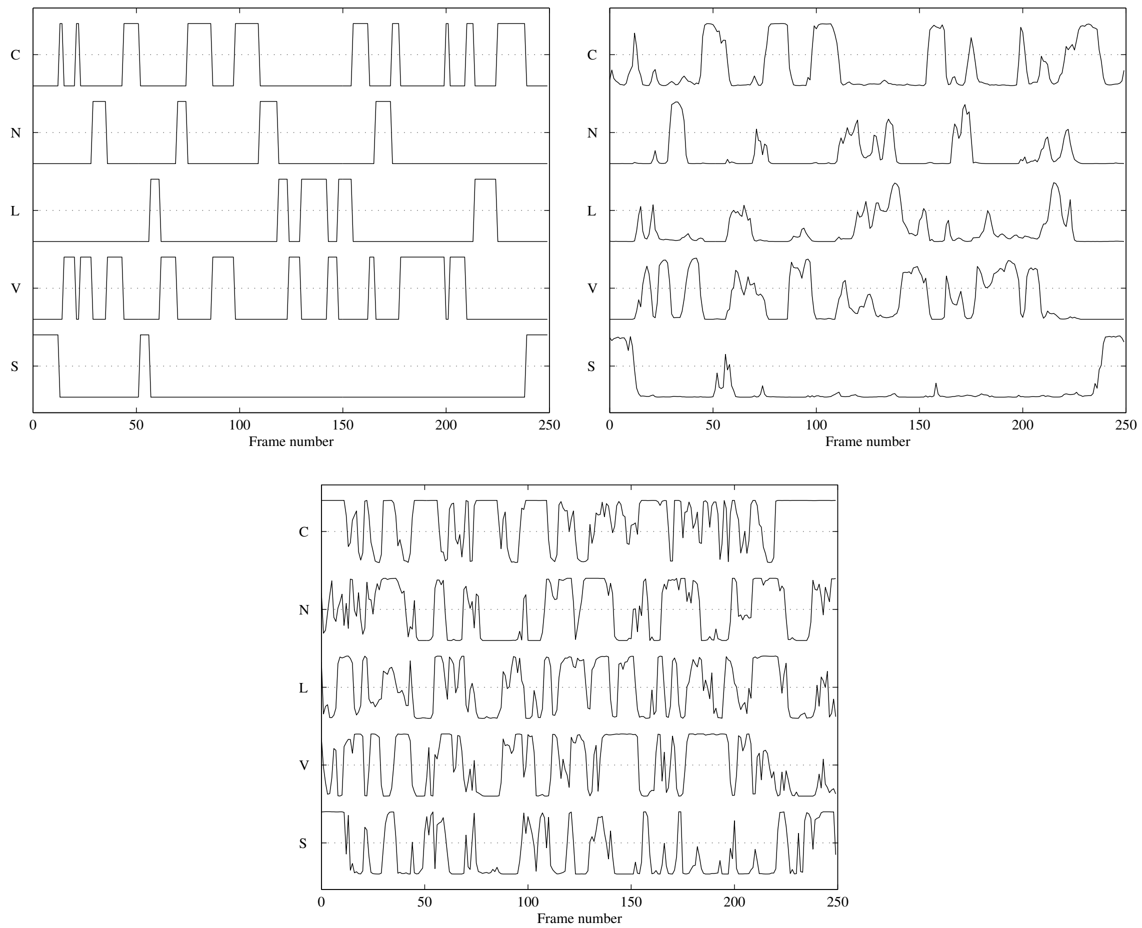

Figure 7.5: Output from match networks in center states of the submodel for each class for an example sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). ***Upper left panel:*** Observed labeling. ***Upper right panel:*** Output before CML training, but after initial classification training. ***Bottom panel:*** Output after CML training. For all three figures the dotted horizontal lines indicate an output level of 0.5.

the match networks in the middle and final states of this submodel can be arbitrary. But if, for a given input frame, the match network in the first state gives a large output a path through the consonant submodel is "opened" and the outputs of the networks in the center and final state now become important. Based on this observation we see that the network in the center state of the consonant submodel gives a large output when we have entered the consonant submodel in good agreement with the observed labeling, see figure 7.6. At some point it is no longer feasible to stay in the consonant submodel and the match network in the last state gives a high output in accordance with the observed labeling. Thus, the recognition of consonants has efficiently been distributed between the three match networks in the consonant submodel. A similar observation can be made for the networks in the nasal submodel.

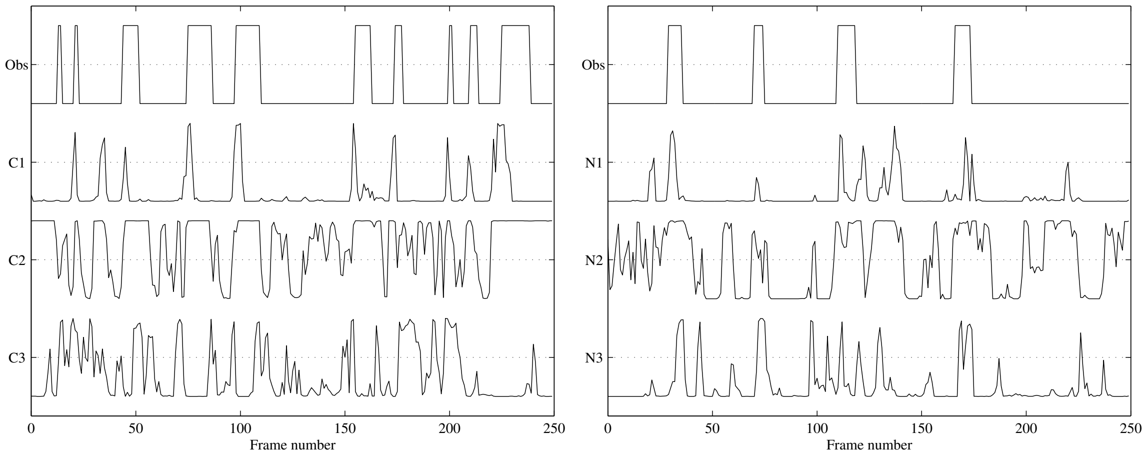

Figure 7.6: Output from the match networks assigned to the first (1), center (2) and last (3) state for an example sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). ***Left panel:*** The consonant model. ***Right panel:*** Nasal model. The top most curve shows the observed complete labeling for the consonant and nasal class, respectively.

7.2.4 Reducing Model Complexity by Weight Sharing

The HNN using match networks with a symmetric input context of one frame and ten hidden units overfits the training data seriously due to the large number of parameters. To reduce overfitting and hopefully improve generalization it was attempted to use a standard weight decay regularizer on the match network weights. However, similar to the regularization experiment for the CHMM a set of initial experiments did not indicate any improvement in performance by using a weight decay. Another approach for improving generalization is to use some automatic method for model complexity optimization like *e.g.*, Optimal Brain Damage [LCDS90]. Optimal Brain Damage is well known to the neural network community and works by removing those parameters of a model that lead to poor a generalization ability according to a so-called saliency measure. However, pruning methods like Optimal Brain Damage have not found widespread use in HMM modeling for speech recognition mainly for two reasons. First of all, they are generally very expensive in terms of computation as they require second order derivative information and because the model often has to be retrained each time a parameter is pruned away. Secondly, because of the normalization constraints on HMM parameters it is difficult to handle the situation where the pruning scheme leads to the removal of *e.g.*, a transition probability. This conceptual problem can, however, be resolved by using globally normalized models. For a very simple speech reading task Optimal Brain Damage pruning was evaluated for a Boltzmann Chain (similar to a globally normalized HMM) in [Ped97].

Even for the simple broad class task the computational complexity is far too large for methods like Optimal Brain Damage because retraining the models used here can take up to one or two days on a fast workstation. An alternative more direct method is to reduce the model complexity a priori by parameter sharing (tying). An obvious method is to tie the match networks within a submodel such that the same match network is used in all three states of a submodel. A more elegant approach is to apply the parameter sharing technique directly to the weights of the match networks instead of to the "outputs". Similar to [RK96b] one can share the weights in the input layer of a match network as illustrated in figure 7.7. By using the same set of weights for each input feature vector, the number of trainable weights becomes less sensitive to the size of the input context. Furthermore, the weight sharing illustrated in figure 7.7 has a very interesting interpretation. It can be viewed as a state specific adaptive input transformation where the high dimensional input feature vector is mapped onto an input vector of lower dimensionality M. Because the transformation is class and state specific the dimension M can be far lower than the dimension of the feature vector.

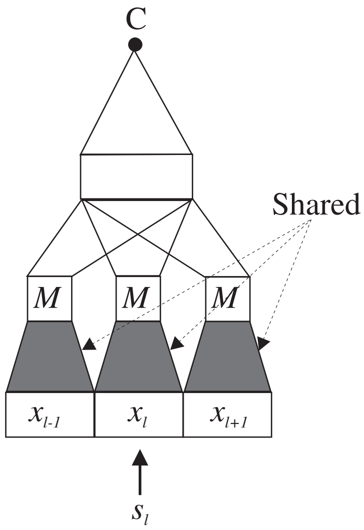

Figure 7.7: Weight sharing in input layer of match networks to reduce complexity.

Model	Param.	Train	Test
No tying	12065	***89.7***	83.8
Tied match	4055	87.4	82.4
Weight sharing, $M = 3$	3020	87.4	83.2
Weight sharing, $M = 5$	4790	88.0	***84.1***

Table 7.5: Effect on recognition accuracy ($\%Acc$) of using parameter sharing. All match networks use a symmetric context as input ($\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$) and have 10 hidden units. The networks with weight sharing have an additional hidden layer of $3M$ units.

Table 7.5 compares the recognition accuracy for HNNs with and without parameter sharing. We see that tying the match networks between the states of each submodel leads to a large parameter reduction, but at the cost of a 1% drop in the accuracy. This indicates that allowing the match networks to adapt to state-specific tasks is important. The accuracies obtained by sharing the weights internally in the match networks instead of tying their "outputs" emphasizes this fact. Thus, for the "adaptive weight sharing" approach an accuracy of 83.2% is obtained for $M = 3$. Although this model contains only one fourth the parameters of the HNN without tying, it obtains a comparable performance. Interestingly, for $M = 5$ the HNN actually outperforms the HNN without weight sharing even though the former contains far fewer parameters. Despite the better generalization ability of the HNN with weight sharing, it still tended to overfit the data if trained for a sufficient number of epochs.

7.2.5 Importance of Joint Training

The use of separate match networks in each HNN state constitutes one of the main differences to other hybrids. As discussed above, this implies that the recognition task can be distributed among highly specialized "expert" networks. Another difference to most early hybrids is that all parameters of the HNN are trained jointly to minimize a global discriminative criterion. To illustrate the importance of joint training the HNN was compared to a scaled likelihood hybrid. As described in chapter 6 the scaled likelihoods are obtained by training a multi-layer perceptron to classify each of the input frames into one of the considered classes. After training, the network outputs are scaled by the corresponding a priori probabilities of the classes and used as replacement for the standard match probabilities in an HMM. To compare this approach with the HNN we used a standard multi-layer perceptron with one output for each of the five classes, a symmetric context input of one frame and 50 hidden units. After classification training, the scaled outputs were used for replacing the match distributions in a CML trained CHMM. Note that the scaled output corresponding to *e.g.*, the consonant class was used for replacing all three match distributions in the consonant submodel. The scaled likelihood hybrid has a total of 4255 parameters and obtains an N-best accuracy of 78.1%, see table 7.6. This is far lower than the best accuracy of 84.1% reported above in table 7.5 for the HNN with separate match networks in each state and a similar number of parameters. However, part of the higher accuracy of the HNN is due to the use of specialized match networks in *each* state. It is therefore more fair to compare the scaled likelihood hybrid to an HNN using tied match networks in each submodel. As shown in table 7.5 such an HNN yields an accuracy of 82.4%. A gain of more than 4% in accuracy is thus achieved simply by adapting the match networks jointly with the transition probabilities instead of separately as in the scaled likelihood hybrid. Despite the difference in accuracies, note from table 7.6 that the HNN and scaled likelihood hybrid obtains almost the same frame classification rate. This indicates the importance of training the "whole" model directly for sequence recognition rather than for frame classification.

Approach	Param.	%Frame	%Sub	%Ins	%Del	%Acc
Raw networks	4205	76.9	7.1	38.7	3.8	51.5
Scaled likelihood	4255	80.0	3.6	8.8	9.6	78.1
HNN, tied match nets	4055	79.5	3.1	4.2	10.2	82.4

Table 7.6: Comparison of different approaches for broad class recognition. The frame classification rate (%Frame) is for the HNN and scaled likelihood hybrid calculated from the complete labeling obtained from a forward-backward decoding pass. Only results for the test set are shown.

It is interesting to see how much the Markov network actually contributes to the recognition accuracy. Based on the complete labeling from the large MLP with 50 hidden units, an incomplete broad class label string was created by "folding" multiple occurrences of the same label into one. Although the "raw network" yields approximately the same frame classification rate as the two hybrids, the obtained accuracy of 51.5% obtained using this method is very low, see table 7.6. The reason for this is that the "raw network" gives a very poor duration modeling. Thus, it is often the case that a few or even a single frame is assigned a class label by the network which is different from that of the surrounding frames. When folding the complete label sequence into an incomplete broad class labeling, such "artifacts" will remain and result in a very large number of insertions in the alignment between observed and recognized labeling. The temporal integration and duration modeling implied by the Markov network is thus crucial for good recognition performance.

7.3 Transition-Based Modeling

A serious limitation of standard HMMs and the above HNN architecture is that they model the speech as sequences of steady state segments connected by "instantaneous" transitions. By using transition networks in the HNN the transitions become "time-varying" and the HNN can thereby model the speech as sequences of steady state segments connected by transitional segments. As discussed in chapter 6 this can be very important for automatic speech recognition because experimental evidence suggests that the regions of the acoustic signal that contain the largest spectral changes are important for human perception of speech.

We start by evaluating a simple topology with only one state per class and then turn to the three-state topology used above and in chapter 5.

7.3.1 1-State Submodels

In these experiments only a single state was used for modeling each class, *i.e.*, the three states for each submodel in figure 5.2 were replaced by a single state. Since each state had a total of five outgoing transitions including the one to the state itself, we used a transition network with five outputs in each state and set all match networks to the trivial mapping $\phi_i(\boldsymbol{s}_l; \boldsymbol{v}^i) = 1.0$ for all l.

Locally Normalized HNN

If all the transition networks use a softmax output function and if the sequences are allowed to end in all states then the model is normalized locally as discussed in chapter 6. Such a locally normalized transition-based HNN is similar to an IOHMM where only one label is allowed in each state.

As for the HNNs evaluated above, it is very important to initialize the networks properly before CML training. Since the softmax normalized networks are used for estimating transition probabilities of the form $P(\pi_l = j|\pi_{l-1} = i, \boldsymbol{s}_{l-1})$, a reasonable initialization is to train the networks to learn these a posteriori probabilities. As usual this can be done by standard classification training with 1/0 targets. Because the transition networks are assigned to a particular state the conditioning on the previous state (*i.e.* π_{l-1}) is given beforehand. In the simple 1-state submodels this implies that we should only train the networks on data having a class label identical to that of the state to which the network is assigned. That is, for frames with other labels the output is undefined. Unfortunately, this results in two problems. First of all, the training data for each network is very sparse, especially for the outputs corresponding to transitions between states with different labels (only the boundary frame between two broad class segments are used for training). Secondly, because the networks are trained only on part of the input space used during recognition they are likely to generalize very poorly to input frames with class labels different from that of the frames used during initial network training. Therefore, the two initialization methods tested for the match networks were also tried. In the classification approach a network with five outputs was simply trained to classify the speech frames into each of the five classes. The same copy of this network was then used as an initial transition network in *all* states.

Initialization	Train	Test
Random	84.3	80.8
Trans. posteriors	85.2	80.7
Viterbi	85.4	81.0
Classification	***85.8***	***81.3***

Table 7.7: Effect on recognition accuracies ($\%Acc$) of different initialization strategies for transition networks in a transition-based, locally normalized HNN with one state per class. All transition networks take a symmetric context of one frame as input ($\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$) and have 10 hidden units. The total number of parameters is 4225.

For the different network initialization methods the results after CML training are shown in table 7.7. As expected, the method of initializing the networks to estimate posteriori probabilities results in a performance that is comparable to that of random initialization. Although the difference between the initialization methods is not large, we see that the classification approach seems to give the best result.

Globally Normalized HNN

Table 7.8 summarizes the results obtained when using globally normalized transition-based HNNs. For these models the data sequences were only allowed to end in the state corresponding to the silence class and global normalization was thus necessary even for softmax normalized network outputs, see chapter 6. Comparing to the locally normalized HNN we observe a small gain by restricting all sequences to end in the silence state, see table 7.8. However, if the transitions are allowed to be unnormalized by using sigmoid output functions, a significant improvement is obtained. Thus, the globally normalized transition-based HNN yields 82.8% accuracy compared to 81.3% for the locally normalized model. By using weight sharing as described above almost the same accuracy is obtained, but with a model containing only 1800 parameters, see table 7.8. The reason for the large difference between the locally and globally normalized transition-based HNN can be found by inspecting figure 7.8 showing the outputs of the transition network assigned to the state modeling the consonant and silence class, respectively. From the figure we first observe that the transition networks in the locally and globally normalized model behave quite differently, especially for the silence class. Similarly, we see that the networks for most frames either output a value close to zero or a value close to one and the models therefore have a somewhat "deterministic behavior". For the globally normalized model we observe that for some frames all the outputs of the transition network in the state for *e.g.*, the silence class are close to zero. In such cases it is not possible to enter or pass through the silence state. On the other hand, this is always possible for the locally normalized model because at least one of the outputs of any transition network in this model will be larger than 0.2 due to the softmax normalization. Thus, with locally normalized transitions we can never prohibit a path from passing through a given submodel at any time. Therefore, using submodels with several states for each class to enforce minimum durations does not really make sense for softmax normalized transition networks.

Model	Output func.	Param.	Train	Test
Transition-based	Softmax	4225	84.9	81.6
Transition-based	Sigmoid	4225	86.4	***82.8***
Transition-based, weight sharing, $M = 5$	Sigmoid	1800	86.1	82.6
Match & transition nets	Sigmoid	8230	***90.2***	82.2
Match nets & standard transitions	Sigmoid	4030	84.6	80.0

Table 7.8: Recognition accuracies ($\%Acc$) by transition-based HNNs, an HNN using match networks and standard transitions and an HNN using both match and transition networks. Only one state is used for each class and all networks take a symmetric context of one frame as input ($\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$) and have 10 hidden units. The network with weight sharing has an additional hidden layer of 15 hidden units, see figure 7.7.

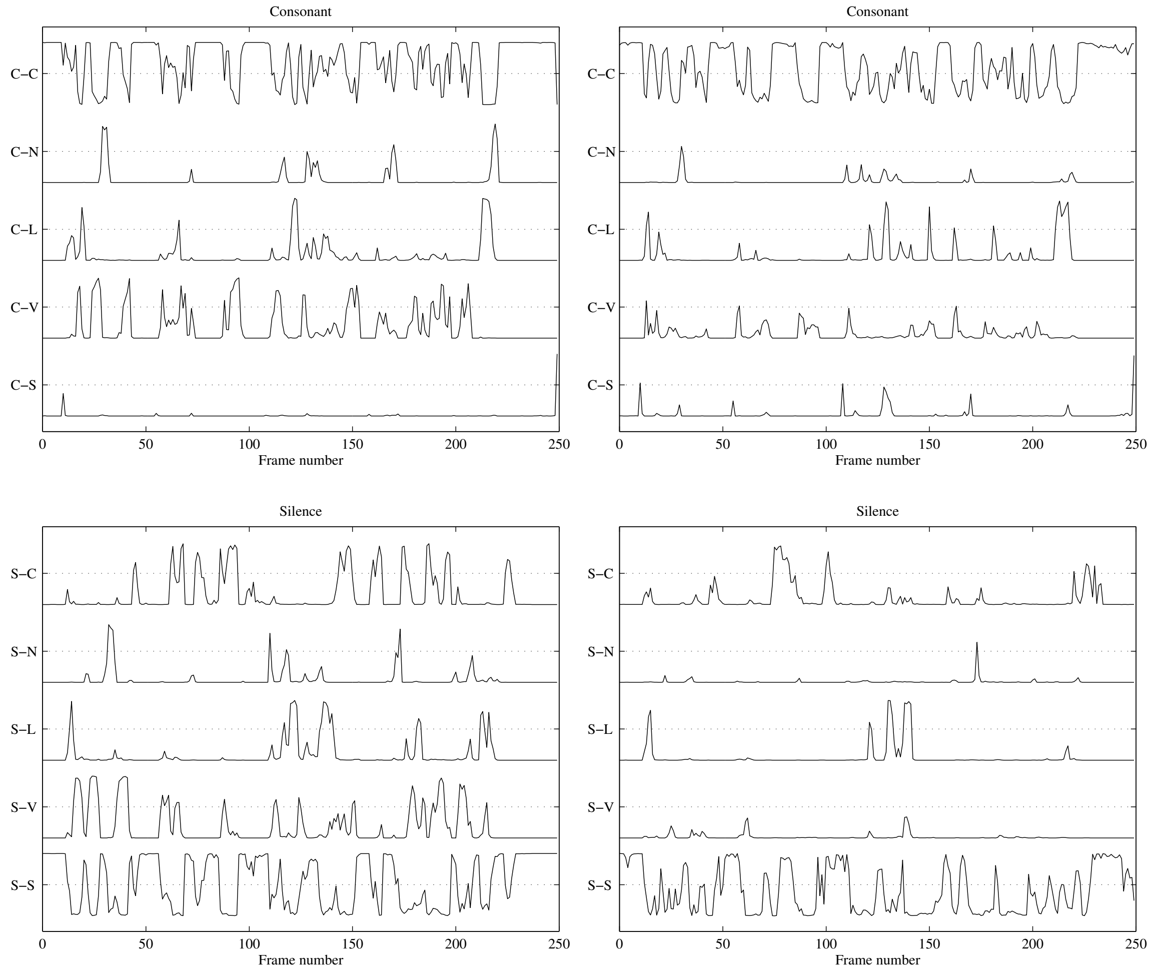

Figure 7.8: Output from the transition networks in the state for the consonant and silence class for an example sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). ***Left panels:*** Locally normalized model. ***Right panels:*** Globally normalized model.

The accuracy of the transition-based HNN using 1-state submodels is somewhat lower than the best accuracy of an HNN using 3-state submodels and match networks, see table 7.5 and 7.8. However, one can argue that it is a bit unfair to compare the 3-state and 1-state models as the former enforces a minimum duration of two frames for each class. If match networks and standard transition probabilities are used instead of transition networks in each 1-state submodel, the accuracy is only 80.0%, see table 7.8. This illustrates the importance of allowing the model to focus on the transitional segments in the speech signal. The different behavior of the transition-based HNN and the HNN using match networks and standard transitions is for an example utterance illustrated by a state posterior plot in figure 7.9. From the figure we observe that the 1-state model using match networks and standard transition is fairly "sure of what it is doing", *i.e.*, only a few paths tend to dominate the probability for the the example utterance. Contrary to this we see that several paths contribute in the globally normalized transition-based HNN. Thus, the transition-based HNN is capable of distributing the recognition task between the class submodels even better than the HNN using match networks and standard transitions.

As a final remark for the 1-state transition-based HNN it should be noted that one can interpret the transition network outputs for the self-loop transitions as modeling the steady-state segments of the speech signal. Thus, as long as the self-loop output is large for a given network we can remain in the corresponding state. Therefore, one can argue that using both transition and match networks in a given state is not necessary. Indeed, as shown in table 7.8 such an approach actually just leads to serious overfitting.

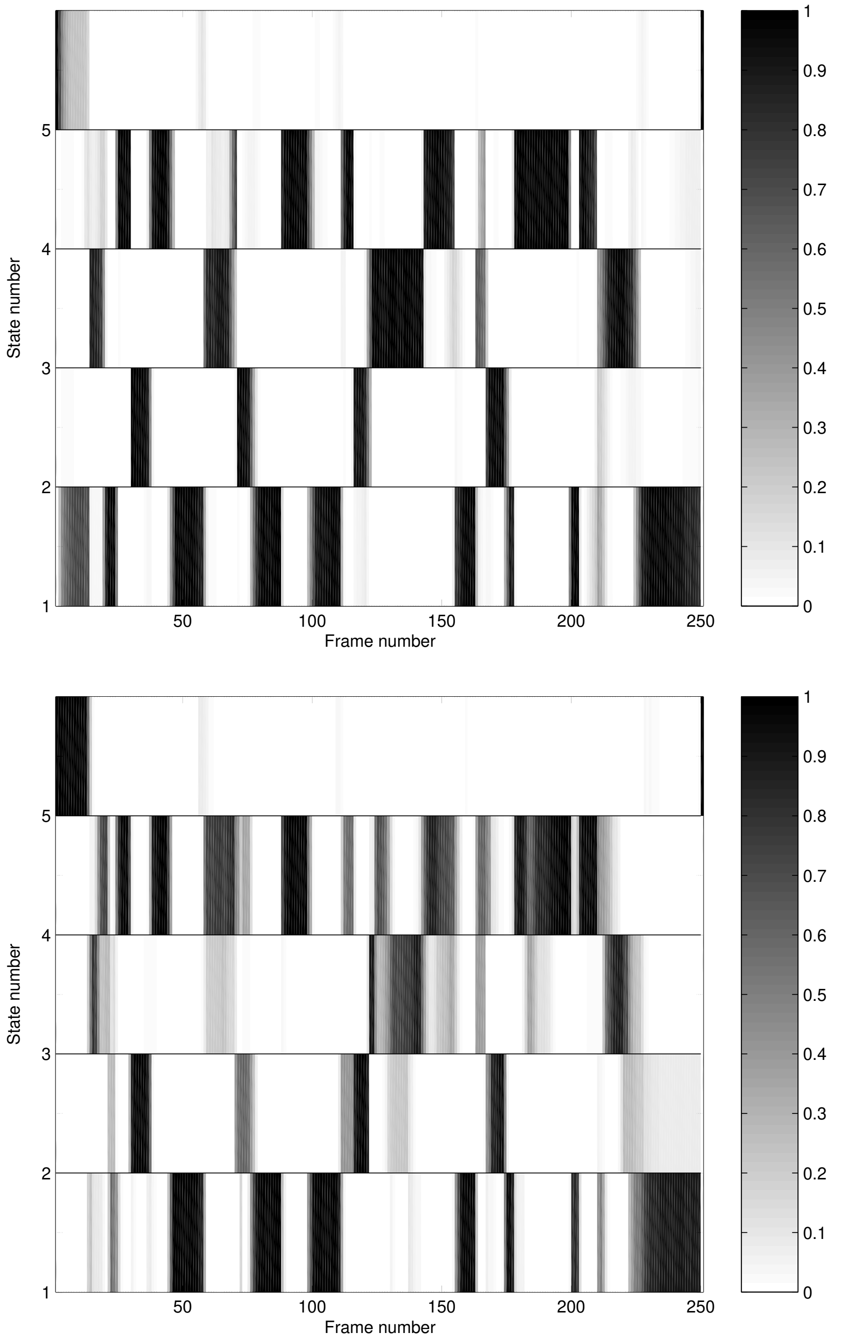

Figure 7.9: Graytone plot of state posteriors $n_i(l) = P(\pi_l = i|\boldsymbol{x}, \boldsymbol{\Theta})$ for the test set utterance "But in this one section we welcomed auditors" (TIMIT id: si1361). State 1 is the consonant, 2 the nasal, 3 the liquid, 4 the vowel and 5 the silence model. ***Upper panel:*** 1-state HNN using match network and standard transitions in each state. Sentence accuracy: $Acc = 75.0\%$. ***Lower panel:*** 1-state globally normalized transition-based HNN. Sentence accuracy: $Acc = 84.4\%$.

7.3.2 3-State Submodels

By comparing the result of 80.0% for the 1-state model using match networks to the result of 82.4% for the HNN using a single (tied) match network in all states of a 3-state submodel we see that enforcing minimum durations is very important. Therefore, an experiment with transition-based HNNs using three-state submodels was made. The transition networks in the last state of each submodel were in these experiments initialized by the above described classification training. This approach does not make sense, however, for the transition networks in the first and center states of the submodels because these networks estimate transition scores between states with identical labels. A method that seems to work well in practice for these networks is to copy the weights of a classification trained match network as illustrated in figure 7.10. This initialization is actually reasonable in the framework of globally normalized HNNs because a state with such a transition network is initially identical to a state with a classification trained match network and unit transitions (*i.e.*, $\theta_{ij} = 1.0$ for all j). If the transition networks in the first and center state are normalized by a softmax output function, the above initialization leads to transition networks with outputs that initially are uniform. Nevertheless, the method of copying weights was observed to work better than just using random weights in softmax normalized transition networks.

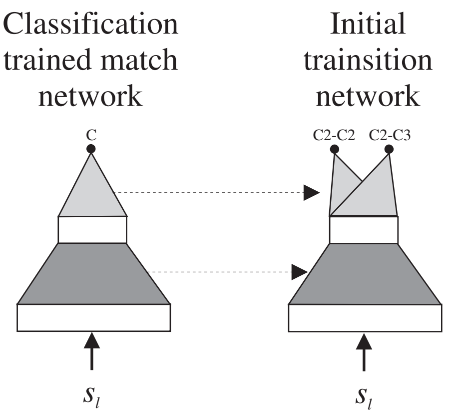

Figure 7.10: Initializing transition networks by copying weights of a classification trained match network. Illustrated for the center state in the consonant submodel (C2 is the center state and C3 the last state).

Table 7.9 summarizes the results obtained by using 3-state submodels. As expected, the result is very poor for both the training and test set when the transition networks are normalized locally by softmax output functions. If sigmoid output activation functions are used instead the accuracy increases to 84.5% or 84.6% when using weight sharing. This compares favorably to the best result of 84.1% obtained by the HNN using match networks and standard transitions.

Instead of using transition networks in all states one can model the steady-state segments by states with standard transitions and match networks. As shown in table 7.9, an HNN using match networks and standard transition probabilities in the first and center states and a transition network in the last state of each submodel yields a performance comparable to the purely transition-based model. The reason for this is that the center states of the two types of HNNs perform a similar task. Thus, if the standard self-loop

Model	Output func.	Param.	Train	Test
Transition-based	Softmax	12400	83.9	82.0
Transition-based	Sigmoid	12400	***89.7***	84.5
Transition-based, $M = 5$	Sigmoid	5125	88.2	***84.6***
Match & transition nets	Sigmoid	12255	89.6	84.4
Match net & standard transitions, $M = 5$	Sigmoid	4790	88.0	84.1

Table 7.9: Recognition accuracies by transition-based HNN and "mixed" HNN with three states per class. The "mixed" HNN uses match networks and standard transition probabilities in the first and center state of each submodel and a transition network in the last state. All networks use a symmetric context of one frame as input ($\boldsymbol{s}_l = \boldsymbol{x}_{l-1}, \boldsymbol{x}_l, \boldsymbol{x}_{l+1}$) and have 10 hidden units. The model with weight sharing ($M = 5$, see figure 7.7) has an additional hidden layer of 15 units.

transition probability is close to one and the transition probability to the final state is close to zero, the only way to exit the center state is by having a match network output close to zero. Similarly, if the match network in the center state outputs a value close to zero for a sequence of frames no path can pass through the corresponding submodel for these frames. A similar operation can be obtained by an unnormalized transition network with two outputs.

7.4 Summary

In this chapter three types of HNNs have been evaluated on the simple broad class task defined in chapter 5. In the first case, only the match distributions were replaced by match networks whereas the other two architectures both used transition networks. For all models it was illustrated that the initialization of the neural networks is very important for good performance. Furthermore, as for the CHMM it was shown that initial complete label training is also very important for good performance.

For the HNN using match networks and standard transitions it was illustrated how the match networks in the different states of a class submodel adapt to state-specific tasks. This was shown to lead to a better performance than when the same network was used in all states of a submodel. Similarly, it was shown that training the match networks jointly with the transition probabilities leads to a very large gain in accuracy compared to a scaled likelihood hybrid where the "match network" and HMM are trained separately.

Contrary to standard HMMs (and HNNs using only match networks), the HNNs with transition networks are capable of modeling the speech as sequences of steady-state segments connected by transitional regions. For transition-based HNNs it was shown that the best performance is obtained when using globally normalized models where all transitions in a state are allowed to be zero simultaneously. Hereby, a transition network in a class submodel can effectively prohibit paths through the submodel which is not possible when using locally normalized transition networks with softmax outputs. This is also important when using multiple states in a class submodel so as to enforce minimum durations. For three-state submodels the transition-based HNN was shown to outperform the HNN using match networks and standard transitions. It was also shown that a "mixed" HNN using match networks to model steady-state segments and transition networks to model transitional regions gives approximately the same accuracy as a transition-based HNN.

A problem with the HNN is that it overfits the training data if match or transition networks with hidden units are used. To handle this, a weight sharing method, which has an interesting interpretation as a state-specific adaptive input transformation, was proposed. Even though the weight sharing reduced the number of parameters considerably it did not completely remove the overfitting tendency. It did, however, lead to better generalization.

The best result of 84.6% on the broad class task was obtained by a purely transition-based globally normalized HNN with approximately 5000 parameters. This compares very favorably to the result of 81.3% obtained by the CHMM with about 4000 parameters. For comparison the best result reported in the literature for a continuous density HMM with a linear adaptive input transformation is 81.3% on this task [Joh94].

In the following chapters we will evaluate the HNN on two more realistic speech recognition tasks, namely the "standard" 39 TIMIT phoneme recognition task and the recognition of isolated-words uttered over American telephone lines.

Chapter 8

TIMIT Phoneme Recognition

The results presented in chapter 7 indicate that the HNN is capable of yielding very good performance on the recognition of broad phoneme classes. However, the acoustic model in any speech recognition system cannot be based entirely on broad phoneme class models since it is not possible to translate broad phoneme class sequences into words. In this chapter we therefore extend the broad class task to the recognition of 39 different English phonemes from which most American English words can be constructed.

The general experimental setup will be described in section 8.1 and section 8.2 gives an evaluation of a baseline discrete CHMM on the phoneme recognition task. The HNN evaluation for this problem is presented in section 8.3 for HNNs using match networks and standard transitions and in section 8.4 for transition-based HNNs. A comparison to results reported in the literature for the TIMIT 39 phoneme task is given in section 8.5.

8.1 Experimental Setup

The 39 phoneme TIMIT task first introduced in [LH89] has been used for preliminary evaluation of acoustic models by several speech groups. Lee and Hon [LH89] defined 39 phoneme classes from the full 61 TIMIT phone set which, in combination with the recommended training and test sets, give a well-defined reference for phoneme recognition experiments. The 39 phoneme inventory is shown in table A.2. As for the broadclass experiments, one context independent monophone submodel was used for each of the 39 phoneme classes. The topology of the submodels and of the overall model was the same as for the broad class task, see chapter 5.

We used the recommended TIMIT training set, but left out 35 of the 462 recommended training set speakers for a validation set. Thus, the validation set contained a total of 280 utterances and the training set a total of 3416 utterances. The validation set was used for adapting the stepsize during gradient descent CML training and for model selection as described in chapter 5. The recommended TIMIT test set of 1344 utterances was used for evaluating the models. Table 8.1 summarizes the dataset details.

The computational complexity of the 39 phoneme task is substantially larger than for the simple broad class problem. Training times of 3-4 CPU weeks on a fast workstation were not uncommon for some of the models evaluated in this experiment. Therefore, the training strategy developed for the broad class task was applied directly to the 39 phoneme task. A maximum number of 10 complete label and 50 incomplete label epochs was enforced in all experiments and the complete labeling was obtained from the phonetic hand-segmentation in the same manner as for the broad phoneme task. Similarly, the

Datasets	
Training set	
Speakers	427
Different sentences	1586
Utterances	3416
10ms frames	1031138 ($\approx$ 2h52min)
Phoneme labels	128988
Test set	
Speakers	168
Different sentences	624
Utterances	1344
10ms frames	407588 ($\approx$ 1h8min)
Phoneme labels	50754
Validation set	
Speakers	35
Different sentences	130
Utterances	280
10ms frames	84520 ($\approx$ 14min)
Phoneme labels	10572

Table 8.1: Summary of TIMIT datasets used for the phoneme recognition experiments. See also table 5.2 and appendix A.

mel-scaled cepstral preprocessor described in chapter 5 was used for extracting feature vectors from the raw speech signal.

As for the broad class experiments it was necessary to use both local and global pruning thresholds in the N-best decoder for computational reasons. Due to the larger number of submodels in the task considered here it was necessary to prune the search somewhat more "aggressively" than for the broad class task. Thus, a local pruning threshold of γ_l=100 and a maximum number $M = 5$ of active hypotheses were used in all phoneme recognition experiments.

8.2 The Discrete CHMM - A Simple Baseline System

As a first experiment, a simple discrete CHMM was trained using the ML and CML criterion. The discrete CHMM made use of the codebook of 256 prototype cepstral vectors described in chapter 5 and was based on the 3-state submodel topology shown in figure 5.1. The CHMM contains a total of 31707 parameters.

Recognition accuracies obtained through ML and CML training are shown in table 8.2. As expected there is a fairly large improvement by using CML training instead of ML training; The CML trained CHMM obtains an N-best accuracy of 56.3% compared to only 49.7% for the ML trained CHMM. Interestingly, a large improvement in accuracy for the ML trained model was obtained by squaring the bigram transition probabilities before decoding, see table 8.2. This was also observed for the broad class task. Also, there is a large difference between Viterbi and N-best decoding results for the discriminatively trained model but not for the ML trained model, see table 8.2. Thus, several paths tend

Criterion	Train	Test	
	N-best	Viterbi	N-best
ML	47.8	49.5	49.7
ML (Squared bigram)	50.7	52.0	51.9
CML	***55.5***	53.0	***56.2***

Table 8.2: Recognition accuracies ($\%Acc$) for discrete CHMM containing a total of 31707 parameters.

to contribute to the probability of the utterance in the CML trained CHMM whereas only a few paths contribute in the ML trained CHMM. This is clearly illustrated by the state posterior plot for the ML and CML trained CHMM given in figure 8.1. Note, that the plots in figure 8.1 have been thresholded so that all posteriors smaller than 10^{-3} are set to zero and all others to one.

Before moving on to the HNN evaluation in the following sections it should be noted that the accuracies for the discrete CHMM are lower than those typically reported for continuous HMMs applied to this task (see section 8.5). However, the discrete CHMM is a simple model based only on a single codebook for both cepstral, Δ-cepstral and log-energy features. In [LH89] an ML estimated discrete HMM employing three separate codebooks, around 92000 parameters and context independent phoneme submodels was reported to yield an accuracy of 53% on the phoneme task.[1] Note that the CML trained CHMM is capable of outperforming the multiple codebook system used be Lee and Hon.

[1] The test set used by Lee and Hon [LH89] is different from the official TIMIT test set and their experiments were done using a prototype of the TIMIT CD-ROM.

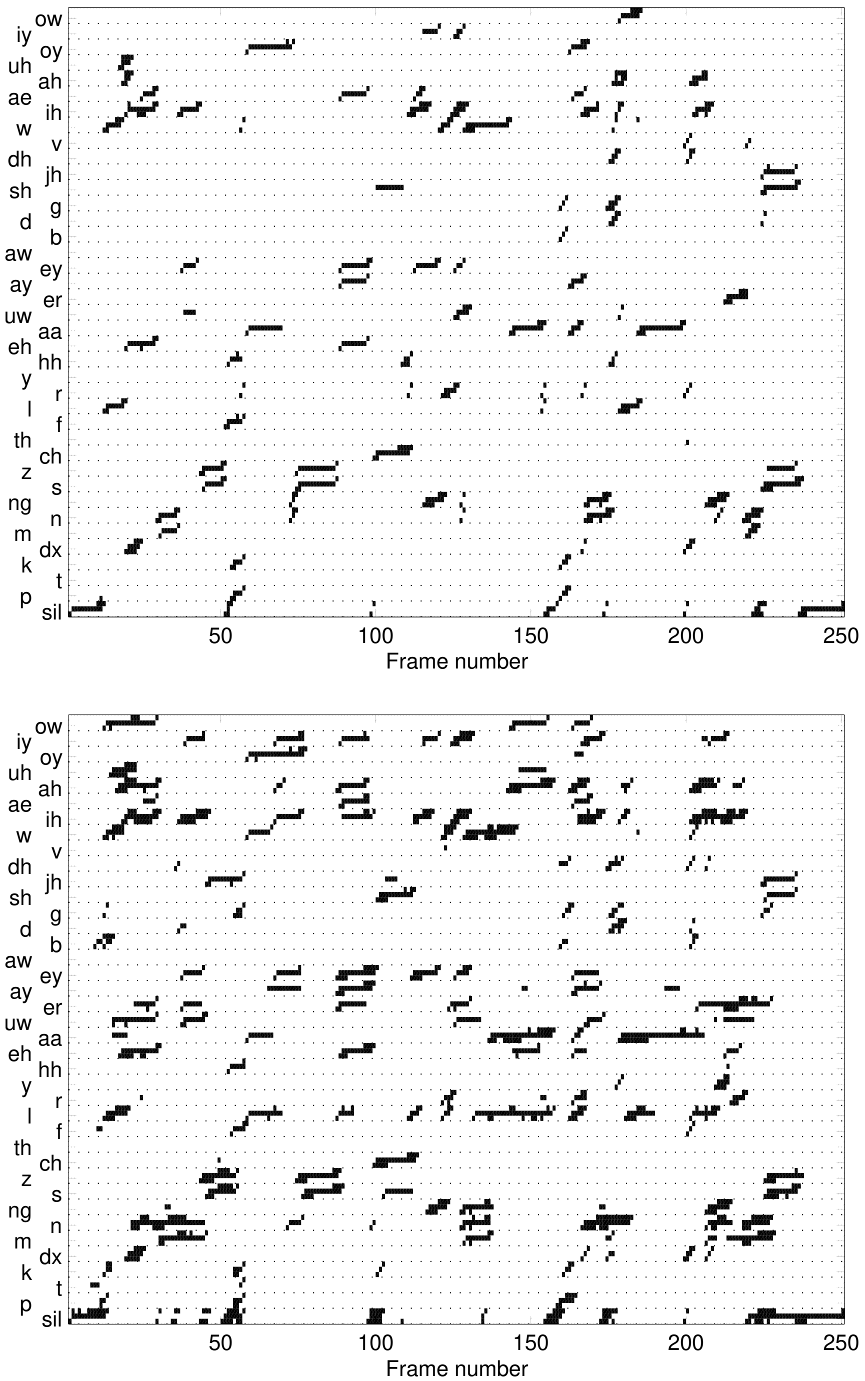

Figure 8.1: "Thresholded" state posteriors $n_i(l) = P(\pi_l = i|\boldsymbol{x}, \boldsymbol{\Theta})$ for the utterance "But in this one section we welcomed auditors" (TIMIT id: si1361). The horizontal lines enclose the three states of each phoneme submodel. To clarify the illustration, posteriors less than a threshold of 10^{-3} were set to zero and all others to 1.0. ***Upper panel:*** ML trained CHMM. Sentence accuracy: $\%Acc = 33.3$. ***Lower panel:*** CML trained CHMM. Sentence accuracy: $\%Acc = 47.2$.

8.3 HNNs using Match Networks and Standard Transitions

In this section we consider HNNs using match networks and standard transitions. For these experiments separate match networks were used in each state of a 3-state phoneme submodel and the networks were fully connected MLPs with sigmoid output functions. The same input s_l was shared by all networks.

Model	Param.	Train	Test	
		N-best	Viterbi	N-best
Context $K = 0$, 0 hidden	4914	57.8	57.5	60.6
Context $K = 1$, 0 hidden	10998	63.7	61.2	63.9
Context $K = 2$, 0 hidden	17082	64.7	63.2	65.8
Context $K = 0$, 10 hidden	34632	74.6	66.8	68.8
Context $K = 1$, 10 hidden	95472	79.0	***67.8***	***69.8***
Context $K = 2$, 5 hidden	79092	75.4	65.8	67.8
Context $K = 2$, 10 hidden	156312	***80.7***	67.8	69.6

Table 8.3: Recognition accuracies ($\%Acc$) for HNNs using match networks and standard transitions. K is the size of the symmetric context used as input to all match networks, *i.e.*, $s_l = x_{l-K}, \ldots, x_l, \ldots, x_{l+K}$.

The results for HNNs using different match network architectures are shown in table 8.3. Although it is a bit unfair to compare models using discrete and continuous inputs it is interesting that the HNNs without hidden units perform so well compared to the discrete CHMM. Thus, the simplest HNN with around 5000 parameters clearly outperforms the CHMM containing almost 32000 parameters. From the table we also note that a symmetric context input increases the accuracy dramatically for the models without hidden units. The best result of 65.8% was obtained by using a symmetric context of 2 frames. This model only contains 17082 parameters, or roughly half that of the discrete CHMM. Larger contexts than $K = 3$ were not evaluated for the reason given below.

When using hidden units in the match networks the accuracy increases, however, at the cost of a much larger number of parameters and consequently overfitting of the training set. The best accuracy of 69.8% was obtained by an HNN using match networks with 10 hidden units and a symmetric input context of one frame. Contrary to the models without hidden units, we see from table 8.3 that a larger context than one frame does *not* improve performance on the test set. This is most likely an effect of overfitting. Note also the large difference between Viterbi and N-best accuracies for all the HNNs shown in table 8.3.

Figure 8.2 illustrates the outputs of the match networks assigned to the states of two different phoneme submodels in the HNN. As for the broad class task we see that the HNN has efficiently learned to distribute the recognition task between the different match networks. Thus, the networks in the first state of the submodels act as "filter"-networks that detect phoneme onsets.

A *confusion matrix* is illustrated in figure 8.3 for the HNN using match networks with ten hidden units and a symmetric input context of one frame . This figure shows how the total number of errors is distributed on phoneme substitutions, insertions and deletions. The strong diagonal in the upper plot represents the 69.8% of the phonemes that are recognized correctly whereas the off diagonal elements indicate errors. The size of each of square at the intersection between two phoneme labels indicates the absolute number of occurrences of this type of error in the test set. Note that the silence "phoneme" is

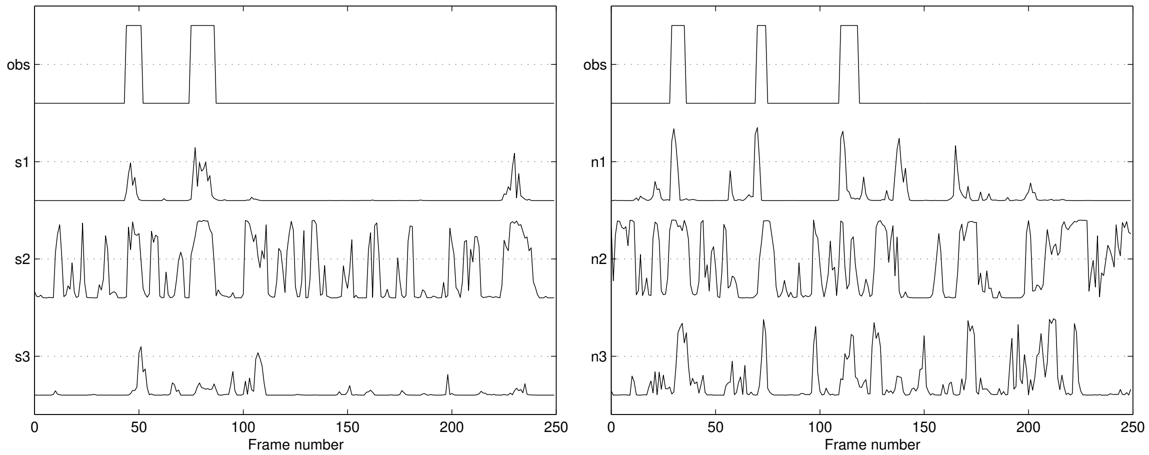

Figure 8.2: Output from the match networks assigned to the first (1), center (2) and last (3) state for an example sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). ***Left panel:*** The submodel for phoneme /s/. ***Right panel:*** The submodel for phoneme /n/. The top most curves show the observed complete labeling for the /s/ and /n/ phonemes, respectively.

Observed label	Recognized label	% of all errors
ah	ih	2.12
ah	*	1.77
ih	*	1.77
r	*	1.73
eh	ah	1.54
ih	ah	1.51
l	*	1.44
z	s	1.34
s	z	1.21
m	n	1.11

Table 8.4: The ten most frequent errors made by the HNN using match networks with 10 hidden units and a symmetric context input of one frame. A "substitution" of an observed phoneme with "*" identifies a deletion error.

not included in the plots because the large number of correct recognitions for this class would dominate the plot if it was included. The confusion plot indicates that a few types of errors tend to dominate the overall error. The ten most common errors are listed in table 8.4. As one might expect, the most common errors beside insertions and deletions are substitutions between highly confusable phonemes from the same phonetic group (see table A.1).

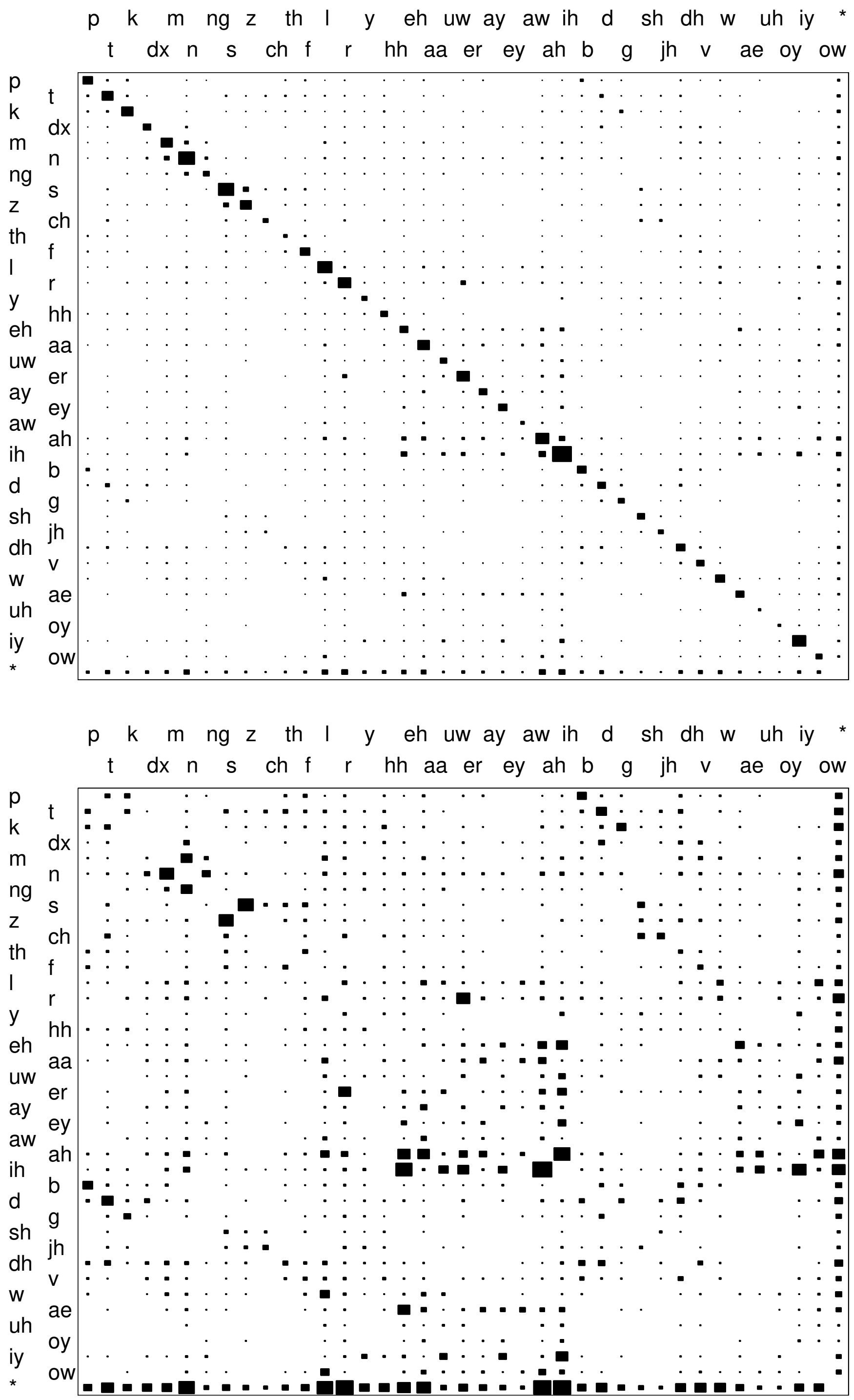

Figure 8.3: ***Upper panel:*** Phoneme confusion matrix for the HNN using match networks with 10 hidden units and a symmetric input context of one frame. The observed phoneme labels are shown on the horizontal axis and the recognized phonemes on the vertical axis. ***Lower panel:*** Same as in the upper panel, but with the diagonal subtracted to better resolve the confusions. For illustrating the insertion and deletion errors the symbol "*" is included so that the column corresponding to "*" indicates insertions and the row corresponding "*" indicates deletions.

8.3.1 Reducing Model Complexity

Similar to the broad class experiments, a reduction in model complexity was also tried by tying of parameters in the phoneme recognition experiments. Table 8.5 shows the results obtained by using various parameter sharing techniques. As seen, tying the match networks within a submodel reduces the number of parameters by a factor of three, but unfortunately also leads to a significant reduction in accuracy. This indicates that it is important to allow the model to distribute the recognition task between the different states of a submodel, see also figure 8.2. For this reason, hardwired minimum duration submodels (see figure 2.8) with a number of states equal to half the observed average duration of the phonemes yield only an insignificant improvement over the 3-state topology with tied match networks, see table 8.5. The result of 69.2% obtained using the "adaptive input transformation" weight sharing described in chapter 7 underlines the importance of having separate match networks in each state. With only 38727 parameters this model performs almost as well as the HNN without tying.

Model	Param.	Train	Test	
		N-best	Viterbi	N-best
3-state, no tying	95472	***79.0***	***67.8***	***69.8***
3-state, tied match nets	32994	69.6	61.2	64.0
3-state, weight sharing, $M = 5$	38727	73.0	66.7	69.2
Min. dur., tied match nets	32760	69.8	61.5	64.8

Table 8.5: Recognition accuracies ($\%Acc$) for HNNs with tying or weight sharing to reduce model complexity. All match networks use a symmetric context input of one frame and have 10 hidden units. The match networks with weight sharing ($M = 5$, see figure 7.7) have an additional hidden layer of 15 hidden units.

8.4 Transition-Based HNNs

In this section a discussion of the results obtained using globally normalized transition-based HNNs is given for the 39 phoneme task.

As for the broad class experiments, both the 3-state topology and the simple 1-state topology were examined. All transition networks used a symmetric input context of one frame, had 10 hidden units and were initialized as described in chapter 7. No softmax normalization was enforced, *i.e.*, all networks used sigmoid output functions.

For 1-state submodels we see from table 8.6 that the transition-based HNN gives a substantially higher accuracy on both training and test set compared to the use of match networks and standard transitions. The gain obtained by the transition-based HNN is somewhat larger for the 39 phoneme task than for the broad phoneme task. This is attributed to the fact that the transitional segments are more well defined in the large task, *i.e.*, they are not "averaged out" by using the same class for a large number of phonemes as in the broad class task.

As for the broad phoneme task, there is a very large difference in accuracy (>8%) between the 1-state and 3-state HNNs using match networks, see table 8.3 and table 8.6. Therefore, 3-state submodel topologies were also tested for the transition-based HNNs. However, with 3-state submodels there is only a small improvement of 0.4% in accuracy for transition-based modeling compared to the more "conventional" approach of using match networks and standard transitions, see table 8.6. In light of the large gain observed for the 1-state submodels, this result is rather disappointing. A possible explanation is that the 3-state submodels in combination with the looped structure of the overall model are not well suited for transition-based modeling. Indeed, this configuration is specifically aimed at modeling speech as a sequence of steady-state segments. Other topologies might be better suited for transition-based modeling.

Model	Param.	Train	Test	
		N-best	Viterbi	N-best
1-state trans nets	47541	68.2	62.6	66.5
1-state match nets	32760	63.9	57.5	61.4
3-state trans nets	111306	***79.6***	***68.0***	***70.2***
3-state match nets	95472	79.0	67.8	69.8
3-state trans nets, $M = 5$	54561	74.5	67.6	69.9
3-state match nets, $M = 5$	38727	73.0	66.7	69.2

Table 8.6: Comparison of recognition accuracies ($\%Acc$) for transition-based HNNs and HNNs using match networks and standard transitions. All networks use a symmetric input context of one frame, have 10 hidden units and sigmoid output functions. The networks with weight sharing ($M = 5$, see figure 7.7) have an additional layer of 15 hidden units.

8.5 Reported Results

A large number of speech groups have evaluated different kinds of standard HMMs and HMM/NN hybrids on the TIMIT 39 phoneme task. Table 8.7 and table 8.8 compare the results obtained in this work to a few of those reported in the literature for discrete and continuous observations, respectively. The experiments quoted in the tables are described in detail below. Note that no validation set was used in any of the quoted works.

Ref.	Model	Param.	%Acc
[LH89]	Discrete HMM (ML)	†92k	53.3
	Discrete HMM (ML), biphones	†3000k	66.1
This work	Discrete CHMM (ML)	32k	51.9
	Discrete CHMM (CML)	32k	56.2

Table 8.7: Summary of reported results for models using discrete observations. †Estimated.

[LH89] The 39 phoneme experiment was originated by Lee and Hon [LH89] almost 10 years ago. They compared context independent monophone and right-context dependent biphone submodels in discrete Mealy type HMMs with multiple input streams, see table 8.7. In the context independent experiments one phoneme submodel was used for each of the 39 phonemes whereas 1450 biphone submodels where used in the right-context dependent experiments. Various smoothing techniques were used to improve generalization in the context dependent models (co-occurrence smoothing and deleted interpolation). The results reported by Lee and Hon were based on a random test set from the December 1988 prototype TIMIT CD-ROM and they used a separately estimated bigram language model.

Ref.	Model	Param.	%Acc
[KVY93]	HMM (ML), full 4 Gaussian mix.	†176k	67.4
	HMM (MMI), full 4 Gaussian mix.	†176k	69.3
[Joh96]	HMM (CML), diag. 1 Gaussian mix.	8k	68.2
[Val95]	HMM (ML), four-gram	273k	70.3
	HMM (MMI), four-gram	273k	71.6
[YW94]	HMM (ML), biphones	†800k	72.3
[Rob94]	Recurrent network hybrid	47k	75.0
This work	HNN	95k	69.8
	HNN, weight sharing ($M = 5$)	39k	69.2
	Transition-based HNN	111k	70.2
	Transition-based HNN, weight sharing ($M = 5$)	55k	69.9

Table 8.8: Summary of reported results for models using continuous features. †Estimated

[KVY93] Kapadia *et al.* [KVY93] pioneered the application of MMI training of continuous HMMs for the TIMIT task. They used one 3-state monophone submodel with Gaussian mixture distributions for each of the 39 phonemes and a separately estimated (squared) bigram language model. Results were reported for a random 330 sentence test set for both diagonal and full covariance Gaussian mixture match distributions. Only the full covariance matrix results are quoted in table 8.8.

[Val95] A similar evaluation of ML and MMI training for 3-state monophone submodels was given by Valtchev [Val95]. For a scaled bigram language model practically the same results as reported by Kapadia *et al.* was obtained. A significant increase in accuracy was observed by using a four-gram instead of a bigram language model, see table 8.8. Valtchev also tried various global and submodel specific adaptive input transformations. For the TIMIT task no significant improvement was observed over the baseline MMI trained HMMs however. The core test set and a language model scaling factor of 2.0 was used in all experiments.

[YW94] Young and Woodland [YW94] applied right-context dependent biphone submodels with Gaussian mixture match distributions in each state. To reduce the number of parameters so-called state-clustering was used for tying match distributions among states of biphone submodels corresponding to the same phone. This resulted in a total of 1176 tied Gaussian mixture distributions. The results quoted in the table are for the 330 sentence random test set also used by Kapadia *et al.*.

[Joh96] An evaluation of CML trained continuous HMMs for the phoneme task was given by Johansen [Joh96]. He used single Gaussian match distributions with diagonal covariance matrices and a submodel topology similar to the one used here. Because of the low number of parameters, Johansen was able to train this model by an approximative second order method for several hundred iterations. Johansen also tried both linear and non-linear (MLP) global adaptive input transformations jointly trained with the HMM using the CML criterion. However, these approaches gave no improvement compared to the result quoted in table 8.8 for the CML trained HMM. Similar to this work Johansen used an N-best decoder and the full TIMIT test set.

[Rob94] The best published results on the TIMIT task to date were obtained using the Cambridge recurrent network hybrid (see chapter 6) in 1991 by Robinson [Rob91]. This network contained about 47000 weights and had 61 outputs — one for each of the 61 TIMIT labels. The 61 outputs were divided by the corresponding class a priori probabilities and used as scaled likelihood estimates in an HMM. For scoring the 61 phones were folded into the 39 phoneme set used here. An often stated reason why Robinson's hybrid is superior to most other methods is that the recurrent network is capable of learning typical phoneme sequences corresponding to the words in TIMIT. It is unclear whether this is the only reason — one can certainly argue that recurrent networks as HMMs have difficulties learning and representing long-term dependencies in the data due to the diffusion of credit problem [BF94, BF95]. The success of the Cambridge hybrid may also be a result of using the test set for designing the model. From an initial accuracy[2] of 68.9% reported in [TF90, RF91] the hybrid has gradually been improved for the TIMIT task by optimizing various parts of the system to yield 75.0% accuracy on the official TIMIT test set [Rob91, Rob94], see table 8.8.

From table 8.7 we see that the CML trained CHMM yields a higher accuracy than the ML trained discrete HMM used by Lee and Hon even though this model contains almost three times as any parameters. However, the monophone based CHMM is clearly inferior to the highly parameter intensive right-context dependent model used by Lee and Hon.

Only a few results for discrete models have been reported in the literature for the TIMIT task. There are essentially two reasons for this. Firstly, the TIMIT database became available about 10 years ago just at the time when continuous HMMs started to become a popular model for speech recognition. Secondly, by comparing the results in table 8.7 and table 8.8 there is no doubt that continuous HMMs yield far better performance with fewer parameters than discrete HMMs. From table 8.8 we see that the HNN yields a comparable performance to models of the same or even higher complexity (in terms of parameters).

8.6 Summary

This chapter has shown that the HNN is capable of obtaining a performance on the TIMIT 39 phoneme task which is comparable to results reported for state-of-the-art HMM and hybrid recognizers. As for the broad class experiments in chapter 7 we found that the weight sharing technique more than halved the number of parameters without affecting the accuracy significantly. Furthermore, for 1-state submodels we observed a large improvement by using a globally normalized transition-based HNN instead of a "conventional" HNN with match networks and standard transitions. A similar improvement was, however, not observed for 3-state submodels. A possible reason for this is that the 3-state submodels in combination with the "looped" topology of the overall model is designed for modeling the speech as a sequence of steady-state segments. Thus, other topologies might be more suitable for transition-based modeling. For example, one could use a single submodel with match networks and standard transitions to take care of *all* steady-state segments and a number of states with transition networks to model transitional regions between steady-state segments. This is similar to the idea of auditory event (avent) modeling as

[2] Based on the December 1988 prototype of the TIMIT CD-ROM.

discussed in chapter 6. The 1-state topology in combination with an explicit parametric or non-parametric duration model might also be more appropriate for transition-based modeling.

The TIMIT task has for almost a decade been used for evaluating the ability of acoustic models to discriminate between different phonemes. Although the database is clearly partitioned into training and evaluation utterances, it has been so widely used that there is a risk of implicitly using the test set for designing the recognition model. Thus, each time a new "trick" is reported to improve performance, it is included in most subsequent recognizers. Typical examples are the 3-state submodel topology and the ad hoc squaring of bigram language model probabilities which are used in many evaluations for this task. A more objective method for evaluating speech recognizers (as well as any other classifier) is to use an independent validation set during the design process and only as a last step evaluate the model on an official test set. In the following chapter, the HNN is evaluated on the task of recognizing isolated words uttered over the existing North American telephone network. This task is fairly new and has only been used a couple of times for evaluating speech recognizers.

Chapter 9

Task Independent Isolated Word Recognition

The phoneme recognition tasks presented in the previous chapters serve as a good test bed for preliminary evaluation of speech recognition systems. The evaluations were, however, based on high fidelity read speech recorded in a noise free environment. Furthermore, only the phoneme recognition ability of the models was considered. Therefore, an evaluation of HNNs on a more "realistic" task is given in this chapter. Specifically, we will consider so-called *task independent* small and medium size vocabulary isolated word recognition over North American telephone lines. The database used for this task was the recently released PhoneBook database, containing isolated words uttered by native American speakers. This database and the preprocessor used for the experiments is described in more detail in section 9.1. Section 9.2 discusses different submodel topologies applicable to task independent isolated word recognition and section 9.3 presents the results for HNNs using match networks and standard transition probabilities. The chapter is concluded in section 9.4 by a comparison to results reported in the literature for the PhoneBook task.

9.1 Task Definition

Isolated word recognizers are often trained using a fixed and task specific vocabulary. For small to medium size vocabularies it is common to train one HMM for each of the vocabulary words. However, such an approach forces that the vocabulary used during recognition is limited to that used during training. For out-of-vocabulary utterances a very poor performance can be expected from such a system unless it incorporates some sort of out-of-vocabulary detection. Furthermore, if the system is to be used for another application than the one it was designed for it has to be retrained or even redesigned to match the new vocabulary. It would therefore be desirable to train the models for *task independent* recognition where the vocabulary used for training can be entirely different to that used during recognition. Thus, once the models have been trained they must be applicable to a wide range of tasks with completely different vocabularies. Typical examples of isolated word recognition tasks with different vocabularies are *e.g.*, telephone home-banking and hands-free operation of mobile phones.

Although most HMM-based state-of-the-art systems for fixed vocabularies use context dependent phone models like *e.g.*, biphones or triphones, context independent monophone models are better suited for task independent recognition. The reason for this is that context dependent phone models are more vocabulary dependent than context independent phone models, *i.e.*, some biphones or triphones may occur only in a single or a few words in the training set even if it covers a very large vocabulary. For this reason all PhoneBook experiments were based on monophone submodels. The topology of the phone submodels and the overall model setup will be discussed in more detail in section 9.2.

9.1.1 Database and Dictionary

The utterances for training and evaluating the HNN for task independent isolated word recognition were taken from the PhoneBook database [PFW+95], see appendix A. PhoneBook is a phonetically rich, isolated word, telephone speech database of American English words. It incorporates all phones in as many segmental/stress contexts as are likely to produce coarticulary variations. Similarly, there is a large variability in talkers and telephone transmission characteristics, since the utterances are recorded over the existing North American telephone lines from a demographically representative set of native American English speakers. Note that the speakers used their own telephone handset for the recordings.

The database is organized into 106 word lists (see appendix A), each composed of 75 or 76 different words, yielding a total of almost 8000 different words. The words in each word list are uttered by an average of 11.7 different speakers so each word list contains approximately 880 isolated word utterances on average. The total number of utterances in PhoneBook is more than 92000, equivalent to 23 hours of speech.

For training we used a vocabulary of 1581 words taken from the 21 PhoneBook word lists `[a-d][a,h,m,q,t]+ea`, see appendix A. These word lists contain a total of about 19000 utterances of which 9000 were selected at random for training. The random selection was done such that each of the 1581 words was uttered at least once. For monitoring performance and for model selection a validation set composed of the utterances corresponding to the two PhoneBook word lists `a[oy]` was used. The validation set was thus based on a 150 word vocabulary and contained 1893 utterances. The test set was composed of eight word lists, `[a,b,c,d][d,r]`, and contained a total of 6598 utterances. This is identical to the test set used in [HRB+97] and [DBD+97] where some of the first recognition results on PhoneBook were presented. Since none of the words in any of the test set word lists are identical, results are reported either as an unweighted average over the 8 lists with a separate 75 (or 76) word dictionary for each list or for the entire test set with a dictionary of 602 words (the total number of different words in all eight word lists). Note that there are no identical words or speakers between any of the three datasets.

Contrary to the TIMIT database the utterances in PhoneBook are not time aligned with the phonetic transcriptions, *i.e.*, only the incomplete labeling for each word is available. The phonetic transcriptions in PhoneBook are based on a 42 phoneme inventory. However, in [DBD+97] a considerably better recognition performance was observed when using the Carnegie Mellon University (CMU) dictionary to phonetically transcribe the words instead of the PhoneBook transcriptions. We therefore used the public domain 110,000-word CMU pronounciation dictionary (`v.0.4`) for phonetically transcribing the words.[1] The phone inventory in the CMU dictionary is identical to the one used for the TIMIT 39 phone recognition experiments, see Table A.2, except that the phone /zh/ is not merged with /sh/. Similar to the experimental setup in [HRB+97, DBD+97] each occurrence of a stop (/b,d,g,k,p,t/) was preceeded by the corresponding closure (/bcl,dcl,gcl,kcl,pcl,tcl/) in the transcriptions of the words from PhoneBook, *i.e.*, /b/

[1] Version `0.4` of the CMU dictionary is available at http://www.speech.cs.cmu.edu/cgi-bin/cmudict

was replaced by /bcl/ /b/. This effectively means that we used a 46 phone set for the PhoneBook experiments. The dataset details are summarized in table 9.1 and table 9.2.

PhoneBook datasets	
General characteristics	
Speech type	Isolated American English words (selected)
Environment	North American phonelines, private handsets
Phone segmentation	Based on NTIMIT
Phonetic transcription	CMU dictionary v. 0.4.
Dialect regions	Demographically representative for USA
Speakers	1358 native, adult
Distinct words	7979 in 106 word lists of 75 or 76 words
Utterances	93667
Training set	
Distinct words	1581 (lists `[a-d][ahmqt]+ea`)
Utterances	9000
10ms frames	1272161 ($\approx$ 3h 32min)
Phone labels	96878
Test set	
Distinct words	602 (lists `[a-d][dr]`)
Utterances	6598
10ms frames	952288 ($\approx$ 2h 39min)
Phone labels	71128
Validation set	
Distinct words	150 (lists `a[oy]`)
Utterances	1893
10ms frames	276907 ($\approx$ 46min)
Phone labels	21014

Table 9.1: Summary of PhoneBook datasets used for isolated word recognition experiments.

Test set	Utterances	Words	Frames	Phones
`ad`	764	75	105152	8121
`ar`	672	75	99169	7326
`bd`	780	75	110129	8333
`br`	1007	75	146103	11054
`cd`	964	75	140599	10565
`cr`	785	75	118356	8417
`dd`	725	76	102291	7690
`dr`	901	76	130489	9622

Table 9.2: Summary of PhoneBook test sets used for isolated word recognition experiments.

9.1.2 Preprocessing

Human perception is generally insensitive to steady-state linear distortions typically encountered in telecommunication channels or resulting from different microphones. Such linear distortions appear as additive constants in the logarithm of the short term spectrum of the speech. Most short term spectral analysis techniques like *e.g.*, cepstral or linear predictive analysis, however, are based on absolute spectral values and are therefore vulnerable to such distortions. Several researchers have shown that a speech recognition system designed using *e.g.*, cepstral features calculated from noise-free speech fails to recognize utterances corrupted by noise, see *e.g.*, [HM94, HMBK91]. A methodology called *RelAtive SpecTrAl* (RASTA) calculation was given in [HMBK91] for reducing the sensitivity of short term spectral features to noise and linear channel distortions. The basic idea in the RASTA method is to replace the conventional short term absolute spectrum used for the speech analysis by a spectral estimate in which each frequency channel is band-pass filtered (in the log domain) [HM94]. This approach suppresses any constant or slowly varying components in each frequency channel. The RASTA methodology was proposed in the context of the so-called Perceptual Linear Predictive (PLP) technique for estimating short term power spectra of speech [Her90]. PLP is similar to standard linear predictive analysis, but in addition it includes a number of perceptual concepts to give an estimate of the auditory spectrum that leads to short term features better suited for speaker-independent speech recognizers [RAB$^+$93]. The PLP coefficients can be further transformed into cepstral parameters by a recursion similar to the one for transforming linear predictive coefficients [HM94, DPH93]. Compared to PLP cepstral coefficients, the RASTA-PLP cepstral coefficients have proven very robust to linear channel distortions [HRB$^+$97, DBD$^+$97, HM94, HMBK91]. Thus, in [HM94] a relative reduction in isolated word error rate of more than 75% was observed by using RASTA-PLP instead of PLP when the environment used for training was different from that during recognition (geographically different telephone switching networks).

Since the PhoneBook database contains multi-speaker utterances recorded over various telephone networks a RASTA-PLP preprocessor seems reasonable. Thus, for the experiments reported in this chapter we used a RASTA-PLP cepstral preprocessor that yielded a 26 dimensional feature vector every 10ms based on a 30ms Hamming weighted window. The 26 dimensional feature vector was composed of 12 RASTA-PLP cepstral features, the corresponding Δ-cepstral features and the Δ- and $\Delta\Delta$-log energy. All feature vectors for a given utterance were normalized so that each element in the feature vector had zero mean and unit variance over the utterance. The freely available RASTA-PLP program (`rasta v. 2.3`)[2] was used for calculating the cepstral coefficients and table 9.3 summarizes the parameters used for the calculations (see also the documentation accompanying the `rasta` program).

9.1.3 Training and Decoding

The HNNs were trained in the same manner as for the phoneme recognition experiments, that is, training passed through initialization of the neural networks, initial complete label training for a maximum of 20 epochs and finally complete label training for a maximum of 50 epochs. Training was done by online gradient descent with a stepsize that was adapted according to the performance on the validation set. The initial stepsize and momentum

[2]The RASTA-PLP program is available at `http://www.icsi.berkeley.edu/real/rasta.html`

RastaPLP cepstral preprocessor	
Amplitude compression	μ-law
Sampling frequency	8 kHz
Window length	30 ms
Weighting function	Hamming
Frame shift	10 ms
Spectral analysis	FFT
Frequency warping	Bark-scale
Filterbanks	17
Filterbank shape	Trapezoidal
PLP all-pole model order	8
Features	12 + Δ's + Δ- & $\Delta\Delta$-log energy
Feature normalization	Zero mean, unit variance
Delta regression window	5 frames

Table 9.3: Summary of RASTA-PLP cepstral preprocessor used for the PhoneBook experiments.

parameter were the same as for the phoneme recognition experiments.

For the network initialization and initial complete label training a phone segmentation (complete labeling) was kindly provided by the speech group at Circuit Theory and Signal Processing Lab (TCTS), Faculté Polytechnique de Mons[3] in Belgium. This complete labeling was obtained by doing a forced Viterbi alignment on the training set using a state-of-the-art scaled-likelihood hybrid trained on the NTIMIT database. The NTIMIT database is a version of the TIMIT database that has been passed through a telephone line, see appendix A. A better approach would probably be to use the NTIMIT database directly for initializing the networks and for initial complete label training of the HNN. By using the phone segmentation from another recognizer there is a risk that the network initialization and initial complete label training will result in the HNN "learning" the errors of the reference recognizer. However, the NTIMIT database was unfortunately not available to us at the time where these experiments were carried out.

As for the TIMIT phoneme recognition experiments presented in the previous chapter the computational demand was also very large for the PhoneBook task. Training times for simple models (*i.e.*, HNNs with networks having no hidden units) were on on the order of 1-2 days whereas more complex models typically required 1-2 weeks of CPU time on a fast workstation (Silicon Graphics Onyx with a MIPS R10000 CPU).

As we are considering vocabularies of a fairly limited size decoding can be based on computing the probability of each of the words in the dictionary. That is, for each word the corresponding temporary model is constructed according to the observed phonetic transcription for that word. Then the probability of the acoustic data given this word model is computed by the standard forward or Viterbi algorithm. Because of the task independence, the a priori probabilities are equal for all words and the recognized word is therefore defined by the word model yielding the largest probability. The computational complexity for this kind of decoding is fairly high, but it can still be done several times faster than real-time on a fast work station due to the limited size of the vocabulary. However, larger vocabularies require a different strategy to achieve real-time decoding. For

[3]http://tcts.fpms.ac.be/

Viterbi decoding an efficient approach is to organize the word pronounciations (phonetic transcriptions) in a tree-like structure and then do Viterbi decoding on this recognition network. Similarly, the stack decoder based on forward probabilities can significantly reduce the decoding complexity of the forward decoder by pursuing the best partial hypothesis first. Note that both of these approaches will yield the same result as the above mentioned direct implementations unless pruning is employed.

The experience from the phoneme recognition experiments given in the previous chapters suggests that all-path forward decoding should be used. However, to emphasize the difference between single-path and all-path decoding for discriminatively trained models, *word error rates* for both Viterbi and forward decoding will be given for the PhoneBook task. The word error rates for the training set were calculated only for those 495 utterances that correspond to word list aa. Hereby, only a 75 word dictionary was used during decoding of part of the training set, which lead to a large computational saving compared to using the entire 1581 word vocabulary associated with the training set.

9.2 Minimum Duration versus 3-State Phone Models

The 3-state class submodels used for the TIMIT phoneme experiments are generally acknowledged to be very good for phoneme recognition. However, for word recognition the duration modeling offered by these models has been observed to give poor results compared to *e.g.*, submodels with hardwired minimum duration modeling, see *e.g.*, [HRB+97, BM94]. Therefore, in a first set of experiments we compared three different phone submodel topologies. The HNNs in these experiments used match networks and standard transition probabilities. The match networks had no hidden units and used only the current feature vector as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$).

The first two HNNs were based on the simple 3-state submodel, which was also used for the phoneme experiments, with either a separate match network in each state or tied match networks for all three states. The third topology was based on submodels with hardwired minimum duration modeling. That is, for each of the 46 phones occurring in the transcriptions the submodels had a number of states equal to half the average duration of the phones as was observed in the complete labeling for the training set. No skips were allowed, and all transition probabilities between states in a phone submodel were fixed to 0.5, refer to figure 2.8. Similarly, all states in the submodel for a phone shared the same match network.

For the three kinds of submodels the overall model topology was identical to that used for the TIMIT experiments, *i.e.*, the phone submodels were connected in a "looped" fashion. Because the words in the training and test sets are different the a priori probabilities for the words should be equal as discussed earlier. For this reason the transition probabilities between phone submodels were fixed to stay uniform throughout training. Using non-uniform transition probabilities would implicitly lead to non-uniform a priori probabilities for the words in the training vocabulary. Consider for example a word with associated phonetic transcription $\boldsymbol{y}_1^S$. Non-uniform "bigram" transitions (see figure 5.2) would imply an a priori probability for this word given by $P(\boldsymbol{y}_1^S) = \prod_s P(y_s|y_{s-1})$, where $P(y_s|y_{s-1})$ is equal to the transition probability between the submodel for phone y_{s-1} and y_s. This also implies that the transition-based HNN models with neural network estimated transition scores between phone submodels used for the TIMIT experiments are not suited for the PhoneBook task. Although one can still use transition networks to estimate transition scores within a submodel this is not believed to lead to better performance as discussed in chapter 7. Other submodel topologies like *e.g.*, broad class context sensitive phone models may be well suited to transition-based modeling. For example, one could use the minimum duration phone submodels to model the steady-state regions of the phones and a set of broad class submodels to take care of the transitional segments. Experiments with such topologies have, however, not been carried out in the present work.

Model	Param.	Train(aa)	Test	
		Forw	Vit	Forw
3-state (tied match)	1564	23.2	37.4	26.4
3-state (no tying)	4048	***10.9***	***19.8***	***13.8***
Minimum duration (fixed trans.)	1242	16.0	23.2	17.5

Table 9.4: Word error rates (%) for the 75 word dictionary obtained by different submodel topologies. The match networks use only the current frame as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$) and have no hidden units.

Table 9.4 shows the average word error rate for the eight test sets obtained by the three different submodel topologies for a 75 word dictionary. It is seen that the 3-state submodel topology obtains the lowest error rate for both the training and test sets. As separate match networks are used in each state of this HNN, however, it contains around three times more parameters than the models with tied match networks. Comparing the 3-state submodel with tied match networks to the minimum duration submodel we see that the latter obtains a considerably lower error rate, even though the two topologies yield approximately the same number of parameters. The low number of parameters in these two models also means that practically no overfitting is present as opposed to the 3-state submodel topology with separate match networks.

The error rates for the simple HNNs without hidden units are fairly high compared to those typically encountered for vocabularies of this size. However, taking into account the low number of parameters and the fact that the match networks are basically linear functions of the observation vectors it is quite an impressive performance.

Similar to the phoneme recognition experiments we see that there is a large difference in the word errors obtained by the Viterbi and forward decoder. For *e.g.*, the minimum duration topology the “all-path” decoding gives a word error rate of 17.5%, whereas the Viterbi decoder yields 23.3%. Thus, the forward decoder is able to effectively utilize the pseudo-Poisson duration distribution implied by the minimum duration submodels. Recall that these submodels have a much weaker exponential distribution when the Viterbi decoder is used, refer to figure 2.8. The fact that several paths contribute to the probability of the utterance is illustrated in figure 9.1, showing the state a posteriori probabilities calculated for a test set utterance (word list `ad`). Note that this plot shows the a posteriori probabilities calculated for the *temporary* model constructed according to the observed phonetic transcription (incomplete labeling). An interesting thing to observe from figure 9.1 is that during Viterbi decoding the minimum duration topology is equivalent to a topology where only one of the states in the submodel has a self-loop transition (see figure 2.8).

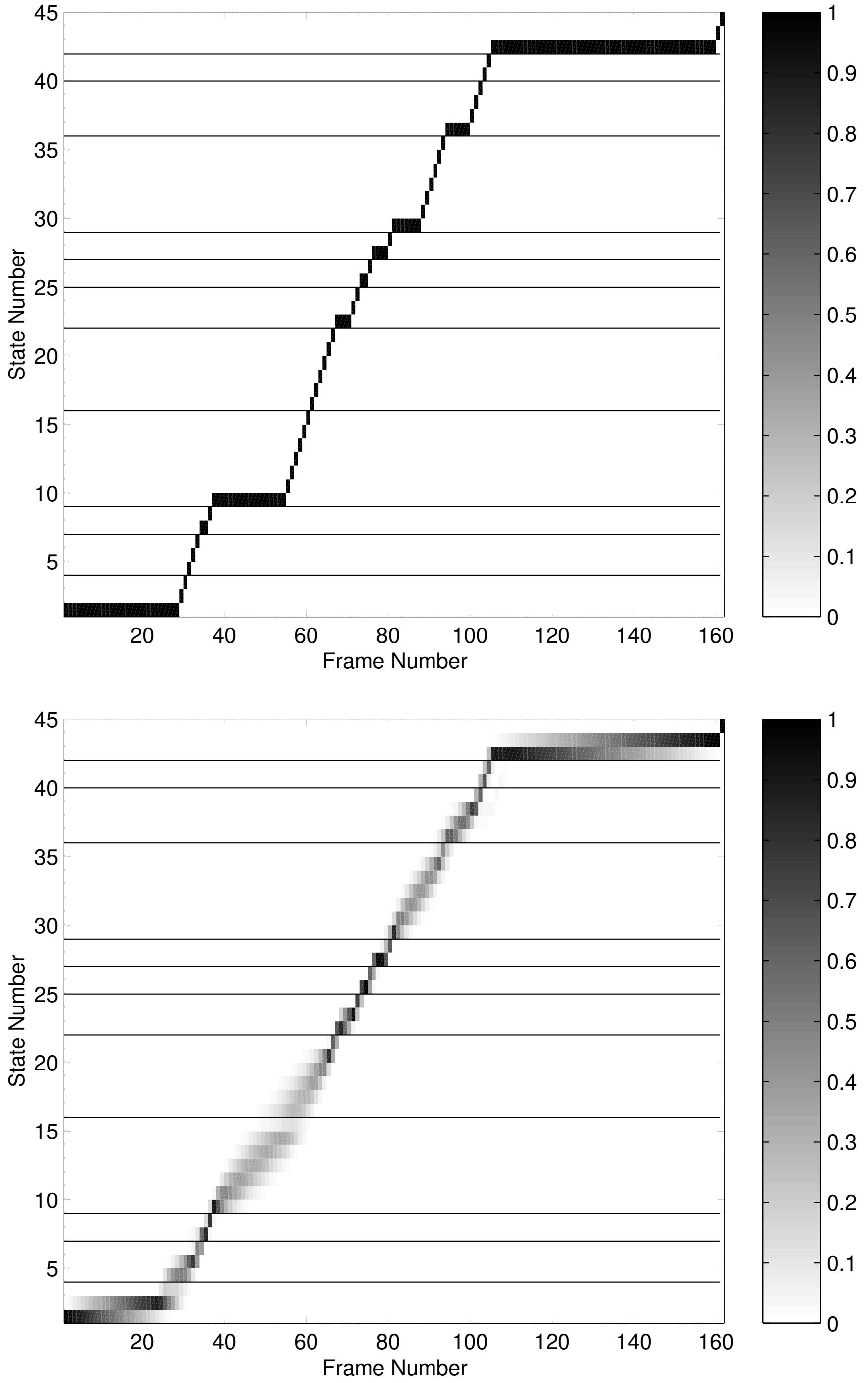

Figure 9.1: Graytone plot of state posterior probabilities for an example word "Bioplasm" from test word list ad (PhoneBook id: ad05-m0k). The state a posteriori probability plots are calculated for the temporary model constructed according to the observed phonetic transcription (/sil bcl b ay ow pcl p l ae z m sil/). The vertical lines indicate the boundary between two phoneme submodels in the temporary model. ***Upper panel:*** Path found by Viterbi decoder (1/0 "posteriors"). ***Lower panel:*** State a posteriori probabilities calculated by the forward-backward algorithm.

9.2.1 Fixed Versus Trained Transitions

The complete labeling from the hybrid trained on NTIMIT is likely to contain a number of errors. This will naturally affect the initial training of the models, but more importantly it might also yield poor estimates of the average duration of each phone. Since the phone durations observed in the training set are used for selecting the number of states in each phone model this can have a crucial impact on performance. To alleviate effects of poor phone duration estimates one can set the transitions in each submodel free instead of fixing them all to 0.5 during training. This will imply that the duration distribution for these models becomes more general and that the average duration they represent becomes different from that observed in the complete labeling.

Transitions	Param.	Train(aa)	Test
All fixed	1242	16.0	17.5
Trained transitions	1576	***12.8***	***15.1***
Trained bigram	3358	15.0	25.6

Table 9.5: Word error rates (%) obtained when training (intra-model) transitions or bigram language model along with the match networks. All match networks only use the current frame as input ($\boldsymbol{s}_l = \boldsymbol{x}_l$) and have no hidden units. Only results for forward decoding are shown.

In table 9.5 the effect of training the (intra submodel) transition probabilities on the word error rate is shown. As can be seen this leads to a significant reduction of the word error and thus illustrates that a better duration model can be learned from the training set (during incomplete label training). A very interesting observation made during this experiment is that the transitions for the states in most of the submodels tend to have identical values even though they are not tied. Thus, for *e.g.*, the submodel corresponding to phone /ey/ all self-loop transitions were approximately equal to 0.75 after training. This submodel contains a total of 6 states and the initial model with $q = 0.5$ implies a pseudo-Poisson duration distribution with an average duration of $\bar{d}_{ey} = 12$ frames (or 120ms). The fact that the self-loop transitions for all states in this model remain "tied" during training indicates that the pseudo-Poisson distribution is appropriate for describing the duration of this phone. However, the value $q \approx 0.75$ after training indicates that the average duration for the /ey/ phone is $\bar{d}_{ey} = 6/0.75 = 8$ frames (or 80ms) rather than 12 frames.

The last row in table 9.5 indicates that training the inter submodel transitions (the "bigram" transitions) leads to a drastic increase of the word error rate for the test set compared to using uniform transitions. This is in agreement with the discussion above.

In all subsequent experiments the transition probabilities within each submodel were trained along with the weights of the match networks.

9.3 Architecture of Match Networks

Table 9.6 and figure 9.2 shows the obtained results for different sizes of the input context to the match networks and for different numbers of hidden units. All match networks are fully connected (except for the one using weight sharing) and the context input is symmetric. From the table we first observe that there is a large difference between Viterbi and forward decoding for all architectures and for both the 75 and 600 word dictionary. Furthermore, it is seen that a large reduction in error rate is obtained when using a symmetric context of one frame as input to the match networks whereas additional context only improves the accuracy slightly. Allowing the match networks to implement non-linear functions by using hidden units gives a further reduction of the error rate. The lowest average error rate of 4.8% for the 75 word and 14.2% for the 600 word dictionary was obtained by an HNN using match networks with 10 hidden units and a context of one frame. This model contains about 37000 trainable parameters and seriously overfits the training data. A better generalization is therefore to be expected if a larger training set is used, *e.g.*, all 19000 utterances in the 21 word list used for training. For task independent training the size of the training set is even more important than for task dependent training because the model should learn to represent "average" phones. If *e.g.*, a particular phone only occurs a few times in the training set in a single or a few words, then the model will tend to learn a context dependent representation of that phone. We will elaborate on this in section 9.4.

Due to the success of the weight sharing technique for the TIMIT experiments this was also attempted for the PhoneBook evaluation. If $M = 5$ hidden units are used for the weight sharing then the number of parameters for the HNN having 10 hidden units is reduced to about 15000, however, at the cost of a slight increase in the error rate, see table 9.6. Since the same match network is used in all states of a submodel this decrease can be explained by the fact that the "adaptive input transform" weight sharing is no longer state specific. Rather, it is now specific to each submodel or phone class. Therefore, the number M of hidden units used for each input feature vector (see figure 7.7) should generally be larger than for submodels with a separate match network in each state. However, if M is too large the weight sharing will only lead to a modest reduction in the number of parameters and the rationale for using this approach is therefore lost.

For the HNN using match networks without hidden units, the 3-state topology was observed to outperform the minimum duration topology. This, however, is not the case for the models using a symmetric context of one frame and hidden units. As shown in table 9.6, the 3-state topology performs considerably worse on both training and test set than the HNN based on minimum duration submodels.

Model	Param.	Train(aa)	Test - 75 word		Test - 600 word	
		Forw	Vit	Forw	Vit	Forw
Minimum duration						
$K = 0$, 0 hidden	1576	12.8	20.8	15.1	43.7	34.5
$K = 1$, 0 hidden	3968	12.4	16.6	13.3	35.3	30.9
$K = 2$, 0 hidden	6360	9.8	15.9	12.8	34.2	30.2
$K = 1$, 5 hidden	18780	3.4	8.4	5.7	21.5	16.1
$K = 1$, 10 hidden	37180	***2.6***	***7.3***	***4.8***	***18.4***	***14.2***
$K = 1$, 20 hidden	73980	2.2	7.9	5.1	17.5	14.5
$K = 1$, 10 hidden, $M = 5$	14536	7.2	8.2	5.7	20.3	16.3
3-state						
$K = 1$, 10 hidden	110814	6.7	15.1	10.1	-	-

Table 9.6: Error rates (%) with a 75 or 600 word dictionary. K is the size of the symmetric context used as input to the match networks, *i.e.*, $\boldsymbol{s}_l = \boldsymbol{x}_{l-K}, \ldots, \boldsymbol{x}_l, \ldots, \boldsymbol{x}_{l+K}$.

The word error rates for the eight test sets obtained by HNN with 10 hidden units and a symmetric input context of one frame are shown in table 9.7 and figure 9.2. We see that the error rates for the two word lists `bd` and `cr` are considerably larger than for any of the other lists. A closer look at the output from the HNN revealed that part of the poor performance for these word lists is due to a few speakers that are hard to recognize. Note that this may very well be due to poor quality handsets used by these speakers or due to atypical channel characteristics for these calls.

List id	75 words		600 words	
	Vit	Forw	Vit	Forw
ad	6.7	3.5	16.0	11.1
ar	4.6	3.6	14.9	12.1
bd	13.6	8.1	27.1	20.1
br	6.4	4.7	19.0	14.9
cd	5.8	3.8	15.6	14.5
cr	9.6	8.2	24.6	20.8
dd	5.4	2.6	12.3	7.3
dr	6.7	4.2	18.0	12.7
ad-dr	7.3	4.8	18.4	14.2

Table 9.7: Error rates (%) for an HNN with match networks having 10 hidden units and a context of one frame. The results are shown for each of the eight word lists.

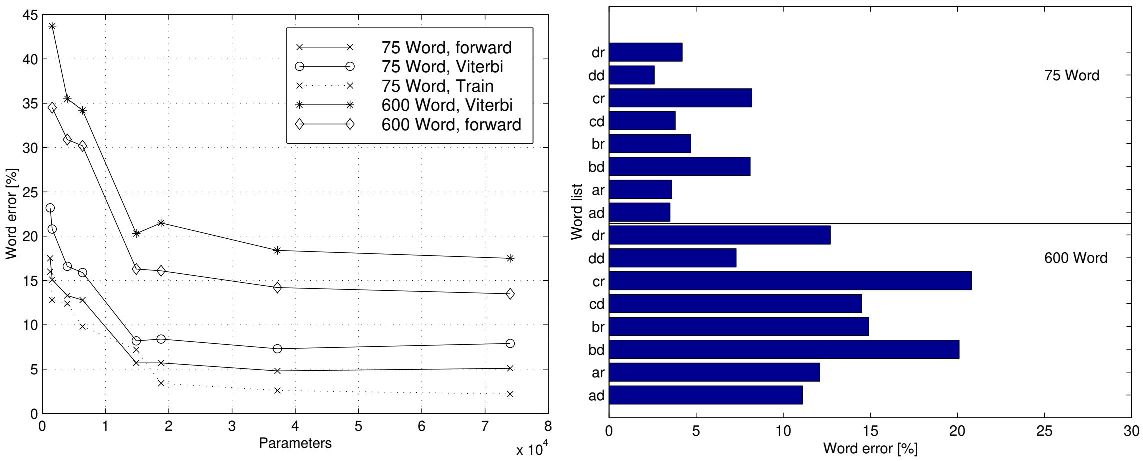

Figure 9.2: Summary of results for HNN based on minimum duration submodel topology. ***Left panel:*** Word error rates as a function of the number of parameters in the HNN. ***Right panel:*** The distribution of word error rates on the eight test sets for the HNN using match networks with 10 hidden units and a symmetric input context of one frame.

9.4 Reported Results

Some of the first results on the PhoneBook database were reported by Dupont *et al.* in [DBD+97]. They compared a Viterbi trained scaled likelihood hybrid to a standard continuous density HMM with diagonal covariance matrices. Both models contained about 160000 parameters and were trained on all 19000 utterances from the same 21 word lists as used in this work. Similarly, the same test set as in this work was used by Dupont *et al.*. For a RASTA-PLP cepstral preprocessor they reported a 75 word dictionary error rate of 5.0% for the HMM and 2.4% for the hybrid, see table 9.8. Similarly, they reported a word error of only 7.6% for the 600 word dictionary. Instead of the RASTA-PLP preprocessor Dupont *et al.* also tried a so-called *Cepstral Means Subtraction* (CMS) preprocessor. The CMS preprocessor aims at reducing the effect of channel distortions and speaker variation by subtracting a speaker specific mean feature vector from all cepstral feature vectors for that speaker. For the training set the mean vector is calculated offline, *i.e.*, using all the utterances from a given speaker. However, during decoding the utterances are only available one at a time and the mean vector must be estimated online, *i.e.*, by updating an initial (zero) mean vector each time an utterance from a given speaker is encountered. The CMS technique substantially reduced the error rate of the Viterbi trained hybrid to 1.5% for the 75 word dictionary and 5.3% for the 600 word dictionary, see table 9.8. The large reduction can by explained by the fact that the CMS technique is actually a very simple implementation of speaker adaption. Speaker adaption during recognition is known to improve performance of speech recognizers significantly because this method basically turns a speaker independent system into a more speaker specific one. Although the CMS method used by Dupont *et al.* does not change any parameters associated with the hybrid itself it *does* change the preprocessor as more data becomes available for a given speaker.

Ref.	Model	N_{train}	Param.	75 word	600 word
[DBD+97]	HMM	19k	162k	5.0	-
	Hybrid (RASTA)	19k	166k	2.4	7.6
	Hybrid (CMS)	19k	166k	***1.5***	***5.3***
[HRB+97]	Viterbi trained	9k	166k	-	13.7
	Viterbi trained	19k	281k	-	9.8
	Forward-backward trained	9k	166k	-	12.2
	Forward-backward trained	19k	281k	-	10.1
This work	HNN, $K = 1$, 0 hidden	9k	***1.6k***	13.3	30.9
	HNN, $K = 1$, 10 hidden	9k	***37k***	4.8	14.2

Table 9.8: HNN and reported word error rates (%) for the PhoneBook task. All models in the table are tested on the same test set and N_{train} is the number of utterances used for training. Hennebert *et al.* used a RASTA-PLP cepstral preprocessor.

The results presented in [DBD+97] are somewhat better than those obtained by the HNN, but the larger training set and the speaker adaptive CMS preprocessing is believed to contribute significantly to the lower error. Indeed, for the same training set as used here, Hennebert *et al.* [HRB+97] reported an error rate for the 600 word dictionary of 13.7% for the same Viterbi trained HMM/NN hybrid as used in [DBD+97], see table 9.8. This is similar to the error rate obtained by the HNN with only 37000 parameters. For the 19,000 word training set [HRB+97] reported an error rate for the 600 word dictionary of 9.8%. This clearly indicates the importance of a large training set for task independent training.

Instead of Viterbi training Hennebert *et al.* [HRB$^+$97] also tried to iteratively reestimate "soft" targets for the neural network as described in chapter 6. For the small training set this resulted in a slightly lower error rate of 12.2% compared to 13.7% for "hard" target Viterbi training. However, for the larger training set no improvement was observed by using "soft" instead of "hard" targets.

9.5 Summary

A large request for commercial isolated word recognizers for many different tasks have recently emerged. Tasks like *e.g.*, telephone home-banking and hands-free operation of mobile phones use a different vocabulary and therefore task independent training is a very important research topic. This chapter described the application of the HNN architecture to speaker and task independent recognition of isolated words uttered over North American telephone lines. For an HNN containing only 37000 parameters a word error rate of only 4.8% was obtained for a 75 word dictionary whereas a larger 600 word dictionary lead to an error rate of 14.2%. These results were obtained by training the model on 9000 utterances of words from a 1581 word vocabulary and compare well to results reported for state-of-the art hybrids and standard HMMs containing at least four times as many parameters. The low number of parameters in the HNNs may be important in practice for hardware implementations and real-time operation.

Chapter 10

Conclusions

This thesis has dealt with various methods for enhancing HMM-based modeling for speech recognition. Specifically, the issues of discriminative training, "all-path" versus "best-path" decoding and HMM/NN hybrid modeling have been investigated. The main conclusions of this work are 1) that discriminative Conditional Maximum Likelihood (CML) training in combination with "all-path" decoding offers a significant advantage over maximum likelihood training and Viterbi decoding and 2) that the HNN hybrid proposed in this work yields better performance than HMMs for speech recognition.

The starting point of this work was on methods for training HMMs to discriminate between classes rather than to learn within-class distributions. Based on a comparison between a number of discriminative training criteria, the conditional maximum likelihood criterion was selected. The arguments in favor of the CML criterion are that it is an intuitive extension of the ML criterion, that it contains no tunable control parameters and that it increases the probability of the correct hypothesis over all competing hypotheses. One potential problem of the CML criterion, however, is that it is very sensitive to mislabelings in the dataset.

A particular extension of the standard HMM, called a Class HMM [Kro94], was proposed as a model that easily allows for CML training. For the CHMM it was discussed how CML training can be implemented by a gradient-based optimization algorithm. Furthermore, it was shown that the CHMM can be normalized globally at the sequence level with no computational overhead during CML training.

To mitigate some of the basic limitations of standard HMMs, the CHMM was combined with neural networks into a CHMM/NN hybrid called a Hidden Neural Network (HNN). The major advantages of the HNN include the ability to use observation context as input, the probabilistic interpretation ensured by global normalization, discriminative CML training and the flexibility of the architecture. CML training of the HNN was in this work implemented by a gradient descent algorithm in which the neural networks are updated by backpropagating errors calculated by a modified forward-backward algorithm. Furthermore, it was found that the HNN can be used as a purely transition-based model and that this particular architecture is very similar to the IOHMM [BF96] and the discriminant HMM/NN hybrid [BKM94]. As opposed to HMMs, which model speech as sequences of steady-state segments, the transition-based HNN can focus on "transitional" regions between steady-state segments. Such "transitional" regions are believed to be important for human perception of speech and the HNN is therefore potentially a better model for speech recognition. The globally normalized transition-based HNN can furthermore prohibit paths through class submodels, which is not possible in locally normalized models like the IOHMM and discriminant HMM/NN hybrid.

10.1 Summary of Experimental Evaluations

In this work three standard speech recognition tasks were used for evaluating discriminative training and the HNN hybrid. The results obtained on these tasks are summarized below.

TIMIT Broad Phoneme Recognition

A comparison between ML and CML training of a discrete CHMM was given for the task of recognizing broad phoneme classes. Through these experiments several topics concerning both discriminative training and decoding were illustrated. It was shown that only one path tends to contribute to the probability of the labeling for ML trained models, whereas several paths contribute for CML estimated models. Thus, the CML trained model reached an "all-path" (N-best) accuracy of 81.3% and a "single-path" (Viterbi) accuracy of only 78.4%. The best result by ML training was 76.1%.

For an HNN using match networks and standard transitions it was illustrated how the match networks in the different states of a 3-state submodel adapt to state-specific tasks. This was found to yield improved performance compared to the use of a single (tied) match network in all states of a submodel. Similarly, it was shown that joint training of match networks and transition probabilities gives a very large gain in accuracy compared to a scaled likelihood hybrid in which the "match network" and HMM are trained separately.

The best performance for transition-based HNNs was obtained by normalizing globally at the sequence level instead of locally at the state level. For 1-state class submodels the transition-based HNN gave a substantially higher accuracy than an HNN using match networks and standard transitions. This illustrates the importance of transition-based modeling for speech recognition. A similar gain in accuracy was, however, not observed for 3-state class submodels. A likely explanation for this is that the 3-state topology is not well suited for transition-based modeling.

The best result of 84.6% on the broad class task was obtained by a globally normalized transition-based HNN containing approximately 5000 parameters. This compares very favorably to the results for the CHMM and to the result of 81.3% reported for a Gaussian mixture density HMM with a linear adaptive input transformation [Joh94].

TIMIT 39 Phoneme Recognition

The findings for the broad class task were also shown to "scale up" to the larger task of recognizing 39 TIMIT phonemes. The best accuracy of 70.2% on the 39 phoneme task was obtained by a globally normalized transition-based HNN containing 111k parameters. This compares well to results reported for standard HMMs. However, the ABBOT recurrent neural network hybrid [Rob94] is still unbeaten with its 75% accuracy on the 39 phoneme task.

PhoneBook Isolated Word Recognition

As a last experiment the HNN was evaluated on the recognition of isolated words from the PhoneBook database. For a 75 word dictionary the HNN yielded an average word error of 4.8%. A larger 600 word dictionary resulted in an average word error of 14.2%. This HNN used minimum duration phone submodels, had 37k parameters and was trained on 9000 utterances. For a scaled likelihood hybrid containing 166k parameters, average word error rates of 1.5% for the 75 word dictionary and 5.3% for the 600 word dictionary were reported in [DBD+97].

10.2 Suggestions for Future Work

The 1-state, 3-state and minimum duration phoneme submodels used in this work are all "developed" in the context of non-discriminative ML training and may not be optimal for CML training. Therefore, other model topologies should be investigated in the context of discriminative CML training. Furthermore, the 1-state and 3-state topologies were used for both "conventional" steady-state modeling and transition-based modeling. Other topologies may, however, be more suitable for transition-based modeling. For example, one could use a single submodel to model *all* steady-state segments and a number of states with transition networks to model "transitional" segments between phonemes. This is similar to the topology proposed in [MBGH96]. One could also try to include explicit parametric or non-parametric duration distributions in the 1-state submodels for transition-based modeling.

Only context independent phoneme submodels were considered in this work. However, context dependent phoneme submodels are known to yield better performance and should consequently be evaluated in the HNN framework. To keep the number of parameters at a reasonable level one could apply tying as commonly done in context dependent modeling with standard HMMs. For example, if biphones are modeled as a left context part and a core part, one could tie match networks between the "core states" across all biphone submodels for the same phoneme. This would lead to a large saving in the number of parameters. An even larger degree of tying can be obtained by allowing some states in the HNN (or CHMM) to model several labels. Thus, with multiple-label-states one can tie "core states" *as well as* "left context states" across all biphone models for the same phoneme. The "core states" should in this case only allow a single label, namely that of the "core" phoneme, whereas the "left context states" should have a class-label distribution over all possible left context phonemes. Note that multiple-label-states are also interesting for "tying" in transition-based modeling because the amount of data that describes "transitional" regions between *e.g.*, phonemes is rather limited. Multiple-label-state CHMMs and HNNs should be fairly straightforward to implement based on the guidelines given in chapter 4 and appendix B.

The HNN has proven very useful for acoustic modeling in speech recognition. It would, however, be interesting to see whether the results reported in this work scale up to large vocabulary continuous speech recognition. Furthermore, it would be interesting to apply the HNN to problems in biological sequence modeling like *e.g.*, protein secondary structure prediction and gene finding.

Appendix A

Databases

In this appendix the TIMIT/NTIMIT and the PHONEBOOK databases used for various experiments are described.

A.1 The TIMIT Database

The TIMIT database [FDGM86, GLF+93] is a corpus of read North American English speech uttered by 630 native speakers from eight major dialect regions in the United States of America. Ten utterances were recorded for each speaker:

- Two dialect sentences, labeled "SA". The dialect sentences should not be used for development and evaluation purposes as the textual content in these sentences is identical for all regions and speakers. The SA sentences are only included to expose the dialectal variations between the speakers in a dialect region.

- Five phonetically-compact sentences labeled "SX" designed to cover all possible bi-phones, which can be generated from the TIMIT phone set, see Table A.1. Each SX sentence is read by seven different speakers.

- Three phonetically-diverse sentences labeled "SI". Each SI sentence is only uttered by one speaker.

Thus, TIMIT contains a total of 6300 sentences of which 5040 can be used for the development and evaluation of speaker independent continuous speech recognition systems. The designers of TIMIT have put some care into defining a recommended training and test set. The recommended training set contains 3696 sentences uttered by 462 speakers, which leaves 1344 sentences uttered by 168 speakers for evaluation purposes. There are no identical speakers or sentences in the training and test portions of the database. Part of the recommended test set is a recommended *core* or "minimum size" test set containing 192 sentences read by 24 speakers, two male and one female from each of the eight dialect regions. As opposed to the full test set all sentences in the core test set are textually different.

All speech in TIMIT is recorded in a quiet studio environment, sampled at 16 kHz and stored in an uncompressed SPHERE format wave-file [FDGM86, GLF+93]. The NTIMIT database [JKBS90] is a version of TIMIT where all the recorded speech in TIMIT has been passed over a North American telephone line. Both databases are available from the Linguistic Data Consortium (LDC), which also distributes a number of other speech databases, text corpera and lexicons.[1]

All sentences in TIMIT have been manually segmented into 61 phones by expert phoneticians, and a short segment of silence has been added to the beginning and end of each sentence. The TIMIT 61 phone set is shown in table A.1 along with example words and phonetic transcriptions. In this thesis we use either a reduced set of 39 phonemes as in *e.g.*, [Lee89, Lee90, KVY93, Joh96, Rob94] for continuous phone recognition experiments or a small set of only five classes for broad phoneme class recognition as in [Joh96, Joh94, JJ94a, You92b]. The reduced 39 phone set is shown in table A.2 and the five broad classes are defined in table A.3. Note, that the glottal stop phone /q/ is deleted from all transcriptions, leaving undefined segments in the speech signal. For complete label training this is handled by simply extending the previous label to the segment covered by /q/, whereas the segment is just left undefined for incomplete label training.

Table A.1: The 61 TIMIT phone set.

Group	Phone	Word	Transcription
Stops	bcl,b	**b**ee	**bcl b** iy
	dcl,d	**d**ay	**dcl d** ey
	gcl,g	**g**ay	**gcl g** ey
	pcl,p	**p**ea	**pcl p** iy
	tcl,t	**t**ea	**tcl t** iy
	kcl,k	**k**ey	**kcl k** iy
	dx	mu**dd**y,dir**t**y	m ah **dx** iy, dcl d er **dx** iy
	q	ba**t**	bcl b ae **q**
Affricates	jh	**j**oke	**dcl jh** ow kcl k
	ch	**ch**oke	**tcl ch** ow kcl k
Fricatives	s	**s**ea	**s** iy
	sh	**sh**e	**sh** iy
	z	**z**one	**z** ow n
	zh	a**zu**re	ae **zh** er
	f	**f**in	**f** ih n
	th	**th**in	**th** ih n
	v	**v**an	**v** ae n
	dh	**th**en	**dh** eh n
Nasals	m	**mom**	**m** aa **m**
	n	**noon**	**n** uw **n**
	ng	si**ng**	s ih **ng**
	em	bott**om**	b aa tcl t **em**
	en	butt**on**	b ah q **en**
	eng	wash**ing**ton	w aa sh **eng** tcl t ax n
	nx	wi**nn**er	w ih **nx** axr
Semivowels and Glides	l	**l**ay	**l** ey
	r	**r**ay	**r** ey
	w	**w**ay	**w** ey
	y	**y**acht	**y** aa tcl t

Continued on next page

[1] See http://www.ldc.upenn.edu/ldc

	hh	**h**ay	**hh** ey
	hv	a**h**ead	ax **hv** eh dcl d
	el	bott**le**	bcl b aa tcl t **el**
Vowels	iy	b**ee**t	bcl b **iy** tcl t
	ih	b**i**t	bcl b **ih** tcl t
	eh	b**e**t	bcl b **eh** tcl t
	ey	b**ai**t	bcl b **eh** tcl t
	ae	b**a**t	bcl b **ae** tcl t
	aa	b**o**tt	bcl b **aa** tcl t
	aw	b**ou**t	bcl b **aw** tcl t
	ay	b**i**te	bcl b **ay** tcl t
	ah	b**u**t	bcl b **ah** tcl t
	ao	b**ou**ght	bcl b **ao** tcl t
	oy	b**oy**	bcl b **oy**
	ow	b**oa**t	bcl b **ow** tcl t
	uh	b**oo**k	bcl b **uh** kcl k
	uw	b**oo**t	bcl b **uw** tcl t
	ux	t**oo**t	tcl t **ux** tcl t
	er	b**ir**d	bcl b **er** dcl d
	ax	**a**bout	**ax** bcl b aw tcl t
	ix	deb**i**t	dcl d eh bcl b **ix** tcl t
	axr	butt**er**	bcl b ah dx **axr**
	ax-h	s**u**spect	s **ax-h** s pcl p eh kcl k tcl t
Others	pau	pause	
	epi	epithentic silence	
	h#	non-speech events (begin/end marks)	

A.2 The PhoneBook Database

The PhoneBook database [PFW+95] is a phonetically-rich, isolated-word, telephone-speech database. The aim of PhoneBook is to serve as a large database of North American English word utterances, where all phones are incorporated in as many segmental/stress contexts as are likely to produce coarticulary variations. Furthermore, there is a large variability in talkers and telephone transmission characteristics, since the words are recorded over the existing North American telephone lines from a set of 1358 native North American speakers. The speakers were chosen to be demographically representative for the United States according to gender (50-50), geography, degree of urbanness of area, age (not under 18), income, level of education and socio-economic status. In addition to the isolated words PhoneBook also contains a number of spontaneous utterances of numbers and money amounts, which can be used for designing spontaneous speech recognizers.

Each speaker made a telephone call using his/her own telephone handset and read up to 76 different isolated words, and uttered three spontaneous digit sequences and one spontaneous money amount. Thus, PhoneBook contains 93667 isolated words and 5081 spontaneous utterances, totaling about 23 hours of speech. This breaks down to 7979 distinct words, each said by an average of 11.7 speakers. All data were collected in 8-bit mu-law digital format.

Table A.2: The reduced 39 TIMIT phone set.

Phone	TIMIT Phone(s)	Phone	TIMIT Phone(s)
b	b	l	l el
d	d	r	r
g	g	w	w
p	p	y	y
t	t	hh	hh hv
k	k	iy	iy
dx	dx	ih	ih ix
None	q	eh	eh
jh	jh	ey	ey
ch	ch	ae	ae
s	s	aa	aa ao
sh	sh zh	aw	aw
z	z	ay	ay
f	f	ah	ah ax ax-h
th	th	oy	oy
v	v	ow	ow
dh	dh	uh	uh
m	m em	uw	uw ux
n	n en nx	er	er axr
ng	ng eng	sil	bcl dcl gcl tcl kcl pau epi h#

Table A.3: TIMIT broad phoneme classes.

Broad class	TIMIT phone
None	q
Vowel (V)	iy ih eh ae ix ax ah ax-h uw uh ao aa ey ay oy aw ow ux
Consonant (C)	ch jh dh b d dx g p t k z zh v f th s sh hh hv
Nasal (N)	m n en ng em nx eng
Liquid (L)	l el r y w er axr
Silence (S)	h# pau

The database is organized in 106 word-lists, each composed of 75 or 76 different words, typically uttered by 11 different speakers. The word-list are labeled $l_1 l_2$ with

$$l_1 \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}\} \tag{A.1}$$

and

$$l_2 \in \begin{cases} \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \ldots, \mathtt{x}, \mathtt{y}, \mathtt{z}\}, & \text{for } l_1 \neq \mathtt{e} \\ \{\mathtt{a}, \mathtt{b}\}, & \text{for } l_1 = \mathtt{e} \end{cases} \tag{A.2}$$

The utterances in PhoneBook are phonetically transcribed using a 42-phone inventory. The phonetic transcriptions contain triphone context information and information about onset, nucleus and coda of simple syllables. Furthermore, three different stress levels are used for the vowels, see [PFW+95] for further details.

Appendix B

Multiple-Label-State Class HMMs

In this appendix we discuss how to do ML estimation for a CHMM with multiple-label-states. It is shown, that for both complete and incomplete labeling this model can be trained by an approach similar to the Baum-Welch algorithm, but the forward-backward algorithms involved are different from the standard case.

This appendix should be read as an extension to chapter 4 describing single-label-state CHMMs and it serves as a reference for future research.

As in chapter 4 we will assume that the observations are discrete symbols from a finite alphabet $\mathcal{A}$, but the discussion below is also valid for continuous observation densities.

B.1 Complete Label ML Estimation

Assume that the observation sequence $\boldsymbol{x}_1^L = x_1, \ldots, x_L$ is associated with a complete label sequence $\boldsymbol{y}_1^L = y_1, \ldots, y_L$. Then the joint likelihood of labels and observation can be written

$$P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\pi}; \boldsymbol{\Theta}). \tag{B.1}$$

Analogous to the standard HMM we find for the complete data likelihood,

$$\begin{aligned} P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\pi}) &= \prod_l P(x_l | \boldsymbol{y}_1^{l-1}, \boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^l) P(y_l | \boldsymbol{y}_1^{l-1}, \boldsymbol{x}_1^l, \boldsymbol{\pi}_1^l) P(\pi_l | \boldsymbol{y}_1^{l-1}, \boldsymbol{x}_1^{l-1}, \boldsymbol{\pi}_1^{l-1}) \\ &\approx \prod_l P(x_l | \pi_l) P(y_l | \pi_l) P(\pi_l | \pi_{l-1}). \end{aligned} \tag{B.2}$$

In addition to the standard HMM assumptions this derivation is based on the assumption that the labels are state conditionally independent. In this way the labels are treated in the same way as the observations, *i.e.*, the probability of matching the observation-label pair (x_l, y_l) at time l in state i is given by the product of match and label probability in state i; $P(x_l | \pi_l = i) P(y_l | \pi_l = i)$. Using $\psi_i(c)$ to denote the probability of label c in state i the joint likelihood becomes,

$$P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}} \prod_l \phi_{\pi_l}(x_l) \psi_{\pi_l}(y_l) \theta_{\pi_{l-1} \pi_l}. \tag{B.3}$$

In line with the EM derivation of the Baum-Welch reestimation equations for the standard HMM given in chapter 2 it is easy to see, that an auxiliary $\mathcal{Q}$-function for the CHMM can be defined by

$$\begin{aligned}
\mathcal{Q}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= E_{\boldsymbol{\pi}}[\log P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\pi};\boldsymbol{\Theta})|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}^{(t)}] \\
&= \sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi}|\boldsymbol{x},\boldsymbol{y};\boldsymbol{\Theta}^{(t)}) \log P(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\pi};\boldsymbol{\Theta}) \\
&= \sum_{li} m_i^{(t)}(l) \log \phi_i(x_l) + \sum_{li} m_i^{(t)}(l) \log \psi_i(y_l) + \sum_{lij} m_{ij}^{(t)}(l) \log \theta_{ij} \\
&= \mathcal{Q}^{\phi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\psi}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\theta}_{\boldsymbol{\pi}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}). \qquad \text{(B.4)}
\end{aligned}$$

Comparing to (2.36) we see that this is almost identical to the auxiliary function for the standard HMM. The only differences are the extra term $\mathcal{Q}^{\psi}_{\boldsymbol{\pi}}$ responsible for the label distribution and a different set of expected counts. The expected count $m_i(l)$ is similar to $n_i(l)$ for a standard HMM and denotes the expected number of times we are in state i at time l *given* the labeling,

$$\begin{aligned}
m_i(l) &= E_{\boldsymbol{\pi}}[\delta_{\pi_l,i}|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}] \\
&= P(\pi_l = i|\boldsymbol{x}_1^L,\boldsymbol{y}_1^L;\boldsymbol{\Theta}). \qquad \text{(B.5)}
\end{aligned}$$

Similarly, $m_{ij}(l)$ denotes the expected number of times we use the transition from state i to state j at time $l-1$ given the labeling,

$$\begin{aligned}
m_{ij}(l) &= E_{\boldsymbol{\pi}}[\delta_{\pi_{l-1},i}\delta_{\pi_l,j}|\boldsymbol{x},\boldsymbol{y},\boldsymbol{\Theta}] \\
&= P(\pi_{l-1} = i, \pi_l = j|\boldsymbol{x}_1^L,\boldsymbol{y}_1^L;\boldsymbol{\Theta}). \qquad \text{(B.6)}
\end{aligned}$$

Since the auxiliary function (B.4) is composed of separate terms for each of the three types of parameters, the Baum-Welch reestimation equations for standard HMM match and transition probabilities are also applicable to the CHMM. All we have to do is to replace the n's by the corresponding m's,

$$\phi_i^{(t+1)}(a) = \frac{\sum_l m_i^{(t)}(l)\delta_{x_l,a}}{\sum_{la'} m_i^{(t)}(l)\delta_{x_l,a'}} = \frac{\bar{m}_i^{(t)}(a)}{\sum_{a'} \bar{m}_i^{(t)}(a')} \qquad \text{(B.7)}$$

and

$$\theta_{ij}^{(t+1)} = \frac{\sum_l m_{ij}^{(t)}(l)}{\sum_{lj'} m_{ij'}^{(t)}(l)} = \frac{\bar{m}_{ij}^{(t)}}{\sum_{j'} \bar{m}_{ij'}^{(t)}}, \qquad \text{(B.8)}$$

where we have defined the accumulated counts $\bar{m}_i(a) = \sum_l m_i(l)\delta_{x_l,a}$ and $\bar{m}_{ij} = \sum_l m_{ij}(l)$. In addition to these reestimation equations, the label probabilities are updated according to

$$\psi_i^{(t+1)}(c) = \frac{\sum_l m_i(l)\delta_{y_l,c}}{\sum_{lc'} m_i(l)\delta_{y_l,c'}} = \frac{\bar{m}_i^{(t)}(c)}{\sum_{c'} \bar{m}_i^{(t)}(c')}. \qquad \text{(B.9)}$$

The expected m-counts can be calculated by a straightforward extension of the standard forward-backward algorithm. By comparing the expression for the CHMM likelihood (B.3) and the standard HMM likelihood (2.9) we see that the expression for the CHMM is obtained by replacing the match probabilities $\phi_{\pi_l}(x_l)$ in the expression for the HMM by $\phi_{\pi_l}(x_l)\psi_{\pi_l}(y_l)$. Based on this observation the modified forward and backward algorithms can be expressed as shown below by making the same substitution.

Algorithm B.1 Forward algorithm $[P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^L; \boldsymbol{\Theta})]$

Definition: $\tilde{\alpha}_j(l) = P(\boldsymbol{y}_1^l, \boldsymbol{x}_1^l, \pi_l = j; \boldsymbol{\Theta})$

Initialization: $\tilde{\alpha}_j(1) = \phi_j(x_1)\psi_j(y_1)\theta_{0j}$,

Recursion: $\tilde{\alpha}_j(l) = \phi_j(x_l)\psi_j(y_l)\sum_i \theta_{ij}\tilde{\alpha}_i(l-1), \quad 1 < l \leq L$

Termination: $P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_i \theta_{iN+1}\tilde{\alpha}_i(L)$

Algorithm B.2 Backward algorithm $[P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^L; \boldsymbol{\Theta})]$

Definition: $\tilde{\beta}_i(l) = P(\boldsymbol{y}_{l+1}^L, \boldsymbol{x}_{l+1}^L | \pi_l = i; \boldsymbol{\Theta})$

Initialization: $\tilde{\beta}_i(L) = \theta_{iN+1}$,

Recursion: $\tilde{\beta}_i(l) = \sum_j \tilde{\beta}_j(l+1)\theta_{ij}\psi_j(y_{l+1})\phi_j(x_{l+1}), \quad 1 \leq l < L$

Termination: $P(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\Theta}) = \sum_{j=1}^N \tilde{\beta}_j(1)\theta_{0j}\psi_j(y_1)\phi_j(x_1)$

As for the standard forward-backward algorithm we note that,

$$P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^L; \boldsymbol{\Theta}) = \sum_i \tilde{\alpha}_i(l)\tilde{\beta}_i(l) \tag{B.10}$$

The expected m-counts are now computed exactly as for the standard HMM (equations (2.46)-(2.47)), but using the forward and backward variables defined in algorithm B.1 and algorithm B.2 and replacing $\phi_i(x_l)$ with $\phi_i(x_l)\psi_i(y_l)$,

$$m_{ij}(l) = \frac{\tilde{\alpha}_i(l-1)\theta_{ij}\phi_j(x_l)\psi_j(y_l)\tilde{\beta}_j(l)}{\sum_{i'} \tilde{\alpha}_{i'}(l)\tilde{\beta}_{i'}(l)} \tag{B.11}$$

and

$$m_i(l) = \frac{\tilde{\alpha}_i(l)\tilde{\beta}_i(l)}{\sum_{i'} \tilde{\alpha}_{i'}(l)\tilde{\beta}_{i'}(l)}. \tag{B.12}$$

B.2 Incomplete Label ML Estimation

The case of incomplete labeling is a bit more difficult to handle than that of complete labels. The problem can be solved by adding an extra *empty* label, $c = \epsilon$, to the set of possible labels. The empty label does not “extend” the label sequence and the label distribution in all states must now satisfy

$$\psi_i(\epsilon) + \sum_{c \in \mathcal{C}} \psi_i(c) = 1. \tag{B.13}$$

This approach is very similar to that proposed by Bengio and Bengio [BB96] for training so-called asynchronous Input/Output HMMs (IOHMMs). In their approach an additional "emit-or-not" distribution is introduced to take care of null labels.

To keep track of the alignment between the observation sequence $\boldsymbol{x} = \boldsymbol{x}_1^L$ and the incomplete label sequence $\boldsymbol{y} = \boldsymbol{y}_1^S$ we will use an additional hidden variable. Let τ_l be equal to s if the s'th label y_s is matched at time l. Since we can either match label y_s or the empty label at any time the difference $\tau_{l-1} - \tau_l$ is either one or zero for any l and $\boldsymbol{\tau} = \boldsymbol{\tau}_1^L = \tau_1, \ldots, \tau_L$ describes the alignment between the two sequences. Using the two hidden variables $\boldsymbol{\pi}$ and $\boldsymbol{\tau}$ the joint likelihood can be expressed in terms of the complete data likelihood as follows

$$P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta}) = \sum_{\boldsymbol{\pi}, \boldsymbol{\tau}} P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S, \boldsymbol{\pi}_1^L, \boldsymbol{\tau}_1^L; \boldsymbol{\Theta}), \tag{B.14}$$

where we have used explicit time super- and subscripts to indicate that the two sequences are of different length. Since we can either match label y_s or the null symbol ϵ in state i at time l the term in the auxiliary function responsible for the labels will now be composed of two terms: One for the null label and one for all other labels,

$$\begin{aligned}
\mathcal{Q}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= E_{\boldsymbol{\pi},\boldsymbol{\tau}}[\log P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\tau}; \boldsymbol{\Theta}) | \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\Theta}^{(t)}] \\
&= \sum_{\boldsymbol{\pi},\boldsymbol{\tau}} P(\boldsymbol{\pi}_1^L, \boldsymbol{\tau}_1^L | \boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta}^{(t)}) \log P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S, \boldsymbol{\pi}_1^L, \boldsymbol{\tau}_1^L; \boldsymbol{\Theta}) \\
&= \mathcal{Q}^{\phi}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\psi}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\theta}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}),
\end{aligned} \tag{B.15}$$

where $\mathcal{Q}^{\phi}_{\boldsymbol{\pi},\boldsymbol{\tau}}$ and $\mathcal{Q}^{\theta}_{\boldsymbol{\pi},\boldsymbol{\tau}}$ are of the same form as for the complete label case, but with a different definition of the expected counts $m_i(l)$ and $m_{ij}(l)$, see below. The label term reads

$$\begin{aligned}
\mathcal{Q}^{\psi}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= \mathcal{Q}^{\psi+}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) + \mathcal{Q}^{\psi-}_{\boldsymbol{\pi},\boldsymbol{\tau}}(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) \\
&= \sum_{lsi} m_i^{(t)+}(l,s) \log \psi_i(y_s) + \sum_{lsi} m_i^{(t)-}(l,s) \log \psi_i(\epsilon).
\end{aligned} \tag{B.16}$$

The expected counts $m_i^+(l,s)$ and $m_i^-(l,s)$ are similar to the expected count $m_i(l)$ defined for the complete label case. $m_i^+(l,s)$ is the expected number of times we are in state i at time l and match the s'th incomplete label y_s given the incomplete label sequence and given that we have matched the partial label sequence $\boldsymbol{y}_1^{s-1}$,

$$\begin{aligned}
m_i^+(l,s) &= E_{\boldsymbol{\pi},\boldsymbol{\tau}}[\delta_{\pi_l,i}\delta_{\tau_l,s} | \tau_{l-1} = s-1, \boldsymbol{x}_1^L, \boldsymbol{y}_1^S, \boldsymbol{\Theta}] \\
&= P(\pi_l = i, \tau_l = s | \tau_{l-1} = s-1, \boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta}).
\end{aligned} \tag{B.17}$$

Similarly, $m_i^-(l,s)$ is the expected number of times we are in state i at time l and match the empty null label given the incomplete label sequence and given that we have matched the partial label sequence $\boldsymbol{y}_1^s$,

$$\begin{aligned} m_i^-(l,s) &= E_{\boldsymbol{\pi},\boldsymbol{\tau}}[\delta_{\pi_l,i}\delta_{\tau_l,s}|\tau_{l-1}=s,\boldsymbol{x}_1^L,\boldsymbol{y}_1^S,\boldsymbol{\Theta}] \\ &= P(\pi_l=i,\tau_l=s|\tau_{l-1}=s,\boldsymbol{x}_1^L,\boldsymbol{y}_1^S;\boldsymbol{\Theta}) \end{aligned} \tag{B.18}$$

These expected counts can be computed by a forward-backward algorithm similar to the one for complete labels. Define the probability of having matched $\boldsymbol{x}_1^l$ and $\boldsymbol{y}_1^s$ and being in state i at time l by $\alpha_i(l,s)=P(\boldsymbol{x}_1^l,\boldsymbol{y}_1^s,\tau_l=s,\pi_l=i;\boldsymbol{\Theta})$. This probability is a sum of two terms because we can either match y_s or the null label in state i at time l,

$$\begin{aligned} \alpha_i(l,s) &= P(\boldsymbol{y}_1^s,\boldsymbol{x}_1^l,\tau_l=s,\tau_{l-1}=s-1,\pi_l=i;\boldsymbol{\Theta})+ \\ &\quad P(\boldsymbol{y}_1^s,\boldsymbol{x}_1^l,\tau_l=s,\tau_{l-1}=s,\pi_l=i;\boldsymbol{\Theta}) \\ &= \alpha_i^+(l,s)+\alpha_i^-(l,s). \end{aligned} \tag{B.19}$$

From this observation the forward algorithm B.3 is easily derived.

Algorithm B.3 Forward algorithm $[P(\boldsymbol{x}_1^L,\boldsymbol{y}_1^S;\boldsymbol{\Theta})]$

Definition: $\alpha_j(l,s)=P(\boldsymbol{y}_1^s,\boldsymbol{x}_1^l,\pi_l=i,\tau_l=s;\boldsymbol{\Theta})$

Initialization: $\alpha_j(1,0)=\phi_j(x_1)\psi_i(\epsilon)\theta_{0j}$,

$\alpha_j(1,1)=\phi_j(x_1)\psi_i(y_1)\theta_{0j}$,

Recursion: $\alpha_j(l,s)=\alpha_j^+(l,s)+\alpha_j^-(l,s)$, $\quad 1<l\leq L, 0\leq s\leq S$

$\alpha_j^+(l,s)=\phi_j(x_l)\psi_j(y_s)\sum_i\theta_{ij}\alpha_i(l-1,s-1)$

$\alpha_j^-(l,s)=\phi_j(x_l)\psi_j(\epsilon)\sum_i\theta_{ij}\alpha_i(l-1,s)$

Termination: $P(\boldsymbol{x}_1^L,\boldsymbol{y}_1^S;\boldsymbol{\Theta})=\sum_i\theta_{iN+1}\alpha_i(L,S)$

Similarly, define $\beta_i(l)=P(\boldsymbol{x}_{l+1}^L,\boldsymbol{y}_{s+1}^S|\pi_l=i,\tau_l=s;\boldsymbol{\Theta})$, *i.e.*, the probability of matching the rest of the observations and labels, given that we are in state $\pi_l=j$ at time l and have matched the first s labels . The backward probability can also be split into a "matching" and "non-matching" term, and the algorithm is shown below.

As for the standard forward-backward algorithm we observe,

$$P(\boldsymbol{x}_1^L,\boldsymbol{y}_1^S;\boldsymbol{\Theta})=\sum_{is}\alpha_i(l,s)\beta_i(l,s). \tag{B.20}$$

The forward-backward algorithm for incomplete labeling has a memory and computational complexity of $\mathcal{O}(SLN)$ which should be compared to $\mathcal{O}(NL)$ for the forward-backward algorithm for standard HMMs. However, it is very unlikely that only a few labels have been matched towards the end of the observation sequence and similarly it is highly unlikely to match all labels within just a few observations. Thus, a considerable decrease in computational complexity can be expected if pruning very unlikely "alignments". In many applications the label sequence will follow the observation sequence linearly on average, and efficient pruning can in such cases reduce the complexity to about that of the standard forward-backward algorithm.

Algorithm B.4 Backward algorithm $[P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta})]$

Define: $\beta_i(l,s) = P(\boldsymbol{y}_{s+1}^S, \boldsymbol{x}_{l+1}^L | \pi_l = i, \tau_l = s; \boldsymbol{\Theta})$

Initialization: $\beta_i(L,S) = \theta_{iN+1}$

Recursion: $\beta_i(l,s) = \beta_i^+(l,s) + \beta_i^-(l,s), \quad 1 \le l < L, 0 \le s \le S$

$\beta_i^+(l,s) = \sum_j \beta_j(l+1,s+1)\theta_{ij}\psi_j(y_{s+1})\phi_j(x_{l+1})$

$\beta_i^-(l,s) = \sum_j \beta_j(l+1,s)\theta_{ij}\psi_j(\epsilon)\phi_j(x_{l+1})$

Termination: $P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S; \boldsymbol{\Theta}) = \sum_j \beta_j(1,1)\theta_{0j}\phi_j(x_1)\psi_j(y_1) + \sum_j \beta_j(1,0)\theta_{0j}\phi_j(x_1)\psi_j(\epsilon)$

From the forward-backward algorithm and our knowledge from the previous sections we can directly write down the expected counts used in the EM-algorithm,

$$m_i^+(l,s) = \frac{\alpha_i^+(l,s)\beta_i(l,s)}{\sum_{i's'} \alpha_{i'}(l,s')\beta_{i'}(l,s')} \tag{B.21}$$

and

$$m_i^-(l,s) = \frac{\alpha_i^-(l,s)\beta_i(l,s)}{\sum_{i's'} \alpha_{i'}(l,s')\beta_{i'}(l,s')}. \tag{B.22}$$

The expected number of times we are in state i at time l is simply the sum over s of $m_i^+(l,s) + m_i^-(l,s)$,

$$\begin{aligned} m_i(l) &= \sum_s m_i^+(l,s) + m_i^-(l,s) \\ &= \frac{\sum_s \alpha_i(l,s)\beta_i(l,s)}{\sum_{i's} \alpha_{i'}(l,s)\beta_{i'}(l,s)}. \end{aligned} \tag{B.23}$$

Because we can either match label y_s or the empty label at time l the expected number of times $m_{ij}(l)$ we use the transition from state i at time $l-1$ to state j at time l is composed of two terms (compare to (2.46) for standard HMMs),

$$\begin{aligned} m_{ij}(l) &= \frac{\sum_s \alpha_i(l-1,s-1)\phi_j(x_l)\psi_j(y_s)\theta_{ij}\beta_j(l,s)}{\sum_{i's} \alpha_{i'}(l,s)\beta_{i'}(l,s)} + \\ &\quad \frac{\sum_s \alpha_i(l-1,s)\phi_j(x_l)\psi_j(\epsilon)\theta_{ij}\beta_j(l,s)}{\sum_{i's} \alpha_{i'}(l,s)\beta_{i'}(l,s)}. \end{aligned} \tag{B.24}$$

As for the standard HMM we see that $m_i(l) = \sum_j m_{ij}(l+1)$ by exploiting the β-recursion and by noting that

$$P(\boldsymbol{x}_1^L, \boldsymbol{y}_1^S, \pi_l = i; \boldsymbol{\Theta}) = \sum_s \alpha_i(l-1,s-1)\beta_i^+(l-1,s-1) + \alpha_i(l-1,s)\beta_i^-(l-1,s) \tag{B.25}$$

The above equality holds because if we are in state i at time $l-1$ then for a particular s we can either have matched $\boldsymbol{y}_1^{s-1}$ or $\boldsymbol{y}_1^s$. The reestimation equations for the match and transition probabilities can be expressed in exactly the same way as for the complete label case, but using $m_i(l)$ defined by (B.23) and $m_{ij}(l)$ defined by (B.24). This is also the case for the label probabilities if we replace $\bar{m}_i(c)$ in (B.9) for the complete label case with,

$$\bar{m}_i(c) = \begin{cases} \sum_{ls} m_i^+(l,s)\delta_{c,y_s} & \text{if } c \in \mathcal{C} \\ \sum_{ls} m_i^-(l,s) & \text{if } c = \epsilon \end{cases}. \tag{B.26}$$

B.3 Decoding Issues for CHMMs

As long as we restrict ourselves to states with only a single label the decoding strategies discussed for standard HMMs still apply with no alterations. When several labels can be modeled in each state the situation is different. For example, it is no longer possible to uniquely map the optimal path found by the standard Viterbi decoder to a sequence of labels. Below we will briefly discuss some possible extensions to the standard decoding algorithms, such that they apply to multiple-label-states.

B.3.1 Viterbi Decoding

The standard Viterbi decoder can by modified to account for states with multiple labels by introducing an additional maximization operation in the Viterbi recursion. This is similar to the Viterbi style decoder proposed for the asynchronous IOHMM in [BB96].

Instead of just selecting the best incoming path we should now select the path with associated partial labeling, which has the highest probability. That is, the Viterbi decoder now finds the most likely path *and* label sequence,

$$(\hat{\boldsymbol{y}}, \hat{\boldsymbol{\pi}}) = \operatorname*{argmax}_{\boldsymbol{y},\boldsymbol{\pi}} P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\pi}; \boldsymbol{\Theta}) = \operatorname*{argmax}_{\boldsymbol{y},\boldsymbol{\pi}} P(\boldsymbol{y}, \boldsymbol{\pi} | \boldsymbol{x}; \boldsymbol{\Theta}). \tag{B.27}$$

The optimal path and associated optimal label sequence can be found by a dynamic programming algorithm similar to the one for the standard Viterbi decoder. The difference is an additional maximization operation in the standard Viterbi recursion,

$$\alpha_i^*(l) = \phi_i(x_l) \max_{c \in \mathcal{C}} [\psi_i(c)] \max_j [\theta_{ji} \alpha_j^*(l-1)], \tag{B.28}$$

where $\mathcal{C}$ is the set of possible labels (including the empty label for incomplete labeling). By keeping the arguments of the maximization operations in the above recursion the most probable path and label sequence can be found by backtracking.

B.3.2 Forward-Backward Decoding

The forward-backward decoder discussed in section 2.7.4 can also be used for multiple-label-state CHMMs by observing that the probability of any label $y_l = c$ (possibly including the empty label) at time l can be expressed as a *mixture* of state-specific label probabilities at time l,

$$P(y_l = c | \boldsymbol{x}; \boldsymbol{\Theta}) = \sum_{i=1}^{N} P(y_l = c | \pi_l = i) P(\pi_l = i | \boldsymbol{x}; \boldsymbol{\Theta}))$$

$$= \sum_{i=1}^{N} \psi_i(c) n_i(l) \tag{B.29}$$

In this mixture distribution, the mixture coefficients are given by the expected counts $n_i(l)$ computed by running the standard forward-backward algorithm on the CHMM. Thus, the forward backward decoding method can be interpreted as a *mixtures of experts* representation of the label distribution at time l. The most probable label at time l is then selected as in section 2.7.4,

$$\hat{y}_l = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, P(y_l = c | \boldsymbol{x}; \boldsymbol{\Theta}). \tag{B.30}$$

Note, that this method applies to both complete and incomplete labeling. For incomplete labeling the label sequence obtained by this method can be folded into an incomplete label sequence by simply removing all occurrences of the null label.

B.3.3 N-best Decoding

The N-best decoder discussed in section 2.7.4 can be extended to the multiple-label-state architecture in a very simple manner. In the N-best decoder for standard HMMs we only extend an incoming partial hypothesis by the label corresponding to the state of the current SMU submodel. For multiple-label-state CHMMs we can extend each incoming partial hypothesis in as many ways as the number of labels the current state can model (possibly including the empty label). Thus, instead of only adding one hypothesis to the token list associated with a state we now add as many new hypothesis as there are different possible labels in the state. The other parts of the algorithm are unchanged.

Appendix C

NORSIG*96 contribution

This appendix contains the paper "Joint Estimation of Parameters in Hidden Neural Networks" presented at the IEEE Nordic Symposium on Signal Processing in Helsinki, Finland. This paper discusses how the neural networks in the HNN are trained by back-propagating errors calculated by the overlying Markov model. An evaluation of complete label CML training is given for the broad class task.

Reference for the paper: [RK96b]

Joint Estimation of Parameters in Hidden Neural Networks

Søren Kamaric Riis[1] and Anders Krogh[2]

[1]Department of Mathematical Modelling
Technical University of Denmark, B305
DK-2800 Lyngby, Denmark
Email: riis@ei.dtu.dk

[2]The Sanger Centre
Hinxton Hall, Hinxton
Cambs CB10 1RQ, UK.
Email: krogh@sanger.ac.uk

ABSTRACT

It has been proven by several authors that hybrids of Hidden Markov Models (HMM) and Neural Networks (NN) yield good performance in speech recognition. However, in many of the current hybrids the HMM and neural networks are trained separately and only combined during decoding. In this paper we propose a new hybrid called Hidden Neural Networks (HNN) where all parameters are trained discriminatively at the same time by maximizing the probability of correct classification. The probability parameters in the HMM are replaced by neural network outputs, and instead of the local normalization of parameters used in standard HMMs the HNN is normalized globally. On the task of classifying TIMIT phonemes into five broad classes the new hybrid obtains a recognition accuracy of 83.7%, whereas a standard HMM obtains 76.1%.

1. INTRODUCTION

It is well known that standard HMMs are based on a number of assumptions which limit their static classification abilities. First of all, it is usually assumed that the Markov process in HMMs is first order and that the observations are independent in the sense that emission probabilities only depend on the current state. Furthermore, the transition probabilities in standard HMMs are time independent and contextual dependencies between observations can only be handled by explicit construction of context dependent models. Some of these assumptions can be relaxed by introducing neural networks to estimate the probability parameters in the HMM. Recently several approaches for combining HMMs and neural networks have been proposed, see *e.g.* [2, 3, 4, 6, 11, 12, 13]. Here we present a new hybrid called Hidden Neural Networks where the usual HMM probabilities are estimated by small neural networks. In the HNN it is possible to assign up to two networks to each state: 1) a *match network* estimating the probability that the current observation matches a given state and 2) a *transition network* that estimates transition probabilities conditioned on observations. One of the two types of networks can be omitted and replaced by standard HMM parameters. In fact all sorts of combinations with standard HMM states are possible.

One of the main ideas in this work is to train the whole HNN supervised, by a joint optimization of parameters (see *e.g.* [3, 4, 5] for similar joint methods). Another important idea is a new way of normalizing the model. Instead of normalizing the model locally, *e.g.*, by using softmax on neural network outputs or by normalizing with class priors, the HNN is normalized globally.

2. THE MODEL

The basic idea of the HNN is to replace the probability parameters of the HMM by neural network outputs that can depend on the context of the observation vector x_l at time l. The emission probability $\phi_i(x_l)$ of observation vector x_l in state i is replaced by a match network $\phi_i(s_l; w^i)$, which is a feed-forward neural network parameterized by weights w^i with input s_l and only one output. The network input s_l corresponding to x_l will usually be a window of context around x_l, *e.g.*, a symmetrical context window of $2K+1$ observation vectors, $x_{l-K}, x_{l-K+1}, \ldots, x_{l+K}$. It can however be any other sort of information related to x_l or the observation sequence in general. Similarly, the probability θ_{ij} of a transition from state i to j is replaced by the output of a transition network $\theta_{ij}(s_l; u^i)$, which is parameterized by weights u^i. The transition network assigned to state i has $\mathcal{J}_i$ outputs, where $\mathcal{J}_i$ is the number of (non-zero) transitions *from* state i.

In complete analogy with the likelihood $p(x|\mathcal{M})$ of a HMM for observation sequence $x = x_1, \ldots, x_L$, we define the quantity

$$q(x|\mathcal{M}) = \sum_\pi q(x, \pi|\mathcal{M}) \qquad (1)$$

with

$$q(x, \pi|\mathcal{M}) = \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l}(s_{l-1}; u^{\pi_{l-1}})\phi_{\pi_l}(s_l; w^{\pi_l}) \qquad (2)$$

where $\mathcal{M}$ denotes the whole model, *i.e.*, all the network parameters, and the state sequence $\pi = \pi_1, \ldots, \pi_L$ is a particular path through the model. We define $\pi_0 = 0$ and $\theta_{0i}(s_0; u^0)$ is the probability of initiating a path in state i. s_0 is the context we choose to associate with the beginning of the sequence. The global normalization is insured by explicit normalization of q,

$$p(x|\mathcal{M}) = \frac{q(x|\mathcal{M})}{\int_{x' \in \mathcal{X}} q(x'|\mathcal{M})dx'} \qquad (3)$$

The integration in the denominator is taken over the space of observation vector sequences $\mathcal{X}$, but as we shall see below, we will never need to calculate this normalization. Apart from making this model much more elegant from a mathematical and computational point of view, we also believe that it may be beneficial to give up local normalization of parameters even for standard HMMs. In fact, non-normalizing parameters are already used frequently in speech recognition by introducing so-called "transition biases" and "stream exponents", see *e.g.* [6, 8, 13]. These heuristic approaches are used in order to reduce the mismatch between transition and emission probabilities in standard HMMs. In [10, 13] these issues are discussed in greater detail.

2.1. Training and decoding

To train the model we assume that the *complete labeling* is available[1] , *i.e.*, that each observation x_l has an associated label y_l corresponding to the class to which it belongs. In order to maximize the prediction accuracy we choose parameters so as to maximize,

$$P(y|x, \mathcal{M}) = \frac{p(x, y|\mathcal{M})}{p(x|\mathcal{M})} \quad (4)$$

as we have previously proposed in [9] (where it was called CHMM for 'class HMM'). This has also been called Conditional Maximum Likelihood (CML) and is equivalent to Maximum Mutual Information estimation (MMI) [1, 7] if the language model is fixed. $p(x, y|\mathcal{M})$ is calculated as a sum over all paths consistent with the labeling, *i.e.*, if observation l is labeled by f only paths in which the l-th state has label f are allowed. If the set of these consistent paths is called $\mathcal{A}(y)$ we have,

$$p(x, y|\mathcal{M}) = \frac{q(x, y|\mathcal{M})}{\sum_{y'} \int_{x' \in \mathcal{X}} q(x', y'|\mathcal{M}) dx'} \quad (5)$$

with,

$$q(x, y|\mathcal{M}) = \sum_{\pi \in \mathcal{A}(y)} q(x, \pi|\mathcal{M}). \quad (6)$$

Since,

$$\sum_{y'} \int_{x' \in \mathcal{X}} q(x', y'|\mathcal{M}) dx' = \int_{x' \in \mathcal{X}} q(x'|\mathcal{M}) dx' \quad (7)$$

equation (4) becomes

$$P(y|x, \mathcal{M}) = \frac{q(x, y|\mathcal{M})}{q(x|\mathcal{M})} \quad (8)$$

and the normalizing factor has conveniently disappeared. Both $q(x|\mathcal{M})$ and $q(x, y|\mathcal{M})$ can be calculated by a straight-forward extension of the forward algorithm, see [9, 10].

[1] In speech recognition this is not always the case. Often only the sequence of utterance symbols, *e.g.* the phoneme transcription, is known (*incomplete labeling*). However, the framework presented here can easily be adapted to the case of incomplete labeling, see [7, 10].

To optimize (8) we use gradient descent. Calculating the derivative of $\log P(y|x, \mathcal{M})$ w. r. t. a weight in the match or transition networks, yields backpropagation training of the neural networks based on an error signal calculated by the forward–backward algorithm, see [10] for details. This has also been observed by several other authors.

In agreement with [6] we have found that Viterbi decoding does not perform well for discriminatively trained models. The reason is that the model is optimized so as to maximize the probability of the labeling, which need not correspond to the most probable path found by Viterbi decoding. Instead one can select the most probable label y_l^* at time l by assuming that consecutive labels are independent. This is done by calculating the sums of state posterior probabilities $P(\pi_l|x, \mathcal{M})$ at time l for those states that carry the same label. The largest of these sums then identify the most probable label. Since the posteriors $P(\pi_l|x, \mathcal{M})$ are readily found using the forward-backward algorithm [7], we call it forward-backward decoding. This type of decoding, however, sometimes results in label sequences that do not correspond to possible paths through the model, and the assumption of independent labels is obviously not good. A better type of decoding which gives label sequences consistent with legal paths in the model is N-best decoding [6, 14]. Results will be reported using both forward-backward and 1-best decoding (in the latter case a maximum of 10 active hypotheses is allowed in each state during decoding).

3. EXPERIMENTS

We have selected the task of predicting five broad phoneme classes in the TIMIT database: Vowels (V), Consonants (C), Nasals (N), Liquids (L) and Silence (S). The five classes are highly confusable and covers all phonetic variations in American English.

We used one sentence from each of the 462 speakers in the TIMIT training set as training set, and the results are reported for the recommended TIMIT core test set. These are also the training and test sets used in [5, 6] for the same task. An independent crossvalidation set is used to monitor the performance of the model during training.

Our preprocessor outputs an observation vector every 10ms consisting of 26 features: 12 mel scaled cepstral coefficients, 1 log energy coefficient and the corresponding derivatives. The observation vectors are normalized to zero mean and unit variance in order to speed up training of the HNN. For the discrete HMMs we used a codebook of 256 prototype feature vectors.

The models used are simple three state left-to-right models for each of the five classes, and only the middle state has a selfloop. In the HNN we only use match networks and let the transitions be the

Table 1: *Test set recognition accuracies for the five broad classes obtained by aligning the predicted and observed phoneme transcriptions. The accuracy is given by:* $\%Acc = 100\% - \%Ins - \%Del - \%Sub$, *where* $\%Ins$, $\%Del$ *and* $\%Sub$ *are the number of insertions, deletions and substitutions respectively. Results for both forward-backward (F-B) and 1-Best decoding are shown.*

Model	#Par.	F-B	1-Best
HMM (ML)	3856	75.5	76.1
HMM (CML)	3856	78.6	79.0
No hidden units			
HNN, $K = 0$	436	77.3	77.3
HNN, $K = 1$	1216	78.4	78.9
HNN, $K = 2$	1996	78.8	79.3
10 hidden units			
HNN, $K = 0$	4246	81.0	83.0
HNN, $K = 1$	12046	82.8	83.7
HNN, $K = 2$	19846	82.1	83.0

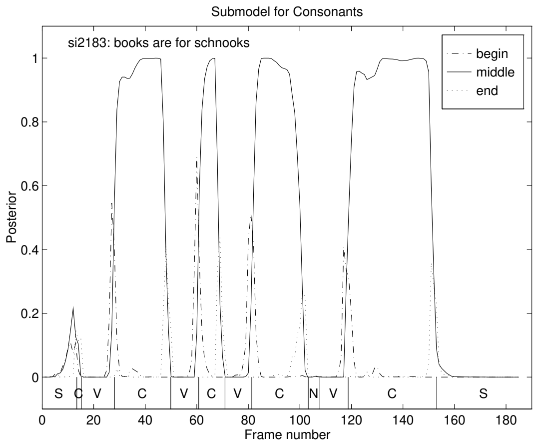

Fig. 1: *State posterior probabilities* $P(\pi_l|x, \mathcal{M})$ *for the three states in the consonant submodel for test sentence "si2183".*

usual time independent transitions. Note however, that the transitions are also trained discriminatively in the HNN as opposed to MMI trained HMMs. The match networks have a symmetric input window of $2K + 1$ frames and a sigmoid output function. With this model setup we found that transition networks did not increase performance. The match networks are initially trained by standard backpropagation to classify the observations into the five broad classes. This speeds up training of the HNN considerably and the models are less prone to getting stuck in local minima. Thus, the performance on the crossvalidation set reaches a maximum within less than 30 epochs (full sweeps through the training set) for all tested models and training times are therefore less than two days on a HP735 workstation. All models except the ML estimated HMM are trained discriminatively by minimizing the negative logarithm of (8) with standard gradient descent.

From table 1 it is observed that 1-best decoding gives slightly higher recognition accuracies compared to forward-backward decoding. This is to be expected since the 1-best decoder selects the transcription that maximizes the full model likelihood (at least approximatively), whereas the forward-backward decoder is based only on "local" label posterior probabilities for each frame.

Compared to the ML trained discrete HMM a gain of about 3% in accuracy is obtained by using discriminative training, see table 1. In [6] an accuracy of 69.3% is reported for a ML trained continuous density HMM with a mixture of six diagonal covariance gaussians per state. Thus, for approximately the same number of parameters the ML trained discrete HMM outperforms the continuous density HMM by more than 6%. For a MMI trained model with a single diagonal covariance gaussian per state they report an accuracy of 72.4%, compared to our 79.0% for the CML trained discrete HMM.

Eventhough the HNN with zero hidden units and no context ($K = 0$) contains almost ten times less parameters than the ML trained HMM it performs considerably better. By increasing the input window width, the HNN performance improves above that of the CML trained HMM, see table 1. No further improvement was observed for contexts larger than $K = 2$. It is interesting to note that the match networks in the HNN without hidden units actually just implements linear weighted sums of input features. Adding hidden units to the HNN drastically increases performance even for small input windows, see table 1. Thus, for approximately the same number of parameters, the HNN with no context ($K = 0$) outperforms the CML trained HMM by 4% in accuracy. The best HNN with a context of one frame ($K = 1$) yields a recognition accuracy of 83.7% which is more than 4% better than the CML trained HMM, and almost 8% better than the ML trained HMM. For comparison, [6] uses a multi-layer perceptron to transform the feature vectors for the continuous density HMM with a single diagonal covariance gaussian per state. This hybrid is trained by MMI and 1-best decoding gives an accuracy of 78.5%. The results in [6] have later been improved to 81.3% by using a linear encoding instead of the multi-layer perceptron [5], and by using multiple CML training passes and non–normalizing transition probabilities.

For a chosen sentence the state posterior probabilities provided by the HNN with 10 hidden units and $K = 1$ are plotted for the the consonant submodel states in fig. 1. It is observed, that the posteriors for the "begin" and "end" states are only large at the class boundaries, whereas the posterior for the "middle" state is large only between these boundaries. Hence, the model is very good at discriminating between different classes. This is also verified in

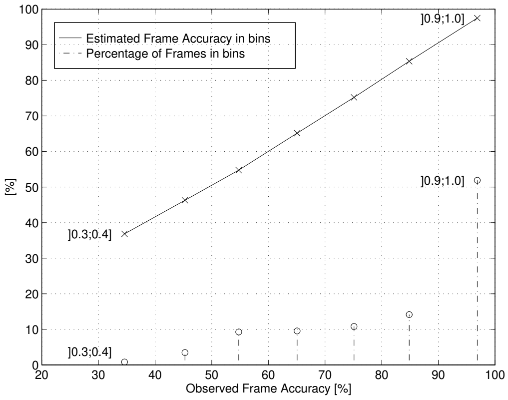

Fig. 2*: The solid line shows the average label posterior probability of the winning class* $\bar{P}(y_l^*|x, \mathcal{M})$ *ranked into 7 equally spaced bins and plotted as a function of the observed frame accuracy in these bins. The dashed lines show the percentage of frames falling into these bins, and are located at the corresponding observed accuracies.*

fig. 2 where it is observed that more than 50% of the frames have a winning-label posterior probability $P(y_l^*|x, \mathcal{M})$ larger than or equal to 0.9. This corresponds to an observed frame recognition rate of more than 95%. Furthermore, there is an almost perfect correlation between the observed frame accuracy and the average label posterior of the winning class. Thus, the larger $P(y_l^*|x, \mathcal{M})$ the more confident is the prediction.

4. CONCLUSION

It has been shown that the HNN combination of neural networks and a hidden Markov model yields a better recognition ability than a standard HMM on the task of classifying TIMIT phonemes into five broad classes. In addition it was illustrated that the HNN provides very accurate estimates of label posterior probabilities. Hence, regions of speech that are predicted with high confidence can easily be identified. The particular architecture introduced here is characterized by: 1) All parameters are trained discriminatively at the same time 2) Proper probability distributions are obtained by global normalization as opposed to local normalization, and 3) Large flexibility in that not all of the HMM parameters need to be replaced by neural networks. In the future we will test the method on larger speech recognition tasks and on problems in molecular biology.

ACKNOWLEDGMENTS

The authors would like to thank Steve Renals for valuable comments and suggestions to this work. The Sanger Centre is supported by the Wellcome Trust.

REFERENCES

[1] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proceedings of ICASSP'86*, pp. 49–52, 1986.

[2] P. Baldi and Y. Chauvin, "Protein modeling with hybrid hidden Markov model/neural network architectures," in *Proceedings of the 3rd ISMB'95*, pp. 39–47, 1995.

[3] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-hidden Markov model hybrid," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 252–9, 1992.

[4] P. Frasconi and Y. Bengio, "An EM approach to grammatical inference: input/output HMMs," in *Proceedings of the 12th ICPR'94*, pp. 289–94, 1994.

[5] F. Johansen, "Global optimisation of HMM input transformations," in *Proceedings of ICSLP'94*, pp. 239–42, 1994.

[6] F. Johansen and M. Johnsen, "Non-linear input transformations for discriminative HMMs," in *Proceedings of ICASSP'94*, pp. 225–28, 1994.

[7] B. Juang and L. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–72, 1991.

[8] S. Kapadia, V. Valtchev, and S. Young, "Mmi training for continuous phoneme recognition on the timit database," in *Proceedings of ICASSP'93*, pp. 491–4, 1993.

[9] A. Krogh, "Hidden Markov models for labeled sequences," in *Proceedings of the 12th ICPR'94*, pp. 140–4, 1994.

[10] A. Krogh and S. Riis, "Hidden neural networks,". In preparation.

[11] E. Levin, "Word recognition using hidden control neural architecture," in *Proceedings of ICASSP'90*, pp. 433–6, 1990.

[12] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in hmm speech recognition," *IEEE Trans. on Speech and Audio Proc.*, vol. 2, no. 1, pp. 161–74, 1994.

[13] T. Robinson, M. Hochberg, and S. Renals, *Automatic speech and speaker recognition – Advanced topics*. Dordrecht, Netherlands: Kluwer, 1995.

[14] R. Schwarz and Y.-L. Chow, "The n-best algorithm: An efficient and exact procedure for finding the n most likely hypotheses," in *Proceedings of ICASSP'90*, pp. 81–84, 1990.

APPENDIX D

ICASSP*97 CONTRIBUTION

This appendix contains the paper "Hidden neural networks: A framework for HMM/NN hybrids" presented at the 1997 International Conference on Acoustics, Speech and Signal processing held in Münich, Germany. In this paper, the results obtained by incomplete label training of the HNN on the broad class task are compared to the complete label training results presented at NORSIG'96 (see appendix C). Furthermore, a set of preliminary results on the TIMIT 39 phoneme task is given.

Reference for the paper: [RK97]

HIDDEN NEURAL NETWORKS: A FRAMEWORK FOR HMM/NN HYBRIDS

Søren Kamaric Riis[1] *Anders Krogh*[2,*]

[1]Dept. of Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, DK (sr@imm.dtu.dk)
[2]The Sanger Centre, Wellcome Trust Genome Campus, Hinxton, Cambs, CB10 1SA, GB (krogh@cbs.dtu.dk)

ABSTRACT

This paper presents a general framework for hybrids of Hidden Markov models (HMM) and neural networks (NN). In the new framework called Hidden Neural Networks (HNN) the usual HMM probability parameters are replaced by neural network outputs. To ensure a probabilistic interpretation the HNN is normalized globally as opposed to the local normalization enforced on parameters in standard HMMs. Furthermore, all parameters in the HNN are estimated simultaneously according to the discriminative conditional maximum likelihood (CML) criterion. The HNNs show clear performance gains compared to standard HMMs on TIMIT continuous speech recognition benchmarks. On the task of recognizing five broad phoneme classes an accuracy of 84% is obtained compared to 76% for a standard HMM. Additionally, we report a preliminary result of 69% accuracy on the TIMIT 39 phoneme task.

1. INTRODUCTION

Among speech research scientists it is widely believed that HMMs are one of the best and most successful modelling approaches for acoustic events in speech recognition. However, common assumptions like state conditional observation independence and time independent transition probabilities limit the classification abilities of HMMs. These assumptions can be relaxed by introducing neural networks in the HMM framework and recently several approaches for HMM/NN hybrids have been proposed, see *e.g.* [1, 2, 3, 4, 5, 6, 7, 8]. In [6] a standard feedforward NN is trained separately to estimate phoneme posterior probabilities. These posteriors are scaled with observed phoneme frequencies and replace the usual emission densities in a HMM during decoding. A similar approach is taken in [7], but here a recurrent NN is used. Instead of training the HMM and the NN separately several authors have proposed architectures where all parameters are estimated simultaneously. In some hybrids [2, 4, 5] a feedforward NN or recurrent NN [8] performs an adaptive input transformation of the observation vectors. Thus, the network outputs are used as new observation vectors in a continuous density HMM and simultaneous estimation of all parameters is performed by backpropagating errors calculated by the HMM into the NN.

Our approach is somewhat similar to the idea of adaptive input transformations, but instead of retaining the computationally expensive mixture densities we propose to *replace* some or all HMM probability parameters by the output of small state specific neural networks. Simultaneous estimation of all parameters can then be performed similar to what is done for adaptive input transformations. A proper probabilistic interpretation is guaranteed by normalizing the model globally as opposed to the often approximate local normalization enforced in many existing hybrids, *e.g.*, using softmax on neural network outputs or using prior scaled network outputs. However, calculating the normalization term can be avoided if the discriminative CML criterion is used for training.

2. THE HNN

In the HNN it is possible to assign up to two networks to each state: 1) a *match network* estimating the probability that the current observation matches a given state and 2) a *transition network* that estimates transition probabilities conditioned on observations. This is very similar to the IOHMM architecture [3] although training and decoding of the IOHMM differs somewhat from that of the HNN. One of the two types of networks in each HNN state can be omitted and replaced by standard HMM parameters. In fact all sorts of combinations with standard HMM states are possible. Instead of using state specific match networks we could use one big network with the same number of outputs as there are states in the HNN. This would correspond to tying weights between input and hidden units in all the match networks.

More formally the HMM emission probability $\phi_i(x_l)$ of observation vector x_l in state i is replaced by a match network $\phi_i(s_l; w^i)$, which is parameterized by weights w^i with input s_l and only one output. The network input s_l corresponding to x_l will usually be a window of context around x_l, *e.g.*, a symmetrical context window of $2K+1$ observation vectors, $x_{l-K}, x_{l-K+1}, \ldots, x_{l+K}$. It can however be any sort of information related to x_l or even the observation sequence in general. Similarly, the probability θ_{ij} of a transition from state i to j is replaced by the output of a transition network $\theta_{ij}(s_l; u^i)$, which is parameterized by weights u^i. The transition network assigned to state i has $\mathcal{J}_i$ outputs, where $\mathcal{J}_i$ is the number of (non-zero) transitions *from* state i. The neural networks in the HNN can be standard feedforward networks or recurrent networks. In fact, they need not even be neural networks — they can be any smooth mapping defined on the space of observation feature vectors.

In complete analogy with the likelihood $p(x|\mathcal{M})$ of a HMM for observation sequence $x = x_1, \ldots, x_L$, we define the quantity

$$q(x|\mathcal{M}) = \sum_\pi q(x, \pi|\mathcal{M}) \qquad (1)$$

*Present address: Center for Biological Sequence Analysis, Technical University of Denmark, 2800 Lyngby, Denmark with

$$q(x,\pi|\mathcal{M}) = \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l}(s_{l-1}; u^{\pi_{l-1}})\phi_{\pi_l}(s_l; w^{\pi_l}) \quad (2)$$

where $\mathcal{M}$ denotes the whole model, *i.e.*, *all* parameters, and the state sequence $\pi = \pi_1, \ldots, \pi_L$ is a particular path through the model. We define $\pi_0 = 0$ and $\theta_{0i}(s_0; u^0)$ is the probability of initiating a path in state i. s_0 is the context we choose to associate with the beginning of the sequence. The probabilistic interpretation is ensured by explicit normalization of q,

$$p(x|\mathcal{M}) = \frac{q(x|\mathcal{M})}{\int_{x'\in\mathcal{X}} q(x'|\mathcal{M})dx'} \quad (3)$$

The integration in the denominator is taken over the space of observation vector sequences $\mathcal{X}$. For the HNN it is generally not possible to calculate the normalization in (3), since this would require knowledge of the network outputs for all possible inputs. However, as shown below, we do not need to compute the normalization term.

Note that the likelihood definition (3) is also applicable to conventional HMMs. Actually, a discrete HMM with non-normalizing parameters is equivalent to the so called Boltzmann Chain introduced in [9].

Apart from making this model elegant from a mathematical and computational point of view, we also believe that it may be beneficial to give up local normalization of parameters even for standard HMMs. In fact, non-normalizing parameters are already used frequently to increase speech recognition accuracy by introducing so-called "transition biases" and "stream exponents", see *e.g.* [7, 10].

2.1. Training and decoding

Assume that the *complete labeling* is available, *i.e.*, that each observation x_l has an associated label y_l corresponding to the class to which it belongs. In speech recognition the classes could be distinct phonemes. Similarly, assume that each state in the HNN is assigned a class label. To maximize the prediction accuracy we choose parameters so as to maximize the conditional likelihood of the observed labeling[1] $y = y_1, \ldots, y_L$,

$$P(y|x,\mathcal{M}) = \frac{p(x,y|\mathcal{M})}{p(x|\mathcal{M})} \quad (4)$$

as we have previously proposed in [11]. Maximizing (4) is known as Conditional Maximum Likelihood estimation (CML) and is equivalent to Maximum Mutual Information estimation (MMI) [12, 13] if the language model is fixed during training. $p(x,y|\mathcal{M})$ is calculated as a sum over all paths consistent with the labeling, *i.e.*, if observation l is labeled f only paths in which the l-th state has label f are allowed. If the set of these consistent paths is called $\mathcal{A}(y)$ we have,

$$p(x,y|\mathcal{M}) = \frac{q(x,y|\mathcal{M})}{\sum_{y'}\int_{x'\in\mathcal{X}} q(x',y'|\mathcal{M})dx'} \quad (5)$$

with,

$$q(x,y|\mathcal{M}) = \sum_{\pi\in\mathcal{A}(y)} q(x,\pi|\mathcal{M}). \quad (6)$$

[1]The extension to multiple training sequences is straightforward by assuming that training sequences are independent.

Since $\sum_{y'} q(x',y'|\mathcal{M}) = q(x'|\mathcal{M})$ the normalization is the same in (3) and (5), so

$$P(y|x,\mathcal{M}) = \frac{q(x,y|\mathcal{M})}{q(x|\mathcal{M})} \quad (7)$$

and the normalizing factor has conveniently disappeared. Both $q(x|\mathcal{M})$ and $q(x,y|\mathcal{M})$ can be calculated by a straightforward extension of the forward algorithm, see *e.g.* [11].

Using the above framework for phoneme recognition requires that the phoneme segmentation (complete labels) of the spoken utterance is known. However, in continuous speech recognition the desired output of the recognizer is not the phoneme label corresponding to each speech frame (*complete labeling*), but rather the phoneme transcription (*incomplete labeling*) [13]. Similar to procedures for standard HMMs (*e.g.* [5, 10]) we can build a "transcription model" by concatenating phoneme submodels according to the observed phoneme transcription of a given training utterance. Analogous to (4) and (7) the goal is now to maximize,

$$P(w|x,\mathcal{M}) = \frac{p(x,w|\mathcal{M})}{p(x|\mathcal{M})} = \frac{p(x|\mathcal{M}_w)}{p(x|\mathcal{M})} = \frac{q(x|\mathcal{M}_w)}{q(x|\mathcal{M})} \quad (8)$$

where w is the phoneme transcription corresponding to x and $\mathcal{M}_w$ is the transcription model. In this work training using the complete or incomplete labeling is called *complete* and *incomplete label training* respectively.

To maximize (7) or (8) we use stochastic online gradient ascent augmented by a momentum term, where the parameter update is performed after each observation sequence. We found that online gradient ascent with an adaptive stepsize yields considerably faster convergence than batch training (updating after presenting the entire training set). Calculating the derivative of $\log P(y|x,\mathcal{M})$ w.r.t. a weight in the match or transition networks yields backpropagation training of the neural networks based on an error signal calculated by the forward–backward algorithm, see *e.g.* [2, 5]. Thus, for each sentence a forward-backward pass is followed by backpropagation training of the neural networks.

In agreement with [5] we have found that Viterbi decoding does not perform well for discriminatively trained models. A probable reason for this is that the Viterbi path through the model need not correspond to the optimal transcription. Ideally the decoder should output the labeling that maximizes the full likelihood of the model. In this work we use the computationally efficient full likelihood based N-best decoder proposed in [14]. This decoder allows multiple active partial transcriptions in each state during a Viterbi style decoding. For computational reasons it is usually necessary to limit the number of active partial transcriptions in each state. We use an upper limit of 10 active partial transcriptions and only the most likely transcription is considered after decoding.

3. GENERAL EXPERIMENTAL SETUP

Initially the HNN has been evaluated on the task of recognizing five broad phoneme classes in the TIMIT database: Vowels (V), Consonants (C), Nasals (N), Liquids (L) and Silence (S). We use one sentence from each of the 462 speakers in the TIMIT training set for training, and the results are reported for the recommended TIMIT core test set. An independent crossvalidation set is used for selecting the best performing model after training.

The preprocessor outputs an observation vector every 10ms consisting of 26 features: 12 mel scaled cepstral coefficients, 1 log energy coefficient and the corresponding delta coefficients. These vectors are normalized to zero mean and unit variance in order to speed up training of the HNN.

Each of the five classes are modelled by a simple left-to-right three state model. The last state in any submodel is fully connected to the first state of all other submodels. Since we did not observe any improvements using transition networks all models use standard HMM transition probabilities. Note however, that transitions between submodels are also trained discriminatively when using the CML criterion as opposed to the MMI criterion.

Our baseline system is a standard discrete HMM using a codebook of 256 prototype vectors (3856 free parameters). In the HNN we replace the emission distribution by fully connected match networks with a symmetric input window of $2K+1$ observation vectors and a sigmoid output function.

CML training is performed using a maximum of 100 online gradient ascent epochs and the Baum-Welch reestimation algorithm is used for ML training. The best model found within 30 epochs of complete label training is used as initial model for incomplete label training. Training times are less than one day on a fast workstation for all models evaluated on the broad class problem.

3.1. Baseline results

In table 1 the results for complete label training are shown for the baseline system. For the ML trained model it is observed that Viterbi and N-best decoding yields approximately the same accuracy[2]. However, for the CML estimated model a considerably higher accuracy is obtained by the N-best decoder. Probably this is because the CML estimated models are trained so as to maximize the probability of the labeling, which need not correspond to the most probable path found by the Viterbi algorithm.

Table 1 shows that the difference between complete and incomplete label training is negligible for the ML estimated models, but significant for the CML estimated models. The reason for this is that the CML criterion is very sensitive to mislabelings, because it is dominated by those training sequences that have an unlikely labeling. Since the phoneme segmentation in TIMIT is set by hand it is very likely to contain mislabelings which degrade performance of the complete label trained model. Thus, an accuracy of 81.3% is obtained using incomplete label CML training compared to 79.0% for complete label CML training. For comparison the ML estimated discrete HMM only reaches 76.1%. In [5] an accuracy of 69.3% is reported for a ML trained continuous density HMM with a mixture of six diagonal covariance gaussians per state. Thus, for approximately the same number of parameters the ML trained discrete HMM outperforms the continuous density HMM by more than 6%. For a MMI trained model with a single diagonal covariance gaussian per state they report an accuracy of 72.4%, compared to our 81.3% for the CML trained discrete HMM.

3.2. HNN Results

The match networks in the HNN are initially trained by a few iterations of standard backpropagation to classify the observations into the five broad classes. This speeds up

Table 1. *Discrete HMM recognition accuracies.*

Complete labels	Vit	N-Best
ML	75.9	76.1
CML	76.6	79.0
Incomplete labels	**Vit**	**N-Best**
ML	75.8	75.2
CML	78.4	*81.3*

Table 2. *HNN recognition accuracies. For $K = 1$ a context of one left and one right frame is used,* i.e., *a total of three frames is used.*

0 hidden units	#Parms	N-Best
$K = 0$	436	80.8
$K = 1$	1216	81.7
10 hidden units	**#Parms**	**N-Best**
$K = 0$	4246	*84.0*
$K = 1$	12046	83.6

training of the HNN considerably and the models are less prone to getting stuck in local minima. All HNNs are estimated using incomplete label CML training.

Even though the HNN with zero hidden units and no context ($K = 0$) contains almost ten times less parameters than the baseline system it achieves a comparable recognition accuracy of 80.8%, see table 2. For a context of one left and right frame ($K = 1$) the HNN outperforms the CML estimated HMM. No further improvement was observed for contexts larger than $K = 1$. It is interesting to note that the match networks in the HNN without hidden units actually just implements linear weighted sums of input features (passed through a sigmoid output function). Adding hidden units to the HNN drastically increases performance even if no context is used, see table 2. Thus, for approximately the same number of parameters as used in the baseline system the HNN with 10 hidden units and no context ($K = 0$) obtains 84.0% recognition accuracy. Context degrades test set performance slightly for the HNN with 10 hidden units. We believe this is due to overfitting, because of the large number of parameters in these models. In [5] a feedforward NN is used as a global adaptive input transformation to a continuous density HMM with a single diagonal covariance gaussian per state. This hybrid is trained by MMI and N-best decoding gives an accuracy of 78.5%. The result has later been improved to 81.3% by using a linear transformation instead of the NN and by using multiple CML training passes [4].

Using the forward-backward algorithm it is straightforward to calculate the posterior probability $P(\pi_l = i|x, \mathcal{M})$ of occupying state i at time l [13]. For a test sentence (si2183: "Books are for schnooks") the state posterior probabilities provided by the HNN with 10 hidden units and no context are shown for the the consonant submodel states in fig. 1. The posteriors for the "begin" and "end" states are only large at the class boundaries, whereas the posteriors for the "middle" state is large only between these boundaries. Hence, the model is very good at discriminating between different classes. This is also verified in fig. 2 showing that more than 50% of the frames have a winning-label posterior probability[3] $P(y_l^*|x, \mathcal{M})$ larger than or equal to 0.9. This

[2] $\%Acc = 100\% - \%Ins - \%Del - \%Sub$, where $\%Ins$, $\%Del$ and $\%Sub$ denote the percentage of insertions, deletions and substitutions used for aligning the observed and the predicted transcription. The NIST standard scoring package "sclite" ver. 1.1 is used in all experiments.

[3] Posterior label probabilities for each frame are obtained by summing state posteriors at time l for states that carry the same label. The winning-label y_l^* at time l is defined by the largest label posterior at time l.

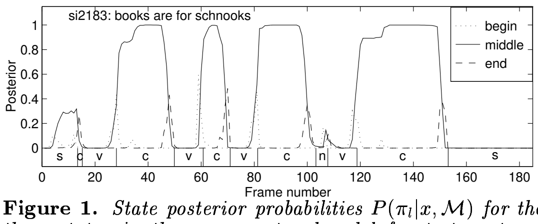

Figure 1. *State posterior probabilities $P(\pi_l|x,\mathcal{M})$ for the three states in the consonant submodel for test sentence "si2183". The broad class segmentation is shown at the bottom of the plot.*

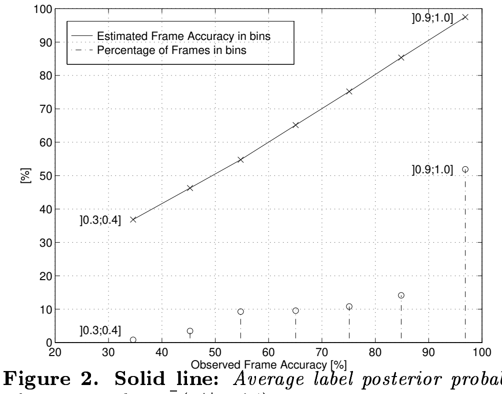

Figure 2. Solid line: *Average label posterior probability of winning class $\bar{P}(y_l^*|x,\mathcal{M})$ ranked into 7 equally spaced bins and plotted as function of the observed frame accuracy in these bins.* **Dashed line:** *Percentage of frames falling into the 7 bins located at the corresponding observed frame accuracies.*

corresponds to an observed frame recognition rate of more than 95%. Furthermore, there is an almost perfect correlation between the observed frame accuracy and the average label posterior of the winning class. Thus, the larger $P(y_l^*|x,\mathcal{M})$ the more confident is the prediction.

3.3. TIMIT 39 phoneme recognition

A set of experiments were carried out on recognizing the reduced 39 TIMIT phoneme set also considered in [4, 7, 10]. In these experiments we used the same preprocessor and submodel setup as in the broad class problem, but a total of 39 phoneme submodels were used. The full TIMIT training and testing set (except all "sa-sentences") were used in these experiments. For a HNN using no context, 10 hidden units in the match networks, standard transition probabilities and incomplete label training a preliminary test set result of 69% accuracy is obtained. This is comparable to results on the same task reported in the literature for standard context independent HMMs [10] and for a HMM using a global linear adaptive input transformation [4]. Currently one of the best results reported is 75% accuracy and is obtained by a HMM/recurrent NN hybrid [7].

4. CONCLUSION

It has been shown that the HNN combination of neural networks and a Hidden Markov model yields a better recognition accuracy than a standard HMM on TIMIT continuous phoneme recognition. In addition it was illustrated that the HNN provides very accurate estimates of phoneme posterior probabilities. The particular architecture introduced here is characterized by: 1) All parameters are trained discriminatively at the same time 2) Proper probability distributions are obtained by global normalization as opposed to local normalization, and 3) Large flexibility in that not all of the HMM parameters need to be replaced by neural networks. Future work includes further experiments on the 39 phoneme task, other optimization methods, *e.g.*, second order methods, and careful network design to reduce model complexity.

ACKNOWLEDGMENTS

The authors would like to thank Steve Renals and Finn T. Johansen for valuable comments and suggestions to this work. The Sanger Centre is supported by the Wellcome Trust.

REFERENCES

[1] P. Baldi and Y. Chauvin, "Hybrid Modeling, HMM/NN Architectures, and Protein Applications," *Neural Computation*, vol. 8, pp. 1541–65, 1996.

[2] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global optimization of a neural network-Hidden Markov model hybrid," *IEEE Trans. on Neural Networks*, vol. 3, no. 2, pp. 252–9, 1992.

[3] Y. Bengio and P. Frasconi, "An input output HMM architecture," in *Advances in Neural Information Processing Systems - 7*, pp. 427–34, Morgan-Kaufmann, 1995.

[4] F. Johansen, "Global optimisation of HMM input transformations," in *Proceedings of ICSLP'94*, vol. I, pp. 239–42, 1994.

[5] F. Johansen and M. Johnsen, "Non-linear input transformations for discriminative HMMs," in *Proceedings of ICASSP'94*, vol. I, pp. 225–28, 1994.

[6] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco, "Connectionist probability estimators in HMM speech recognition," *IEEE Trans. on Speech and Audio Proc.*, vol. 2, no. 1, pp. 161–74, 1994.

[7] A. Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Trans. on Neural Networks*, vol. 5, pp. 298–305, 1994.

[8] V. Valtchev, S. Kapadia, and S. Young, "Recurrent input transformations for Hidden Markov models," in *Proceedings of ICASSP'93*, pp. 287–90, 1993.

[9] L. Saul and M. Jordan, "Boltzman Chains and Hidden Markov models," in *Advances in Neural Information Processing Systems - 7*, pp. 435–42, Morgan-Kaufmann, 1995.

[10] S. Kapadia, V. Valtchev, and S. Young, "MMI training for continuous phoneme recognition on the TIMIT database," in *Proceedings of ICASSP'93*, vol. 2, pp. 491–4, 1993.

[11] A. Krogh, "Hidden Markov models for labeled sequences," in *Proceedings of the 12th ICPR'94*, pp. 140–4, 1994.

[12] L. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of Hidden Markov model parameters for speech recognition," in *Proceedings of ICASSP'86*, pp. 49–52, 1986.

[13] B. Juang and L. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, no. 3, pp. 251–72, 1991.

[14] R. Schwarz and Y.-L. Chow, "The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses," in *Proceedings of ICASSP'90*, pp. 81–84, 1990.

Appendix E

ICASSP*98 contribution

This appendix contains the paper "Hidden Neural Networks: Application to Speech Recognition" to be presented at the 1998 International Conference on Acoustics, Speech and Signal Processing in Seattle, USA. The paper gives an evaluation of the HNN on two speech recognition tasks, namely the TIMIT broad phoneme class task and the PhoneBook isolated word recognition task. For the broad class task transition based modeling is shown to lead to better performance than HNNs based on match networks and standard transition probabilities.

Reference for the paper: [Rii98]

HIDDEN NEURAL NETWORKS: APPLICATION TO SPEECH RECOGNITION

Søren Kamaric Riis

Department of Mathematical Modelling
Section for Digital Signal Processing
Technical University of Denmark
2800 Lyngby, Denmark
Email: sr@imm.dtu.dk

ABSTRACT

In this paper we evaluate the Hidden Neural Network HMM/NN hybrid presented at last years ICASSP on two speech recognition benchmark tasks; 1) task independent isolated word recognition on the PHONEBOOK database, and 2) recognition of broad phoneme classes in continuous speech from the TIMIT database. It is shown how Hidden Neural Networks (HNNs) with much fewer parameters than conventional HMMs and other hybrids can obtain comparable performance, and for the broad class task it is illustrated how the HNN can be applied as a purely transition based system, where acoustic context dependent transition probabilities are estimated by neural networks.

1. INTRODUCTION

Although the HMM is good at capturing the temporal nature of processes such as speech it has a very limited capacity for recognizing complex patterns involving more than first order dependencies in the observed data. This is primarily due to the first order state process and the assumption of state conditional observation independence. Neural networks and in particular multi-layer perceptrons (MLP) are almost the opposite: they cannot model temporal phenomena very well, but are good at recognizing complex patterns in a very parameter efficient way.

The Hidden Neural Network hybrid introduced in [16] is a very flexible architecture, where the probability parameters of an HMM are replaced by the outputs of small state specific neural networks. The model is estimated by the discriminative Conditional Maximum Likelihood (CML) criterion and is normalized globally. The global normalization works at the sequence level and ensures a valid probabilistic interpretation as opposed to the often approximate local normalization enforced in many other hybrids. Furthermore, instead of training the HMM and NNs separately, all parameters in the HNN are estimated simultaneously.

2. THE HNN

The HNN is a very natural extension of the so-called class HMM (CHMM) introduced in [10], which is basically a standard HMM where each state, in addition to the emission or *match* distribution, also has assigned a distribution over labels (classes). The basic idea is to *replace* the probability parameters of the CHMM by the outputs of state specific neural networks. Thus, it is possible to assign up to three networks to each state: 1) a *match network* $\phi_i(s_l; w^i)$ estimating the "probability" that the current observation matches a given state, 2) a *transition network* $\theta_{ij}(s_l; u^i)$ that estimates transition "probabilities" conditioned on observations, and finally 3) a *label network* $\psi_{ik}(s_l; v^i)$ estimating the probability of label $y_l = k$ in state i at time l. We have put probabilities in quotes because we do not require that the network outputs normalize locally in the HNN, *e.g.* the outputs of the transition network assigned to a state need not sum to one. The match network is parameterized by weights w^i and has only one output, which replaces the usual emission probability. Similarly the transition network is parametrized by weights u^i, and has the same number of outputs as there are non-zero transitions from state i to state j. The label network is parameterized by weights v^i and has one output for each of the possible labels in this state. Note that a given state can be restricted to model only a subset of the possible labels, *i.e.*, some labels have probability zero. The input s_l to the networks will usually be a window of context around the current observation x_l, *e.g.*, a symmetrical context window of $2K+1$ observation vectors, $s_l = x_{l-K}, x_{l-K+1}, \ldots, x_{l+K}$. It can, however, be any sort of information related to x_l or even the observation sequence in general. We will call s_l the context. Each of the three types of networks in each HNN state can be omitted and replaced by standard CHMM parameters. In fact all sorts of combinations with standard CHMM states are possible. In this work we assume, that the label networks are just delta-functions, *i.e.*, each state can only model one particular label and $\psi_{ik}(s_l; v^i) = \delta_{k,c_i}$, where c_i is the label of state i. Similarly we here restrict ourselves to use MLP match and transition networks, although they can in principle be any kind of mapping defined on the space of observations.

It is well known that Maximum Likelihood (ML) estimation is not optimal when the models are used for recognition especially when the training data is limited. We therefore choose parameters so as to maximize the probability of the correct labeling y associated with observation sequence x,

$$P(y|x, \mathcal{M}) = \frac{P(x, y|\mathcal{M})}{P(x|\mathcal{M})} \tag{1}$$

as we have previously proposed in [10]. Maximizing (1) is known as Conditional Maximum Likelihood estimation (CML) and is equivalent to Maximum Mutual Information estimation (MMI) [1, 8] if the language model is fixed during training. For an observation sequence of length L, the labeling y can be either *complete*, *i.e.*, there is one label for each observation ($y = y_1, \ldots, y_L$), or *incomplete*, *i.e.*, the label sequence $y = y_1, \ldots, y_S$ is shorter than the observation sequence ($S < L$). The latter case is more common in speech recognition since we usually only know the spoken words in the training set (and thereby the phonetic transcription), whereas the former is more common in *e.g.* biological sequence analysis. From (1) we see, that in order to compute the probability of the labeling we need to do two forward passes; once in the *free-running* or *recognition* time phase to compute $P(x|\mathcal{M})$ and once in the *clamped phase* to compute $P(y, x|\mathcal{M})$.

Similar to the likelihood for a standard HMM, we define for the HNN

$$R(x|\mathcal{M}) = \sum_\pi R(x,\pi|\mathcal{M}) = \sum_\pi \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l}(s_{l-1}; u^{\pi_{l-1}})\phi_{\pi_l}(s_l; w^{\pi_l}) \quad (2)$$

where $\pi = \pi_1, \ldots, \pi_L$ is a path through the model corresponding to the observation sequence $x = x_1, \ldots, x_L$. Since we do *not* require, that the neural network outputs normalize locally, $R(x|\mathcal{M})$ *will generally not be a probability*. However, if we only allow one label in each state we can define $R(x, y|\mathcal{M})$ in a way similar to equation (2) by only extending the sum over paths that are *consistent* with the observed complete or incomplete labeling. Then it is straightforward to show that,[1]

$$P(y|x,\mathcal{M}) = \frac{R(x,y|\mathcal{M})}{R(x|\mathcal{M})} \quad (3)$$

and the model is normalized at a global level, see [11, 15] for further details. Both $R(x|\mathcal{M})$ and $R(x, y|\mathcal{M})$ can be calculated by a straight-forward extension of the forward algorithm for complete as well as incomplete labeling, see *e.g.* [10, 11]

To maximize (3) we use stochastic online gradient ascent augmented by a momentum term, where the parameter update is performed after each observation sequence. As shown in [11] it turns out that the networks in the HNN are trained by standard backpropagation where the error to backpropagate is computed by running two forward-backward passes on the model; once for the free-running phase, and once for the clamped phase.

In agreement with results reported in [7], we have observed an increased performance for CML estimated models when using full-likelihood based decoders instead of a Viterbi best-path decoder. The reason for this is primarily that several paths have been observed to contribute significantly to the optimal labeling in the CML estimated models, see [11]. In the case of small vocabulary isolated word recognition the likelihood of the model for each word can be computed using the forward algorithm, whereas in the case of continuous speech recognition or large vocabulary isolated word recognition one can use stack decoding [13] or approximative algorithms like the N-best decoder [17]. In this work we use standard full-forward decoding for the PHONEBOOK experiments and N-best decoding for the broad class experiments. The N-best decoder allows 10 active hypothesis during decoding, and only the top-scoring hypothesis is used for recognition at the end of decoding.

3. COMPARISON TO OTHER HYBRIDS

Instead of training the HMM and the NN separately as in the work by *e.g.* Renals *et al.* [14] several authors have recently proposed architectures where all parameters are estimated simultaneously as in the HNN, see *e.g.* [2, 3, 5, 7, 9]. An example of such an approach is to use the neural network as an adaptive input transformation, where the network outputs are used as new observation vectors in a continuous HMM [7]. Our approach is somewhat similar to the idea of adaptive input transformations, but instead of retaining the computationally expensive mixture densities we *replace* these by match networks. This is also done in [3], where a large network with the same number of outputs as there are states in the HMM is optimized using the CML criterion by backpropagating errors calculated by the HMM. Instead of backpropagating errors from the HMM into the neural network some researchers [5] have proposed to let the HMM iteratively reestimate new "soft" targets for the network and then train the network to learn these targets. This method extends the approach by Renals *et al.* [14] to use global estimation where training can be done by a *Generalized* EM (GEM) algorithm.

The HNN is very closely related to the IOHMM [2]. In fact the IOHMM can be considered a special case of the HNN where each state has assigned a label network and a transition network, but no match network. If the transition and label networks all have a softmax output function then, in the free-running phase, $R(x|\mathcal{M}) = \sum_\pi R(x,\pi|\mathcal{M}) = 1$ independent of the observations, and thus $P(y|x,\mathcal{M}) = R(x, y|\mathcal{M})$. For this model only one forward pass is needed to compute the probability of the labeling and similarly only one forward-backward pass is needed to find the gradient. Furthermore, this model can be trained by a GEM algorithm. An additional assumption of only one label in each state renders the HNN similar to the purely transition based discriminant HMM/NN hybrid discussed in [9].

4. PHONEBOOK EXPERIMENTS

Often speech recognizers are trained using a fixed vocabulary, *i.e.*, the models are designed only to recognize words also used for estimation. For utterances containing out-of-vocabulary (OOV) words a very poor performance can be expected from such models unless the model incorporates some sort of OOV word detection. Therefore it would be desirable to train the models for task independent recognition, where the vocabulary used for training can be entirely different from that used during recognition. In this section we evaluate the HNN on task independent recognition of isolated words, where there are no identical words in the training and test set. The words are taken from the PHONEBOOK database [12], which is a phonetically-rich isolated word database. The words in PHONEBOOK are uttered by 1300 native American English speakers over a public American telephone line. In PHONEBOOK each of almost 8000 different words are uttered by an average of more than 11 speakers yielding a total of about 92,000 utterances.

We use a training set of 9,000 words randomly selected from the 21 PHONEBOOK wordlists ([a-d][a,h,m,q,t]+ea), see *e.g.* [4, 12], and a crossvalidation set of 1,893 utterances (wordlists ao and ay). The test set is composed of 8 wordlists ([a,b,c,d][d,r]) and results are reported as an unweighted average over the 8 lists. This is identical to the test sets used in [4, 5]. The results are reported for a dictionary size of either about 75 words (one dictionary for each of the 8 wordlists) or a larger dictionary of about 600 words (all 8 wordlists). The 110,000-word CMU 0.4 dictionary was used for phonetically transcribing the words.

For speech corrupted by linear additive channel noise RASTA-PLP cepstral features have been shown to be very robust [6]. We therefore use a RASTA-PLP cepstral preprocessor yielding a feature vector each 10ms based on a 30ms window. The 26 dimensional feature vector is composed of 12 RASTA-PLP cepstral features, the corresponding Δ-features and the Δ- and $\Delta\Delta$-energy.

For each of the 46 phonemes occurring in the transcriptions we use a phoneme submodel with a number of states equal to half the average duration of the phonemes as obtained from an initial forced Viterbi alignment of the training set. No skips are allowed, and the transition probabilities between states in a phoneme submodel are fixed to 1/2. We use a zero-gram grammar between phoneme models to avoid unintended introduction of priors for

[1] If more than one label have non-zero probability in each state equation (3) is still valid provided that the outputs of the label networks normalize locally, $\sum_k \psi_{ik}(s_l; v^i) = 1$ for $\forall i, l$, see [15].

Table 1: *HNN error rates on PHONEBOOK. K is the context-size,* i.e., $s_l = x_{l-K}, \ldots, x_l, \ldots, x_{l+K}$.

75 word dictionary	#Parms	Vit	Forw
Context $K = 0$, 0 hidden	1,242	20.8	15.1
Context $K = 1$, 0 hidden	3,634	16.6	13.3
Context $K = 1$, 10 hidden	36,846	7.3	4.8
600 word dictionary	#Parms	Vit	Forw
Context $K = 1$, 10 hidden	36,846	18.4	14.2

words in the training vocabulary due to a limited size training set. All states in a phoneme submodel share *one* match network, which replaces the usual emission distribution. All 46 match networks are fully connected MLPs, have the same number of hidden units, share the same input s_l and use sigmoid output functions. The match networks are initially trained to classify the observation vectors into each of the 46 phonemes by a few iterations of standard backpropagation. This speeds up training of the HNN and the model is less prone to getting stuck in local minima. Furthermore the HNN is bootstrapped by a few iterations (usually less than five) of complete label training.

In Table 1 the results obtained by the HNN is shown. First of all it is observed that full-forward decoding gives considerably better results than Viterbi decoding. This can be attributed to two facts: 1) in the CML estimated models several paths contribute to the optimal labeling as discussed above, and 2) the architecture of the phoneme submodels implies a Poisson-like duration distribution when using full-forward decoding, whereas a much weaker exponential duration distribution is implied when using Viterbi decoding. From the table it is also observed that contextual input increases performance considerably. Thus, for a model using a context of one left and right frame and no hidden units an error rate of 13.3% is obtained for the 75 word dictionary. With only 3634 parameters this model is very small and it is interesting that the match networks in this model actually just implement linear weighted sums of the input features (passed through a sigmoid output). No futher improvements in performance was observed for lager contexts. The best model with a context of one frame and 10 hidden units obtains an error rate of 4.8% for the 75 word dictionary. In [4] a continuous density HMM with more than four times as many parameters (162k) is reported to have an error rate of 5% when using a training set of 19,000 words. Similarly, for a Viterbi trained HMM/NN hybrid with 166k parameters an error rate of 1.5% is reported. This is somewhat better than the HNN, but the larger training set contributes significantly to the lower error rate as discussed below.

The effect of using the larger 600 word dictionary is reflected in the higher error rates shown in Table 1. For a model with context $K = 1$ and 10 hidden units an error rate of 14.2% is achieved. For the same training set as used here, Hennebert *et al.* [5] reports an error rate of 13.7% for a Viterbi trained HMM/NN hybrid containing 166k parameters. By iteratively reestimating "soft"-targets for the neural network instead of Viterbi "hard"-target training the error rate drops to 12.2%. For a 19,000 word training set Dupont *et al.* [4] reports an error rate of only 5.3% for the Viterbi trained HMM/NN hybrid. Thus, the size of the training set is indeed very important for the performance of the models.

5. BROAD PHONEME CLASS EXPERIMENTS

In this section we discuss some improvements on results reported at last years ICASSP [16] on the recognition of five broad phoneme classes in continuous speech from the TIMIT database. The broad classes are Vowels (V), Consonants (C), Nasals (N), Liquids (L) and Silence (S) and cover all phonetic variations in American English. We use one sentence from each of the 462 speakers in the TIMIT training set for training, and the results are reported for the recommended TIMIT core test set. The preprocessor is a standard mel cepstral preprocessor, which gives a 26 dimensional feature vector each 10ms (12 mel cepstral coefficients+1 log energy coefficient and the corresponding Δ-coefficients).

Table 2: *HNN accuracies on TIMIT broad class experiments.*

Model	#Parms	Accuracy
1-state, Match	4,030	80.0
1-state, Transition (sigmoid)	4,225	81.6
1-state, Transition (softmax)	4,225	81.6
3-state, Match	12,055	83.8
3-state, Transition (softmax)	12,290	82.0
3-state, Transition (sigmoid)	12,290	83.7
3-state, Mixed (sigmoid)	12,200	84.4

In the experiments reported in [16] we used a simple left-to-right three state model for each of the five classes, where the the match distributions were replaced by match networks. However, one major advantage of the HNN is that it allows for using transition probabilities that can depend on the observation context s_l. This can be important in tasks like speech recognition, where the acoustics of one phoneme is indeed highly influenced by the phonetic context in which it is uttered. Here we report results for two different kinds of HNNs with acoustic context dependent transitions: 1) a purely transition based model and 2) a mixed model. In the transition based HNN the transition probabilities in each state are modeled by a transition network and the match distributions are replaced by a constant. The mixed model is based on the 3-state left-to-right submodels also used in [16]. The first two states use standard transitions and match networks and in the last state transitions are modeled by a transition network. The transition based HNN is similar to the discriminant HMM/NN hybrid proposed in [9], except that we do not have to enforce locally normalizing transitions.

For the broad class experiments all match and transition networks have 10 hidden units and use a context of one left and right frame as input ($s_l = x_{l-1}, x_l, x_{l+1}$). The match networks use sigmoid output functions and the transition networks use either a locally normalizing softmax output function or sigmoid outputs. Initialization of the match networks is done in the same way as for the PHONEBOOK experiments, whereas the transition networks are initialized by duplicating the hidden to output weights of a pre-trained match network as many times as there are non-zero transitions. For a state with only a transition network assigned, this initially corresponds to a state with a match network and uniform standard HMM transition probabilities.

In table 2 the results for the TIMIT broad class experiments are shown. For a very simple model with only one state per class, where the match distributions are replaced by match networks, an accuracy of 80.0% is obtained. This compares favorably with a result of 69.3% accuracy on the same test set reported for a 3-state ML estimated HMM with six diagonal covariance Gaussians in each state (4,799 parameters) [7]. The purely transition based HNN with one state per class and non-normalizing transitions obtains an accuracy of 81.6%. The same result is obtained if locally normalizing transitions are enforced by softmax output functions. These results indicate that the purely transition based 1-state model is much better capable of modelling the five broad classes than the model with standard transition probabilities and match networks. It is interesting that this very simple model outperforms an HMM/adaptive linear input transformation hybrid with three states per class, which was reported to have an accuracy of 81.3% in [7] for the same training and test set. If we use 3-state submodels with match networks and standard transition probabilities in all states the accuracy increases to 83.8%. Similar to the transition based model above, we tried a 3-state model where the match distributions and standard transition probabilities are replaced by transition networks. If we enforce local normalization of the transitions by using a softmax output function the accuracy drops significantly compared to the 3-state model with match networks and standard transitions, see table 2. This can be explained as follows: assume that in a particular path we have just entered the consonant submodel. Then the next label in this path will also be a consonant with probability one because of the softmax function. That is, after entering a submodel, the path through this submodel will never be terminated due to a very low probability, since no matter what state we make a transition to this will always be with a fairly high probability. This is a fundamental problem, which makes minimum duration modeling very difficult in transition based models. However, the problem can be eliminated in practice by using non-normalizing transition "probabilities", whereby a path can be terminated if all outgoing transition "probabilities" from a state are close to zero. Such a model obtains a performance of 83.7%, which is practically identical to the 3-state model with standard transitions and match networks, see table 2. By replacing the match network and standard transition probabilities in the last state of each 3-state submodel with a transition network, the transitions between submodels become dependent on the acoustic context around the boundaries between the broad phoneme classes. This increases the performance of the HNN to 84.4% accuracy.

These results clearly illustrate the advantage of using transition networks. However, an even larger gain is expected for tasks, where there is a more pronounced context dependency between the phoneme classes like, *e.g.*, the well known TIMIT 39 phoneme recognition task. We are currently investigating this issue.

6. CONCLUSION

The globally normalized HNN has been introduced as a very flexible HMM/NN hybrid that allows for acoustic context dependent transitions. Furthermore, a comparison to other hybrids was given, and the similarity of the HNN to the IOHMM [2] and the discriminant HMM/NN hybrid [9] was discussed. Through a series of experiments it was shown how the HNN in a very parameter efficient way can yield state-of-the-art performance on two different speech recognition benchmarks. The results obtained on PHONEBOOK are slightly inferior to results reported for the HMM/NN hybrid discussed in [4, 5], where a much larger training set is used. On the TIMIT broad class experiments the HNN has been evaluated as a purely transition based model and as a "mixed" model. These experiments clearly illustrated the advantage of using transitions that depend on the acoustic context.

7. ACKNOWLEDGMENTS

The author would like to thank Christoffe Ris for details of his own work and for providing the initial forced Viterbi alignment of the PHONEBOOK database and phonetic transcriptions of words not in the CMU 0.4 dictionary. Also I thank DCS at Sheffield University and in particular Steve Renals for kindly hosting a five months visit in the Spring of 1997 during which much of this work was done. This work was partially supported by ESPRIT LTR project SPRACH (20077).

8. REFERENCES

[1] BAHL, L. R., BROWN, P. F., DE SOUZA, P. V., AND MERCER, R. L. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of ICASSP'86* (1986), pp. 49–52.

[2] BENGIO, Y., AND FRASCONI, P. Input/output HMMs for sequence processing. *IEEE Transactions on NN 7*, 5 (1996), 1231–1249.

[3] BENGIO, Y., LECUN, Y., NOHL, C., AND BURGES, C. Lerec: A NN/HMM hybrid for on-line handwritting recognition. *Neural Computation 7*, 5 (1995).

[4] DUPONT, S., BOURLARD, H., DEROO, O., FONTAINE, V., AND BOITE, J. Hybrid HMM/ANN systems for training independent tasks: Experiments on phonebook and related improvements. In *Proceedings of ICASSP '97* (1997), pp. 1767–1770.

[5] HENNEBERT, J., RIS, C., BOURLARD, H., RENALS, S., AND MORGAN, N. Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. In *Proceedings of EUROSPEECH'97* (1997).

[6] HERMANSKY, H., AND MORGAN, N. RASTA processing of speech. *IEEE Transactions on SAP 2*, 4 (1994), 578–589.

[7] JOHANSEN, F. T., AND JOHNSEN, M. H. Non-linear input transformations for discriminative HMMs. In *Proceedings of ICASSP'94* (1994), vol. I, pp. 225–28.

[8] JUANG, B. H., AND RABINER, L. R. Hidden Markov models for speech recognition. *Technometrics 33*, 3 (August 1991), 251–272.

[9] KONIG, Y., BOURLARD, H., AND MORGAN, N. REMAP: Experiments with speech recognition. In *Proceedings of ICASSP'96* (1996), pp. 3350–3353.

[10] KROGH, A. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR ICPR'94* (1994), pp. 140–144.

[11] KROGH, A., AND RIIS, S. Hidden Neural Networks. Submitted.

[12] PITRELLI, J., FONG, C., WONG, S., SPITZ, J., AND LEUNG, H. Phonebook: A Phonetically-Rich Isolated-Word Telephone-Speech Database. In *Proceedings of ICASSP'95* (1995), pp. 101–104.

[13] RENALS, S., AND HOCHBERG, M. Efficient evaluation of the LVCSR search space using the NOWAY decoder. In *Proceedings of ICASSP '96* (1996), pp. 149–152.

[14] RENALS, S., MORGAN, N., BOURLARD, H., COHEN, M., AND FRANCO, H. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on SAP 2*, 1 (1994), 161–74.

[15] RIIS, S. *Hidden Markov Models and Neural Networks*. PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 1998. To appear May 1998.

[16] RIIS, S. K., AND KROGH, A. Hidden neural networks: A framework for HMM/NN hybrids. In *Proceedings of ICASSP '97* (1997), pp. 3233–3236.

[17] SCHWARZ, R., AND CHOW, Y.-L. The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses. In *Proceedings of ICASSP'90* (1990), pp. 81–84.

Appendix F

Neural Computation*98 submission

This appendix contains the paper "Hidden Neural Networks" submitted to Journal of Neural Computation, 1998. The CHMM theory and the extension to the HNN hybrid is discussed in this paper. Furthermore, a comparison between standard HMMs, CHMMs, HNNs and the Input Output HMM by Bengio *et al.* is given in terms of graphical models of probabilistic independence networks. The paper is concluded by a brief experimental evaluation of different HNN architectures on the broad phoneme task.

Reference for the paper: [KR98].

Hidden Neural Networks

Anders Krogh
CBS, Building 206
Technical University of Denmark
2800 Lyngby, Denmark
Email: krogh@cbs.dtu.dk

Søren Kamaric Riis
IMM, Building 321
Technical University of Denmark
2800 Lyngby, Denmark
Email: sr@imm.dtu.dk

Abstract

A general framework for hybrids of Hidden Markov models (HMMs) and neural networks (NNs) called Hidden Neural Networks (HNNs) is described. The paper begins by reviewing standard HMMs and estimation by conditional maximum likelihood, which is used by the HNN. In the HNN the usual HMM probability parameters are replaced by the outputs of state specific neural networks. As opposed to many other hybrids, the HNN is normalized globally and therefore has a valid probabilistic interpretation. All parameters in the HNN are estimated simultaneously according to the discriminative conditional maximum likelihood criterion. An evaluation of the HNN on the task of recognizing broad phoneme classes in the TIMIT database shows clear performance gains compared to standard HMMs tested on the same task.

1 Introduction

Hidden Markov models is one of the most successful modeling approaches for acoustic events in speech recognition (Rabiner 1989; Juang & Rabiner 1991), and more recently they have proven useful for several problems in biological sequence analysis like protein modeling and gene finding, see *e.g.* (Durbin *et al.* 1998; Eddy 1996; Krogh *et al.* 1994). Although the HMM is good at capturing the temporal nature of processes such as speech it has a very limited capacity for recognizing complex patterns involving more than first order dependencies in the observed data. This is due to the first order state process and the assumption of state conditional independence of observations. Multi-layer perceptrons are almost the opposite: they cannot model temporal phenomena very well, but are good at recognizing complex patterns. Combining the two frameworks in a sensible way can therefore lead to a more powerful model with better classification abilities.

The starting point for this work is the so-called class HMM (CHMM) which is basically a standard HMM with a distribution over classes assigned to each state (Krogh 1994). The CHMM incorporates conditional maximum likelihood (CML) estimation (Juang & Rabiner 1991; Nádas 1983; Nádas, Nahamoo, & Picheny 1988). In contrast to the widely used Maximum Likelihood (ML) estimation, CML estimation is a discriminative training algorithm that aims at maximizing the ability of the model to discriminate between different classes. The CHMM can be normalized globally, which allows for non-normalizing parameters in the individual states, and this enables us to generalize the CHMM to incorporate neural networks in a valid probabilistic way.

In the CHMM/NN hybrid, which we call a *Hidden Neural Network* or HNN, some or all CHMM probability parameters are replaced by the outputs of state-specific neural networks which take the observations as input. The model can be trained as a whole from observation sequences with labels by a gradient descent algorithm. It turns out that in this algorithm the neural networks are updated by standard back-propagation, where the errors are calculated by a slightly modified forward-backward algorithm.

In this paper we first give a short introduction to standard HMMs. The CHMM and conditional maximum likelihood are then introduced and a gradient descent algorithm is derived for estimation. Based on this, the HNN is described next along with training issues for this model, and finally we give a comparison to other hybrid models. The paper is concluded with an evaluation of the HNN on the recognition of five broad phoneme classes in the TIMIT database (Garofolo *et al.* 1993). Results on this task clearly show a better performance of the HNN compared to a standard HMM.

1.1 Hidden Markov Models

To establish notation and set the stage for describing CHMMs and HNNs, we start with a brief overview of standard hidden Markov models. For a more comprehensive introduction we refer the reader to (Rabiner 1989; Juang & Rabiner 1991). In this description we consider discrete first-order HMMs, where the observations are symbols from a finite alphabet $\mathcal{A}$. The treatment of continuous observations is very similar, see *e.g.* (Rabiner 1989).

The standard HMM is characterized by a set of N states and two concurrent stochastic processes; a first order Markov process between states modeling the temporal structure of the data and an emission process for each state modeling the locally stationary part of the data. The state process is given by a set of transition probabilities, θ_{ij}, giving the probability of making a transition from state i to state j, and the emission process in state i is described by the probabilities, $\phi_i(a)$, of emitting symbol $a \in \mathcal{A}$ in state i. The ϕ's are usually called emission probabilities, but below we use the term match probabilities. We only observe the sequence of outputs from the model, and not the underlying (hidden) state sequence; therefore the name *Hidden* Markov Model. The set Θ of all transition and emission probabilities completely specifies the model.

Given an HMM, the probability of an observation sequence, $x = x_1 \ldots x_L$, of L symbols from the alphabet $\mathcal{A}$ is defined by

$$P(x|\Theta) = \sum_{\pi} P(x,\pi|\Theta) = \sum_{\pi} \prod_{l=1}^{L} \theta_{\pi_{l-1}\pi_l} \phi_{\pi_l}(x_l). \tag{1}$$

Here $\pi = \pi_1, \ldots, \pi_L$ is a state sequence; π_i is the number of the i'th state in the sequence. Such a state sequence is called a *path* through the model. An auxiliary start state, $\pi_0 = 0$, has been introduced such that θ_{0i} denotes the probability of starting a path in state i. In the following we assume that state N is an end state, *i.e.*, a non-matching state with no out-going transitions.

The probability (1) can be calculated efficiently by a dynamic programming-like algorithm known as the *forward algorithm.* Let $\alpha_i(l) = P(x_1, \ldots, x_l, \pi_l = i \mid \Theta)$, *i.e.* the probability of having matched observations $x_1, \ldots, x_l$ and being in state i at time l. Then the following recursion holds for $1 \leq i \leq N$ and $1 < l \leq L$,

$$\alpha_i(l) = \phi_i(x_l) \sum_j \alpha_j(l-1)\theta_{ji}, \tag{2}$$

and $P(x|\Theta) = \alpha_N(L)$. The recursion is initialized by $\alpha_i(1) = \theta_{0i}\phi_i(x_1)$ for $1 \leq i \leq N$.

The parameters of the model can be estimated from data by a maximum likelihood method. If multiple sequences of observations are available for training, they are assumed independent, and the total likelihood of the model is just a product of probabilities of form (1) for each of the sequences. The generalization from one to many observation sequences is therefore trivial and we will only consider one training sequence in the following. The likelihood of the model, $P(x|\Theta)$ given in (1), is commonly maximized by the Baum-Welch algorithm, which is an Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin 1977) guaranteed to converge to a local maximum of the likelihood. The Baum-Welch algorithm iteratively re-estimates the model parameters until convergence, and for the transition probabilities the reestimation formulas are given by

$$\theta_{ij} \leftarrow \frac{\sum_l n_{ij}(l)}{\sum_{j'l'} n_{ij'}(l')} = \frac{n_{ij}}{\sum_{j'} n_{ij'}}, \tag{3}$$

where $n_{ij}(l) = P(\pi_{l-1} = i, \pi_l = j \mid x, \Theta)$ is the expected number of times a transition from state i to state j is used at time l. The reestimation equations for the match probabilities can be expressed in a similar way by defining $n_i(l) = P(\pi_l = i \mid x, \Theta)$ as the expected number of times we are in state i at time l. Then the reestimation equations for the match probabilities are given by,

$$\phi_i(a) \leftarrow \frac{\sum_l n_i(l)\delta_{x_l,a}}{\sum_{la'} n_i(l)\delta_{x_l,a'}} = \frac{n_i(a)}{\sum_{a'} n_i(a')} \tag{4}$$

The expected counts can be computed efficiently by the *forward-backward* algorithm. In addition to the forward recursion a similar recursion for the backward variable $\beta_i(l)$ is introduced. Let $\beta_i(l) = P(x_{l+1}, \ldots, x_L \mid \pi_l = i, \Theta)$, *i.e.* the probability of matching the rest of the sequence $x_{l+1}, \ldots, x_L$ given that we are in state

i at time l. After initializing by $\beta_N(L) = 1$ the recursion runs from $l = L - 1$ to $l = 1$ as

$$\beta_i(l) = \sum_{j=1}^{N} \theta_{ij}\beta_j(l+1)\phi_j(x_{l+1}), \tag{5}$$

for all states $1 \le i \le N$. Using the forward and backward variables, $n_{ij}(l)$ and $n_i(l)$ can easily be computed,

$$n_{ij}(l) = P(\pi_{l-1}=i, \pi_l=j \mid x, \Theta) = \frac{\alpha_i(l-1)\theta_{ij}\phi_j(x_l)\beta_j(l)}{P(x|\Theta)} \tag{6}$$

$$n_i(l) = P(\pi_l=i \mid x, \Theta) = \frac{\alpha_i(l)\beta_i(l)}{P(x|\Theta)}. \tag{7}$$

1.2 Discriminative training

In many problems the aim is to predict what class an input belongs to or what sequence of classes it represents. In continuous speech recognition, for instance, the object is to predict the sequence of words or phonemes for a speech signal. To achieve this a (sub)model for each class is usually estimated by maximum likelihood independently of all other models and using only the data belonging to this class. This procedure maximizes the ability of the model to reproduce the observations in each class, and can be expressed as

$$\hat{\Theta}^{\mathrm{ML}} = \operatorname*{argmax}_{\Theta} P(x, y|\Theta) = \operatorname*{argmax}_{\Theta}[P(x|\Theta_y)P(y|\Theta)], \tag{8}$$

where y is the class or sequence of class labels corresponding to the observation sequence x and Θ_y is the model for class y or a concatenation of submodels corresponding to the observed labels. In speech recognition $P(x|\Theta_y)$ is often denoted the acoustic model probability and the language model probability $P(y|\Theta)$ is often assumed constant during training of the acoustic models. If the true source producing the data is contained in the model space, maximum likelihood estimation based on an infinite training set can give the optimal parameters for classification (Nádas, Nahamoo, & Picheny 1988; Nádas 1983), provided that the global maximum of the likelihood can be reached. However, in any real-world application it is highly unlikely that the true source is contained in the space of HMMs, and the training data is indeed limited. This is the motivation for using discriminative training.

To accommodate discriminative training we use one big model and assign a label to each state; all the states that are supposed to describe a certain class C are assigned label C. A state can also have a probability distribution $\psi_i(c)$ over labels, so that several labels are possible with different probabilities. This is discussed in (Krogh 1994) and (Riis 1998), and it is somewhat similar to the Input/Output HMM (IOHMM) (Bengio & Frasconi 1996), as discussed later. For brevity however, we here limit ourselves to consider only one label for each state, which we believe is the most interesting for many applications. Because each state has a class label or a distribution over class labels this sort of model was called a class HMM (CHMM) in (Krogh 1994).

In the CHMM the objective is to predict the labels associated with x, and instead of ML estimation we therefore choose to maximize the probability of the correct labeling,

$$\hat{\Theta}^{\mathrm{CML}} = \operatorname*{argmax}_{\Theta} P(y|x,\Theta) = \operatorname*{argmax}_{\Theta} \frac{P(x,y|\Theta)}{P(x|\Theta)}, \tag{9}$$

which is also called Conditional Maximum Likelihood (CML) estimation (Nádas 1983). If the language model is assumed constant during training, CML estimation is equivalent to maximum mutual information estimation (Bahl *et al.* 1986).

From (9) we observe that computing the probability of the labeling requires computation of: 1) the probability $P(x,y|\Theta)$ in the *clamped phase* and 2) the probability $P(x|\Theta)$ in the *free-running phase*. The term free-running means that the labels are not taken into account, so this phase is similar to the decoding phase, where we wish to find the labels for an observation sequence. The constraint by the labels during training gives rise to the name *clamped phase*; this terminology is borrowed from the Boltzmann machine literature (Ackley, Hinton, & Sejnowski 1985; Bridle 1990). Thus CML estimation adjusts the model parameters so as to make the free-running 'recognition model' as close as possible to the clamped model. The probability in the free-running phase is computed using the forward algorithm described for standard HMMs, whereas the probability in the clamped phase is computed by only considering paths $\mathcal{C}(y)$ that are consistent with the observed labeling,

$$P(x,y|\Theta) = \sum_{\pi \in \mathcal{C}(y)} P(x,\pi|\Theta). \tag{10}$$

This quantity can be calculated by a variant of the forward algorithm to be discussed below.

Unfortunately the Baum-Welch algorithm is not applicable to CML estimation, see *e.g.* (Gopalakrishnan *et al.* 1991). Instead, one can use a gradient descent based approach, which is also applicable to the HNNs discussed later. To calculate the gradients we switch to the negative log likelihood, and define

$$\begin{aligned} \mathcal{L} &= -\log P(y|x,\Theta) = \mathcal{L}_c - \mathcal{L}_f &(11)\\ \mathcal{L}_c &= -\log P(x,y|\Theta) &(12)\\ \mathcal{L}_f &= -\log P(x|\Theta). &(13) \end{aligned}$$

The derivative of $\mathcal{L}_f$ for the free-running model w.r.t. a generic parameter $\omega \in \Theta$ can be expressed as,

$$\begin{aligned} \frac{\partial \mathcal{L}_f}{\partial \omega} &= -\frac{1}{P(x|\Theta)} \frac{\partial P(x|\Theta)}{\partial \omega} \\ &= -\sum_{\pi} \frac{1}{P(x|\Theta)} \frac{\partial P(x,\pi|\Theta)}{\partial \omega} \\ &= -\sum_{\pi} \frac{P(x,\pi|\Theta)}{P(x|\Theta)} \frac{\partial \log P(x,\pi|\Theta)}{\partial \omega} \\ &= -\sum_{\pi} P(\pi|x,\Theta) \frac{\partial \log P(x,\pi|\Theta)}{\partial \omega}. \end{aligned} \tag{14}$$

This gradient is an expectation over all paths of the derivative of the *complete data* *log likelihood* $\log P(x, \pi|\Theta)$. Using (1) this becomes

$$\frac{\partial \mathcal{L}_f}{\partial \omega} = -\sum_{l,i} \frac{n_i(l)}{\phi_i(x_l)} \frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{l,i,j} \frac{n_{ij}(l)}{\theta_{ij}} \frac{\partial \theta_{ij}}{\partial \omega}. \tag{15}$$

The gradient of the negative log likelihood $\mathcal{L}_c$ in the clamped phase is computed similarly, but the expectation is taken only for the allowed paths $\mathcal{C}(y)$,

$$\frac{\partial \mathcal{L}_c}{\partial \omega} = -\sum_{l,i} \frac{m_i(l)}{\phi_i(x_l)} \frac{\partial \phi_i(x_l)}{\partial \omega} - \sum_{l,i,j} \frac{m_{ij}(l)}{\theta_{ij}} \frac{\partial \theta_{ij}}{\partial \omega}, \tag{16}$$

where $m_{ij}(l) = P(\pi_{l-1} = i, \pi_l = j \,|\, x, y, \Theta)$ is the expected number of times a transition from state i to state j is used at time l for the allowed paths. Similarly, $m_i(l) = P(\pi_l = i \,|\, x, y, \Theta)$ is the expected number of times we are in state i at time l for the allowed paths. These counts can be computed using the modified forward-backward algorithm discussed below.

For a standard model the derivatives in (15) and (16) are simple. When ω is a transition probability we obtain

$$\frac{\partial \mathcal{L}}{\partial \theta_{ij}} = -\frac{m_{ij} - n_{ij}}{\theta_{ij}}. \tag{17}$$

The derivative $\frac{\partial \mathcal{L}}{\partial \phi_i(a)}$ is of exactly the same form, except that m_{ij} and n_{ij} are replaced by $m_i(a)$ and $n_i(a)$, and θ_{ij} by $\phi_i(a)$.

When minimizing $\mathcal{L}$ by gradient descent it must be ensured that the probability parameters remain positive and properly normalized. Here we use the same method as (Bridle 1990; Baldi & Chauvin 1994) and do gradient descent in another set of unconstrained variables. For the transition probabilities we define

$$\theta_{ij} = \frac{e^{z_{ij}}}{\sum_{j'} e^{z_{ij'}}}, \tag{18}$$

where z_{ij} are the new unconstrained auxiliary variables, and θ_{ij} always sum to one by construction. Gradient descent in the z's by $z_{ij} \leftarrow z_{ij} - \eta \frac{\partial \mathcal{L}}{\partial z_{ij}}$ yields a change in θ given by

$$\theta_{ij} \leftarrow \frac{\theta_{ij} \exp(-\eta \frac{\partial \mathcal{L}}{\partial z_{ij}})}{\sum_{j'} \theta_{ij'} \exp(-\eta \frac{\partial \mathcal{L}}{\partial z_{ij'}})}. \tag{19}$$

The gradients with respect to z_{ij} can be expressed entirely in terms of θ_{ij} and $m_{ij} - n_{ij}$,

$$\frac{\partial \mathcal{L}}{\partial z_{ij}} = -[m_{ij} - n_{ij} - \theta_{ij} \sum_{j'} (m_{ij'} - n_{ij'})], \tag{20}$$

and inserting (20) into (19) yields an expression entirely in θs. Equations for the emission probabilities are obtained in exactly the same way. This approach is slightly more straightforward than the one proposed in (Baldi & Chauvin 1994), where the auxiliary variables are retained and the parameters of the model calculated explicitly from (18) after updating the auxiliary variables. This type of gradient descent is very similar to the exponentiated gradient descent proposed and investigated in (Kivinen & Warmuth 1997; Helmbold *et al.* 1997).

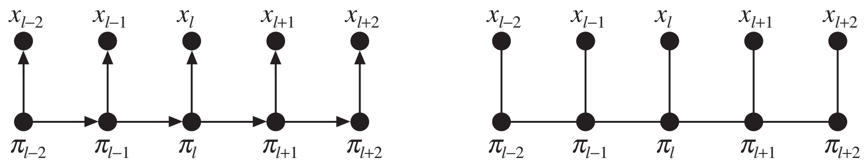

Figure 1: The DPIN (left) and UPIN (right) for an HMM.

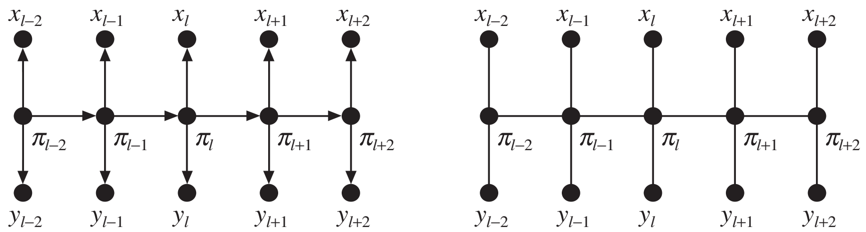

Figure 2: The DPIN (left) and UPIN (right) for a CHMM.

1.3 The CHMM as a probabilistic independence network

A large variety of probabilistic models can be represented as graphical models (Lauritzen 1996) including the HMM and its variants. The relation between HMMs and probabilistic independence networks is thoroughly described in (Smyth, Heckerman, & Jordan 1997), and here we follow their terminology and refer the reader to that paper for more details.

An HMM can be represented both as a directed probabilistic independence network (DPIN) and an undirected one (UPIN), see fig. 1. The DPIN shows the conditional dependencies of the variables in the HMM, both the observable ones (x) and the unobservable ones (π). For instance, the DPIN in fig. 1 shows that conditioned on π_l, x_l is independent of $x_1 \ldots x_{l-1}$ and $\pi_1 \ldots \pi_{l-1}$, *i.e.*, $P(x_l|x_1 \ldots x_{l-1}, \pi_1 \ldots \pi_l) = P(x_l|\pi_l)$. Similarly, $P(\pi_l|x_1 \ldots x_{l-1}, \pi_1 \ldots \pi_{l-1}) = P(\pi_l|\pi_{l-1})$. When 'marrying' unconnected parents of all nodes in a DPIN and removing the directions, the moral graph is obtained. This is a UPIN for the model. For the HMM the UPIN has the same topology as shown in fig. 1.

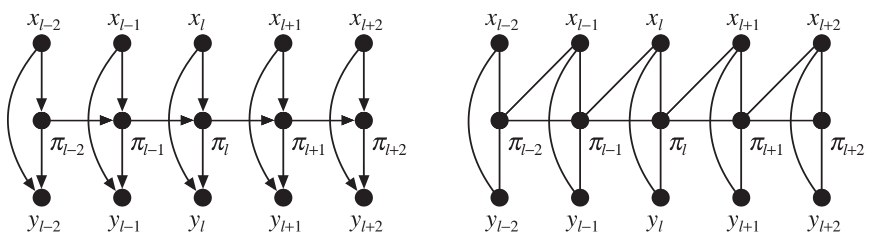

Figure 3: The DPIN for an IOHMM (left) is adapted from (Bengio & Frasconi 1996). The 'moral' graph to the right is a UPIN for an IOHMM.

In the CHMM there is one more set of variables (the y's), and the PIN structures are shown in fig. 2. In a way the CHMM can be seen as an HMM with two streams of observables, x and y, but they are usually not treated symmetrically. Again the moral graph is of the same topology, because no node has more than one parent.

It turns out that the graphical representation is the best way to see the difference between the CHMM and the IOHMM. In the IOHMM, the output y_l is conditioned on both the input x_l and the state π_l, but more importantly, the state is conditioned on the input. This is shown in the DPIN of fig. 3 (Bengio & Frasconi 1996). In this case the moral graph is different, because π_l has two unconnected parents in the DPIN.

It is straightforward to extend the CHMM to have the label y conditioned on x, meaning that there would be arrows from x_l to y_l in the DPIN for the CHMM. Then the only difference between the DPINs for the CHMM and the IOHMM would be the direction of the arrow between x_l and π_l. However, the DPIN for the CHMM would still not contain any 'unmarried parents' and thus their moral graphs would be different.

1.4 Calculation of quantities consistent with the labels

Generally there are two different types of labeling: incomplete and complete labeling (Juang & Rabiner 1991). We describe the modified forward-backward algorithm for both types of labeling below.

Complete Labels

In this case each observation has a label, so the sequence of labels denoted $y = y_1, \ldots, y_L$ is as long as the sequence of observations. Typically the labels come in groups, *i.e.*, several consecutive observations have the same label. In speech recognition the complete labeling corresponds to knowing which word or phoneme each particular observation x_l is associated with.

For complete labeling the expectations in the clamped phase are averages over "allowed" paths through the model, *i.e.*, paths in which the labels of the states agree with the labeling of the observations. Such averages can be calculated by limiting the sum in the forward and backward recursions to states with the correct label. The new forward and backward variables, $\tilde{\alpha}_i(l)$ and $\tilde{\beta}_i(l)$, are defined as $\alpha_i(l)$ (2) and $\beta_i(l)$ (5), but with $\phi_i(x_l)$ replaced by $\phi_i(x_l)\delta_{y_l,c_i}$. The expected counts $m_{ij}(l)$ and $m_i(l)$ for the allowed paths are calculated exactly as $n_{ij}(l)$ and $n_i(l)$, but using the new forward and backward variables.

If we think of $\alpha_i(l)$ (or $\beta_i(l)$) as a matrix, the new algorithm corresponds to masking this matrix such that only allowed regions are calculated, see fig. 4. Therefore the calculation is faster than the standard forward (or backward) calculation of the whole matrix.

Incomplete Labels

When dealing with incomplete labeling the whole sequence of observations is associated with a shorter sequence of labels $y = y_1, \ldots, y_S$, where $S < L$. The label of each individual observation is unknown—only the order of labels is available.

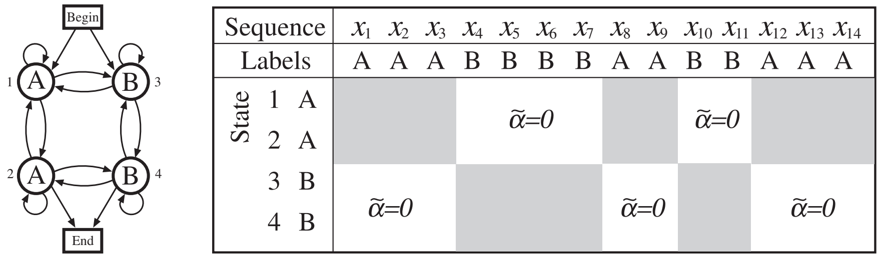

Figure 4: **Left:** A very simple model with four states, two labeled 'A' and two labeled 'B'. **Right:** The $\tilde{\alpha}$ matrix for an example of observations $x_1, \dots, x_{14}$ with complete labels. The grey areas of the matrix are calculated as in the standard forward algorithm whereas $\tilde{\alpha}$ is set to zero in the white areas. The $\tilde{\beta}$ matrix is calculated in the same way, but from right to left.

In continuous speech recognition the correct string of phonemes is known (because the spoken words are known in the training set), but the time boundaries between them are unknown. In such a case the sequence of observations may be considerably longer than the label sequence. The case $S = 1$ corresponds to classifying the whole sequence into one of the possible classes (*e.g.* isolated word recognition).

To compute the expected counts for incomplete labeling one has to ensure that the sequence of labels matches the sequence of *groups of states with the same label.*[1] This is less restrictive than the complete label case. An easy way to ensure this is by rearranging the (big) model temporarily for each observation sequence and collect the statistics (the m's) by running the standard forward-backward algorithm on this model. This is very similar to techniques already used in several speech applications, see *e.g.* (Lee 1990), where phoneme (sub)models corresponding to the spoken word or sentence are concatenated. Note however, that for the CHMM the transitions between states with different labels retain their *original value* in the temporary model, see fig. 5

2 Hidden Neural Networks

Hidden Markov models are based on a number of assumptions which limit their classification abilities. Combining the CHMM framework with neural networks can lead to a more flexible and powerful model for classification. The basic idea of the Hidden Neural Network (HNN) presented here is to replace the probability parameters of the CHMM by state specific multi-layer perceptrons which take the observations as input. Thus, in the HNN it is possible to assign up to three networks to each state: 1) a *match network* outputting the 'probability' that the current observation matches a given state, 2) a *transition network* that outputs transition 'probabilities' dependent on observations and finally 3) a *label network* which outputs the probability of the different labels in this state. We have put probabilities in quotes, because the output of the match and transition networks need not be properly normalized probabilities, because global normalization is used. For brevity we here

[1]If multiple labels are allowed in each state an algorithm similar to the forward-backward algorithm for asynchronous IOHMMs (Bengio & Bengio 1996) can be used, see (Riis 1998).

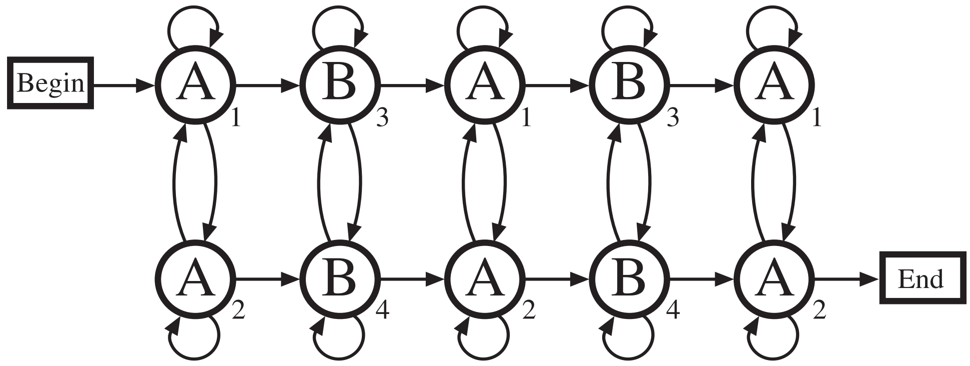

Figure 5: For the same model as in fig. 4 this example shows how the model is temporarily rearranged for gathering statistics (*i.e.* calculation of m values) for a sequence with incomplete labels $ABABA$.

limit ourselves to one label per state, *i.e.* the label networks are not present. The case of multiple labels in each state are treated in more detail in (Riis 1998).

The CHMM match probability $\phi_i(x_l)$ of observation x_l in state i is replaced by the output of a match network, $\phi_i(s_l; w^i)$, assigned to state i. The match network in state i is parameterized by a weight vector w^i and takes the vector s_l as input. Similarly, the probability θ_{ij} of a transition from state i to j is replaced by the output of a transition network $\theta_{ij}(s_l; u^i)$, which is parameterized by weights u^i. The transition network assigned to state i has $\mathcal{J}_i$ outputs, where $\mathcal{J}_i$ is the number of (non-zero) transitions *from* state i. Since we only consider states with one possible label, the label networks are just delta functions as in the CHMM described earlier.

The network input s_l corresponding to x_l will usually be a window of context around x_l, *e.g.*, a symmetrical context window of $2K + 1$ observations,[2] $x_{l-K}, x_{l-K+1}, \ldots, x_{l+K}$. It can however be any sort of information related to x_l or the observation sequence in general. We will call s_l the *context* of observation x_l, but the reader should bear in mind that it can contain all sorts of other information and that it can differ from state to state. The only limitation is that it cannot depend on the path through the model, because then the state process is no longer first order Markovian.

Each of the three types of networks in an HNN state can be omitted or replaced by standard CHMM probabilities. In fact, all sorts of combinations with standard CHMM states are possible. If an HNN only contains transition networks (*i.e.* $\phi_i(s_l; w^i) = 1$ for all i, l) the model can be normalized locally by using a softmax output function as in the IOHMM. However, if it contains match networks it is usually impossible to make $\sum_{x \in \mathcal{X}} P(x|\Theta) = 1$ by normalizing locally even though the transition networks are normalized. A probabilistic interpretation of the HNN is instead ensured by global normalization. We define the joint probability

$$P(x, y, \pi|\Theta) \quad = \quad \frac{1}{Z(\Theta)} R(x, y, \pi|\Theta)$$

[2]If the observations are inherently discrete (as in *e.g.* protein modeling) they can be encoded in binary vectors an then used in the same manner as continuous observation vectors.

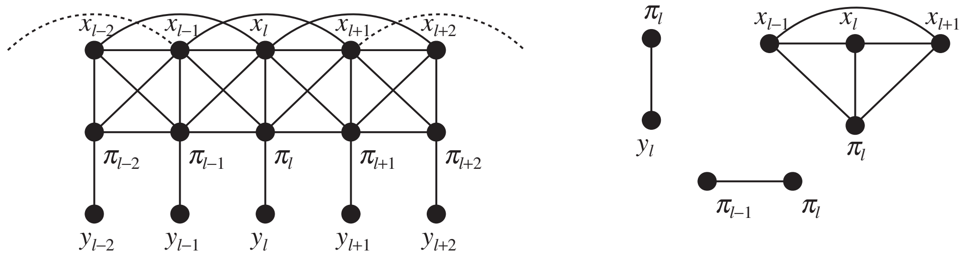

Figure 6: To the left is shown the UPIN of an HNN with no transition networks and match networks having a context of one to each side. To the right, the three different clique types contained in the graph.

$$= \frac{1}{Z(\Theta)} \prod_l \theta_{\pi_{l-1}\pi_l}(s_l; u^{\pi_{l-1}}) \phi_{\pi_l}(s_l; w^{\pi_l}) \delta_{y_l, c^{\pi_l}}, \tag{21}$$

where the normalizing constant is $Z(\Theta) = \sum_{x,y,\pi} R(x,y,\pi|\Theta)$. From this,

$$\begin{aligned} P(x,y|\Theta) &= \frac{1}{Z(\Theta)} R(x,y|\Theta) = \frac{1}{Z(\Theta)} \sum_\pi R(x,y,\pi|\Theta) \\ &= \frac{1}{Z(\Theta)} \sum_{\pi \in \mathcal{C}(y)} R(x,\pi|\Theta), \end{aligned} \tag{22}$$

where

$$R(x,\pi|\Theta) = \prod_l \theta_{\pi_{l-1}\pi_l}(s_l; u^{\pi_{l-1}}) \phi_{\pi_l}(s_l; w^{\pi_l}). \tag{23}$$

Similarly,

$$\begin{aligned} P(x|\Theta) &= \frac{1}{Z(\Theta)} R(x|\Theta) = \frac{1}{Z(\Theta)} \sum_{y,\pi} R(x,y,\pi|\Theta) \\ &= \frac{1}{Z(\Theta)} \sum_\pi R(x,\pi|\Theta). \end{aligned} \tag{24}$$

In (23) and (24) it was used that the label distribution is a delta function. It is sometimes possible to compute the normalization factor Z, but not in all cases. However for CML estimation the normalization factor cancels out,

$$P(y|x,\Theta) = \frac{R(x,y|\Theta)}{R(x|\Theta)}. \tag{25}$$

The calculation of $R(x|\Theta)$ and $R(x,y|\Theta)$ can be done exactly as described above for $P(x|\Theta)$ and $P(x,y|\Theta)$, because the forward and backward algorithms are not dependent on the normalization of probabilities.

Because one cannot usually normalize the HNN locally, there exists no *directed* graph (DPIN) for the general HNN. For UPINs, however, local normalization is not required. For instance the Boltzmann machine can be drawn as a UPIN, and the Boltzmann chain (Saul & Jordan 1995) can actually be described by a UPIN identical to the one for a globally normalized discrete HMM in fig. 1. A model with a UPIN is characterized by its *clique* functions and the joint probability is the product of all the clique functions (Smyth, Heckerman, & Jordan 1997). The three different clique functions are clearly seen in (21). The graphical representation as a UPIN for an HNN with no transition networks and match networks having a context of one to each side is shown in fig. 6 along with the three types of cliques.

The fact that the individual neural network outputs do not have to normalize give us a large freedom in selecting the output activation function. A natural choice is a standard (asymmetric) sigmoid or an exponential output activation function, $g(h) = \exp(h)$, where h is the input to the output unit in question.

Even though the HNN is a very intuitive and simple extension of the standard CHMM it is a much more powerful model. First of all, neural networks can implement complex functions using far fewer parameters than *e.g.*, a mixture of Gaussians. Furthermore, the HNN can directly use observation context as input to the neural networks and thereby exploit higher order correlations between consecutive observations, which is difficult in standard HMMs. This property can be particularly useful in problems like speech recognition, where the pronunciation of one phoneme is highly influenced by the acoustic context in which it is uttered. Finally, the observation context dependency on the transitions allows the HNN to model the data as successive steady-state segments connected by "non-stationary" transitional regions. For speech recognition this is believed to be very important, see *e.g.* (Bourlard, Konig, & Morgan 1994; Morgan *et al.* 1994).

2.1 Training an HNN

As for the CHMM it is not possible to train the HNN using an EM algorithm and instead we suggest to train the model using gradient descent. From (15) and (16) we find the following gradients of $\mathcal{L} = -\log P(y|x, \Theta)$ w.r.t. a generic weight ω^i in the match or transition network assigned to state i,

$$\frac{\partial \mathcal{L}}{\partial \omega^i} = -\sum_l \frac{m_i(l) - n_i(l)}{\phi_i(s_l; w^i)} \frac{\partial \phi_i(s_l; w^i)}{\partial \omega^i} - \sum_{lj} \frac{m_{ij}(l) - n_{ij}(l)}{\theta_{ij}(s_{l-1}; u^i)} \frac{\partial \theta_{ij}(s_{l-1}; u^i)}{\partial \omega^i}, \quad (26)$$

where it is assumed that networks are not shared between states. In the back-propagation algorithm for neural networks (Rumelhart, Hinton, & Williams 1986) the squared error of the network is minimized by gradient descent. For an activation function g this gives rise to a weight update of the form $\Delta w \propto -\mathcal{E} \times \frac{\partial g}{\partial w}$. We therefore see from (26) that the neural networks are trained using the standard backpropagation algorithm where the quantity to backpropagate is $\mathcal{E} = [m_i(l) - n_i(l)]/\phi_i(s_l; w^i)$ for the match networks and $\mathcal{E} = [m_{ij}(l) - n_{ij}(l)]/\theta_{ij}(s_{l-1}; u^i)$ for the transition networks. The m and n counts are calculated as before by running two forward-backward passes; once in the clamped phase (the m's) and once in the free-running phase (the n's).

The training can be done either in batch mode, where all the networks are updated after the entire training set has been presented to the model, or in sequence online mode, where the update is performed after the presentation of each sequence. There are many other variations possible. Because of the l dependence of $m_{ij}(l)$, $m_i(l)$ and the similar n's the training algorithm is not as simple as for standard HMMs, *i.e.*, we have to do a backpropagation pass for each l. Because the expected counts are not available before the forward-backward passes have been completed we must either store or recalculate all the neural network unit activations for each input s_l before running backpropagation. Storing all activations can require large amounts of memory even for small networks if the observation sequences are very long (which they typically are in continuous speech). For such tasks it is necesarry to recalculate the network unit activations before each backpropagation pass. Many of the standard modifications of the backpropagation algorithm can be incorporated, such as momentum and weight-decay (Hertz, Krogh, & Palmer 1991). It is naturally also possible to use conjugate gradient descent or approximative second order methods like pseudo Gauss-Newton. However, in a set of initial experiments for the speech recognition task reported in section 4, online gradient methods consistently gave the fastest convergence.

3 Comparison to other work

Recently several HMM/NN hybrids have been proposed in the literature. The hybrids can roughly be divided into those estimating the parameters of the HMM and the NN separately, see *e.g.* (Renals *et al.* 1994; Robinson 1994; Le Cerf, Ma, & Compernolle 1994; McDermott & Katagiri 1991) and those applying simultaneous or joint estimation of all parameters as in the HNN, see *e.g.* (Baldi & Chauvin 1996; Konig, Bourlard, & Morgan 1996; Bengio *et al.* 1992; Johansen 1994; Valtchev, Kapadia, & Young 1993; Bengio & Frasconi 1996; Hennebert *et al.* 1997; Bengio *et al.* 1995).

In (Renals *et al.* 1994) a multi-layer perceptron is trained separately to estimate phoneme posterior probabilities which are scaled with the observed phoneme frequencies and then used instead of the usual emission densities in a continuous HMM. A similar approach is taken in (Robinson 1994), but here a recurrent NN is used. A slightly different method is used in (McDermott & Katagiri 1991; Le Cerf, Ma, & Compernolle 1994), where the vector quantizer front-end in a discrete HMM is replaced by a multilayer perceptron or an LVQ network (Kohonen, Barna, & Chrisley 1988). In contrast, our approach uses only one output for each match network whereby continuous and discrete observations are treated the same.

Several authors have proposed methods in which all parameters are estimated simultaneously as in the HNN. In some hybrids a big multi-layer perceptron (Bengio *et al.* 1992; Johansen & Johnsen 1994) or recurrent network (Valtchev, Kapadia, & Young 1993) performs an adaptive input transformation of the observation vectors. Thus, the network outputs are used as new observation vectors in a continuous density HMM and simultaneous estimation of all parameters is performed by backpropagating errors calculated by the HMM into the neural network in a way similar to the HNN training. Our approach is somewhat similar to the idea of adaptive input transformations, but instead of retaining the computationally expensive mixture densities we *replace* these by match networks. This is also done in (Bengio *et al.* 1995), where a large network with the same number of outputs as there are states in the HMM is trained by backpropagating errors calculated by the HMM. Instead of backpropagating errors from the HMM into the neural network some researchers (Hennebert *et al.* 1997; Senior & Robinson 1996) use a two step iterative procedure to train the networks.

In the first step the current model is used for estimating a set of "soft" targets for the neural networks, and then the network is trained on these targets. This method extends the scaled likelihood approach by (Renals *et al.* 1994) to use global estimation where training is performed by a *Generalized* EM (GEM) algorithm (Hennebert *et al.* 1997).

The IOHMM (Bengio & Frasconi 1996) and the CHMM/HNN have different graphical representations, as seen in the figures 2, 3, and 6. However, the IOHMM is very similar to a locally normalized HNN with a label and transition network in each state, but no match network. An important difference between the two is in the decoding, where the IOHMM uses only a forward pass, which makes it insensitive to future events, but makes the decoding 'real-time'. See (Riis 1998) for more details.

4 Experiments

In this section we give an evaluation of the HNN on the task introduced in (Johansen 1994) of recognizing five broad phoneme classes in continuous read speech from the TIMIT database (Garofolo *et al.* 1993). The five broad classes are vowels (V), consonants (C), nasals (N), liquids (L) and silence (S), see table 1. We use one

Table 1: Definition of broad phoneme classes.

Broad class	TIMIT phoneme label
Vowel (V)	iy ih eh ae ix ax ah ax-h uw uh ao aa ey ay oy aw ow ux
Consonant (C)	ch jh dh b d dx g p t k z zh v f th s sh hh hv
Nasal (N)	m n en ng em nx eng
Liquid (L)	l el r y w er axr
Silence (S)	h# pau

sentence from each of the 462 speakers in the TIMIT training set for training, and the results are reported for the recommended TIMIT core test set containing 192 sentences. An additional validation set of 144 sentences has been used to monitor performance during training. The raw speech signal is preprocessed using a standard mel cepstral preprocessor which outputs a 26 dimensional feature vector each 10ms (13 mel cepstral features and 13 delta features). These vectors are normalized to zero mean and unit variance. Each of the five classes are modeled by a simple left-to-right three state model. The last state in any submodel is fully connected to the first state of all other submodels. Further details are given in (Riis & Krogh 1997).

4.1 Baseline results

In table 2 the results for complete label training are shown for the baseline system, which is a discrete CHMM using a codebook of 256 codebook vectors. The results are reported in the standard measure of percent accuracy, $\%Acc = 100\% - \%Ins - \%Del - \%Sub$, where $\%Ins$, $\%Del$ and $\%Sub$ denote the percentage of insertions, deletions and substitutions used for aligning the observed and the predicted transcription.[3] In agreement with results reported in (Johansen 1994), we have observed an increased performance for CML estimated models when using a

[3]The NIST standard scoring package "sclite" ver. 1.1 is used in all experiments.

forward or all-paths decoder instead of the best-path Viterbi decoder. In this work we use an N-best decoder (Schwarz & Chow 1990) with ten active hypotheses during decoding. Only the top-scoring hypothesis is used at the end of decoding. The N-best decoder finds (approximatively) the most probable labels, which depends on many different paths, whereas the Viterbi algorithm only finds the most probable path. For ML trained models the N-best and Viterbi decoder yield approximately the same accuracy, see table 2. As shown by an example in fig. 7, several paths contribute to the optimal labeling in the CML estimated models, whereas only a few paths contribute significantly for the ML estimated models.

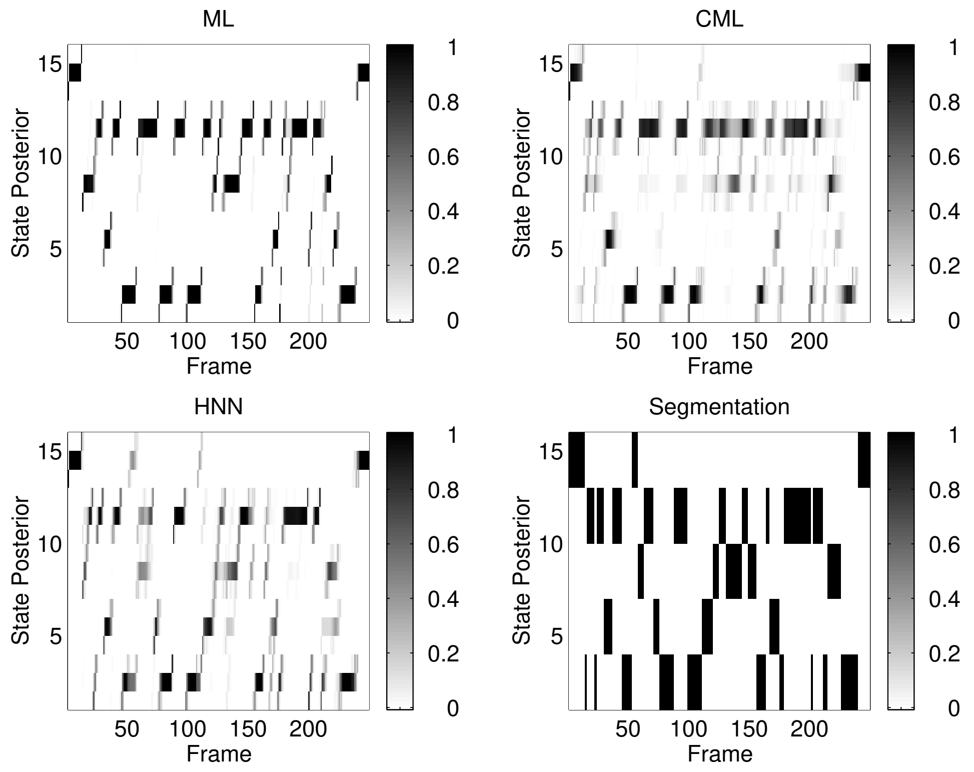

Figure 7: State posterior plots ($P(\pi_l = i \,|\, x, \Theta)$) for baseline and HNN for the test sentence "But in this one section we welcomed auditors" (TIMIT id: si1361). States 1–3 belong to the consonant model, 4–6 to the nasal model,7–9 to the liquid model, 10–12 to the vowel model and 13–15 to the silence model. **Top left:** ML trained baseline, which yield $\%Acc$ = 62.5 for this sentence. **Top right:** CML trained baseline ($\%Acc$ = 78.1). **Bottom left:** HNN using both match and transition networks with 10 hidden units and context $K = 1$ ($\%Acc = 93.7$). **Bottom Right:** The observed segmentation.

Table 2 shows that additional incomplete label training of a complete label trained model does not improve performance for the ML estimated model. However, for the CML estimation there is a significant gain in accuracy by incomplete label training. The reason for this is that the CML criterion is very sensitive to mislabelings, because it is dominated by training sequences with an unlikely labeling. Although the phoneme segmentation (complete labeling) in TIMIT is done by hand, it is imperfect. Furthermore, it is often impossible — or even meaningless — to assign exact boundaries between phonemes.

CML gives a big improvement from an accuracy of around 76% for the ML estimated models to around 81%. Statistical significance is hard to asses, because of the computational requirements for this task, but in a set of 10 CML training sessions of random initial models we observed a deviation of no more than ±0.2% in accuracy. For comparison a MMI trained model with a single diagonal covariance Gaussian per state achieved a result of 72.4% accuracy in (Johansen & Johnsen 1994).

Table 2: Baseline recognition accuracies. The baseline system contains 3856 free parameters.

Complete labels	Viterbi	N-Best
ML	75.9	76.1
CML	76.6	79.0
Incomplete labels	**Viterbi**	**N-Best**
ML	75.8	75.2
CML	78.4	*81.3*

4.2 HNN results

For the HNN two series of experiments where conducted. In the first set of experiments, only a match network is used in each state and the transitions are standard HMM transitions. In the second set of experiments we also use match networks, but the match distribution *and* the standard transitions in the last state of each submodel is replaced by a transition network. All networks use the same input s_l, have the same number of hidden units, are fully connected and have sigmoid output functions. Note that this also applies for the transition networks, *i.e.* a softmax output function is *not* used for the transition networks.

Even though the HNN with match networks and no hidden units has far fewer parameters than the baseline system, it achieves a comparable performance of 80.8% accuracy using only the current observation x_l as input ($K = 0$) and 81.7% accuracy for a context of one left and right observation ($K = 1$), see table 3. No further improvement was observed for larger contexts. Note that the match networks without hidden units just implement linear weighted sums of input features (passed through a sigmoid output function). For approximately the same number of parameters as used in the baseline system the HNN with 10 hidden units and no context ($K = 0$) yields 84.0% recognition accuracy. Increasing the context or number of hidden units for this model yields a slightly lower accuracy due to overfitting.

In (Johansen 1994) a multi-layer perceptron was used as a global adaptive input transformation to a continuous density HMM with a single diagonal covariance Gaussian per state. Using N-best decoding and CML estimation a result of 81.3% accuracy was achieved on the broad phoneme class task.

When using a transition network in the last state of each submodel the accuracy increases as shown in table 3. Thus for the model with context $K = 1$ and no hidden units an accuracy of 82.3% is obtained compared to 81.7% for the same model with only match networks. The best result on the five broad class task is an accuracy of

Table 3: Recognition accuracies for HNNs. "HNN, match networks" are models using only match networks and standard CHMM transitions, whereas "HNN, match & trans. networks" use both match and transition networks. Decoding is done by N-best.

No hidden units	Context K	Number of Parameters	Accuracy
HNN, match networks	0	436	80.8
HNN, match networks	1	1216	81.7
HNN, match & trans. networks	1	2411	82.3
10 hidden units			
HNN, match networks	0	4246	84.0
HNN, match networks	1	12046	83.8
HNN, match & trans. networks	1	12191	*84.4*

84.4% obtained by the HNN with context $K = 1$, match and transition networks and 10 hidden units in all networks, see table 3.

5 Conclusion

In this paper we described the hidden neural network (HNN) which in a very natural way replaces the probability parameters of an HMM with the output of state-specific neural networks. The model is normalized at a global level, which ensures a proper probabilistic interpretation of the HNN. All the parameters in the model are trained simultaneously from labeled data using gradient descent based CML estimation. The architecture is very flexible in that all combinations with standard CHMM probability parameters are possible.

Finally, it was shown that the HNN improves on the results of a speech recognition problem with a reduced set of phoneme classes.

Acknowledgments

The authors would like to thank Steve Renals and Finn T. Johansen for valuable comments and suggestions to this work. We also thank the anonymous referees for pointing our attention to graphical models. This work was supported by the Danish National Research Foundation.

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for Boltzmann machines. *Cognitive Science* 9:147–169.

Bahl, L. R.; Brown, P. F.; de Souza, P. V.; and Mercer, R. L. 1986. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of ICASSP'86*, 49–52.

Baldi, P., and Chauvin, Y. 1994. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation* 6(2):307–318.

Baldi, P., and Chauvin, Y. 1996. Hybrid modeling, HMM/NN architectures, and protein applications. *Neural Computation* 8:1541–65.

Bengio, S., and Bengio, Y. 1996. An EM algorithm for asynchronous input/output hidden Markov models. In *Proceedings of the ICONIP'96*.

Bengio, Y., and Frasconi, P. 1996. Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks* 7(5):1231–1249.

Bengio, Y.; De Mori, R.; Flammia, G.; and Kompe, R. 1992. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks* 3(2):252–9.

Bengio, Y.; LeCun, Y.; Nohl, C.; and Burges, C. 1995. Lerec: A NN/HMM hybrid for on-line handwritting recognition. *Neural Computation* 7(5).

Bourlard, H.; Konig, Y.; and Morgan, N. 1994. REMAP: Recursive estimation and maximization of a posteriori probabilities. Technical Report TR-94-064, International Computer Science Institute, Berkeley, CA.

Bridle, J. S. 1990. Alphanets: A recurrent 'neural' network architecture with a hidden Markov model interpretation. *Speech Communication* 9:83–92.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Royal Statistical Society B* 39:1–38.

Durbin, R. M.; Eddy, S. R.; Krogh, A.; and Mitchison, G. 1998. *Biological Sequence Analysis*. Cambridge University Press. To appear.

Eddy, S. R. 1996. Hidden Markov models. *Current Opinion in Structural Biology* 6:361–365.

Garofolo, J. S.; Lamel, L. F.; Fisher, W. M.; Fiscus, J. G.; Pallet, D. S.; and Dahlgren, N. L. 1993. *DARPA TIMIT Acoustic-Phonetic Continuos Speech Corpus CD-ROM*. National Institute of Standards.

Gopalakrishnan, P. S.; Kanevsky, D.; Nádas, A.; and Nahamoo, D. 1991. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory* 37(1):107–113.

Helmbold, D. P.; Schapire, R. E.; Singer, Y.; and Warmuth, M. K. 1997. A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning* 27(1):97–119.

Hennebert, J.; Ris, C.; Bourlard, H.; Renals, S.; and Morgan, N. 1997. Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. In *Proceedings of EUROSPEECH'97*.

Hertz, J. A.; Krogh, A.; and Palmer, R. 1991. *Introduction to the Theory of Neural Computation*. Redwood City: Addison-Wesley.

Johansen, F. T., and Johnsen, M. H. 1994. Non-linear input transformations for discriminative HMMs. In *Proceedings of ICASSP'94*, volume I, 225–28.

Johansen, F. T. 1994. Global optimisation of HMM input transformations. In *Proceedings of ICSLP'94*, volume I, 239–42.

Juang, B. H., and Rabiner, L. R. 1991. Hidden Markov models for speech recognition. *Technometrics* 33(3):251–272.

Kivinen, J., and Warmuth, M. K. 1997. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation* 132(1):1–63.

Kohonen, T.; Barna, G.; and Chrisley, R. 1988. Statistical pattern recognition with neural networks: benchmarking studies. In *Proceedings of ICNN'88*, volume I, 61–68.

Konig, Y.; Bourlard, H.; and Morgan, N. 1996. REMAP: Recursive estimation and maximization of a posteriori probabilities — application to transition-based connectionist speech recognition. In Touretzky, D. S.; Mozer, M. C.; and Hasselmo, M. E., eds., *Advances in Neural Information Processing Systems*, volume 8, 388–394.

Krogh, A.; Brown, M.; Mian, I. S.; Sjölander, K.; and Haussler, D. 1994. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology* 235:1501–1531.

Krogh, A. 1994. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR ICPR'94*, 140–144.

Lauritzen, S. L. 1996. *Graphical Models.* Oxford University Press.

Le Cerf, P.; Ma, W.; and Compernolle, D. V. 1994. Multilayer perceptrons as labelers for hidden Markov models. *IEEE Transactions on Speech and Audio Processing* 2(1):185–193.

Lee, K.-F. 1990. Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 38(4):599–609.

McDermott, E., and Katagiri, S. 1991. LVQ-based shift-tolerant phoneme recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 39:1398–1411.

Morgan, N.; Bourlard, H.; Greenberg, S.; and Hermansky, H. 1994. Stochastic perceptual auditory-event-based models for speech recognition. In *Proceedings of Interternational Conference on Spoken Language Processing*, 1943–1946.

Nádas, A.; Nahamoo, D.; and Picheny, M. A. 1988. On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 36(9):814–817.

Nádas, A. 1983. A decision-theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing* 31(4):814–817.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Procedings of IEEE* 77(2):257–286.

Renals, S.; Morgan, N.; Bourlard, H.; Cohen, M.; and Franco, H. 1994. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing* 2(1):161–74.

Riis, S. K., and Krogh, A. 1997. Hidden neural networks: A framework for HMM/NN hybrids. In *Proceedings of ICASSP '97*, 3233–3236.

Riis, S. 1998. *Hidden Markov Models and Neural Networks.* Ph.D. Dissertation, Department of Mathematical Modelling, Section for Digital Signal Processing, Technical University of Denmark. In preparation.

Robinson, A. J. 1994. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks* 5:298–305.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.

Saul, L. K., and Jordan, M. I. 1995. Boltzman chains and hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 7, 435–42. Morgan-Kaufmann.

Schwarz, R., and Chow, Y.-L. 1990. The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses. In *Proceedings of ICASSP'90*, 81–84.

Senior, A., and Robinson, T. 1996. Forward-backward retraining of recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 8, 743–49. Morgan-Kaufmann.

Smyth, P.; Heckerman, D.; and Jordan, M. I. 1997. Probabilistic independence networks for hidden Markov probability models. *Neural Computation* 9:227–269.

Valtchev, V.; Kapadia, S.; and Young, S. 1993. Recurrent input transformations for hidden Markov models. In *Proceedings of ICASSP'93*, 287–90.

Bibliography

[AHS85] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[Bak75a] J. K. Baker. The DRAGON system — An overview. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23:24–29, February 1975. Reprinted in [WL90].

[Bak75b] J. K. Baker. Stochastic modeling for automatic speech understanding. In D. R. Reddy, editor, *Speech Recognition*, pages 521–542. Academic Press, New York, 1975. Reprinted in [WL90].

[Bau72] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[BB96] S. Bengio and Y. Bengio. An EM algorithm for asynchronous input/output hidden Markov models. In *Proceedings of the International Conference on Neural Information Processing*, 1996.

[BBdSM86] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 49–52, 1986.

[BBdSM93] L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. Estimating hidden Markov model parameters so as to maximize speech recognition accuracy. *IEEE Transactions on Speech and Audio Processing*, 1(1):77–83, Jan 1993.

[BC94] P. Baldi and Y. Chauvin. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6(2):307–318, 1994.

[BCHM94] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure. Hidden Markov models of biological primary sequence information. *Proceedings of the National Academy of Sciences of the U.S.A.*, 91(3):1059–63, Feb 1 1994.

[BDMFK92] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transactions on Neural Networks*, 3(2):252–9, 1992.

[BE67] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, 73:360–363, 1967.

[BF94] Y. Bengio and P. Frasconi. Credit assignment through time: Alternative to backpropagation. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann, 1994.

[BF95] Y. Bengio and P. Frasconi. Diffusion of context and credit information in Markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

[BF96] Y. Bengio and P. Frasconi. Input/output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249, 1996.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.

[BJM83] L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179–90, March 1983.

[BKM94] Herve Bourlard, Yochai Konig, and Nelson Morgan. REMAP: Recursive estimation and maximization of a posteriori probabilities. Technical Report TR-94-064, International Computer Science Institute, Berkeley, CA, November 1994.

[BLNB95] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A NN/HMM hybrid for on-line handwritting recognition. *Neural Computation*, 7(5), 1995.

[BM94] H. A. Bourlard and N. Morgan. *Connnectionist Speech Recognition — A Hybrid Approach.* Kluwer Academic Publishers, Boston, MA, 1994.

[BN88] J. Bellagarda and D. Nahamoo. Tied-mixture continuous parameter models for large vocabulary isolated speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 1988.

[BP66] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[BPS70] L. E. Baum, T. Petrie, and G et al. Soules. A maximization technique in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

[Bri90] J. S. Bridle. Alphanets: A recurrent 'neural' network architecture with a hidden Markov model interpretation. *Speech Communication*, 9:83–92, Feb 1990.

[CH96] G. I. Chiou and J. N. Hwang. Lipreading from color motion video. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 2156–2159, Atlanta, Georgia, May 1996. IEEE, IEEE.

[Cho90] Y. L. Chow. Maximum mutual information estimation of hmm parameters for continuous speech recognition using the N-best algorithm. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, 1990.

[CNM91] R. Cardin, Y. Normandin, and R. D. Mori. High-performance connected digit recognition using maximum mutual information estimation. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 533–536, 1991.

[CS94] J. K. Chen and F. K. Soong. An N-best candidates-based discriminative training for speech recognition applications. *IEEE Transactions on Speech and Audio Processing*, 2(1):206–216, jan 1994.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[DBD+97] S. Dupont, H. Bourlard, O. Deroo, V. Fontaine, and J.-M. Boite. Hybrid HMM/ANN systems for training independent tasks: Experiments on phonebook and related improvements. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 1767–1770, Munich, Germany, April 1997.

[DEKM98] R. M. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, 1998.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley & Sons, New York, 1973.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–88, 1977.

[DPH93] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-Time Processing of Speech Signals.* Maxwell Macmillan, 1993.

[Edd96] Sean R. Eddy. Hidden Markov models. *Current Opinion in Structural Biology*, 6:361–365, 1996.

[ER89] A. Ephraim, Y. amd Dembo and L. R. Rabiner. A minimum discrimination information approach for hidden Markov modeling. *IEEE Transactions on Information Theory*, 35(5):1001–1013, Sep 1989.

[ER90] Y. Ephraim and L. R. Rabiner. On the relations between modeling approaches for speech recognition. *IEEE Transactions on Information Theory*, 36(2):372–380, March 1990.

[Fah88] S. E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie-Mellon Univeristy, 1988.

[FD93] A. M. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden Markov models with mixed states. In A. S. Weigend and N. A. Gershenfeld, editors, *Times Series Prediction: Forecasting the Future and Understanding the Past*, volume XV of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 265–282. Addison-Wesley, 1993.

[FDGM86] W. Fisher, G. Doddington, and K. Goudie-Marshall. The DARPA speech recognition research database: Specifications and status. In *Proceeding of DARPA Workshop on Speech Recognition*, pages 93–99, 1986.

[FLW90] M. Franzini, K. F. Lee, and A. Waibel. Connectionist Viterbi training: A new hybrid method for continuous speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 425–428, 1990.

[For73] G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, March 1973.

[FR95] S. Forchhammer and J. Rissanen. Coding with partially hidden Markov models. In *Proceeding of DCC*, Utah, March 1995.

[FRL+96] V. Fontaine, C. Ris, H. Leich, J. Vantieghem, S. Accaino, and D. Van Compernolle. Comparison between two hybrid HMM/MLP approaches in speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA, May 1996. IEEE.

[Fur86a] S. Furui. On the role of spectral transition for speech perception. *Journal of Acoustical Society of America*, 80:1016–1025, 1986.

[Fur86b] S. Furui. Speaker independent isolated word recognizer using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34:52–59, 1986.

[GKN+98] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, D. Nahamoo, and M. A. Picheny. Decoder selection based on cross-entropies. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 20–23, 1998.

[GKNN91] P. S. Gopalakrishnan, D. Kanevsky, A. Nádas, and D. Nahamoo. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory*, 37(1):107–113, 1991.

[GL94] J.-L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for mutlivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, april 1994.

[GLF+93] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallet, and N. L. Dahlgren. *DARPA TIMIT Acoustic-Phonetic Continuos Speech Corpus CD-ROM.* National Institute of Standards, Feb 1993.

[Hay94] S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan College Publishing, New York, 1994.

[HBAH93] X. Huang, M. Belin, F. Alleva, and M. Hwang. Unified stochastic engine (USE) for speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume II, pages 636–639. IEEE, 1993.

[HCL95] Q. Huo, C. Chan, and C.-H. Lee. Bayesian adaptive learning of the parameters of hidden Markov model for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 3(5):334–344, september 1995.

[Her90] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 4(87):1738–1752, April 1990.

[HJ89] X. D. Huang and M. A. Jack. Semi-continuous hidden Markov models for speech signals. *Computer, Speech and Language*, 3, july 1989.

[HKP91] J. A. Hertz, A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, 1991.

[HM94] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing*, 2(4):578–589, 1994.

[HMBK91] H. Hermansky, N Morgan, A. Bayya, and P. Kohn. RASTA-PLP speech analysis. Technical report, International Computer Science Institute (ICSI), Berkeley, December 1991.

[HRB+97] J. Hennebert, C. Ris, H. Bourlard, S. Renals, and N. Morgan. Estimation of global posteriors and forward-backward training of hybrid HMM/ANN systems. In *Proceedings of EUROSPEECH'97*, 1997.

[HRRC95] M. M. Hochberg, S. J Renals, A. J. Robinson, and G. D. Cook. Recent improvements to the ABBOT large vocabulary CSR system. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 69–72, Detroit, 1995.

[HSSW97] D. P. Helmbold, R. E. Schapire, Y. Singer, and M. K. Warmuth. A comparison of new and old algorithms for a mixture estimation problem. *Machine Learning*, 27(1):97–119, 1997.

[Hua92] X. D. Huang. Phoneme classification using semicontinuous hidden Markov models. *IEEE Transactions on Signal Processing*, 40(5), May 1992.

[Ita75] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23:67–72, Feb 1975. Reprinted in [WL90].

[JBM75] F. Jelinek, L. R. Bahl, and R. L. Mercer. Design of a linguistic statistical decoder for the recognition of continuous speech. *IEEE Transactions on Information Theory*, 21:250–256, May 1975.

[Jel76] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of IEEE*, 64:532–556, Apr 1976.

[JJ94a] F. T. Johansen and M. H. Johnsen. Non-linear input transformations for discriminative HMMs. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 225–28, 1994.

[JJ94b] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[JJNH91] R. Jacobs, M. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[JK92] B. H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40:3043–3054, Dec 1992.

[JKBS90] C. Jankowski, A. Kalyanswamy, S. Basson, and J. Spitz. NTIMIT: A phonetically balanced, continuous speech, telephone bandwidth speech database. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 109–112, April 1990.

[Joh94] F. T. Johansen. Global optimisation of HMM input transformations. In *Proceedings of ICSLP'94*, volume I, pages 239–42, 1994.

[Joh96] F. T. Johansen. *Global Discriminative Modelling for Automatic Speech Recognition*. PhD thesis, Technical Norwegian University of Science and Technology, Trondheim, Norway, May 1996.

[JR91] B. H. Juang and L. R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, August 1991.

[KBC88] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: benchmarking studies. In *Proceedings of ICNN*, volume I, pages 61–68, 1988.

[KBM+94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, Feb. 1994.

[KBM96] Yochai Konig, Hervé Bourlard, and Nelson Morgan. REMAP: Recursive estimation and maximization of a posteriori probabilities — application to transition-based connectionist speech recognition. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 388–394, 1996.

[KCD88] F. Kubala, Y. Chow, and A. et al. Derr. Continuous speech recognition results of the BYBLOS system on the DARPA 1000-word resource management database. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 291–294, New York, 1988.

[KGJV83] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[KKLT92] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola. LVQ-PAK: A program package for the correct application of learning vector quantization algorithms. In *Proceedings of IJCNN*, pages 725–730, Baltimore, June 1992.

[KLJ91] S. Katagiri, C. H. Lee, and B. H. Juang. New discriminative training algorithms based on the generalized probabilistic descent method. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 299–308, 1991.

[KMH94] A. Krogh, I. S. Mian, and D. Haussler. A hidden Markov model that finds genes in *e. coli* DNA. *Nucleic Acids Research*, 22:4768–4778, 1994.

[Kon96] Y. Konig. *REMAP: Recursive Estimation and Maximization of A Posteriori Probabilities in Transition-based Speech Recognition*. PhD thesis, University of California at Berkley, 1996.

[KOR94] A. Kannan, M. Ostendorf, and J. R. Rohlicek. Maximum likelihood clustering of gaussians for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(3):453–455, 1994.

[KR98] A. Krogh and S. K. Riis. Hidden neural networks. *Neural Computation*, 1998. Submitted.

[Kro94] A. Krogh. Hidden Markov models for labeled sequences. In *Proceedings of the 12th IAPR ICPR'94*, pages 140–144, 1994.

[Kro97] A. Krogh. Two methods for improving performance of an HMM and their application for gene finding. In *Proceedings of Fifth International Conference on Intelligent Systems for Molecular Biology*, pages 179–188, Menlo Park, CA., 1997. AAAI Press.

[Kro98] A. Krogh. An introduction to hidden Markov models for biological sequences. In S. Salzberg, D. Searls, and S. Kasif, editors, *Machine Learning and Pattern Analysis Methods in Computational Biology*. Elsevier, 1998. To appear.

[KVY93] S. Kapadia, V. Valtchev, and S. J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 2, pages 491–4, 1993.

[KW97] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[LCBB97] Y. Le Cun, L. Bottou, and Y. Bengio. Reading checks with multilayer graph transformer networks. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 151–155, 1997.

[LCDS90] Y. Le Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan-Kaufmann, 1990.

[LCMC94] P. Le Cerf, W. Ma, and D. V. Compernolle. Multilayer perceptrons as labelers for hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 2(1):185–193, jan 1994.

[Lee89] Kai-Fu Lee. *Automatic Speech Recognition: Development of the SPHINX System*. Kluwer Academic Publishers, 1989.

[Lee90] K-F. Lee. Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(4):599–609, April 1990.

[LER90] A. Lolje, Y. Ephraim, and L. R. Rabiner. Estimation of hidden Markov model parameters by minimizing empirical error rate. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 709–712, 1990.

[Lev85] S. E. Levinson. Structural methods in automatic speech recognition. *Proceedings of IEEE*, 73:1625–1650, Nov 1985.

[Lev86] S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1(1):29–45, March 1986.

[LG87] R. P. Lippmann and B. Gold. Neural-net classifiers useful for speech recognition. In *Proceedings of IEEE first International Conference on Neural Networks*, volume IV, pages 417–425, San Diego, CA, June 1987.

[LG89] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, Reading, Mass., 1989.

[LH89] K. F. Lee and H. W. Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648, 1989.

[Lip89] R. P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1:1–38, 1989.

[LRS83] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal*, 62:1035–1074, Apr 1983.

[Mac96] D. J. C. Mackay. Equivalence of linear Boltzmann chains and hidden Markov models. *Neural Computation*, 1(8):178–181, 1996.

[MBGH94] N. Morgan, H. Bourlard, S. Greenberg, and H. Hermansky. Stochastic perceptual auditory-event-based models for speech recognition. In *Proceedings of Interternational Conference on Spoken Language Processing*, pages 1943–1946, Yokohama, Japan, 1994.

[MBGH96] N. Morgan, H. Bourlard, S. Greenberg, and H. Hermansky. Stochastic perceptual models for speech. 1996.

[ME91] N. Merhav and Y. Eprahim. Maximum likelihood hidden Markov modeling using a dominant sequences of states. *IEEE Transactions on Signal Processing*, 39:2111–2115, Sept 1991.

[Mer88] B. Merialdo. Phonetic recognition using hidden Markov models and maximum mutual information training. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 111–114, 1988.

[MHJ96] C. D. Mitchell, M. P. Harper, and L. H. Jamieson. Stochastic observation hidden Markov model. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Atlanta, USA, May 1996. IEEE.

[MK91] E. McDermott and S. Katagiri. LVQ-based shift-tolerant phoneme recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 39:1398–1411, 1991.

[MS91] D. P. Morgan and C. L. Scofield. *Neural Networks for Speech Processing.* Kluwer Academic Publishers, Norwell, Massachussets, 1991.

[Nád83] Arthur Nádas. A decision-theoretic formulation of a training problem in speech recognition and a comparison of training by unconditional versus conditional maximum likelihood. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 31(4):814–817, August 1983.

[NCM94] Y. Normandin, R. Cardin, and R. D. Mori. High-performance connected digit recognition using maximum mutual information estimation. *IEEE Transactions on Speech and Audio Processing*, 2:299–311, April 1994.

[NM91] Y. Normandin and S. D. Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 537–540, 1991.

[NNP88] A. Nádas, D. Nahamoo, and M. A. Picheny. On a model-robust training method for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(9):814–817, September 1988.

[NSB90] L. T. Niles, H. F. Silverman, and M. A. Bush. Neural networks, maximum mutual information training, and maximum likelihood training. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 493–496, 1990.

[Ped97] M. W. Pedersen. *Optimization of Recurrent Neural Network for Time Series Modeling.* PhD thesis, Technical University of Denmark, Department of Mathematical modeling, Lyngby, DK, august 1997.

[PFW+95] J. F. Pitrelli, C. F. Fong, S. H. Wong, J. R. Spitz, and H. C. Leung. PhoneBook: A phonetically-rich isolated-word telephone-speech database. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 101–104, Detroit, Michigan, April 1995.

[PGMDF86] J. Picone, K. M. Goudie-Marshall, G. Doddington, and W. Fisher. Automatic text alignment for speech system evaluation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):780–784, august 1986.

[Pic90] J. Picone. Continuous speech recognition using hidden Markov models. *IEEE Acoustics, Speech and Signal Processing Magazine*, 7:26–41, Jul 1990.

[Rab89] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Procedings of IEEE*, 77(2):257–286, 1989.

[RAB+93] A. J. Robinson, L. Almeida, J. M. Boite, H. Bourlard, F. Fallside, M. Hochberg, D. Kershaw, P. Kohn, Y. Konig, N. Morgan, J. P. Neto, S. Renals, M. Saerens, and C. Wooters. A neural network based, speaker independent, large vocabulary, continuous speech recognition system: The *wernicke* project. In *Proceedings of EUROSPEECH*, Berlin, September 1993.

[RBFC92] N. Renals, S. Morgan, H. Bourlard, H. Franco, and M. Cohen. Connectionist optimization of tied mixture hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 4, pages 167–174. Morgan-Kaufmann, 1992.

[RF88] A. J. Robinson and F. Fallside. A dynamic connectionist model for phoneme recognition. In *Neural Networks from Models to Applications: Proceedings of nEuro*, pages 541–550, Paris: I.D.S.E.T., 1988.

[RF91] T. Robinson and F. Fallside. A recurrent error propagation network speech recognition system. *Journal of Computer, Speech and Language*, 5(3), july 1991.

[RH95] S. Renals and M. Hochberg. Decoder technology for connectionist large vocabulary speech recognition. Technical Report CS-95-17, Department of Computer Science, University of Sheffield, Sheffield, UK, 1995.

[RHR96] T. Robinson, M. Hochberg, and S. Renals. *Automatic speech and speaker recognition – Advanced topics*, chapter 7. Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[Rii95] S. Riis. Combining neural networks for protein secondary structure prediction. In *Proceedings of International Conference on Neural Networks*, Perth, Australia, 1995.

[Rii98] S. K. Riis. Hidden neural networks: Application to speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, Seattle, May 1998. To appear.

[RJ93] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[RK96a] S. Riis and A. Krogh. Improving protein secondary structure prediction using structured neural networks and multiple sequence profiles. *Journal of Computational Biology*, pages 163–183, 1996.

[RK96b] S. Riis and A. Krogh. Joint estimation of parameters in Hidden Neural Networks. In *Proceeding of Nordic Signal Processing Symposium*, pages 431–434, Sept 1996.

[RK97] S. K. Riis and A. Krogh. Hidden neural networks: A framework for HMM/NN hybrids. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 3233–3236, 1997.

[RL91] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian *a posteriori* probabilities. *Neural Computation*, 3:461–483, 1991.

[RM85] M. J. Russell and R. K. Moore. Exlicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 5–8, March 1985.

[RMB+94] S. Renals, N. Morgan, H. Bourlard, M. Cohen, and H. Franco. Connectionist probability estimators in HMM speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(1):161–74, 1994.

[RMCF92] S. Renals, N. Morgan, M. Cohen, and H. Franco. Connectionist probability estimators in the Decipher speech recognition system. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 601–604, 1992.

[Rob91] T. Robinson. Several improvements to a recurrent error propagation network phone recognition system. Technical Report CUED/F-INFENG/TR.82, Cambridge University, Department of Engineering, Cambridge, UK, September 1991.

[Rob92] T. Robinson. A real-time recurrent error propagation network word recognition system. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 1992.

[Rob94] A. J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5:298–305, 1994.

[RWJ86] L. R. Rabiner, J. G. Wilpon, and B. H. Juang. A segmental k-means procedure for connected word recognition. *AT& T Technical Journal*, 64(3):21–40, May 1986.

[SA91] R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (N-BEST) sentence hypothesis. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, 1991.

[Sam93] F. Samaria. Face segmentation for identification using hidden Markov models. In *Proceedings of 4th British Machine Vision Conference*, volume 3, 1993.

[SC90] R. Schwarz and Y.-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely hypotheses. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, 1990.

[SH94] F. Samaria and A. C. Harter. Parameterisation of a stochastic model for human face identification. In *Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, Dec 1994.

[SH96] D. G. Stork and M. E. Hennecke, editors. *Speechreading by humans and machines*. Springer-Verlag, New York, 1996.

[SHJ97] P. Smyth, D. Heckerman, and M. I. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation*, 9:227–269, 1997.

[SJ95] L. K. Saul and M. I. Jordan. Boltzman chains and hidden Markov models. In *Advances in Neural Information Processing Systems*, volume 7, pages 435–42. Morgan-Kaufmann, 1995.

[SL92] E. Singer and R. P. Lippmann. Improved hidden Markov model speech recognition using radial basis function networks. In *Advances in Neural Information Processing Systems*, volume 4, pages 159–166. Morgan-Kaufmann, 1992.

[SR96] A. Senior and T. Robinson. Forward-backward retraining of recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 8, pages 743–49. Morgan-Kaufmann, 1996.

[Teb95] J. Tebelskis. *Speech recognition using neural networks.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1995.

[TF90] Robinson. T. and F. Fallside. Phoneme recognition from the TIMIT database using recurrent error propagation networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University, Department of Engineering, Cambridge, UK, March 1990.

[THPF90] Robinson. T., J. Holdsworth, R. Patterson, and F. Fallside. A comparison of preprocessors for the Cambridge recurrent error propagation network speech recognition system. In *Proceeding of the International Conference on Spoken Language Processing*, Kobe, Japan, Nov 1990.

[Tor94] K. Torkkola. LVQ as a feature transformation for HMMs. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, volume IV, pages 299–308, Piscataway, New Jersey, 1994. IEEE.

[Val95] V. Valtchev. *Discriminative methods in HMM-based speech recognition.* PhD thesis, Cambridge University, Department of Engineering, Cambridge, UK, March 1995.

[Vit67] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, Apr 1967.

[VKY93] V. Valtchev, S. Kapadia, and S. Young. Recurrent input transformations for hidden Markov models. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 287–90, 1993.

[WL90] A. Waibel and K.-F. Lee. *Readings in speech recognition.* Morgan-Kauffmann, Palo Alto, California, USA, 1990.

[YHS98] G. Yoshihiko, M. Hochberg, and H. F. Silverman. Efficient training algorithms for HMMs using incremental estimation. *IEEE Transactions on Speech and Audio Processing*, 1998. To appear.

[YNC94] R. Lacouture Y. Normandin and R. Cardin. MMIE training for large vocabulary continuous speech recognition. In *International Conference on Spoken Language Processing*, pages 1367–1370, Apr 1994.

[Yos96] G. Yoshihiko. *Incremental Algorithms and MAP Estimation: Efficient HMM Learning of Speech Signals.* PhD thesis, Brown University, Providence, May 1996.

[You92a] S. J. Young. The general use of tying in phoneme-based hmm speech recognisrs. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 569–572, March 1992.

[You92b] S.J. Young. *HTK: Hidden Markov Model Toolkit V1.4*. Cambridge University, Department of Engineering, Speech group, Cambridge, September 1992.

[You96] S. Young. A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, pages 45–57, Sep 1996.

[YRT89] S. J. Young, N. H. Russel, and J. H. S. Thornton. Token passing: A simple conceptual model for connected speech recognition. Technical Report F-INFENG/TR38, Cambridge University, Department of Engineering, Cambridge, UK, July 1989.

[YW94] S. Young and P. C. Woodland. State clustering in hmm-based continuous speech recogntion. *Computer Speech and Language*, 8(4):369–384, 1994.