CONNECTIONIST SPEECH RECOGNITION

A Hybrid Approach

THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER
SCIENCES

CONNECTIONIST SPEECH RECOGNITION

A Hybrid Approach

by

Hervé Bourlard

Nelson Morgan

Foreword by Richard Lippmann

KLUWER ACADEMIC PUBLISHERS

Copyright Page

Contents

List of Figures

List of Tables

Notation

- $x_n = (x_{n1}, x_{n2}, \ldots, x_{nd})^T$: (acoustic) vector at time n
- d: dimension of acoustic vectors
- $\overline{x}_n = (1, x_{n1}, x_{n2}, \ldots, x_{nd})^T$: augmented acoustic vector at time n
- $()^T$: transpose operation
- ω_k: a category or class
- K: number of classes, HMM states, probability density functions or MLP output classes
- $w_k = \{w_{k0}, \ldots, w_{kd}\}^T$: parameters of a discriminant function associated with class ω_k
- w_{k0}: bias of class ω_k
- $W = \{w_1, \ldots, w_k, \ldots, w_K\}$: weight matrix
- $X = \{x_1, \ldots, x_n, \ldots, x_N\}$: acoustic vector sequence of length N
- $X_{n-c}^{n+c} = \{x_{n-c}, \ldots, x_n, \ldots, x_{n+c}\}$: a subsequence of X of length $2c+1$
- $\mathcal{Y} = \{y_1, \ldots, y_I\}$: set of prototype vectors
- I: total number of prototype vectors
- $Y = \{y_{i_1}, \ldots, y_{i_n}, \ldots, y_{i_N}\}$: prototype vector sequence associated with X; each $y_{i_N} \in \mathcal{Y}$
- $\mathcal{M} = \{m_1, \ldots, m_U\}$: the set of possible elementary speech unit HMMs

- M, M_i: Hidden Markov models built up by concatenating m_u's
- Λ, Λ_i: parameter set of hidden Markov models M and M_i; sometime these will also be used as Lagrange multipliers
- $\mathcal{L}$: parameter set of all the HMMs in $\mathcal{M}$
- $\mathcal{Q} = \{q_1, \ldots, q_k, \ldots, q_K\}$: the set of HMM states described in terms of different probability density functions
- q_k: a HMM state or an MLP output class (associated with HMM state q_k)
- L: total number of state (with possible repetitions of the same q_k's) in a Markov model M
- μ_k: mean vector associated with ω_k or q_k in the case of Gaussian distribution
- Σ_k, σ_k: full covariance matrix or covariance "vector" (if diagonal covariance matrix) associated with ω_k or q_k
- q_k^n: HMM state q_k observed at time n
- q^n: HMM state observed at time n
- q_k^-: HMM observed at the previous time frame
- $Q = \{q^1, \ldots, q^n, \ldots, q^N\}$: a HMM state sequence (of length N)
- $p(\cdot)$: local probability or likelihood (or density function)
- $\hat{p}(\cdot)$: estimate of the local probability or likelihood (or density function)
- $p(q|x)$: local posterior probability
- $p(x|q)$: local likelihood
- $p(q)$: prior probability of class (or HMM state) q
- $P(\cdot)$: global probability or likelihood (relative to a sequence or subsequence of states or acoustic vectors)
- $P(X|M)$: likelihood of X given the Markov model M

- $\overline{P}(X|M)$: Viterbi approximation of the likelihood of X given the Markov model M

- $P(M|X)$: posterior probability of a Markov model M given the acoustic vector sequence X

- $\overline{P}(M|X)$: Viterbi approximation of the posterior probability of M given X

- n_{ik}: number of times a prototype vector y_i of $\mathcal{Y}$ has been observed on a HMM state q_k

- n_k: total number of time state q_k has been observed

- n_i: total number of times prototype y_i has been observed

- $\alpha_n(\ell)$: "forward" probability $= P(q_\ell^n, X_1^n|M)$

- $\beta_n(\ell)$: "backward" probability $= P(X_{n+1}^N|q_\ell^n, M)$

- η: number of layers in a MLP

- W_ℓ: weight matrix between layer $\ell - 1$ and layer ℓ of an MLP

- $\overline{W}_\ell$: augmented weight matrix between layer $\ell - 1$ and layer ℓ of an MLP (taking the bias of layer $\ell - 1$ into account)

- n_ℓ: number of units on layer ℓ

- $g(x_n) = \{g_1(x_n), \ldots, g_k(x_n), \ldots, g_K(x_n)\}^T$: output vector of an MLP as a function of the input pattern x_n

- $g_k(x_n)$: activation values of the k-th output unit of an MLP associated with a class ω_k or a HMM state q_k

- $h_\ell(x_n)$: activation vector of the ℓ-th hidden layer of an MLP, given x_n at the input; $\ell = 1, \ldots, \eta$

- $\overline{h}_\ell(x_n)$: augmented activation vector of the ℓ-th hidden layer of an MLP (to take the bias into account)

- $F(\cdot)$: sigmoid function (applied componentwise if the argument is a vector)

- E: error function minimized for MLP training; usually, this is a mean square error function or a (relative) entropy criterion

- $d_k(x_n)$: desired MLP output vector for pattern x_n at the input

- Δ_k: index vector used as desired output vector for MLP training in classification mode; this is a K-vector with all components equal to zero except the k-th one, which is equal to 1.

- $\Delta(x_n)$: index vector of the class assigned to x_n

- $2c+1$: width of the contextual acoustic information at the input of an MLP (c frames of left context, c frames of right context, and the current frame)

- $\phi(x)$: radial basis function

- $F_k(X_{n-p}^{n-1}, \theta_k)$: linear or nonlinear autoregressive function associated with class q_k and applied, at time n, on the p previous acoustic vectors

- θ_k: parameters of the linear or nonlinear predictor associated with class q_k

- p: order of prediction in AR models

- $\epsilon_{n,k}$: prediction error at time n for state (or class) q_k

- J: number of phonemic contexts

- $c_j,\ j = 1, \ldots, J$: the possible phonemic contexts

- $c_j^\ell, c_j^r,\ j = 1, \ldots, J$: left and rigth phonemic contexts $= c_j$

Foreword

Over the past five years, Hervé Bourlard and Nelson Morgan have been engaged in an international collaboration aimed at determining whether neural networks could be used to improve talker-independent continuous speech recognition. This book provides a detailed review of their successful collaboration. It describes how large multi-layer perceptron networks containing more than 150,000 weights were trained and integrated into a state-of-the-art *Hidden Markov Model* (HMM) recognizer to provide improved acoustic-phonetic modeling and improved recognition accuracy. The lessons learned along the way form a case study which demonstrates how hybrid systems can be developed that combine neural networks with more traditional statistical approaches. The book illustrates both the advantages and limitations of neural networks as seen by researchers who understand both neural networks and alternative statistical approaches.

The book first describes early research and theory which demonstrate that neural networks estimate Bayesian a posteriori probabilities. This allows multi-layer perceptrons to be integrated into hybrid HMM speech recognizers via a common statistical framework. The book then describes problems that were encountered and solved in developing hybrid speaker-dependent and speaker-independent continuous speech recognizers. It describes the Ring Array Processor that was required to obtain practical training times on databases with more than a million input feature vectors. It also describes how cross-validation testing was used to prevent over training; how network outputs were normalized by class prior probabilities to provide scaled likelihoods; and techniques that improved training times, including random sampling, correctly initializing output node biases, gradually decreasing the step-size, and trial-by-trial weight adaptation.

The book presents some unique network designs and proofs motivated by the connection between networks and statistics. A context-dependent neural network is described which estimates context-dependent class probabilities with many fewer weights than would be required if one output was provided for each class-context combination. This approach requires three small networks instead of one much larger network. A modification of radial basis function networks is also described which forces outputs to sum to one. Proofs are presented which demonstrate that network outputs estimate Bayesian a posteriori probabilities and that auto-association networks used for data compression perform a function similar to that performed by principal components analysis.

This book is useful to anyone who intends to use neural networks for speech recognition or within the framework provided by an existing successful statistical approach. It requires some knowledge of speech recognition and signal processing but provides a helpful case study which demonstrates that neural networks are a useful tool that can be used side-by-side with other more accepted statistical approaches.

Richard Lippmann
MIT Lincoln Labs, April 13, 1993

Preface

Since Leibniz there has been no man who has had a full command of all the intellectual activity of his day. There are fields of scientific work which have been explored from the different sides of pure mathematics, statistics, electrical engineering and neurophysiology; in which every single notion receives a separate name from each group, and in which important work has been triplicated or quadruplated, while still other important work is delayed by the unavailability in one field of results that may have already become classical in the next field.
– Norbert Wiener –

At one time, scientific knowledge was held by a relatively small group with a command of an encyclopedic range of topics. Today, it may no longer be possible to do scientific research as a lone wolf. Advances can only be achieved through discussion and worldwide exchanges with scientific colleagues. This is actually a positive development for a number of reasons. First, many ideas can come through the group activity of complementary minds. Past a certain point, participants in such a "group think" exercise are no longer clearly aware of the boundaries of origin of any particular idea. The work reported here is an example of such a collaboration. Secondly, when these advances are obtained through the interaction of colleagues across national lines, this communication is an important component of global awareness. We grow to understand and appreciate each other's cultures, becoming better citizens of our global village. Finally, widespread exchanges force us toward more effective self-criticism, since we know that others viewing our work will certainly see its flaws.

There is, however, a negative side to this distribution of expertise. Innumerable sub-disciplines have developed, often each with its own jargon. This often means that complementary experts may not even share the same technical language, making collaboration difficult. Speech communication with machine is certainly such an area, requiring contributions from linguists, computer scientists, electrical engineers, psychologists, and mathematicians (to name just a few major fields). Even within the engineering aspects of this pursuit, there is a diverse mixture of signal processing, pattern recognition, probability theory, speech science, and system design. For complete systems, these aspects must be augmented by incorporating knowledge of language. The diversity of these topics is one of the greatest problems for machine speech communication.

The work described in this volume is the result of a strong collaboration and friendship between the authors. The joint work began with an extended visit Bourlard made to ICSI starting in 1988, shortly after some seminal work that he had done with Christian Wellekens at Philips in Belgium. After this visit, the research continued both individually and jointly, assisted by frequent short visits, electronic mail, fax, and the occasional expensive phone call. The result was a body of work that we report here, all based on the use of multi-layer perceptrons to estimate the probabilities of sound units for use in continuous speech recognition. While only a small part of the overall problem of speech recognition, it nonetheless brings together a range of subjects. We have tried to describe these pieces using consistent terminology, but hope that we still give the reader a sense of the diversity of the fields required to explore this subject.

Science moves quickly, and the ideas in these pages may well seem naive in a few short years after they were written. This can't be helped. On the other hand, some of these ideas may prove to be important, in which case they will almost certainly be misused (that is, be used to Humankind's detriment). This also cannot be helped. However, we express the hope that the sense of internationalism that was responsible for the research progress will be reflected in the use of this technology for peaceful purposes.

Hervé Bourlard and Nelson Morgan
Berkeley, CA, May 1993.

Acknowledgments

Many people have contributed directly and indirectly to both the work reported here and to this book itself. Some helpful parties will, alas, be unavoidably omitted from any such list.[1] Some of the people who we feel most strongly about thanking are (in no particular order):

- Richard Lippmann for his Foreword and helpful comments.
- Luis Almeida, René Boite, Henri Leich, and John Lazzaro for their early readings and criticism.
- Chuck Wooters for the years of work that he has put into developing the practical instantiations of the systems described here
- Steve Renals for his work with us, both on writing, theory, and practice.
- Christian Wellekens, who collaborated on the work described in parts of Chapters 3 and 7, while he and Bourlard were at Philips.
- Marco Saerens, who collaborated on Chapter 13.
- Yves Kamp, who collaborated with Bourlard on the work reported in Chapter 14.
- James Beck, Phil Kohn, and Jeff Bilmes, who made the RAP machine a reality for the heavy computation required to develop the methods described here.
- Yochai Konig, whose criticism was very helpful during our rewrites, and who has been a contributor in the more recent stages of our work.

[1] We have also gracefully neglected to mention the names of anyone who hindered us.

- Our friends at SRI: Mike Cohen, Horacio Franco, and Victor Abrash, with whom we have collaborated over the last three years, particularly on the context-dependent advances that they have made; and Hy Murveit, with whom we had early discussions (in 1988) that started off our own collaboration.

- ICSI in general and Jerry Feldman in particular for putting up with us while we puzzled through this work and this book.

- Ditto for Jo Lernout and LHS.

- Beatrice Benjamin, for the cartoons that brighten up this sober treatise.

- Vivian Balis, for her help with the other figures.

- Molly Walker, for her careful editing of the final version.

Of course, we also gratefully acknowledge the recent support from the European Community's ESPRIT program (Basic Research Project no. 6487), as well as ARPA (and in particular, Barbara Yoon) for support of our collaboration with SRI under ARPA contract MDA904-90-C-5253.

We also thank the world's leaders for not blowing up the planet while we were writing this, as we REALLY wanted to finish it.

We thank our families for their support.

Finally, we would regretfully like to say farewell to a friend and colleague that left us this year: Frank Fallside, who will be missed.

Part I

BACKGROUND

Chapter 1

INTRODUCTION

New opinions are always suspected, and usually opposed, without any other reason, but because they are not already common.
– John Locke –

For thirty years, *Artificial Neural Networks* (ANNs) have been used for difficult problems in pattern recognition [Viglione, 1970]. Some of these problems, such as the pattern analysis of brain waves, have been characterized by a low signal-to-noise ratio; in some cases it was not even known what was signal and what was noise.

More recently, ANNs have been applied to *Automatic Speech Recognition* (ASR). Despite the relatively deep knowledge that we have about the speech signal, ASR is still difficult. This is so for a number of reasons, but partly because the field is motivated by the promise of human-level performance under realistic conditions, and this is currently an unsolved problem. For speech communication, statistically significant classification accuracies are of no interest if they are low compared to human listeners.

So ASR is a hard problem and ANNs can be helpful for hard pattern recognition problems. However, we are wary of assuming that neural "magic" can solve this or any other problem. Practical pattern recognition systems are not realized simply by one monolithic element, either in the form of a single ANN or any other homogeneous component. Solving a real-world problem almost always requires the crafting of a heterogeneous system from modules that the engineer has in his toolkit. This is at least partly because the structure of hard problems themselves is typically heterogeneous. One might have a component that is best described as signal processing, another as a classification or pattern matching module, and yet another might incorporate syntactic or semantic knowledge. ASR is no exception; even for the "standard" statistical systems that we have used as our starting point, complete recognizers consist of a number of critical pieces. This modular property is fortuitous for researchers. We can pick some aspect of the whole problem that we consider suboptimal, and attempt to improve it with a new technique (or an old one that has not been applied to this subtask). Ultimately, there are strong advantages to greater homogeneity – for instance, doing the entire task with ANN modules; in principle, this would provide greater flexibility for global optimization of the system. At the moment, we don't know how to do this.

The focus of this book, as suggested by the title,[1] is on the integration of ANNs into an ASR system for continuous speech. This has been done for an important recognition subtask, phonetic probability estimation. These probabilities are used as parameters for *Hidden Markov Models* (HMMs), currently the framework of choice for state-of-the-art recognizers. Keeping the overall framework of a conventional recognizer has permitted us to make controlled comparisons to evaluate new techniques.

1.1 Automatic Speech Recognition (ASR)

The dominant technological approaches for speech recognition systems are based on pattern matching of statistical representations of the acoustic speech signal, such as HMM whole word and subword (e.g., phoneme) models. However, although significant progress has been made in the field of ASR these last years, the typical vocabulary size is still very limited[2] and the performance of the resulting systems is still not comparable to that achieved by human beings.

Considering the immense effort that has gone into studying speech recognition over the last four decades, one might wonder why this area is still a topic for research. As early as the 1950s, researchers built

[1] See the front cover if you've forgotten it.

[2] While some existing systems work with very large vocabularies, there is always some compensating restriction, such as the limitation to a very task-specific grammar, for any system that works well enough to be useful.

simple recognizers with credible performance for restricted tasks (such as isolated digits spoken by a single talker). Unfortunately, the techniques used in these systems were not sufficient to solve the general problem of ASR. The difficulties of this problem can be described in terms of a number of characterizations of the task, including:

1. *Intra- and inter-speaker variabilities*: Is the system *speaker dependent* (i.e., optimized for a single talker), or *speaker independent* (can recognize anyone's voice)? Typically, speaker-dependent systems can achieve better recognition performance than speaker-independent systems because the variability of the speech signal (i.e., the way words and subwords are pronounced) is more limited. However, this is achieved at the cost of a new enrollment (training) session that has to be done for every new speaker; the memory requirements will also be larger since one has to store specific models for every user.

   For most applications (e.g., the public telephone network), only speaker-independent systems are useful. In this case, speaker independence is usually achieved by using the same baseline models (e.g., HMMs) that are trained on databases containing a large population of representative talkers. In this case, the ASR system does not require specific training and uses the same set of models of every user.

   A solution between speaker-dependent and speaker-independent systems consists in doing fast *speaker adaptation*. In this case, starting from pre-trained (e.g., speaker-independent) models, one tries to adapt the parameters of the models quickly to better match the characteristics of the user's voice. This can be performed in a supervised way (e.g., prompting the speaker for a small set of utterances, or taking a corrective action each time the user detects an error) or unsupervised way (from a few unconstrained utterances from the speaker).

2. Is the system able to recognize *isolated or continuous speech*? With *isolated word recognition* systems, the talker is required to say words with short pauses in between. This is the simplest case, since word boundaries are detected fairly easily, and since the words are not strongly coarticulated.

Continuous Speech Recognition (CSR) systems can recognize a sequence of words that are spoken without requiring pauses between the words. In this case, the words in the utterances can be strongly coarticulated, which makes recognition considerably more difficult. Typically these systems assume speech with a predefined lexicon and syntax. A very challenging extension of such systems achieves recognition of *natural* or *spontaneous speech*, for which the talker is not constrained by vocabulary size or artificial grammatical constraints. This is still an open research issue since, in this case, the system has to deal with speech disfluencies, hesitations, non-grammatical sentences, out-of-vocabulary words, etc.

Another problem that is neither isolated word recognition nor continuous speech recognition (but which is as difficult as CSR) is often referred to as *keyword spotting* (KWS). In this case, one wants to detect "keywords" from unconstrained speech, while ignoring all other words or non-speech sounds. This is a kind of generalization of an isolated word recognition system in which the user is not constrained to pronounce the words in isolation; this leads to more user-friendly systems.[3] The challenging problem of rejecting utterances with no keywords is also usually addressed in a KWS system.

3. *Vocabulary size and confusability*: Is the system able to recognize a vocabulary of only a few words, or can it handle large vocabularies of thousands of words? What is the potential confusability between words? In general, it is difficult to get good recognition results with a large vocabulary, and the computation time can also be an issue in this case. Adding more words increases the probability of confusion between words. Also, since more recognition time is required for a larger lexicon, faster search techniques (e.g., beam search and fast look-ahead) are required, and can degrade the final performance of the system. For large vocabularies, it is also not generally feasible to work with whole word models, since this would require a prohibitive amount of training data, particularly for a speaker-independent system. A natural subword unit is the phoneme, but such units are strongly coarticulated; that is, the

[3] Diabolical system designers can also use this approach to create a user-nasty system, such as one that reboots whenever the user says "pizza".

pronunciation of each linguistically based sound unit is strongly dependent on its acoustic-phonetic context. In general, ASR performance is significantly affected by the acoustic confusability of the vocabulary to be recognized, which can lead to difficulties even for small vocabularies. This is the case, for example, with the "E" set of the alphabet, that is, for the spoken names of the letters that rhyme with "E" (b, c, d, e, g, p, t, v, and z).

4. Are there any *task and/or language constraints*? In most cases, the task of continuous speech recognition is simplified by restricting the possible utterances. This is usually done by using syntactic (and, sometimes, semantic) information to reduce the complexity of the task and the ambiguity between words. However, this is still a very active research area since it is not known how to properly interface general grammars and natural speech constraints with acoustic recognizers. The use of semantic information is still an open issue. Since the degree to which these non-acoustic knowledge sources limit the possible utterances can differ, vocabulary size is not a good measure of a CSR task's difficulty. The constraining power of a language model is usually measured by its *perplexity*, roughly the geometric mean of the number of words that can occur at any decision point.[4] If N is the number of lexicon words, the perplexity $P \in [1, N]$. A high perplexity generally implies a high level of difficulty for a task, since many competing word candidates must be examined by the acoustic recognizer. The CSR tasks that will be described in this book will usually have roughly a 1,000 word lexicon and a perplexity equal to 60.

5. Does the system work in *adverse conditions*? Several variables that can alter the performance of ASR systems have been identified:

   - environmental noise, i.e., stationary or nonstationary additive noise (e.g., in car, cockpit or on factory floor);
   - distorted acoustics and speech correlated noise (e.g., reverberant room acoustics, nonlinear distortions);
   - different microphones (e.g., telephone set, superdirectional or close-talking microphones,) and different filter characteristics

[4] See [Jelinek, 1990] for a more precise description.

(for which the telephone channel is a particular case), which usually lead to convolutional noise;

- limited frequency bandwidth (e.g., telephone channels where the transmitted frequencies are limited between approximately 350 Hz and 3,200 Hz);
- altered speaking manner, (e.g., Lombard effect, differing speaking rate, speaker stress, breath and lip noise, pitch, uncooperative talker, etc.); or
- some combination of the above (which is, unfortunately, the most frequent case).

Some systems can be more robust than others in response to some of these factors, but in general recognizers are overly sensitive in this regard.[5]

6. Will the system be trained to recognize read or *natural, spontaneous speech*? Virtually all practical applications require the recognition of natural, spontaneous speech. On the other hand, nearly all research experiments have used read text as input (see the ATIS task [DARPA, 1991], an experimental airline reservation system, for a notable exception). The practical difference is a host of disfluencies that people produce – for instance, filled pauses ("um" or "er") or false starts. Additionally, in natural speech, talkers will almost certainly use some words that are outside of the recognizer lexicon. All of these factors make CSR much more difficult.

In this book, we will address the problem of speaker-dependent and speaker-independent CSR for "read" speech, with moderate size lexicons (typically 1,000 words) and a simplified language model with moderate to high perplexities (commonly 60, but much higher for some tasks). This has been chosen as the reference domain for the approach proposed in the book for several reasons:

- Tests will be performed on standard databases for which results obtained with state of the art systems, highly optimized for these tasks, are available for comparison;

[5] See [Furui, 1992] for a good overview of these variables and the methods that are currently used or investigated to cope with them.

- These tasks are easy enough to get some "promising" results, but also are hard enough to identify statistically significant improvements or degradations resulting from a new approach; and

- After the developments reported in this book, we were able to integrate our new modules into a state-of-the-art system to boost its general recognition performance.

1.2 Limitations in Current ASR Systems

An excellent early report [Davis et al., 1952] described one of the first successful speech recognition systems. While the recognizer worked well, it was limited to recognition of isolated digits for a single speaker. A reading of the text suggests that the authors believed that unrestricted recognition of natural speech was a short hop away. It has been stated for many years that the solution to the speech recognition problem was only five years away.[6]

Unfortunately, the problem is not that simple. Words that have entirely different meanings (and consequences!) can have very similar phonetic structure. Additionally, words uttered in connected speech are said in very different ways, often including the complete deletion of phonemes that are clearly stated in isolated words (words bounded by silence). There are an infinite number of sentences that may be spoken (since there is no restriction on sentence length), and even an artificial restriction on sentence length (say, to less than 20 seconds) permits a gigantic number of possible word combinations. Each such combination affects the way words and phonemes are spoken (especially due to contextual effects from neighboring sounds). Moreover, as noted earlier, variations in the speech collection environment, such as room acoustics, channel spectral characteristics, or microphone response can all make major changes in the speech spectrum, all of which can seriously degrade recognizer performance. When a system must be speaker-independent, the variability for dialect, speaking speed, and other talker-dependent aspects further increase the difficulty of the task, typically doubling the error rate, even within the same dialect or accent group.

Most of these difficulties can be summarized fairly simply: variability of the speech acoustic and variation of additive and convolutional noise.

[6] Since it has been repeated for so long, clearly it must be true ...

Additionally, however, we instinctively expect a high level of recognition performance, much as would be achieved by a human (or for a keyboard input, for instance), and have very little interest in a recognizer that makes frequent mistakes. For these reasons, speech recognition must achieve a very high level of performance to be of general interest as a man-machine interface.

What is the performance currently available from the best speech recognition system? Certainly such systems can give extremely impressive results in the laboratory, particularly with a task-restricted grammar; 95-98% accuracies have been reported for the recognition of 1000 words in continuous speech [Weintraub et al., 1989; Lee et al., 1989]. However, unconstrained background noise, microphone characteristics, and grammars that are more characteristic of real applications result in far lower performance. Even the relatively simple case of telephone digits, for which rates greater than 99% have been achieved, becomes extremely difficult with the channel variation from real telephone lines; a recent experiment contrasted a 97% accuracy for speaker-independent isolated digit recognition with a 40% score when changes in the telephone channel were not handled [Hermansky, 1990b; Hermansky et al., 1991]. Furthermore, without restrictive grammars (which are not realistic for general speech), even the best speaker-independent systems currently give about 20% word error and about 80% sentence error. This is far from what one would require in a dictation system, for instance.

In short, despite encouraging progress over the last few years, speech recognition by machine still has a long way to go to be good enough to be generally useful.

1.3 Book Overview

In this work we will show how neural network techniques can complement traditional approaches to improve state-of-the-art continuous speech recognition systems. However, we will also show that this is not easy; we should not rely too much on the "magical" problem-solving abilities of ANNs without a clear understanding of the underlying principles and tasks.

After a basic review of statistical pattern classification (Chapter 2), Chapter 3 recalls the main features and underlying hypotheses of *Hidden Markov Models (HMMs)*, stochastic models that are now widely used for automatic isolated word and connected speech recognition. One of their main advantages lies in their ability to represent the time sequential order of speech signals and the variability of speech production by the use of powerful matching techniques such as the *Baum Welch* or *Viterbi* algorithms. The parameters of these models are learned from large data bases that are assumed to be statistically representative. For their training, a critical problem is the choice of a learning criterion. A model is said to be discriminant if it maximizes the probability of producing an associated set of features while minimizing the probability that they have been produced by rival models. Problems related to the choice of a criterion are discussed in Section 3.5.

Chapter 4 focuses on linear discriminant functions and a kind of ANN called a *Multilayer Perceptron* (MLP). Like HMMs, these are learning machines, but they also provide discriminant-based learning; that is, models are trained to suppress incorrect classification as well as to model each class separately. MLPs can acquire pattern knowledge by learning, and can then recognize patterns that are similar to those presented in the learning set. This stands in contrast with the HMM approach, which uses a separate model for each phonemic class and usually makes assumptions about the distribution of the associated feature vectors. Further, in contrast to HMMs, MLPs can capture high-order constraints between data while discriminating between classes.

However, the sequential nature of the speech signal, which is inherent to the HMM formalism, remains difficult to handle with neural networks. Chapter 5 reviews the different approaches that have been proposed so far to handle sequential signals: models with buffered inputs or recurrent neural networks. Some of these models have already proved useful in recognizing isolated speech units. By their dynamical properties, these models are able to include some kind of time warping, or at least some integration over time. However, ANNs by themselves have not been shown to be effective for large scale continuous speech recognition. It currently appears that a good solution is to integrate them into a statistical framework using standard approaches such as HMMs.

To properly interface ANNs with HMMs, it was necessary to understand what these networks are actually computing. In Chapter 6, it is shown that the output values of MLPs may be considered to be estimates of Bayes (a *posteriori*, or posterior) probabilities when trained for pattern classification. In other words, the MLPs can estimate the conditional probability of each class of speech sound, given the speech data.

This property is proved theoretically and demonstrated experimentally on a realistic task.

This was an important link between MLPs and HMMs. However, showing that an MLP can generate *a posteriori* probabilities is not sufficient for recognition, since HMMs require the estimation of likelihoods. That is, standard HMM recognizers require the estimation of the probability of producing the observed data assuming each sound class. Accordingly, a new kind of HMM, referred to as a *discriminant* HMM, using posterior probabilities instead of likelihoods, is introduced in Chapter 7. While improving the HMM discriminant capabilities, these models preserve the algorithmic aspects of HMMs (e.g., Viterbi-like training and recognition procedures) and have the potential to overcome some of the major weaknesses of standard HMMs. However, it appears that it is not easy to properly interface MLPs and HMMs. Consequently, the modifications of the basic scheme that were necessary to get good word recognition performance are presented and discussed in Chapter 7.

In Chapter 7, only discrete features are used. In Chapter 8, the hybrid HMM/MLP approach is extended and tested with continuous acoustic vectors and dynamic features, and this is shown to improve recognition performance. In addition, we show how our approach was successfully integrated into DECIPHER [Cohen et al., 1990], currently (1993) one of the best large vocabulary, speaker-independent, continuous speech recognition systems. This integration improved the performance of a state-of-the-art system.

The initial hybrid HMM/MLP approach focused on HMMs that are independent of phonetic context, and for these models the MLP approaches have consistently provided significant improvements. In Chapter 9, this approach is extended to context-dependent models. The technique presented is not really constrained to speech recognition; it is a general approach to split any large net used for pattern classification into smaller nets without any assumptions of independence.

Since recognizer accuracy is only one measure of a practical speech recognition system, Chapter 10 examines the system tradeoffs for MLP probability estimation and compares the resources requirements (storage, memory bandwidth, and computation) for HMMs, both with and without an MLP. Chapter 11 describes the development of some ANN acceleration hardware and software that were required to speed the research work described in this book.

The next part of this book (Part III) discusses related problems or approaches that have been investigated in the framework of this work. In Chapter 12, we describe the approach of cross-validation that has been instrumental in our achievement of good recognition results with a hybrid HMM/MLP hybrid structure. Although this has been done in the framework of speech recognition, cross-validation is a general technique to avoid overfitting when the number of training patterns is not large in comparison with the number of ANN parameters to be learned.

In Chapter 13, another kind of hybrid HMM/MLP approach using MLPs as autoregressive models (as initially introduced in, e.g., [Levin, 1990; Tebelskis et al., 1991]) are discussed in the general framework of linear and nonlinear predictive models.

In Chapter 14, we investigate the possibility of using MLPs for feature extraction. Auto-associative MLPs have sometimes been proposed for feature extraction or dimensionality reduction of the feature space in information processing applications. This chapter shows that, for auto-association with linear output units, the nonlinearities at the hidden units are unnecessary. This is so because the optimal transformation and the corresponding parameter values can be derived directly by purely linear techniques, relying on singular value decomposition and low rank matrix approximation, similar in spirit to the Karhunen-Loève transform.

We conclude the book with Part IV, consisting of a general overview of the final system and the Conclusions. Chapter 15 summarizes the hybrid HMM/MLP approach (including training) that led to our basic system and discusses the possible extensions that are the subject of current research. Chapter 16 contains the conclusions and some speculation about future research trends in the field of ANN-based pattern recognition and speech recognition.

In summary, this work describes a first attempt at incorporating neural network approaches for continuous speech recognition. In this attempt we are largely constrained to use neural networks for well defined subtasks. However, the principles established here should be applicable to other parts of speech recognition, and to some extent to quite different tasks in statistical pattern recognition.[7]

[7] But time will tell.

Chapter 2

STATISTICAL PATTERN CLASSIFICATION

In this world, nothing can be said to be certain,
except death and taxes.
– Benjamin Franklin –

2.1 Introduction

No new engineering or scientific technique, however novel, evolves in isolation. Both *Hidden Markov Model* (HMM) and *Multilayer Perceptron* (MLP) based approaches have been developed in the context of a long history of pattern recognition technology. Though specific methods are changing, the pattern recognition perspective continues to be useful for the description of many problems and their proposed solutions.

While pattern recognition is in general more complex than simply categorizing an input into one of a number of possible classes, the classification perspective is useful. The notation and definitions for this model of pattern recognition will provide tools that permit us to characterize more complex problems, such as speech recognition. In this chapter we will briefly describe the classical elements of pattern classification, with an emphasis on the statistical approach. The definitions and concepts will be useful for understanding the later chapters. Readers with a strong background in pattern recognition may want to scan this chapter quickly.[1] Beginners who would like to see a more thorough treatment are referred to [Fukunaga, 1972] and [Duda & Hart, 1973], two truly wonderful books.

2.2 A Model for Pattern Classification

Many pattern recognition problems can be partially or completely described in terms of the goal of classifying some set of measurements into categories. Many cases are more complex, e.g., deriving a description for a visual scene or translating a speech utterance into a database query. While these problems still are in some sense recognizing a pattern of symbols or values, their solution is far more than the identification of a pattern from some fixed set of classes. Nonetheless, even in such cases there are almost always components of the problem that are well described as either the hard or soft classification of measurements; in the latter case, a graded decision (such as a probability) is required.

Generically, a pattern classifier consists of two major pieces: a *feature extractor* and a *classifier*. These two modules ultimately have the same goal, namely to transform the input into a representation that is trivially convertible into a class decision. The difference between the two modules is one of perspective. Traditionally the feature extractor is used to convert the raw input into a form that is easily classified; this is a common place to incorporate domain-specific knowledge that will greatly enhance performance over the blind use of automatic techniques. For example, for speech recognition, conversion from a continuous waveform to a series of short-term spectral representations has been shown to be beneficial. This is true despite the fact that, in theory, automatic transformation techniques that are part of the classifier (e.g., neural networks) can learn this mapping.

In addition to the representational transformation of the feature extraction stage, this piece of the system typically (though not universally) introduces some coarser granularity. For instance, while the speech waveform might be sampled at 8-16 kHz, the time-varying spectral measure is typically sampled at 100 Hz. This is not necessarily a dimensionality reduction, although transforming to a smaller representation is frequently a motivation for feature extraction; some representations (such as the more complex auditory models for the case of a speech signal) can even

[1] Or skip it entirely. It's a short chapter so you won't miss many jokes.

increase the signal dimensionality. In either case, the feature extraction stage must produce a representation that is good for separation or *discrimination* between classes.

For the purposes of this book, we will assume a data representation that is discretized in time.[2] The data corresponding to each chunk of data (e.g., a speech segment) is referred to as a *pattern*. From these patterns, the feature extractor will produce a sequence of vectors, where each vector component is a variable that has been computed from the original data. We denote this *feature vector* as $x = (x_1, x_2, \ldots, x_d)$, a d-dimensional vector. [3]

Given a set of feature vectors, the classification stage is designed to discriminate between them on the basis of class. *Trained classifiers* are developed using a set of feature vector examples, with the object of developing a numerical or symbolic rule that can make the desired distinction for data that has not yet been seen. Typically the training develops a set of *discriminant functions*, one for each class, which are optimized in an attempt to produce the largest value for the function corresponding correct class. For the purposes of this chapter we will denote ω_k, $k = 1, \ldots, K$, as the list of pattern classes.

In general, then, we wish to find the best feature extractor and classifier to discriminate between patterns that belong to different classes. For simplicity's sake, we can define "best" here as the one which provides the fewest classification errors on a new sample of data.[4]

We have described feature extraction and the classification decision as being distinct processes. We briefly note that it is almost always desirable to integrate these steps as much as possible. For instance, the best features are the ones that are chosen to give good discrimination between classes, and the best classifiers are those that do not critically rely on incorrect assumptions about the feature representation.

There are many kinds of pattern classifiers. For the purposes of this book, we will be limited to statistical methods. Statistical pattern

[2] In general, patterns do not need to be part of a sequence, but can be any set of examples that we wish to categorize, such as the set of height and weight values for all American and Belgian speech researchers.

[3] In later sections, the feature vectors will also be indexed over time n and will be denoted $x_n = \{x_{n1}, x_{n2}, \ldots, x_{nd}\}$.

[4] More generally, there is some cost associated with each different kind of error, so that the optimum system in the sense of least cost will not necessarily be the optimum system in the sense of minimizing the number of errors.

classification has the advantage of a powerful mathematical framework.[5] Given this framework, we can develop new methods and relate them to existing approaches. It is the capability for such insights that interests us most.

2.3 Statistical Classification

As noted above, statistical approaches provide us with a strong body of relevant theory.[6] In the statistical framework, the concept of optimum classification that was discussed in the previous section can be made much more concrete. For some restricted cases, this formulation can even tell us precisely what the optimum classifier must be. More generally, we are given no such prescription; nonetheless, the ideal cases can provide us with some intuition to help understand what new approaches are doing.

Using the notation of the previous section, we can denote the probability that a pattern belongs to a class as $p(\omega_k)$ (where $0 \leq p(\omega_k) \leq 1$). Since this expression contains no dependence on feature vectors, and thus can be expressed before new data is seen, it is generally referred to as the *prior* or *a priori* probability of class ω_k. If a class probability is conditioned on the feature vector, we can express the probability of class membership given the pattern. In the notation of the previous section, for feature vector x this conditional probability is denoted $p(\omega_k|x)$. Since this probability can only be estimated after the data has been seen, it is generally referred to as the *posterior* or *a posteriori* probability of class ω_k.

The generic pattern classification problem can now be stated in a statistical framework: given a set of measurements, vector x, what is the probability of it belonging to a class ω_k, i.e., $p(\omega_k|x)$? Intuitively, if one knew these probabilities for all classes and for every possible feature vector, a good decision could be made. As it turns out, such a decision can be formalized and proven to be optimum in the same sense described in the last section. That is, assuming that the pattern must come from one class of $\omega_1, \omega_2, \ldots, \omega_K$, if we construct a classifier that assigns x to

[5] Unfortunately, the assumptions required for the relevant theory to be valid are almost never realistic. This hasn't stopped any of us.

[6] We assume here that the reader is familiar with the basic ideas of probability, including probability density functions. If not, go directly to jail, do not pass Go, and do not collect \$200.

class ω_k if:

$$p(\omega_k|x) \;>\; p(\omega_j|x), \forall\ j = 1, 2, \ldots, K, j \neq K \tag{2.1}$$

then we can show that such a rule will provide the minimum probability of error for any classifier.[7] This would mean that the expected number of errors for a new unseen set of feature vectors would be minimized. In the terminology of the previous section, we would be using $p(\omega_k|x)$ (over all values of k) as the discriminant functions. This optimum strategy is often called *Bayes' Decision Rule*. Simply put, it requires that we assign a pattern to the class that has the highest posterior probability (i.e., given the vector x). Because of the optimality of this decision rule, the posterior is also sometimes referred to as the *Bayes probability*.

Alas, life is not so simple.[8] How do we actually get these probabilities? In general they cannot be computed directly, and can only be estimated from the data (and from assumed prior knowledge). The accuracy of the estimates will ultimately determine the performance of the classifier. Commonly, this estimation is simplified by making some assumptions about the pattern data. For instance, the unknown distributions are often characterized by some parametrized model. In other words, we define the form of the statistical distribution, so that we only have to estimate the parameters, rather than a complete (often continuous) probability density function. Most commonly, we assume this distributional form separately for each class, and for the probability density $p(x|\omega_k)$ [as opposed to the posterior probability $p(\omega_k|x)$]. This is the probability density function for the feature vector x among those patterns that belong to class ω_k, and is often referred to as the data *likelihood*. This class-conditional data likelihood is simply related to the posterior probability by Bayes' rule, which is:

$$p(\omega_k|x) \;=\; \frac{p(x|\omega_k)p(\omega_k)}{p(x)} \;=\; \frac{p(x|\omega_k)p(\omega_k)}{\sum_{j=1}^{K} p(x|\omega_j)p(\omega_j)} \tag{2.2}$$

where K is the total number of classes. In this expression, $p(x|\omega_k)$ and $p(x)$ are both probability functions or both density functions, according as x is discrete or continuous, and $p(\omega_k|x)$ and $p(\omega_k)$ are likewise both probability functions or both density functions, according as ω_K

[7]See Duda & Hart for the proof, which takes about 2 lines of math, but there are some nice text and pictures to make it easy.

[8]Or maybe it is, but we just don't know how to look at it.

is discrete or continuous. In this book, we will mainly make references to probabilities or likelihoods, but we'll have to keep in mind that these can be actual probabilities or probability density functions. This relation tells us that the posterior probability (or posterior density) is essentially the product of the likelihood function $p(x|\omega_k)$ for the observed feature x and the prior probability $p(\omega_k)$. The denominator in the formula is just a normalization constant to make the sum over all posterior probabilities equal to 1. The notion of using Bayes' rule to modify prior beliefs is the essence of "Bayesian" statistics. This somewhat indirect way of calculating the posterior probabilities is not without drawbacks:

- Some assumptions are still required about the form of the parametric model of $p(x|\omega_k)$ (see below).
- *A priori* probabilities are generally very difficult to estimate reliably.
- Using only the likelihood during training (i.e., estimation of the model's parameters) results in poor discrimination between models.

Using Bayes' rule, we can reformulate the Bayes' Decision Rule as follows: assign x to class ω_k if:

$$\frac{p(x|\omega_k)}{p(x|\omega_j)} > \frac{p(\omega_j)}{p(\omega_k)}, \forall\ j = 1, 2, \ldots, K, j \neq K \tag{2.3}$$

Note that in this formulation the factor $p(x)$ has been factored out. The fraction on the left is commonly referred to as the *likelihood ratio.* This is also equivalent to using a discriminant function that is the product $p(x|\omega_k)p(\omega_k)$ for each value of k.

In most systems, likelihoods are estimated using the parametric model of a "normal" (or Gaussian) distribution:

$$\begin{aligned} p(x|\omega_k) &= N(x, \mu_k, \Sigma_k) \\ &= \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right) \end{aligned} \tag{2.4}$$

where μ_k and Σ_k respectively denote the mean vector and the covariance matrix associated with class ω_k. If the diagonal elements of the covariance matrix are represented by σ_{ki}^2, and we assume that the covariance matrix is diagonal (i.e., the measurements of our feature vector are assumed to be uncorrelated), this expression reduces to

$$p(x|\omega_k) = N(x, \mu_k, \Sigma_k) = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_{ki}^2}} \exp\left(-\frac{(x_i - \mu_{ki})^2}{2\sigma_{ki}^2}\right) \quad (2.5)$$

where μ_{ki} denotes the i-th component of μ_k.

Although it may appear that the choice of this distributional form is arbitrary, normal densities have many useful properties that make them reasonable models. The most obvious is the fact that this distribution is a reasonable approximation to the actual distributions found in many real data sets. This is a consequence of a rule known as the Central Limit Theorem, which says that observable phenomena that are a consequence of many unrelated causes, with no single cause predominating over the others, tend to a Gaussian distribution.[9] The Gaussian is also a good approximation of many other distributions. Its most endearing quality is the fact that it is easy to work with – its distribution has been well researched and there is a large pool of knowledge from which to draw when using it. The Gaussian is *unimodal*; in other words, there is a single "bump" or mode (maximum), which occurs at the mean. More complex distributions with multiple modes can be approximated by weighted sums of Gaussian distributions called *Gaussian mixtures*. These can be expressed as:

$$p(x|\omega_k) = \sum_{j=1}^{J} c_{kj} \; N_j\left(x, \mu_j, \Sigma_j\right) \quad (2.6)$$

where J is the total number of Gaussian densities. Parameters c_{kj} are the mixture gain coefficients and are additional parameters to be trained, with the constraints:

$$c_{kj} \geq 0,$$

and

$$\sum_{j=1}^{J} c_{kj} = 1, \forall\; k = 1, \ldots, K$$

For each of these expressions, there are a number of parameters that must be estimated from the data. In the case of a single Gaussian, this

[9]This was proven by Markov, which shows that he did more than assemble chains.

is trivial if the classes are known; once the means and covariances are estimated for each class, the work is done. In the case of the mixture, the parameters cannot be determined so directly, since the means and covariances do not in this case are not *a priori* associated with a particular subset of the feature vectors. However, all of the parameters of (2.4), (2.5) and (2.6) can be iteratively determined by the powerful "Estimate and Maximize" (EM) algorithm [Dempster et al., 1977; Baum, 1972]. This approach is guaranteed to iteratively converge to a local minimum of some measure of error between the model and the sample distribution of the data.

By adopting models for the conditional probability $p(x|\omega_k)$ based on standard distributions like (2.4), (2.5) or (2.6) and applying Bayes' rule, we can define a relatively straightforward statistical classifier. The performance will depend on how closely the pattern data does actually fit the model selected, but generally Bayesian classifiers based on simple parametric assumptions can perform well. In passing, it might be important to recall here that if the likelihoods $p(x|\omega_k)$ are assumed to follow Gaussian distributions, the posterior probabilities $p(\omega_k|x)$ generally do not have Gaussian-like distributions at all [Duda & Hart, 1973].

For the case of Gaussian distributions like (2.4), it can be shown that a quadratic decision surface is provided by the Bayes classifier; this can be seen by using log likelihoods and priors in the discriminant function rather than the densities themselves. In this case the exponential of the normal distribution drops away, leaving only first and second order functions of the feature vector. If a further assumption is added, namely that the two class distributions have equal covariance matrices, it can also be shown that a linear classifier is optimal; this is demonstrated in Section 4.2.3. Note that this constraint is equivalent to the assumption that the distributions both have the same overall shape, spread and correlation between features. In general, a linear classifier can be useful for other distributions, though it is not optimal. Because if its simplicity, it can be derived in many ways, including through deterministic procedures.

In later chapters it will be shown that neural networks can be seen as generalizations of these simpler pattern classifiers. They can form more complex and unconstrained decision surfaces, and so do not require strong assumptions about the pattern distributions. However, in many cases it is useful to view the network's operation in terms of the simpler techniques described in this chapter, in other words, as approximations to a Bayes' classifier.

2.4 Pattern Classification with Realistic Data

The distinction between training and testing data is a critical one for pattern classification. In some simple examples of function learning (such as the famous XOR problem) a relationship that can be explicitly stated for the complete set of possible feature vectors is learned. However, in classification problems we often have a huge or even infinite number of potential feature vectors, and the system must be trained using a very limited subset that has been labeled with class identity. Thus, even though the relationship between feature vector and class might be learned perfectly for some training set of data, this will not in general mean that the likelihoods or posteriors are well-estimated for the general population of possible samples.

In general, the classification error on the training set patterns should be viewed as a lower bound. A better estimate of the classifier error is obtained using an independent test set. The larger this test set, the better the representation of the general population of possible test patterns. Conventional significance tests (e.g., a normal approximation to the binomial distribution for correctness on the test set) should be done as a sanity check. For instance, a 49% error on a million-pattern test set is significantly different from chance (50% error) on a two-class problem; it represents 10,000 patterns. On the other hand, the same error percentage on a 100-pattern test set is indistinguishable from chance performance.[10] One way to effectively increase the size of a test set is to use a "jackknife" procedure, in which each split of the data (e.g., fifths) is used in turn for test after using the remaining part for training. Thus, all of the available data is ultimately used for the test set.

Training set size is also a major concern for real problems. The larger the training set, the better the classifier will do on representing the underlying distributions. Also, the more complex the recognizer (e.g. the larger the number of independent parameters), the greater the risk of over-specializing to the training sample, and generalizing poorly to independent test sets. Both of these factors push pattern recognition researchers toward using as much training data as possible. Unfortunately, in most practical situations, there are strong limits on the available data.

[10] For the normal approximation to a binomial distribution, the equivalent standard deviation is $\sqrt{npq}$, where n is the number of patterns, p is the probability of getting the class correct by chance, and q is the probability of getting the class incorrect by chance. For the examples above, this would be 500 and 5 patterns, respectively.

In this case, clever approaches to squeezing out information from limited data are essential, whether they are based on domain-specific knowledge, or on general constraints that diminish the tendency for the classifier to "over-learn" the training set.[11]

Another difficulty associated with the finite size of the training set concerns the *discriminant* character of a statistical classifier. As noted in the previous section, likelihoods and posteriors are simply transformable between one another (by Bayes' rule). However, in the case of probability estimates (as opposed to true probabilities), the training criterion will produce different results. For instance, if the probabilities are estimated in a procedure that attempts to maximize the discrimination between classes, the classification error will be minimized. This might actually produce poorer estimates of the likelihoods than would be observed after training by a criterion that attempts to best model the underlying densities $p(x|\omega_k)$. This would not be the case for an infinite amount of training data, for which the estimates could converge to the true underlying distributions, so that Bayes' rule would literally be satisfied and the criteria would be equivalent.

In general, making strong assumptions about the data (such as the Gaussian assumption discussed in this chapter) improves the quality of the estimate – IF the assumption is correct. To the extent that these assumptions are wrong, the resulting estimates are poor. Approaches such as EM estimation of mixture Gaussians and gradient-based neural network training incorporate successively weaker constraints, and thus have largely supplanted the earlier, simpler models (e.g., single Gaussians per class).

Speech recognition requires the incorporation of "soft" classification decisions as part of the more global decision about the utterance. This requires a dynamic model that represents sequences, rather than the static pattern classification described in this chapter. A general statistical framework for this more general case is presented next.

[11] In chapter 12, we describe such a general form of training constraint, called *cross-validation*. In this approach, training is modified so that generalization performance is guaranteed never to become worse.

2.5 Summary

In this chapter, we have very briefly introduced the basic notions (and notations) of feature extraction and statistical pattern classification that will be used in this book. Ideally, the feature extractor and the classifier should be jointly optimized. However, in most cases, these processes are developed independently of each other. Statistical classification should rely on the posterior probability. However, since this probability cannot generally be estimated directly from the data (parametric models are not known, and there is not enough training data to estimate the priors implicitly included in it), it is usually replaced by a likelihood criterion, which will be good only if the assumptions made about the distribution of the data are correct. Also, the use of likelihoods instead of posterior probabilities will result in poorer discrimination properties.

Chapter 3

HIDDEN MARKOV MODELS

Everything should be made as simple as possible,
but not simpler.
– Albert Einstein –

3.1 Introduction

The most efficient approach developed so far to deal with the statistical variations of speech (in both the frequency and temporal domains) consists in modeling the lexicon words or the constituent speech units by *Hidden Markov Models* (HMMs) [Baker 1975a,b; Jelinek 1976; Bahl et al., 1983; Levinson et al., 1983; Rabiner & Juang, 1986]. First described in [Baum & Petrie, 1966; Baum & Eagon, 1967; Baum, 1972], this formalism was proposed as a statistical method of estimating the probabilistic functions of Markov chains.[1] Shortly afterwards, they were extended to automatic speech recognition independently at CMU [Baker, 1975b] and IBM [Bakis, 1976; Jelinek, 1976].

Speech is a non-stationary process, but it is assumed to be "quasi-stationary." That is, it is assumed that over a short period of time the statistics of the speech do not differ from sample to sample. Un-

[1] Reminder: a k-th order Markov chain is a sequence of discrete random variables that depends only on the preceding k variables. For HMM systems, typically k is 1.

der this assumption a preprocessor can extract statistically meaningful acoustic parameters (feature vectors) at regular intervals from a sampled speech waveform. Typically the original speech is sampled at 8 to 16 kHz (8 for telephone speech and 16 or so for high-quality microphone input). The features are commonly extracted once per 8-16 msec, and are calculated from a 20 to 30 msec analysis window. Many variants of spectral analysis have been used, including Linear Predictive Coding (LPC), Perceptual Linear Prediction (PLP) [Hermansky, 1990a], and log power spectral or cepstral coefficients computed from a spectrum with "mel scale" spacing, which roughly corresponds to auditory critical bands. Physiologically-based auditory models have also been developed by a number of researchers and are summarized in [Greenberg, 1988]. Simpler models based on some gross psychoacoustic characteristics that display some robustness to the channel characteristic were described in [Hermansky et al., 1991].

First and second time derivatives of spectral or cepstral parameters between successive time slots are also often added to the speech features to represent some of the speech dynamics [Furui, 1986; Lee, 1989]. These derivatives may be calculated by a simple differencing operation or by polynomial approximation to the derivative (since differencing is inherently very noisy). These "delta features" provide information about spectral changes over time, although their exact interpretation is still not entirely clear.

Whatever the acoustic features are, the feature extraction process converts the speech signal into a sequence of acoustic vectors $X = \{x_1, x_2, \ldots, x_N\}$. For a good review of the basics of digital processing of speech signals, see [Rabiner & Schafer, 1978].

Hidden Markov modeling assumes that the sequence of feature vectors is a *piecewise stationary* process. That is, an utterance is modeled as a succession of discrete stationary states, with instantaneous transitions between these states. Essentially, a HMM is a stochastic finite state machine with a stochastic output process attached to each state[2] to describe the probability of occurrence of some feature vectors. A simple HMM is illustrated in Figure 3.1. Thus we have two concurrent stochastic processes: the sequence of HMM states modeling the temporal structure of speech; and a set of state output processes modeling the

[2]More generally, the stochastic process could be regarded as being attached to the transitions – see Section 3.3 for more details about this.

(locally) stationary character of the speech signal. The HMM is called "hidden" because the sequence of states is not directly observable, but affects the observed sequence of events.

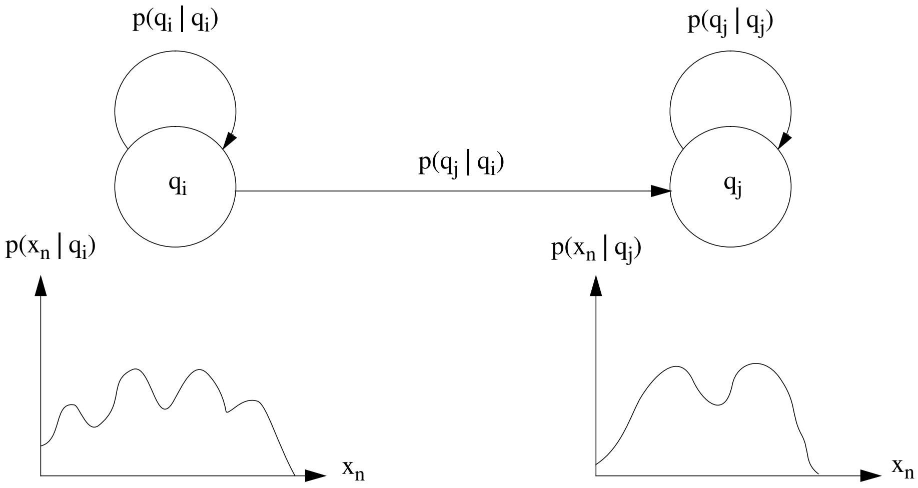

Figure 3.1: A schematic of a two state, left-to-right hidden Markov model (HMM). A hidden Markov model is a stochastic automaton, consisting of a set of states and corresponding transitions between states. HMMs are hidden because the state of the model, q_i, is not observed; rather the output, the acoustic vector x_n of a stochastic process attached to that state, is observed. This is described by a probability distribution $p(x_n|q_i)$. The other set of pertinent probabilities are the state transition probabilities $p(q_j|q_i)$.

Ideally, there should be one HMM for every sentence allowed in the recognition task. However, this is clearly infeasible, so that a hierarchical scheme must be adopted. First, a sentence is modeled as a sequence of words. The total number of models required is then much smaller. However, to further reduce the number of parameters (and, consequently, the required amount of training material) and to avoid the need of a new training each time a new word has to be added to the lexicon, sub-word units are usually preferred to word models. Although there are good linguistic arguments for choosing units such as syllables or demi-syllables, the unit most commonly used is the phoneme. This is the unit used in this work, in which we will use between 60 and 70 phoneme models.[3] In this case, word models consist of concatenations of phoneme models (constrained by pronunciations from a lexicon), and sentence models consist of concatenations of word models (constrained by a grammar).

By the late 1980's, a number of laboratories were able to demonstrate large-vocabulary (1,000 words), speaker-independent, continuous speech recognition systems based on HMMs [Lee, 1989; Weintraub et al., 1989; Paul, 1989]. These systems extended the earlier HMM approaches in several ways, such as:

- better modeling of speech dynamics, by extending the feature vectors to first and second time derivatives;
- adding phonological rules; and
- better speech units such as context-dependent phoneme models (e.g., biphones and triphones).

Although this kind of approach is very efficient and can lead to very impressive results in the laboratory, the numerous implicit assumptions that make optimization of these models possible limit their generality. In this chapter, we will briefly recall the fundamentals of standard HMM approaches by making explicit all the underlying assumptions. The algorithms used to evaluate their parameters and to perform decoding will also be recalled. For reviews and further reading on the fundamentals of HMMs, see [Baker 1975a,b; Jelinek, 1976; Bahl et al., 1983; Levinson et al., 1983; Rabiner & Juang, 1986; Rabiner, 1989; Lee, 1989].

This chapter is not intended to present a deep description of HMM-based speech recognition and training algorithms, but only to give the preliminary body of knowledge necessary for this book. Its primary goal is also to list the many implicit assumptions that are made when using HMMs and that led us to use neural networks to try to alleviate some of these. For more details about the subject, we recommend [Deller Proakis & Hansen, 1993], a recent book that gives a very good (and detailed) overview of the most common speech analysis methods, and a very clear description of the theoretical developments related to HMMs.

[3] In fact, we generally use categories that are more like phones, i.e., acoustic categories, whereas phonemes are linguistic categories.

3.2 Definition and Underlying Hypotheses

In the HMM formalism, the speech signal is assumed to be produced by a finite state automaton built up from a set of K states $\mathcal{Q} = \{q_1, q_2, \ldots, q_K\}$ governed by statistical laws and (non-emitting) initial and final states q_I and q_F. Each speech unit (e.g., each vocabulary word or each phoneme) is associated with a Markov model made up of states from $\mathcal{Q}$ according to a predefined topology. Markov models of words or sentences can then be built up by concatenating (according to phonological rules) elementary speech unit models.

Let $\mathcal{M} = \{m_1, \ldots, m_U\}$ represent the set of possible elementary speech unit HMMs and $\mathcal{L} = \{\lambda_1, \ldots, \lambda_U\}$ the set of associated parameters. In the following, M (or M_j) will represent the HMM associated with a specific word or sentence X (or X_j) and obtained by concatenating elementary HMMs of $\mathcal{M}$ associated with the speech units constituting X and made up of L states $q_\ell \in \mathcal{Q}$ with $\ell = 1, \ldots, L$. The same state may occur several times and with different indices ℓ, so that $L \neq K$. The set of parameters present in M (or M_j) will be denoted Λ (or Λ_j) and is a subset of $\mathcal{L}$

The key training and decoding criterion of HMMs is based on the *posterior probability* $P(M|X)$ that the acoustic vector sequence X has been produced by M. During training, we want to determine the set of parameters $\mathcal{L}$ that will maximize $P(M|X, \mathcal{L})$ for every X associated with M; this is usually referred to as the *Maximum A Posteriori* (MAP) probability criterion. During recognition, we have to find the best M that maximizes $P(M|X, \mathcal{L})$ given a fixed set of parameters $\mathcal{L}$ and an observation sequence X. Unfortunately, the training process usually does not permit characterization of $P(M|X)$ directly but only the probability that a given model will generate certain acoustic vector sequences, i.e., $P(X|M)$.[4] Using Bayes' law, $P(M|X)$ can be expressed in terms of $P(X|M)$ as

$$P(M|X) = \frac{P(X|M)P(M)}{P(X)} \tag{3.1}$$

in which $P(X|M)$ is usually referred to as the *likelihood* of the data given the model M, $P(M)$ is the prior probability of the model (usually given by the language model), and $P(X)$ is the prior probability of the acoustic vector sequence.

[4]See discussion in Chapter 2 and consequences of this in Section 3.6.

First set of hypotheses:

- **H1:** It is usually assumed that $P(M)$ can be calculated separately (i.e., without acoustic data). In continuous speech recognition, M usually represents a sequence of word models for which the probability $P(M)$ can be estimated from a language model, usually formulated in terms of a stochastic grammar. However, if M represents elementary HMMs representing phonemes, the lexicon and the grammar *together* make up the language model, specifying prior probabilities for sentences, words and phones. This will be discussed further in Section 7.8.

- **H2:** For a known sequence of observations, $P(X)$ can be assumed constant since it does not depend on the models. However, we will see in the next section that this is only true if the model's parameters are fixed.

In this case, estimation of (3.1) amounts to calculating $P(X|M)$, the probability that M has generated the acoustic vector sequence X. When used for training this criterion is usually referred to as the *Maximum Likelihood Estimate* (MLE) criterion, emphasizing that optimization (i.e., maximization of $P(X|M)$) is performed in the parameter space of the probability density functions or likelihoods.[5] As discussed in Section 3.6, a consequence of this assumption is that, during training, $P(X|M)$ will depend only on the set of parameters Λ present in M [which will sometimes be made explicit by writing $P(X|M,\Lambda)$] while $P(M|X)$ was dependent on the whole set of parameters $\mathcal{L}$ [which will sometimes be made explicit by writing $P(M|X,\mathcal{L})$].

Given this formulation, three problems have to be addressed when using HMMs for speech recognition (see [Rabiner & Juang, 1986; Lee, 1989] for an extended development on this theme):

- **Parametrization and estimation of probabilities** – How should $P(X|M)$ be computed, and what are the necessary assumptions about the HMMs to define a useful parameter set $\mathcal{L}$?

- **Training problem** – Given a set of observation sequences X_j, $j = 1, \ldots, J$, associated with their respective Markov models M_j, how should the model's parameters be determined so that each model

[5] $P(X|M)$ is not a probability, but rather a probability density function (pdf).

has the highest possible probability $P(X_j|M_j, \Lambda_j)$ of generating its associated observation sequences, i.e.,

$$\operatorname*{argmax}_{\mathcal{L}} \prod_{j=1}^{J} P(X_j|M_j, \Lambda_j)$$

However, it is important to remember here that, ideally, the parameters should be determined so that they maximize $P(M_j|X_j, \mathcal{L})$ since this is a discriminant criterion that actually minimizes the error rate (recalling Chapter 2; also see discussion in Section 3.6).

- **Decoding problem** – Given the set of Markov models with their trained parameters and a sequence of observations X, how should the best sequence M_k of elementary Markov models be found to maximize the probability that M_k generated the observations?

3.3 Parametrization and Estimation

3.3.1 General Formulation

The estimation problem can be stated as: given a Markov model M, compute the probability $P(X|M)$ that it will generate the sequence of acoustic vectors $X = \{x_1, x_2, \ldots, x_N\}$. Starting from the actual likelihood criterion we will show the hypotheses that are necessary to make the model computationally tractable.

We define a path C of length N in the model as an ordered sequence $Q_1^N = \{q^0 = q_I, q^1, q^2, \ldots, q^N, q^{N+1} = q_F\}$ of N states (non-emitting initial and final states excluded) entering via the initial state q_I and ending via the final state q_F of M. Both initial and final states are assumed to be non-emitting states (i.e., no acoustic vectors are associated with them).[6] Along this path, let q^n represent the HMM state at time n, with $q^n \in \mathcal{Q}$, and let q_k^n mean that specific state q_k is visited at time n. If Γ denotes the set of all possible paths C in M, the actual calculation of $P(X|M)$ involves summing the probabilities of all possible paths $C \in \Gamma$, i.e.:

$$P(X|M) = \sum_{\Gamma} P(C, X|M) = \sum_{\Gamma} P(Q_1^N, X|M) \tag{3.2}$$

[6]As shown later, these states will be used only to initialize recursions.

To make apparent all possible paths, (3.2) can also be rewritten (without any simplifying assumptions) as:

$$P(X|M) = \sum_{\ell_1=1}^{L} \cdots \sum_{\ell_N=1}^{L} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, X|M) \tag{3.3}$$

for all possible $\{q_{\ell_1}^1, \ldots, q_{\ell_N}^N\} \in \Gamma$. Since events q_ℓ^n are mutually exclusive, probability $P(X|M)$ can also be written for any arbitrary n and without any simplifying assumptions:

$$P(X|M) = \sum_{\ell=1}^{L} P(q_\ell^n, X|M), \text{ for any } n \in [1, N] \tag{3.4}$$

where $P(q_\ell^n, X|M)$ denotes the probability that X is produced by M while visiting state q_ℓ at time n and can also be written as:[7]

$$P(q_\ell^n, X|M) = P(q_\ell^n, X_1^n|M)P(X_{n+1}^N|q_\ell^n, X_1^n, M) \tag{3.5}$$

where X_m^n denotes the partial vector sequence $\{x_m, x_{m+1}, \ldots, x_n)$. In this expression, $P(q_\ell^n, X_1^n|M)$ represents the joint probability of having generated the partial observation sequence X_1^n and having arrived at state q_ℓ at time n given HMM M; this probability will be defined as the *forward probability* $\alpha_n(k)$ in Section 3.3.4 [equation (3.13)]. $P(X_{n+1}^N|q_\ell^n, X_1^n, M)$ represents the probability of the partial observation sequence X_{n+1}^N given that the partial observation sequence X_1^n has already been observed while arriving at state q_ℓ at time n; this probability will be defined as the *backward probability* in Section 3.3.4 [equation (3.15)]. It is then easy to show that, without any assumptions, the following recursion holds:

$$P(q_\ell^n, X_1^n|M) = \sum_{k=1}^{L} P(q_k^{n-1}, X_1^{n-1}|M)p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, M) \tag{3.6}$$

which will be defined later on as the forward recursion of the *Forward-Backward algorithm* [Baum, 1972; Bourlard et al., 1985; Lee, 1989] (see Section 3.3.4 for further discussion).

[7] Throughout much of this book, the following "trick" is used, which is essentially a restatement of the definition of conditional probability: $p(a, b|c) = p(a|b, c)p(b|c)$. This decomposition proves very useful in understanding the progression through many of the equations.

In many texts, notations $P(\cdot)$ and $p(\cdot)$ stand for actual probabilities and probability density functions respectively. However, in this book, we will assume that the reader can infer which is referred to, and will instead use $P(\cdot)$ to denote "global" or "accumulated" probabilities or densities (where the left-hand side of the conditional contains a sequence of acoustic vectors or a sequence of HMM states) while $p(\cdot)$ will stand for local probabilities (i.e., relative to a specific acoustic vector and/or a specific HMM state).

The conditional probability $p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, M)$ in (3.6) is usually referred to as the *local contribution*. To make the models computationally tractable, the influence of the conditional events are usually limited by some simplifying hypotheses. For example, the dependence is restricted to the last emitted vector; this could be achieved by concatenating two successive vectors as in [Furui, 1986] and in [Marcus, 1981, 1985], while an explicit formulation is given in [Wellekens, 1987]. However, in standard HMMs [Bourlard et al., 1985; Jelinek, 1976], the local contribution is assumed independent of the whole previously emitted vector sequence X_1^{n-1} and thus reduces to $p(q_\ell^n, x_n|q_k^{n-1}, M)$. This is equivalent to assuming that, not only are the acoustic vectors uncorrelated, but also the Markov models are first-order Markov models. Indeed, to make these new assumptions more explicit, let us rewrite (first without any of these assumptions) the local contribution in (3.6) as:

$$
\begin{aligned}
&p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, M) \\
&\quad = p(q_\ell^n|q_k^{n-1}, X_1^{n-1}, M)p(x_n|q_\ell^n, q_k^{n-1}, X_1^{n-1}, M)
\end{aligned}
\tag{3.7}
$$

Actually, equation (3.7) reflects the property that hidden Markov models can in fact be described in terms of two correlated Markovian processes, one corresponding to the state sequence, and the second one to the acoustic vector sequence. This explains why HMMs are called "hidden" Markov models since there is an underlying Markov process (i.e., the state sequence) which is not observable, but affects the observed Markov process describing the sequence of events. It is then clear that, when assuming the independence of the local contribution in (3.6), two hypotheses are actually made (one for each underlying Markov process), i.e.:

Second set of hypotheses:

- **H3:** It is assumed that Markov models are *first-order Markov models*, i.e.:
$$p(q_\ell^n|q_k^{n-1}, X_1^{n-1}, M) \;=\; p(q_\ell^n|q_k^{n-1}, M) \qquad (3.8)$$
which states that the probability that the Markov chain is in state q_ℓ at time n depends only on the state of the Markov chain at time $n-1$, and is conditionally independent of the past (both the past acoustic vector sequence and the states before the previous one). Probabilities $p(q_\ell^n|q_k^{n-1}, M)$, $\forall k, \ell = 1, \ldots, L$, are usually referred to as *transition probabilities* and represent the probability to go from q_k to q_ℓ in a Markov model M. Since these transition probabilities are usually assumed to be stationary (models are usually assumed to be time invariant), the upper time indices will be dropped in the following when there is no risk of confusion. Although these transition probabilities are usually dependent on M, this conditional will sometimes be overlooked in the following when considering a specific HMM. These transition probabilities are nonnegative (of course!), and the probabilities of all transitions that leave a state sum to one, i.e.,
$$\sum_{\ell=1}^{L} p(q_\ell|q_k, M) = 1, \; \forall k = 1, \ldots, L$$
for every specific M. These probabilities can be estimated during training by counting the relative occurrence of state transitions.

- **H4:** Another common related assumption is the *observation-independence assumption*, i.e.:
$$p(x_n|q_\ell^n, q_k^{n-1}, X_1^{n-1}, M) \;=\; p(x_n|q_\ell^n, q_k^{n-1}, M) \qquad (3.9)$$
which states that the acoustic vectors are not correlated or, in other words, that the probability that a particular acoustic vector will be emitted at time n depends only on the transition taken at that time (from state q_k^{n-1} to q_ℓ^n), and is conditionally independent of the past. Probabilities $p(x_n|q_\ell^n, q_k^{n-1}, M)$ are usually referred to as *emission-on-transition probabilities* [Jelinek, 1976] and represent the probability of observing x_n while doing a transition from state q_k to q_ℓ.

Given these two assumptions, the local contribution in (3.6) thus reduces to

$$p(q_\ell^n, x_n | q_k^{n-1}, X_1^{n-1}, M) = p(q_\ell | q_k, M) p(x_n | q_\ell^n, q_k^{n-1}, M) \qquad (3.10)$$

Given this formulation, one other assumption is commonly made:

- **H5:** Although emission-on-transition probabilities are sometimes used as such [Jelinek, 1976; Lee, 1989], they are often assumed to depend only on the current state to reduce the number of parameters, i.e.,

  $$p(x_n | q_\ell^n, q_k^{n-1}, M) \;=\; p(x_n | q_\ell) \qquad (3.11)$$

  in which case the $p(x_n|q_\ell)$'s are simply referred to as *emission probabilities*. Since these emission probabilities are usually assumed independent of the model, M has been removed from the conditional.[8] Also, since the models are assumed time invariant, the upper time indices of the states will generally be dropped when they are not required for clarity. Notation like q_ℓ and q_k^- will sometimes be used to remind the reader that state q_k was the state visited before q_ℓ.

Taking hypotheses H3-H5 into account, (3.6) reduces to:

$$P(q_\ell^n, X_1^n | M) = \left[\sum_{k=1}^{L} P(q_k^{n-1}, X_1^{n-1} | M) p(q_\ell | q_k, M) \right] p(x_n | q_\ell) \qquad (3.12)$$

which is the basic recursion of HMMs.

To compute the emission probabilities of (3.12), each state q_k of $\mathcal{Q}$ (or each pair of states in the case of emission on transitions) has to be associated with a probability density function describing $p(x_n|q_k)$. Since the acoustic vectors x_n are often in the form of *d-dimensional real-valued feature vectors*, additional hypotheses about the underlying probability density function describing these emission probabilities are usually required.

[8] This is usually referred to as *state tying* between models, i.e., a state q_k is described in terms of the same probability density function independently of the model M it belongs to. If we do not assume that the emission probabilities are independent of M, this is then equivalent to specific states with probability density functions for every possible Markov model M.

3.3.2 Continuous Input Features

Let X represent a sequence of acoustic vectors $\{x_1, \ldots, x_n, \ldots, x_N\}$, where each x_n belongs to a predefined d-dimensional feature space (e.g., cepstral vectors). To estimate the emission probabilities, some assumptions about the underlying distribution associated with each state q_k are required. In the case of continuous input, it is particularly useful to split the local contribution as done in (3.10), since this leaves only the continuous random variable x_n on the left-hand side of the conditional in the emission probabilities.

- **H6:** As mentioned in Chapter 2, in many practical situations $p(x_n|q_\ell)$ in (3.12) is estimated by assuming it to be in the form of a multivariate Gaussian distribution (2.4). To reduce the number of free parameters, the components of the feature vector are often assumed to be uncorrelated (i.e., the covariance matrix is assumed to be diagonal), so we can compute $p(x_n|q_k)$ by (2.5). Since the normal distribution with a diagonal covariance matrix is generally not a good enough model, emission probabilities are often estimated from multivariate Gaussian mixture densities (2.6) [Rabiner et al., 1985]. Other forms include the Gaussian autoregressive mixture density [Juang & Rabiner, 1985], the Richter mixture density [Richter, 1986], and the Laplace mixture density [Ney & Noll, 1988]. Whatever solution is chosen, assumptions are made about the underlying probability density functions. Although it can be proved that a mixture of Gaussians can model any kind of distribution, we still do not know how many Gaussians are needed per state, and the number that we can effectively train will be dependent on the size of the available training set.

  A problem worth noting in all these approaches is that the probability $p(x_n|q_k)$ is usually not computed but is simply replaced by the value of the probability density function on x instead of estimating the actual probability, i.e., the integral of the pdf on $[x - \epsilon, x + \epsilon]$. However, if ϵ is small enough, the actual probability is then equivalent to the density within a scaling factor equal to 2ϵ. This problem does not appear in the case of discrete input features.

3.3.3 Discrete Input Features

In discrete HMMs, the acoustic vector sequence X is quantized in a front-end processor where each acoustic vector x_n is replaced by the closest (in the sense of Euclidian or Mahalanobis distance) prototype vector y_i selected in a predetermined finite set $\mathcal{Y} = \{y_1, \ldots, y_i, \ldots, y_I\}$ of cardinality I. Usually, $\mathcal{Y}$ is referred to as the *codebook* of prototype vectors. This prototype vector set is usually determined independently by using clustering algorithms, e.g., K-means or binary splitting algorithms, on a large number of acoustic vectors [Linde et al., 1980; Makhoul et al., 1985]. In this case, the acoustic vector sequence X is replaced by a (prototype) vector sequence $Y = \{y_{i_1}, \ldots, y_{i_n}, \ldots, y_{i_N}\}$, where i_n represents the label of the closest prototype vector of $\mathcal{Y}$ associated with x_n.

Consequently, emission probabilities can be described in terms of discrete probability density functions, and every L-state Markov model is characterized by $I \times L^2$ parameters

$$p(y_i|q_\ell, q_k^-)$$

for $i = 1, \ldots, I$ and $k, \ell = 1, \ldots, L$, if emission-on-transition probabilities are used [i.e., the left-hand side of (3.11)], or by $I \times L$ parameters

$$p(y_i|q_\ell)$$

if we assume that the emission probabilities depend only on the current state [right-hand side of (3.11)].[9]

- **H7:** In the discrete case, we do not require hypothesis H6 about the underlying distribution of the emission probabilities. However, there is a similar assumption about the distribution of the feature space that is implicit in the choice of distance metric for the clustering procedure (e.g., the Euclidian distance measure in the case of the K-means clustering algorithm).

Additionally, if multiple features are used, quantization distortion is reduced if separate codebooks are used; however, since the resulting probabilities are combined by multiplication, there is an implicit assumption of feature independence.

[9] Again, if these discrete emission probabilities are assumed to be independent of the Markov model M (i.e., tied emission probabilities), the total number of parameters in these two cases will be respectively equal to $I \times K^2$ and $I \times K$.

In the case of continuous HMMs, there is no need to choose the distance metric or vector quantization heuristics. The parameters of the models are computed directly but at the cost of limiting assumptions regarding the form of the probability density function. In a common form of these assumptions, Gaussians or Gaussian mixtures are used with covariance matrices having nonzero elements only on the diagonal. This is equivalent to assuming that the features are uncorrelated.

In today's speech recognition systems, the choice between discrete and continuous HMMs often depends on several factors:

- Compared with discrete models, continuous distributional models usually require fewer parameters, which limits their memory requirements.

- Since they have fewer parameters, continuous distributional models usually require less training data to achieve good generalization performance. However, if enough training data is available (which is rarely the case!), discrete models can give a better estimate of the observation density function.

- Generally, continuous HMMs require longer training and recognition time. Continuous models require the calculation of several Gaussian or multi-Gaussian densities. For discrete HMMs, probability calculation is replaced by a look-up table and only quantization of the acoustic vectors has to be performed (which can be optimized by tree-search approaches like binary splitting).

An intermediate approach uses what are referred to as semi-continuous HMMs, and aims at incorporating the advantages of both discrete and continuous HMMs while keeping the number of parameters and the computation time acceptable. Essentially, semi-continuous HMMs use a "codebook" of Gaussians shared by all output pdfs associated with $q_k \in \mathcal{Q}$. See Section 6.7.2 for a further discussion and extension to the relationship with ANNs.

3.3.4 Maximum Likelihood Criterion

The problem considered here is the following: given a set of model parameters, how do we compute the likelihood $P(X|M)$ (3.2) of an acoustic vector sequence X given a Markov model M? Unfortunately, direct computation of (3.2) is clearly infeasible since the number of operations is $O(2NL^N)$ (see [Rabiner, 1989] for the full calculation).[10]

The so-called *Forward-Backward algorithm* [Baum & Eagon, 1967; Baum, 1972] can be used to efficiently compute $P(X|M)$. This is achieved by splitting $P(X|M)$ into a sum of products of a "forward" and a "backward" probability [as done in (3.4) and (3.5)]. Considering (3.5), we may define the *forward probability*

$$\alpha_n(\ell) = P(q_\ell^n, X_1^n|M), \forall \ell \in [1, L] \tag{3.13}$$

as the joint probability of having generated the partial observation sequence $X_1^n = \{x_1, \ldots, x_n\}$ and having arrived at state state q_ℓ at time n given HMM M. In this case, recursion (3.12) [i.e., recursion (3.6) after hypotheses H3-H5] can be rewritten as:

$$\alpha_{n+1}(\ell) = \left[\sum_{k=1}^{L} \alpha_n(k) p(q_\ell|q_k)\right] p(x_{n+1}|q_\ell), \forall \ell \in [1, L] \tag{3.14}$$

and is often referred to as the *forward recursion* of the Forward-Backward algorithm. By definition of $\alpha_n(\ell)$, it is clear that recursion (3.14) is initialized by setting

$$\alpha_1(\ell) = p(x_1|q_\ell)p(q_\ell|q_I), \forall \ell \in [1, L]$$

where q_I is the initial (non emitting) state of M.

In a similar way, and considering the second factor in the right-hand side of (3.5), we can define the *backward probability*

$$\beta_n(\ell) = P(X_{n+1}^N|q_\ell^n, X_1^n, M) \tag{3.15}$$

as the probability of the partial "backward" observation sequence $X_{n+1}^N = \{x_{n+1}, \ldots, x_N\}$, given that the partial observation sequence X_1^n has already been observed while arriving at state q_ℓ at time n. The corresponding *backward recursion* of the Forward-Backward algorithm is then:

$$\beta_n(\ell) = \sum_{k=1}^{L} p(q_k|q_\ell)p(x_{n+1}|q_k)\beta_{n+1}(k), \forall \ell \in [1, L] \tag{3.16}$$

[10]There are L^N possible state sequences, and for each of them about $2N$ computations are necessary.

with the initialization

$$\beta_N(\ell) = 1 \text{ or } 0, \forall \ell \in [1, L] \tag{3.17}$$

depending on whether q_ℓ is a legal ending state (i.e., directly connected to q_F, $\beta_N(\ell) = 1$) or not ($\beta_N(\ell) = 0$).

According to (3.4), we then have:

$$P(X|M) = \sum_{\ell=1}^{L} \alpha_n(\ell)\beta_n(\ell), \text{ for any } n \in [1, N] \tag{3.18}$$

and the global likelihood can therefore be calculated at any time slot n. However, we can also work at the final time $n = N$ and, inserting (3.17) into (3.18), we obtain:

$$P(X|M) = \sum_{\forall \ell \in \mathcal{F}} \alpha_N(\ell) \tag{3.19}$$

where $\mathcal{F}$ stands for the set of legal final states (directly connected to q_F). In the same way, and incorporating symbols x_0 and x_{N+1} (which are virtual extrapolations of the actual observation sequence) for convenience, we obtain:

$$P(X|M) = \alpha_{N+1}(F) = \beta_0(I) \tag{3.20}$$

Although this backward recursion is not actually required to estimate $P(X|M)$, it was defined here since it will be used for training (Section 3.4.1).

Given that $\alpha_n(k)$ accounts for X_1^n and state q_k at time n, while $\beta_n(k)$ accounts for X_{n+1}^N given q_k at time n, it is also interesting to note that the forward and backward recursions together allow us to estimate the probability of any state q_k, $\forall k \in [1, L]$, at time n (and as the full likelihood of the complete vector sequence):

$$P(q_k^n, X|M) = \alpha_n(k)\beta_n(k) \tag{3.21}$$

It can also be shown that, using the forward recursion, the number of operation required to compute $P(X|M)$ has been reduced to $O(NL^2)$. In practice, however, most models are not fully connected, so the complexity is more like $O(NL)$ with a small constant for the average number of connections into a state.

3.3.5 Viterbi Criterion

The Viterbi criterion can be viewed as an approximation to the full likelihood (MLE) criterion where, instead of taking account of all possible state sequences in M capable of producing X, one merely considers the most probable path. This criterion can be used to provide an explicit segmentation of the observation sequence, which can sometime be very useful.

An explicit formulation of the Viterbi criterion is obtained by replacing all summations in (3.3) by a "*max*" operator, yielding

$$\overline{P}(X|M) = \max_{\ell_1,\ldots,\ell_N} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, X|M) \tag{3.22}$$

where $\overline{P}(\cdot)$ represents the Viterbi approximation of the full likelihood. In this case, (3.6) then becomes:

$$\overline{P}(q_\ell^n, X_1^n|M) = \max_k \left[\overline{P}(q_k^{n-1}, X_1^{n-1}|M) p(q_\ell^n, x_n|q_k^{n-1}, X_1^{n-1}, M)\right] \tag{3.23}$$

Using hypotheses H3-H5, (3.12) becomes:

$$\overline{P}(q_\ell^n, X_1^n|M) = \max_k \left[\overline{P}(q_k^{n-1}, X_1^{n-1}|M) p(q_\ell|q_k, M)\right] p(x_n|q_\ell) \tag{3.24}$$

which gives us the basic recursion of the Viterbi algorithm [Viterbi, 1967]. The global likelihood $P(X|M)$ is then approximated as:

$$\overline{P}(X|M) = \max_{\ell \in \mathcal{F}} \overline{P}(q_\ell^N, X_1^N|M) \tag{3.25}$$

$$= \overline{P}(q_F^{N+1}, X_1^{N+1}|M) \tag{3.26}$$

which represents the probability of the best path or, in other words, the maximum likelihood state sequence. Recurrence (3.24) can be reformulated in terms of *Dynamic Programming* (DP), also referred to as *Dynamic Time Warping* (DTW),[11] where distances are defined as the negative logarithm of probabilities [Sankoff & Kruskal, 1983; Bourlard et al., 1985].

[11]In some usages, DTW is distinguished from the Viterbi in that the former is a deterministic application of dynamic programming to speech, with nothing like the transition probabilities of the Viterbi. However, in both cases the best warping of the time axis is found to match the acoustic inputs with the models, so the distinction is probably academic.

3.4 Training Problem

The parameters to be determined in an HMM-based ASR system are the topology of the model (number of states and allowable transitions), the state transition probabilities, and the parameters related to the emission probabilities (i.e., symbol output probability matrices in the discrete case, or means and variances in the continuous (Gaussian) case). Large amounts of training data (per speech unit) are needed to obtain good estimates of these probabilities.

The choice of a topology to describe the observed sequence of events for a given speech unit is often ad hoc. For word models, one might select a number of states which roughly corresponds to the number of phonemes in the word to be modeled, or else to the number of pseudo-phonemes, which can be determined by vector quantization of an utterance of that word. In the case of phoneme models, the number of states in the word models is automatically proportional to the number of phonemes (since word models are built up by concatenating constituting phoneme models). However, in this case, we still have to choose the topology of each phonemic HMM, often chosen to be a strictly left-to-right 3-state model. In the case of sub-unit HMMs (like phonemes) it is useful to perform embedded training [Bourlard et al., 1985] to take account of the statistical variations in speaking rate and pronunciation. In this case, we assume that training is performed on a set of (known) training utterances X_j, $j = 1, \ldots, J$, labeled (but not necessarily segmented) in terms of speech units. As a consequence, each of the training utterance can be associated with a specific Markov models M_j of parameters $\Lambda_j \in \mathcal{L}$ obtained by concatenating elementary HMMs $m_u \in \mathcal{M}$ of parameters λ_u associated with the speech sub-units constituting the utterance. In the case of MLE criterion, the goal of the training is then to find the best set of parameters $\mathcal{L}$ such that:

$$\underset{\mathcal{L}}{\text{argmax}} \prod_{j=1}^{J} P(X_j | M_j, \Lambda_j) \tag{3.27}$$

Since MLE is not a discriminant criterion, maximization of this product can be done independently for each factor (each model M_j).[12] As a

[12] In fact, this is not completely true since two Markov models M_j and M_i (respectively parametrized by Λ_j and Λ_i) can have a common subset of elementary speech units m_u and, consequently, the same subset of parameters λ_u. In this case we will

consequence, in the following, we will consider only a particular training utterance X and its associated Markov model M (modification of the training algorithms to take multiple training utterances into account is straightforward).

Unfortunately, maximization of $P(X|M,\Lambda)$ in the parameter space Λ does not have a direct analytical solution. However, as briefly shown in the following, iterative procedures known as the *Forward-Backward algorithm* and the *Viterbi algorithm* may be used to locally maximize $P(X|M,\Lambda)$ (MLE criterion) or $\overline{P}(X|M,\Lambda)$ (Viterbi criterion).

3.4.1 Maximum Likelihood Criterion

The most popular approach to iteratively maximize $P(X|M,\Lambda)$ has been described by Baum and his colleagues in a number of classic papers [Baum & Petrie, 1966; Baum et al., 1970; Baum, 1972]. Starting from initial guesses Λ^0, the model parameters Λ are iteratively updated according to the Forward-Backward algorithm (or equivalently the Estimate-and Maximize (EM) algorithm [Dempster et al., 1977]) so that $P(X|M,\Lambda)$ is maximized at each iteration. This kind of training algorithm, often referred to as *Baum-Welch* training, can also be interpreted in terms of gradient techniques [Levinson et al., 1983; Levinson, 1985].

Though backward recursions were not both necessary for decoding, both the forward and backward recursions are required to derive the re-estimation formulas. Although this will not be fully described in this book, the general concept is recalled here.[13]

For a heuristic derivation of the re-estimation procedure of the HMM parameter, we must define the probability of being in state q_k at time n, given the complete observation sequence X and the model parameters Λ, i.e.,

$$\gamma_n(k) = p(q_k^n|X,M,\Lambda) \;=\; \frac{P(q_k^n, X|M,\Lambda)}{P(X|M,\Lambda)} \tag{3.28}$$

where the second equality comes from the definition of conditional probability (the joint probability divided by the marginal probability). Given

---

have to make sure that, during training, parameters λ_u remain the same for both models M_j and M_i (referred to as "parameter tying"). However, this does not significantly modify the following reasoning.

[13]As usual, if you are either too much of an expert or a greenhorn, you might want to skip this section. Then again, you might be missing something ...

(3.21), we have:

$$\gamma_n(k) \;=\; \frac{\alpha_n(k)\beta_n(k)}{P(X|M,\Lambda)} \;=\; \frac{\alpha_n(k)\beta_n(k)}{\sum_{\ell=1}^{L}\alpha_n(\ell)\beta_n(\ell)} \tag{3.29}$$

in which the normalization factor guarantees that

$$\sum_{k=1}^{K}\gamma_n(k) = 1,\, \forall n = 1,\ldots,N$$

As a consequence, the probability of producing X while visiting q_k (but without specifying at what time n, i.e., without specifying the emitted vector x_n) is given by:

$$\gamma(k) = \sum_{n=1}^{N}\gamma_n(k) \tag{3.30}$$

For re-estimation of the transition probabilities, we also need to define the probability of being on state q_k at time n and state q_ℓ at time $n+1$, given the X and the model parameters, i.e.,

$$\xi_n(k,\ell) = P(q_k^n, q_\ell^{n+1}|X,M,\Lambda) \tag{3.31}$$

Given the definitions of $\alpha_n(k)$ and $\beta_n(\ell)$, we have:

$$\begin{aligned}\xi_n(k,\ell) &= \frac{P(q_k^n, q_\ell^{n+1}, X|\Lambda)}{P(X|M,\Lambda)} \\ &= \frac{\alpha_n(k)p(q_\ell|q_k)p(x_{n+1}|q_\ell)\beta_{n+1}(q_\ell)}{\sum_{k=1}^{L}\sum_{\ell=1}^{L}\alpha_n(k)p(q_\ell|q_k)p(x_{n+1}|q_\ell)\beta_{n+1}(q_\ell)}\end{aligned} \tag{3.32}$$

If $\xi(k,\ell)$ denotes the probability of producing X while taking a transition from state q_k to state q_ℓ but without specifying at what time (i.e., without specifying for which acoustic vector x_n), we have:

$$\xi(k,\ell) = \sum_{n=1}^{N}\xi_n(k,\ell) \tag{3.33}$$

Generalizing the concept of counting occurrences to the expected number of occurrences (i.e., counting where each count is weighted by its probability of occurrence), re-estimation formulas for the transition and emission probabilities can be derived from (3.29), (3.30), (3.32) and (3.33); for continuous observation density HMMs, see [Liporace, 1982;

Juang et al., 1986]; for discrete HMMs, see [Rabiner, 1989; Lee, 1989]. The very elegant proof that they lead to a convergent process can be found in [Baum, 1972].

As an example of these re-estimation formulas, let us consider a HMM in which each state q_k is described in terms of a single Gaussian distribution of mean μ_k and covariance matrix Σ_k. The re-estimation formulas for the parameters of emission probability density function are then:

$$\hat{\mu_k} = \frac{\sum_{n=1}^{N} x_n \gamma_n(k)}{\gamma(k)} \tag{3.34}$$

and

$$\hat{\Sigma_k} = \frac{\sum_{n=1}^{N} \gamma_n(k)(x_n - \hat{\mu_k})(x_n - \hat{\mu_k})^T}{\gamma(k)} \tag{3.35}$$

In the case of Viterbi training (see next section) only the best path will be considered and probabilities $\gamma_n(k)$ degenerate into 1's and 0's (since every vector is assumed to be associated with only one state with probability 1). Given (3.30), $\gamma(k)$ represents the number of times an acoustic vector has been observed on q_k, and re-estimation formulas (3.34) and (3.35) simply degenerate into standard estimates of Gaussians pdfs.

For the transition probabilities, then, we have :

$$\hat{p}(q_\ell|q_k) = \frac{\xi(k,\ell)}{\gamma(k)} \tag{3.36}$$

3.4.2 Viterbi Criterion

When using the Viterbi criterion we are only interested in the best state sequence. In this case, the parameters of the models are optimized iteratively to find the best parameters *and the best state sequence* (i.e., the best segmentation in terms of the speech units used) maximizing

$$\overline{P} = \prod_{j=1}^{J} \overline{P}(X_j|M_j,\Lambda_j) \tag{3.37}$$

Each training iteration consists of two steps. In the first step, we use the old parameter values (or initial values) to determine the new best path matching the training sentences against the associated sequence of Markov models [by using (3.24)]. In the second step, we use this path to re-estimate the new parameter values (see below); backtracking of the optimal paths provides us with the number of observed transitions between states (to update the transition probabilities) and the acoustic vectors that have been observed on each state (to update the parameters describing the emission probabilities). This process can be proved to converge to a local minimum.

For clarity's sake, the re-estimation formulas (given the current matching path) are only provided for the discrete case in the following discussion. We also discuss the consequences of the different assumptions about the emission and transition probabilities on the generality and (local) discriminant properties of the models. In the continuous (Gaussian) case, re-estimation formulas directly follow from (3.34) and (3.35) (in which $\gamma_n(k)$ is simply equal to 1 or 0 according to whether or not x_n is associated with q_k for the best matching path), and similar conclusions remain valid.

We start from the most general form of the local contribution [left-hand side of (3.10)] in which we assume independence of the previous acoustic vectors X_1^{n-1}. When using the Viterbi criterion, each training vector x_n (replaced by its closest prototype vector y_i in the discrete case) is then uniquely associated with a single transition. Let $n_{ik\ell}$ denote the number of times each prototype vector y_i has been associated with a transition $\{q_k \rightarrow q_\ell\}$ between two states $\in \mathcal{Q}$ for the whole training set X. According to this definition, $n_{ik\ell}$ sums up all transitions wherever they appear in an HMM associated with X. The estimates of the probabilities $p(q_\ell, y_i|q_k, M)$ used in (3.10) (in which we assume independence of the previous acoustic vectors) and which guarantee the convergence of the Viterbi training are given by:

$$\hat{p}(q_\ell, y_i|q_k^-) = \frac{n_{ik\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jkm}}, \forall i \in [1, I], \forall k, \ell \in [1, K] \tag{3.38}$$

and thus:

$$\sum_{i=1}^{I}\sum_{\ell=1}^{K} \hat{p}(q_\ell, y_i|q_k^-) = 1, \forall k \in [1, K]$$

As done in (3.10), the local probability (3.38) is often split into a product of a transition probability and an emission probability (transition emitting models) [Jelinek, 1976] for which the respective estimators are:

$$\hat{p}(q_\ell|q_k) = \frac{\sum_{j=1}^{I} n_{jk\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jkm}} \tag{3.39}$$

and

$$\hat{p}(y_i|q_\ell, q_k) = \frac{n_{ik\ell}}{\sum_{j=1}^{I} n_{jk\ell}} \tag{3.40}$$

A further common simplification is to assume that the emission probabilities (3.40) only depend on the current state q_ℓ (state emitting models). Its estimator is then

$$\hat{p}(y_i|q_\ell) = \frac{\sum_{m=1}^{K} n_{im\ell}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jm\ell}} = \frac{n_{i\ell}}{n_\ell} \tag{3.41}$$

where $n_{i\ell}$ represents the number of times y_i has been observed on q_ℓ and n_ℓ the number of times state q_ℓ has occurred. The product of (3.39) by (3.41) is different from (3.38) due to the additional assumption on the emission probability.

If the models are trained using this formulation of the Viterbi algorithm, no discrimination is used. For instance, the local probability (3.38) is not the right measure to use if the goal is obtaining the most probable label associated with prototype vector y_i (or, in other words, to find the most probable associated state given a specified previous state). As noted in Chapter 2, this decision should ideally be based on the Bayes decision rule [Fukunaga, 1972] in which case the most probable state $q_{\ell_{opt}}$ should be defined according to:

$$\ell_{opt} = \underset{\ell}{\text{argmax}} \;\; p(q_\ell|y_i, q_k^-) \tag{3.42}$$

and not, from (3.38), by

$$\ell_{opt} = \underset{\ell}{\text{argmax}} \;\; p(q_\ell, y_i|q_k^-)$$

Thus, it is necessary that $\sum_{\ell=1}^{K} p(q_\ell|y_i, q_k^-) = 1$. As for the classical techniques summarized earlier, these probabilities are related to local contributions. Indeed, this discriminant local probability can be written

$$p(q_\ell|y_i, q_k^-) = \frac{p(q_\ell, y_i|q_k^-)}{p(y_i|q_k^-)} \tag{3.43}$$

Summing (3.38) on ℓ yields an estimator of $p(y_i|q_k^-)$

$$\hat{p}(y_i|q_k^-) = \frac{\sum_{m=1}^{K} n_{ikm}}{\sum_{j=1}^{I}\sum_{m=1}^{K} n_{jkm}} \tag{3.44}$$

By (3.38), (3.43) and (3.44), an estimator of the discriminant local probability is:

$$\hat{p}(q_\ell|y_i, q_k^-) = \frac{n_{ik\ell}}{\sum_{m=1}^{K} n_{ikm}} \tag{3.45}$$

and sums to unity as required. Such discriminant probabilities were already used in the ERIS system described in [Marcus, 1981, 1985]. It will be shown in Chapter 6 that the optimal output values of a particular neural network, the multilayer perceptron, are estimates of these discriminant local probabilities.

3.5 Decoding Problem

3.5.1 Maximum Likelihood Criterion

According to the likelihood criterion, an utterance X will be recognized as the word (or word sequence) associated with M_k if:

$$P(X|M_k) \geq P(X|M_j), \forall j \neq k \tag{3.46}$$

For continuous speech recognition, if a grammar is available to provide us with prior probabilities associated with every possible model sequence M_j, the recognition criterion will then be based on:

$$P(X|M_k)P(M_k) \geq P(X|M_j)P(M_j), \forall j \neq k \tag{3.47}$$

Statistical grammars (e.g., bigrams or N-grams) can be trained on large text corpora to estimate $P(M_k)$ for every possible word sequence [Jelinek, 1990].

For isolated word recognition, the full likelihood of the input sequence for each word model can be computed by using the forward recursion (3.14). However, for continuous speech recognition, the forward recursion would have to be computed for every possible word sequence, which in general is computationally intractable. A known solution to this problem is the *stack decoding* approach [Bahl et al., 1983] and is particularly well suited to tree grammars. This is a modification of the forward recursion for continuous speech which is derived from the A^* Search [Nilsson, 1980]. Although stack decoding is very attractive, there are some potential drawbacks [Lee, 1989]:

- Stack decoding is a tree search. For this case, only the top candidates (at different nodes of the tree, corresponding to different partial sentences and different utterance lengths) are stored in a stack and are extended (by using the forward recursion) and pruned. During the search, we need an evaluation function that estimates the full likelihood of the complete utterance from the known scores of partial utterances. We also need an estimate of the likelihood of the remaining path. This estimate must be lower than the actual likelihood of the remaining path, or the solution will be suboptimal. However, the more this is underestimated, the larger the stack must be.

- Since this algorithm extends paths of different lengths, it is no longer time-synchronous (as opposed to Viterbi decoding – see next section); during pruning it is more difficult to compare scores (that have to be time normalized).

- To improve the efficiency of the search, it is necessary to have a tree grammar. This approach is very difficult (and quickly becomes intractable) if the grammar is not tree-based.

Given all these implementation difficulties, a Viterbi search is often preferred for continuous speech recognition, even though it is suboptimal. However, some researchers (e.g., IBM [Bahl et al., 1983]) have successfully used a stack decoder for continuous speech recognition. More recently, there is a resurgence of interest in this algorithm since it also has several advantages compared with Viterbi decoding, e.g., decoding in terms of the actual likelihood word sequence, and an easy way to get the top N-best sentences [Paul and Necioglu, 1993].

3.5.2 Viterbi Criterion

According to the Viterbi criterion, an utterance X will be recognized as the word (or word sequence) associated with M_k if:

$$\overline{P}(X|M_k) \geq \overline{P}(X|M_j), \forall j \neq k \tag{3.48}$$

i.e., the Markov model for which the state sequence that has the highest probability of being taken while generating the observation sequence is maximum [computed by using recursion (3.24)].

Continuous speech recognition is achieved by searching for the best Markov model sequence leading to the maximum likelihood state sequence. This can also be formulated as the "one-stage" dynamic programming algorithm [Bridle et al., 1982; Ney, 1984]. This algorithm has several advantages:

- The Viterbi search is a time synchronous search algorithm, particularly well suited to real-time implementation.

- Because of the efficient dynamic programming procedure, the computational requirements of the Viterbi search are moderate and linearly proportional to the length of the utterance and the number of nodes present in the search space to describe the lexicon. It is also possible to prune the search space by using beam-search and fast look-ahead techniques [Lowerre, 1976; Schwartz et al., 1985; Ney et al., 1987].

- Simple syntax (such as word-pair or bigram grammars) can be incorporated quite easily. More complex models (such as augmented transition networks and context-free grammars) can also be used, but at the cost of much greater computational requirements.

- As a by-product, the recognition process also provides us with a segmentation of the sentences, which sometimes can be very useful (for example to get target outputs to train a neural network – see Chapter 7).

However, the probability obtained from this procedure is an approximation of the actual likelihood, and is thus suboptimal. This is because the Viterbi search finds the *optimal state sequence* instead of the *optimal word sequence*. A solution to this problem was proposed in [Schwartz et al., 1985] in which the full likelihood was computed within each word model while a Viterbi criterion was used for between-word transitions.

3.6 Likelihood and Discrimination

In this chapter we have used a MLE approach to the training of our HMMs. However, there are two fundamental conceptual problems with this approach (apart from all the hypotheses developed in this chapter):

- It is implicitly assumed that the model (with all its assumptions relative to its topology and probability density functions) is accurate and actually reflects the structure of the data (although the data might not adhere to the constraints imposed by the HMMs). If we had enough training data (which is not certain even in the case of MLE), it would probably be more preferable to infer all the parameters of the models (including topology and non-parametric probability density functions) directly from the data, which can be seen as implicitly using a Bayes or MAP criterion (i.e., $P(M|X)$) during training instead of MLE. Since MAP includes the effects of prior information, the language model (as defined in H1) would also be inferred from the training data. However, it appears that this would require a prohibitive amount of training data.

- By training with MLE instead of MAP, we strongly reduce the discriminant properties of HMMs. Ideally, each HMM should be trained not only to generate high probabilities for its own class, but also to discriminate against rival models.

These two points (but especially the second one) are related to the discussion that follows on discriminant criterion for HMM training.

As shown in Section 3.2, the actual criterion that should be maximized during training of HMMs is the posterior probability $P(M_i|X)$ that a Markov model M_i generates the acoustic vector sequence X. However, the parameter space on which this optimization is performed provides the difference between independently trained models and *discriminant* ones. Recalling Bayes' rule (from Chapter 2), the probability $P(M_i|X)$ can be written as

$$P(M_i|X) = \frac{P(X|M_i)P(M_i)}{P(X)} \tag{3.49}$$

In a recognition phase, $P(X)$ may be considered as a constant since the model parameters are fixed. However, during training, this probability depends on the parameters of all possible models. Indeed, denoting by $\mathcal{L}$ the parameter set for all the models, and taking into account that the models are mutually exclusive, $P(X)$ can be rewritten as:

$$P(X|\mathcal{L}) = \sum_k P(X|M_k, \mathcal{L})P(M_k|\mathcal{L}) \tag{3.50}$$

where the summation extends over all possible word or phoneme sequences. During training, the sum in (3.50) includes the correct model as well as all possible rival models.

As explained in Section 3.2, the prior probabilities $P(M_k)$ are usually estimated independently of the model parameters from a language model (hypothesis H1) and can thus be assumed independent of the parameters $\mathcal{L}$, i.e., $P(M_k|\mathcal{L}) = P(M_k)$. Furthermore, the likelihoods $P(X|M_i)$ depend only on the parameters Λ_i present in M_i. As a consequence, (3.49) may be written as:

$$P(M_i|X,\mathcal{L}) = \frac{P(X|M_i,\Lambda_i)P(M_i)}{P(X|M_i,\Lambda_i)P(M_i) + \sum_{k\neq i} P(X|M_k,\Lambda_k)P(M_k)} \tag{3.51}$$

Maximization of $P(M_i|X,\mathcal{L})$ is usually simplified by restricting it to the subspace of the M_i parameters. This restriction leads to the MLE criterion as used in this chapter. In this case, the summation term in the denominator is constant over the parameter space of M_i and maximization of $P(X|M_i,\Lambda_i)$ implies that of its bilinear transform $P(M_i|X,\mathcal{L})$ (3.51). This "model by model" optimization allows important simplifications in the training algorithms by avoiding the computations of all rival sequences, but at the cost of discrimination.

On the other hand, maximization of $P(M_i|X,\mathcal{L})$ with respect to the whole parameter space (i.e., the parameters of all possible models) leads to discriminant models since it implies that the contribution of $P(X|M_i,\Lambda_i)P(M_i)$ should be enhanced while the contribution of all possible rival models, represented by

$$\sum_{k\neq i} P(X|M_k,\Lambda_k)P(M_k),\ \forall k \neq i$$

should be reduced.[14] In this case, maximization of (3.51) is equivalent to maximization of

$$\frac{1}{1 + \frac{\sum_{k\neq i} P(X|M_k,\Lambda_k)P(M_k)}{P(X|M_i,\Lambda_i)P(M_i)}}$$

[14] In the case of isolated word recognition, this is quite easy since it is enough to consider all the possible word models of the lexicon. For continuous speech recognition, the sum should include the probability of all possible HMM sentence models (excluding M_i) that are allowed by the syntax.

or maximization of

$$\frac{P(X|M_i, \Lambda_i)P(M_i)}{\sum_{k \neq i} P(X|M_k, \Lambda_k)P(M_k)} \tag{3.52}$$

If we assume equal priors $P(M_k)$ for all possible k, this maximization with respect to the whole parameter space is then equivalent to *Maximum Mutual Information* (MMI) criterion [Bahl et al., 1986; Brown, 1987; Merialdo, 1988] which aims at maximizing

$$\frac{P(X|M_i, \Lambda_i)}{\sum_{k \neq i} P(X|M_k, \Lambda_k)} \tag{3.53}$$

which is also a discriminant criterion.

These considerations still hold in the case of embedded training where discrimination is increased between the various word or sentence models M_k by adjusting the parameters of the constituting sub-unit models which may appear several times and in several different M_k.

As shown in this chapter, the Forward-Backward algorithm simply maximizes likelihoods and, consequently, does not provide discriminant models. However, this is a very efficient training procedure which iteratively provides parameter estimates based on partial path probabilities computed by forward and backward recursions and which is guaranteed to converge (at least to local optimum). This advantage is unfortunately lost if the MMI criterion is used [Bahl et al., 1986] and a gradient method is generally preferred for the optimization; the forward and backward recursions can still be used to compute the gradient [Brown, 1987] but additional constraints are required to guarantee that probabilities are positive and between 0 and 1. If phoneme models are trained, a *looped phonetic model,* i.e., a word model that allows any possible phoneme sequence [Wellekens, 1986; Merialdo, 1988], may generate all possible phoneme sequences and, by running the forward recursion for full likelihood estimation through it, may provide the summed probability in the denominator of (3.53). The numerator of (3.53) is obtained in a separate step via a serial model. This method could, in principle, also apply to word model training, but would require an excessive computation time for a large lexicon due to the size of the looped word model. The MMI criterion is thus particularly unsuitable for embedded training of word models; severe hypotheses must be accepted to cope with the complexity [Brown, 1987].

The Viterbi algorithm, which is the main tool for the recognition task, is also often used in the training phase. In this case, the parameters are updated so as to increase the probability of the most probable state sequence and $P(X|M)$ is thus not actually maximized. It is somewhat simpler than the Forward-Backward algorithm and convergence (at least to a local optima) is also guaranteed. It is well adapted to a simplified form of MLE optimization (considering the best path only) but not at all to MMI maximization since it requires taking account of *all* paths (not only the best one) in *all* possible rival sequence models.

To circumvent the lack of discrimination, other methods than MMI have been proposed. In a two-pass method for isolated word recognition [Rabiner & Wilpon, 1981], the local distances between acoustic vectors of a test word and of reference templates are weighted to enhance the discriminating parts of the matching path. Another two-stage method to improve the discrimination in isolated word recognition has been proposed by [Martin et al., 1987]. In this case, information on the rival models (including duration information) are collected during the training phase and used in the second stage to get rid of ambiguities. The pragmatic use of an iterative *corrective training* has been proposed in [Bahl et al., 1988] and has yielded improved models. However, for all these approaches, there is no clear theoretical justification, and it is usually difficult to guarantee the convergence of the related training procedures.

In this book, it is shown how neural networks can be used to improve the discriminant properties of HMMs and to overcome some of their limiting hypotheses.

3.7 Summary

The unpredictable and sequential nature of the human speech production system makes automatic speech recognition difficult. Hidden Markov Models (HMMs) provide a good representation of these characteristics and are now the dominant technology for continuous speech recognition. They benefit from powerful training and decoding algorithms. However, the assumptions that make optimization of these models possibly limit their generality; these are, among others:

- Poor discrimination due to the training algorithm, which maximizes likelihoods instead of posterior probabilities (i.e., the models are trained independently of each other).

- *A priori* choice of model topology and statistical distributions, e.g., assuming that the probability density functions associated with the HMM state can be described as multivariate Gaussian densities or as mixtures of multivariate Gaussian densities, each with a diagonal-only covariance matrix.

- Assumption that the state sequences are first-order Markov chains.

- Typically, no acoustical context is used, so that possible correlations between successive acoustic vectors is overlooked.

In fact, the HMM we end up with after these assumptions is very crude compared with the ideal model, in which emission and transition probabilities should be dependent on a fixed window back into the recent past on both states and observations.

It should also be noted here that there is a fundamental weakness in the HMM representation of speech production, even if none of these assumptions are too harmful: we must assume that speech is well represented by a succession of steady-state (locally stationary) segments. In fact it is more likely that speech should be represented as a sequence of significant changes. This issue is not addressed in this book, but we mention it here for perspective.

In later chapters we will show how neural network technology can be integrated into HMM systems to alleviate some of their more restrictive limitations. But first, we proceed to a short background on a useful form of neural network, the Multilayer Perceptron.

Figure 3.2: Who is this Markov fellow, and why is he hiding?

Chapter 4

MULTILAYER PERCEPTRONS

Physical models are as different from the world as a geographical map is from the surface of the earth
– L. Brillouin –

4.1 Introduction

In this section, we will describe the perceptron and *Multilayer Perceptron* (MLP) classes of Artificial Neural Networks. MLPs can be used for tasks such as feature extraction (see Chapter 14) and prediction (see Section 6.8 and Chapter 13) with applications ranging from signal processing to stock market forecast. For reviews and further reading on the fundamentals of neural networks, see [Rumelhart, Hinton, & Williams, 1986b; Pao, 1989; Beale & Jackson, 1990; Hertz, Krogh, & Palmer, 1991; Zurada, 1992]. For more information on learning algorithms, performance evaluation, and applications, see [Karayiannis & Venetsanopoulos, 1993]. For more references and application areas, see [Simpson, 1991].

Since this book is mainly concerned with speech recognition,[1] the next sections will consider MLPs as pattern classifiers. More specifically,

[1]You can look for a hidden message if you want, but as far as we can tell this is pretty much all that it's about.

we will consider the problem of classifying acoustic vectors into (phonemic) classes. As in Chapter 2, these classes will be denoted ω_k, with $k = 1, \ldots, K$. Ultimately, these classes will be associated with HMM states q_k. The goal of this chapter is to provide some basic theoretical foundation for the use of MLPs that will be described in the later chapters.

4.2 Linear Perceptrons

4.2.1 Linear Discriminant Function

We assume here that each class ω_k, $\forall k = 1, \ldots, K$ (that will ultimately be associated with a HMM state) can be associated with a *linear discriminant function* [Nilsson, 1965] defined as:

$$g_k(x_n) = \overline{w}_k^T \overline{x}_n \tag{4.1}$$

where

$$\overline{x}_n = (1, x_{n1}, x_{n2}, \ldots, x_{nd})^T$$

is the acoustic vector x_n at time n augmented by an additional element with a value of 1, T the transpose operation, and

$$\overline{w}_k = (w_{k0}, w_{k1}, w_{k2}, \ldots, w_{kd})^T$$

represents the parameter set describing ω_k, consisting of the augmented form of the weight vector w_k (including 0-th component of $\overline{x}_n$). Parameter w_{k0} is usually referred to as the *bias* of class ω_k. Collecting equations (4.1) for all classes leads to the matrix notation:

$$g(x_n) = \overline{W}^T \overline{x}_n$$

where $g(x_n) = (g_1(x_n), \ldots, g_K(x_n))^T$, $\overline{W} = (\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_K)$ is a $(d+1) \times K$ weight matrix, and K is the number of classes.

Classification is then based on the rule:

$$\text{if } g_k(x_n) \geq g_j(x_n), \ \forall j \neq k \text{ then } x_n \in \omega_k \tag{4.2}$$

For a 2-class problem, an equivalent formulation of (4.2) is

$$\text{if } h_{12}(x_n) = g_1(x_n) - g_2(x_n) \geq 0 \text{ then } x_n \in \omega_1 \text{ else } x_n \in \omega_2$$

This rule is often replaced by a simple binary decision via a *threshold logic unit*. Such a system is then referred to as a *perceptron*. There is a simple training algorithm (known as the perceptron training algorithm) that is guaranteed to converge if the data set is linearly separable [Nilsson, 1965; Minsky & Papert, 1969, 1988; Rosenblatt, 1958, 1962].

After training, perceptrons cut the feature space $\mathcal{R}^d$ into regions corresponding to different classifications. The resulting decision boundaries are composed of segments of hyperplanes defined by: $g_k(x_n) - g_j(x_n) = 0$. Augmenting the feature vector allows discriminant surfaces that do not necessarily contain the origin of the parameter space; this is the main goal of the biases w_{k0} that are used in connectionist systems.

The entries of $\overline{W}$ can be determined by training on a preclassified vector set according to the reciprocal of the decision rule (4.2)

$$\text{if } x_n \in \omega_k \text{ then } g_k(x_n) \geq g_j(x_n), \ \forall j \neq k \tag{4.3}$$

Early work by Rosenblatt and his students [Rosenblatt, 1962] showed some of the versatility of the perceptron. Unfortunately, simple perceptrons cannot solve classification problems for data sets that are not linearly separable, in which case the perceptron training algorithm simply does not converge [Minsky & Papert, 1969].[2]

4.2.2 Least Mean Square Criterion

As noted above, a perceptron can only partition data sets linearly (i.e., with a hyperplane) and the perceptron training rule will not converge if the data set is not linearly separable. However, even for this simple architecture, an approach to learning can be used that will at least converge to a reasonable solution for data sets that are not linearly separable. In this approach, a *Least Mean Square* (LMS) criterion can be used for the determination of discriminant function parameters w_{kj}. In this case, however, it is no longer guaranteed that requirements (4.2) will be satisfied for all individual vectors of the training set even for linearly separable sets. Indeed we do not minimize the actual classification error rate (via the decision logic that simply counts the number of errors)

[2]It should be noted that some of Rosenblatt's students did in fact develop multilayer versions that were quite successful at generating more complex partitions. However, in this case the algorithms were not guaranteed to converge, although they were used successfully for a number of difficult problems in pattern recognition (see [Viglione, 1970]).

anymore but just a standard *Mean Square Error* (MSE) which, as we will see later on, approximates a Bayes decision (see pages 193-195 and Figure 12.3 in [Minsky & Papert, 1988] for more discussions about this). In [Minsky & Papert, 1988] it is shown that in different situations either the perceptron or Bayes approach may be superior.

The parameter values w_{kj} may be calculated from a preclassified training vector set, minimizing a cost function E expressing the sum over all training vectors and all classes of the squared errors between $g_k(x_n)$ and the desired outputs, typically equal to 1 if $x_n \in \omega_k$ and 0 if $x_n \notin \omega_k$. This cost function, usually referred to as *Mean Square Error* (MSE), is explicitly written as:

$$E = \sum_{k=1}^{K} \sum_{x_n \in \omega_k} \| g(x_n) - \Delta_k \|^2 \tag{4.4}$$

where Δ_k, which we will call the index vector, is a K-vector with all zero components except the k-th one. For each training vector, correct classification is reinforced while incorrect classification is punished. However, the use of an LMS criterion leads to discriminant functions that have real outputs approximating the values 1 or 0, as opposed to making a simple logical decision between these two alternatives. Minimization of E with respect to the parameters w_{kj} leads to the following equation (see, for example, [Devijver & Kittler, 1982]):

$$\overline{X}\overline{X}^T\overline{W} = \overline{M}D \tag{4.5}$$

where $\overline{M}$ is a $(d+1) \times K$ matrix where the k-th column is the augmented mean vector $\overline{\mu}_k = (1, \mu_{k1}, \ldots, \mu_{kd})^T$ of all the vectors associated with class ω_k, D is a diagonal $K \times K$ matrix with $(n_1, \ldots, n_K)$ on its diagonal, with n_k representing the number of vectors classified into ω_k; $\overline{X} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_\eta)$ is a $(d+1) \times N$ matrix whose columns contain the whole set of augmented vectors $\overline{x}_n$ in the training set.

4.2.3 Normal Density

It can easily be shown that, during classification, using linear discriminant functions is equivalent to estimating Bayes Gaussian densities in which it is assumed that all densities are sharing the same covariance matrix Σ.

As noted in Chapter 2, decisions made using estimates of the Bayes probability $p(\omega_k|x_n)$, $\forall k = 1, \ldots, K$, are equivalent to decisions made using estimates of $p(x_n|\omega_k)p(\omega_k)$, since $p(x_n)$ is independent of the class during classification (this is not true during training – see discussion in Section 3.6). These decisions can also be made on the basis of the log probabilities, since the log is a monotonic function. In the case of Gaussian densities of mean μ_k and covariance matrix Σ, The Bayes decision can be made on the basis of the function:

$$g_k(x_n) = -\frac{1}{2}(x_n - \mu_k)^T \Sigma^{-1}(x_n - \mu_k) + \log\, p(\omega_k) \qquad (4.6)$$

Since the quadratic term $x_n^T \Sigma^{-1} x_n$ is independent of the class, we can use a slightly different discriminant function, dropping this term:

$$g_k(x_n) \;=\; w_k^T x_n + w_{k0} \qquad (4.7)$$

with:

$$\begin{aligned} w_k &= \Sigma^{-1}\mu_k \\ w_{k0} &= -\frac{1}{2}\mu_k^T \Sigma^{-1}\mu_k + \log\, p(\omega_k) \end{aligned} \qquad (4.8)$$

Thus, in this special case, the optimal discriminant function is linear. If the covariance matrix Σ in equation (4.8) is made class-dependent (the more general case), the quadratic terms are still relevant for classification; but since there are no higher-order terms, Gaussian densities can always be optimally classified by quadratic discriminant functions [Nilsson, 1965].

4.3 Multilayer Perceptrons (MLP)

4.3.1 Some History

One of the technical developments sparking the recent resurgence of interest in neural networks has been the rediscovery and popularization of multi-layer networks of perceptrons, particularly after the publication of a lucid book on the subject [Rumelhart, Hinton, & Williams, 1986b]. However, as noted earlier, systems based on multilayer perceptrons were widely used for problems in biomedical pattern classification and image processing in the 1960's [Viglione, 1970]. These systems were designed to overcome the well-known limitations of single-layer perceptrons, and already used approaches such as constructive and destructive topology learning, selective or weighted sampling of the input, and modular construction.[3] In particular, they built up a layer of units that were viewed as feature extractors, and whose outputs were further processed by a final layer of units. In current terminology, the feature extractor layer would be called a *hidden* layer, since it is neither an input nor an output layer; its outputs are not accessible outside of the network.

These systems used hard decisions in the hidden layer. The use of softer decision functions, such as the sigmoid, is the major structural innovation of the more modern perceptron-related networks[4] and resulted in the derivation of simple and powerful learning rules that (in principle) apply over any number of perceptron layers.

It is these more recent architectures that we will review here, as we have found them useful for the speech recognition systems described in this book.

4.3.2 Motivations

Even for linear perceptrons, multiclass discriminations require a kind of added layer. The class decision can be based on (4.2) as follows: $K \times (K-1)/2$ discriminant functions $h_{jk}(x_n) = g_j(x_n) - g_k(x_n)$, $\forall j, k,\ k > j$, are defined and computed; their thresholded values are then entered into a very simple logical net providing the suitable index vector Δ at the output. These thresholded elementary units are perceptrons so that logical decisions occur. It is easily shown that the logical net can also be realized as a second layer of perceptrons. Piecewise linear discriminant networks have a similar structure but the h_{jk} are independent [Fukunaga, 1972].

However, in this case (as opposed to the case of multilayer perceptrons described by Viglione), the perceptrons inherit the limitations of the linear discriminant functions (separating the K classes of patterns by hyperplanes), and these training algorithms unfortunately fail to converge for sets of vectors that are not linearly separable.

[3] It may not be true that there is nothing new under the sun, but there sure are a lot of things that have been lying there getting a tan for 30 years.

[4] Although sigmoids were suggested by a number of researchers at an early date [Cowan, 1967], they were not widely used for pattern recognition problems until the 1980's.

Convergence problems can be avoided by using a LMS criterion as (4.4) but at the cost of losing the power of the nonlinear logical decision (i.e., minimization of the error rate). A compromise approach is to approximate the threshold logic unit by a smooth differentiable function F, which is generally chosen as being the *sigmoid function*

$$F(x) = \frac{1}{1+e^{-x}}$$

More precisely, a new LMS criterion could be the minimization of

$$E = \sum_{k=1}^{K} \sum_{x_n \in \omega_k} \parallel F[g(x_n)] - \Delta_k \parallel^2 \tag{4.9}$$

where the modified discriminant vector function $F[g(x_n)]$ is component wise related to $g(x)$ by

$$F_i[g(x_n)] = \frac{1}{1+\exp(-g_i(x_n))} \tag{4.10}$$

However, even when criterion (4.9) is restricted to linear discriminant functions (and linear decision surfaces), we must minimize a nonlinear function. This requires the use of an iterative gradient-like method, which is just a simplified version of the error back-propagation algorithm used to train multilayer perceptrons (see below). This solves the problem of non-convergence of the perceptron algorithm for non-linearly separable sets. For linearly separable sets, one can still expect that minimization of this MSE will converge to the perceptron solution, since there are no local minima in this case.[5]

Nonlinear mappings are often required to achieve logical decisions, and single-layer perceptrons are not sufficient to discriminate vector sets that are not linearly separable. However, there is no utility in building multilayer linear discriminant nets by adding layers of *linear hidden units* since such a network is always equivalent to a single linear system (within a possible drop of rank).[6] However, it is shown in [Minsky & Papert,

[5]See [Sontag and Sussman, 1989a] for a proof of this. These authors also show in [Sontag and Sussman, 1989b] that without the separability assumption, there are counterexamples to this assertion.

[6]This is a critical point. Once a network has been trained, several layers of linear transformations are equivalent to a single linear transformation. During training, however, the transformations are constrained to be equivalent to multiplication by matrices that are limited in rank by the layer size.

1969] that, if a nonlinear operator is added at the output of each unit of intermediate processing layers (referred to as *hidden layers*), it is in principle possible to perform any nonlinear mapping from the input to the output – provided the set of hidden units is large enough.[7] This kind of network is known as *Multilayer Perceptron* (MLP). Such MLPs have several important properties:

- They are discriminant when they are trained with discriminant criteria (like LMS).
- They can generate nonlinear decision boundaries.
- Given the sigmoid function at the output, they represent a good compromise between simultaneous minimization of the MSE and the actual classification error rate. Along this line, the role of the sigmoid functions on the hidden layers and on the output layer is different. The nonlinear function on the hidden layer generates higher order moments of the input patterns that are subsequently used by later layers. This could be accomplished by many other functions than the sigmoid, including ones that are not at all an approximation to a threshold logic unit (see, for example, radial basis functions in Section 6.7). The nonlinear function at the output, on the other hand, must be an approximation of the decision logic and, consequently, is restricted to a sigmoid-like function[8] (see Section 4.4.1 for further discussion).

4.3.3 Architecture and Training Procedure

An η-layered perceptron consists of $(\eta + 1)$ layers $L_\ell\,(\ell = 0, \ldots, \eta)$ of several units, where L_0 corresponds to the input layer, L_η to the output layer and $L_\ell\,(\ell = 1, \ldots, \eta - 1)$ to the hidden layers. Hidden and output units are computational units and their output values are determined by first summing all of their inputs and then passing the results through the sigmoid function (4.10). The output values of layer L_ℓ form a n_ℓ-vector $h_\ell(x_n)$ which is a function (4.11) of the input vector x_n; n_ℓ is the number of units in L_ℓ. Input vector $h_0(x_n)$ and output vectors $h_\eta(x_n)$ are also

[7] Of course, for some functions the required layer length might be Avogodro's number.

[8] Or equivalently a *softmax* function, which is described in Section 6.4.2. Statistically this function acts much like the sigmoid.

denoted x_n and $g(x_n)$ in the following. Vector $\overline{h}_\ell(x_n)$ ($\ell = 0, \ldots, \eta - 1$) stands for the $(n_\ell + 1)$-augmented vector, where the 0-th unit will be fixed to 1 and will account for the biases of the following layer. As the biasing unit is irrelevant for the output layer, we have $\overline{h}_\eta = h_\eta$. Layer $L_{\ell-1}$ is fully connected to layer L_ℓ by a $(n_{\ell-1} + 1) \times n_\ell$ weight matrix $\overline{W}_\ell$. Matrix W_ℓ denotes $\overline{W}_\ell$ deprived of its first row (corresponding to the biases). The state propagation is thus described by:

$$h_\ell(x_n) = F(\overline{W}_\ell^T \overline{h}_{\ell-1}(x_n)), \forall \ell = 1, \ldots, \eta \qquad (4.11)$$

where F is a nonlinear function, typically a sigmoid function (4.10) that operates componentwise. Finally, we may write symbolically:

$$h_\eta(x_n) = g(x_n) \qquad (4.12)$$

where g is now a nonlinear function of x_n depending on the parameters $\overline{W}_\ell$, $\forall \ell \in [1, \ldots, \eta]$.

The model parameters (the weight matrices $\overline{W}_\ell$) are obtained from a set of training input and associated (or desired) output pairs by minimizing, in the parameter space, the error criterion defined as:

$$E = \sum_{n=1}^{\eta} \parallel g(x_n) - d(x_n) \parallel^2 \qquad (4.13)$$

where, given each training input vector x_n, $g(x_n)$ represents the output vector $h_\eta(x_n)$ generated by the system. If there is at least one hidden layer, given (4.11) and (4.12), $g(x_n)$ is a nonlinear function of the input x_n (defining nonlinear decision surfaces)[9] and contains the sigmoid function (4.10); $d(x_n)$ is the desired output associated with x_n. The total number of training patterns is denoted by N.

As explained in [Rumelhart et al., 1986a; Sejnowski & Rosenberg, 1986], the weight matrices are iteratively updated via a gradient correction procedure to reduce the error (4.13). By simply using the chain rule for differentiation, it has been shown in [Rumelhart et al., 1986a; Sejnowski & Rosenberg, 1986] that the gradient of the error criterion versus every weight in the network can be calculated by recursively back-propagating the error at the output layer. The corresponding training

[9] It has been shown in [White, 1988] that an MLP with only one hidden layer, but containing "enough" hidden units, was enough to generate any kind of nonlinear function. This is also true for multi-Gaussian classifiers and radial basis functions (see Section 6.7).

procedure will not be described here since it has been widely presented in the literature. It can, however, be summarized as follows. For each training iteration t ($t = 1, \ldots, T$):

1. Presentation of all the training input vectors $x_n, n = 1, \ldots, N$, forward computation of the output vectors $g(x_n)$ [using (4.11)], and calculation of the error function E.

2. Backward propagation of the error (using the chain rule) to compute the partial derivative of the error criterion with respect to every weight, and update of the weights according to:

$$w_{kj}(t+1) \;=\; w_{kj}(t) - \alpha \frac{\partial E}{\partial w_{kj}(t)} \tag{4.14}$$

   in which α is usually referred to as the *step size parameter* or *learning rate* and has to be small enough to guarantee the convergence of the process (see Section 4.3.5 for further discussions about this).

This procedure, known as the *error back-propagation* (EBP) algorithm, is iterated and stopped, e.g., when the absolute value of the relative correction on the parameters falls under a given threshold.[10] This EBP training algorithm is now the most commonly used algorithm for training MLPs [Parker, 1982; Parker, 1985; Rumelhart et al., 1986a; Werbos, 1974]. However, this algorithm was already used in [Amari, 1967; Rosenblatt, 1960; Widrow & Hoff, 1960] for a single-layer network, in [Bryson & Ho, 1969] in control theory, and certainly credit for the chain rule must go to Newton [Newton, 1687].[11]

If the input units are directly connected to the output units and if $d(x_n)$ is an index vector, criterion (4.13) becomes equivalent to (4.9), which can be solved by the same procedure.

4.3.4 Lagrange Multipliers

In the following, it is shown that the EBP algorithm can also be derived by interpreting the problem as a constrained minimization problem. This

[10] It can be argued that the so-called *stopping criterion* should be based on the performance or error in an independent data set and not on the training patterns. See Chapter 6 for a procedure to do this, and Chapter 12 for an extensive discussion of this topic.

[11] Disputed, however, by Leibniz, circa 1675.

approach, initially proposed in [le Cun, 1988], regards MLP training as a constrained minimization problem involving the network state variables $\overline{h}_\ell(x_n)$ and the weights. The problem may be specified by a single Lagrange function L containing the objective function E and constraint terms, each multiplied by a Lagrange multiplier Λ_ℓ. In this particular case, the constraint terms describe the network architecture, i.e., the forward equations of the network. However, this formulation will permit easier generalization to other kinds of constraints (see Section 6.7.4). It also can clarify parallels between MLPs and other connectionist models.

For each training pattern x_n, we want $h_\eta(x_n) = d(x_n)$ under the η constraints represented by equations (4.11). By introducing η vectorial Lagrange multipliers Λ_ℓ, $\ell = 1, \ldots, \eta$, the problem is transformed to minimizing a modified error function L versus the parameters $\overline{W}_\ell$, $h_\ell(x_n)$ and Λ_ℓ, for $\ell = 1, \ldots, \eta$, with:

$$\begin{aligned} L &= (d(x_n) - h_\eta(x_n))^T (d(x_n) - h_\eta(x_n)) \\ &\quad + \sum_{\ell=1}^{\eta} \Lambda_\ell \left[h_\ell(x_n) - F(\overline{W}_\ell^T \overline{h}_{\ell-1}(x_n)) \right] \end{aligned} \tag{4.15}$$

A constraint is met when the corresponding term in L is zero. It may be shown that:

$$\nabla L(\Lambda_\ell, h_\ell(x_n), \overline{W}_\ell) = 0, \forall \ell = 1, \ldots, \eta \tag{4.16}$$

corresponds to a minimum of E while meeting the constraints. We may split condition (4.16) into its constituent partials:

Condition 1:

$$\frac{\partial L}{\partial \Lambda_\ell} = 0, \forall \ell = 1, \ldots, \eta \tag{4.17}$$

where the derivative is applied to each component of its argument. This leads to:

$$h_\ell(x_n) = F[\overline{W}_\ell^T \overline{h}_{\ell-1}(x_n)] \tag{4.18}$$

which comprises the forward recurrences (4.11) of the EBP algorithm.

Condition 2:

$$\frac{\partial L}{\partial \overline{h}_\ell(x_n)} = 0, \forall \ell = 1, \ldots, \eta \tag{4.19}$$

Setting to zero the derivative with respect to h_η leads to:

$$\Lambda_\eta = 2(d(x_n) - h_\eta(x_n))$$

Differentiation with respect to the other h_ℓ's yields

$$\Lambda_\ell = \overline{W}_{\ell+1} F'[\overline{W}_\ell^T \overline{h}_\ell(x_n)]\Lambda_{\ell+1} \ , \quad \forall \ell = 1, \ldots, \eta - 1$$

By defining

$$b_\ell(x_n) = F'[\overline{W}_\ell^T \overline{h}_{\ell-1}(x_n)]\Lambda_\ell, \ \forall \ell = 1, \ldots, \eta - 1$$

we finally obtain:

$$b_\eta(x_n) = 2F'[\overline{W}_\eta^T \overline{h}_{\eta-1}(x_n)](d(x_n) - h_\eta(x_n)) \tag{4.20}$$

and the backward recurrence

$$b_\ell = F'[\overline{W}_\ell^T \overline{h}_{\ell-1}]\overline{W}_{\ell+1} b_{\ell+1}, \ \forall \ell = 1, \ldots, \eta - 1 \tag{4.21}$$

In comparison with the EBP algorithm described in [Rumelhart et al., 1986a; Sejnowski & Rosenberg, 1986], equations (4.20) and (4.21) correspond to the computation rule of the gradients where the back-propagated variables b_ℓ are the Lagrange multipliers within a simple scaling matrix.

Condition 3:

$$\frac{\partial L}{\partial \overline{W}_\ell} = 0, \ \forall \ell = 1, \ldots, \eta \tag{4.22}$$

This leads to:

$$F'[\overline{W}_\ell^T \overline{h}_{\ell-1}(x_n)]\Lambda_\ell \overline{h}_{\ell-1}^T(x_n) = 0 \tag{4.23}$$

or

$$b_\ell(x_n)\overline{h}_{\ell-1}^T(x_n) = 0 \ , \quad \forall \ell = 1, \ldots, \eta \tag{4.24}$$

The weight matrices $\overline{W}_\ell$ satisfying these equations can be obtained by an iterative gradient procedure making weight changes according to $\alpha \frac{\partial L}{\partial \overline{W}_\ell}$. The parameters at training step $t+1$ are then calculated from their value at iteration t by:

$$\overline{W}_\ell^T(t+1) = \overline{W}_\ell^T(t) + \alpha b_\ell(x_n)\overline{h}_{\ell-1}^T(x_n) \tag{4.25}$$

which is the standard weight update formula of the EBP algorithm.[12]

These three conditions, when met, give a complete specification of the back-propagation training of the network: optimizing with respect to

[12]See Section 4.3.6 about on-line and off-line training.

the Lagrange multipliers gives the forward propagation equations; optimization with respect to the state variables gives the backward equations (the gradients); and optimization with respect to the weights gives the weight update equations.

This approach is more flexible for possible generalizations. For example, to keep the weights small, the term $\mu \sum_{\ell=1}^{\eta} \| \overline{W}_\ell \|^2$ can be added to the function L (4.15) that is minimized. For the weight update (condition 3), (4.25) then becomes:

$$\overline{W}_\ell^T(t+1) = (1-2\alpha\mu)\overline{W}_\ell^T(t) + \alpha b_\ell(x_n)\overline{h}_{\ell-1}^T(x_n) \tag{4.26}$$

so that the extra condition will be satisfied for a sufficiently small value of μ. This approach will be used in Section 6.7.4, where additional constraints will be introduced in the EBP algorithm.

4.3.5 Speeding up EBP

The EBP algorithm is a simple gradient-based optimization procedure and, consequently, can suffer from the limitations typical of this kind of algorithm, i.e., slow convergence and difficulty in the choice of the learning rate α. The techniques usually used to improve the convergence of gradient searches in general can then can be applied directly to EBP. Among the simplest of these are second order methods (variants of Newton's method) [Becker & le Cun, 1988; Parker, 1987; Watrous, 1987] that use the information contained in the second derivative (Hessian) matrix to speed up convergence. However, the major drawback to using these methods is that, although they actually reduce the required number of training iterations, they are more computationally expensive since they require computing the inverse of the Hessian at each iteration. In this case, if we have K parameters, we have to invert a $(K \times K)$ Hessian matrix at each iteration. As a consequence, most commonly (particularly for a large number of weights) only the diagonal terms of the Hessian matrix are usually used, which is equivalent to performing Newton's rule separately for each weight. Other approaches using adaptive learning rates (which can be equal or different for each MLP parameter) can also be used [Silva & Almeida, 1990; Kesten, 1957; Jacobs, 1988]. In [Watrous, 1987], line search routines were used to determine the optimal change during each training iteration. Another approach is to increment the gradient by a fraction of the previous weight change, referred to as momentum term. This term also has its own step size β in the EBP update formula. In this case, (4.14) becomes:

$$w_{kj}(t+1) \;=\; w_{kj}(t) - \alpha \frac{\partial E}{\partial w_{kj}(t)} + \beta \Delta w_{kj}(t) \qquad (4.27)$$

This tends to smooth the weight changes according to the "average" gradient, which will allow an increased learning rate without incurring divergent oscillations. A related gradient procedure known as *conjugate gradient* can also result in a faster convergence (at least in the case of quadratic error surfaces). In this case, the new search direction is defined from the new gradient and the previous search direction and attempts to retain the gains from the previous minimization. As a particular case of this approach, the Polak-Ribière rule can be used to compute the optimal proportion of the previous search direction that should be added to the new gradient. For quadratic error surfaces in a parameter space of dimension n, it can be proved that this method will reach the minimum within n iterations. This conjugate gradient approach has been used quite successfully to speed up EBP[13] in [Kramer and Sangiovanni-Vincentelli, 1989; Makram-Ebeid et al., 1989].

Other methods and/or heuristics have been used to reduce the training time of EBP. For example, in [Lehman et al., 1988], it was shown that adding noise on the training patterns could also decrease the training time, while sometimes helping to escape from poor local minima. Finally, alternative cost functions like the entropy or relative entropy measure can also have better training properties [Solla et al., 1988].

In our experience, some simple choices (relative entropy error criterion rather than LMS, random pattern presentation, pre-setting of biases to expected values based on probabilistic interpretation or on previous experience, and use of a declining learning rate based on cross-validation – see Chapter 12) lead to very fast convergence using simple on-line EBP with the problems we have been studying. For instance, with a popular speech database (the Resource Management speaker-independent training set – see Section 7.7), we currently (1992) use about 5 iterations. This is, of course, a relatively large and varied training set (over a million patterns from over a hundred different speakers), and this result may not apply to problems with less varied training data; however, future speech data sets that we will use will be even larger.

[13] For the off-line, full gradient case; the next section will discuss the on-line approach, which is also much faster for realistic data sets than the off-line method.

4.3.6 On-Line and Off-Line Training

During training, the update of the MLP parameters can be done in two different ways:

- Off-line training: In this case, we accumulate the weight updates over all the training patterns and we modify the weights only when all the training patterns have been presented to the network. The actual gradient of (4.13) is then estimated for the complete set of training patterns, which guarantees, under the usual conditions of standard gradient procedures, the convergence of the algorithm.

- On-line training: In this case, the MLP parameters are updated after each training pattern according to the "local" gradient. However, while this does not actually minimize (4.13) directly, it can be shown [Widrow & Stearns, 1985] that this process will stochastically converge to the same solution.[14] In practice, the on-line training exhibits several advantages compared with the off-line procedure: it is generally acknowledged that it converges much faster and that it can more easily avoid local minima. This can be explained by the fact that the use of "local" gradients introduces noise in the training process, which usually improves the behavior of gradient searches because of the lowering of the risk of getting stuck in a suboptimal local minimum; in some sense a large step-size corresponds to a high "temperature" in simulated annealing. Additionally, for large and varied training sets (such as the Resource Management training sentences mentioned previously), on-line training implies multiple passes through similar data for each single pass through the whole set.

As noted above, on-line training frequently shows a practical advantage. The enhancements discussed previously appear to improve off-line performance significantly for at least some examples. However, the on-line techniques, which converge so quickly in our experience, have the additional advantage of simplicity. Consequently, all the experiments that will be reported in this book have been done using on-line MLP training.

[14] Since the local gradient can be viewed as a random variable whose mean is the true gradient, such an approach is sometimes called a "stochastic gradient" procedure.

4.4 Nonlinear Discrimination

4.4.1 Nonlinear Functions in MLPs

In this section, the "generalization" properties of the MLP and some discussions regarding the use of nonlinear functions on the hidden and output units will be illustrated by two particularly simple but illustrative examples.

The basic architecture of the network which will be used in the following examples is a simple two-layered perceptron[15] containing d input units representing the d-dimensional input vector x_n, one layer of n_h hidden units and one layer of n_o output units; $\overline{W}_1$ denotes the augmented $(d+1) \times n_h$ weight matrix between the input layer and the hidden layer, $\overline{W}_2$ denotes the augmented $(n_h+1) \times n_o$ weight matrix between the hidden layer and the output layer. Generally, n_o is equal to K, the number of classes. For an input vector x_n, the output values of the hidden layer units form a n_h-vector denoted $h(x_n)$ and given by:

$$h(x_n) = F(\overline{W}_1^T \overline{x}_n) \tag{4.28}$$

where F is a nonlinear function, typically a sigmoid function, that is applied to each component of its argument. Vector $\overline{x}_n$ is the augmented input vector taking the biases into account. In the same way, the values of the output layer form a n_o-vector given by:

$$g(x_n) = F(\overline{W}_2^T \overline{h}(x_n)) \tag{4.29}$$

where $\overline{h}(x_n)$ is the augmented form of $h(x_n)$.

As noted earlier, the role of the nonlinear function F is different for the hidden and output units. At the output, it simulates logic decisions (to minimize the classification error rate) and, consequently, must approximate the threshold function of the perceptron. On the hidden units, it generates nonlinear functions of the input. In the case of binary inputs, it can be shown[16] that a sigmoid function will, in theory, generate all possible high order moments of the input. However, its form is

[15] There is an ambiguity in the literature as to whether an MLP with one hidden layer is to be called a 2-layered or a 3-layered network. We will use the former convention since the input layer performs no computation. In any event, a single hidden layer is sufficient to perform any nonlinear mapping from the input to the output, provided the set of hidden units is large enough.

[16] And it will. Just you wait.

certainly not restricted to a saturating nonlinearity such as the sigmoid function. A sinusoidal function could also be used or, as is usually done with radial basis functions (see Section 6.7), a quadratic function might be used, in which case we generate only second-order moments on the hidden units.

This property of MLPs is shown here for the case of binary inputs. Let x_{nj}, $j = 0, \ldots, d$, denote the components of the augmented input vector $\overline{x}_n$. When using a sigmoid function on the hidden units, the i-th component of $h(x_n)$ is then expressed as:

$$h_i(x_n) = \frac{1}{1 + e^{-\sum_{j=0}^{d} w_{ij} x_{nj}}} \tag{4.30}$$

or also, by using the Taylor expansion:

$$h_j(x_n) = \sum_{k=0}^{\infty} a_k (\sum_{j=0}^{d} w_{ij}.x_{nj})^k \tag{4.31}$$

where w_{ij} represents the weight between j-th input unit and the i-th hidden unit. In the case of binary inputs, we have $x_{nj}^k = x_{nj}$, $\forall k > 0$, and expression (4.31) can then be rewritten as:

$$\begin{aligned} h_j(x_n) &= \alpha_0 + \sum_{i=0}^{d} \alpha_i x_{ni} + \sum_{i \neq k}^{d} \alpha_{ik} x_{ni} x_{nk} \\ &\quad + \sum_{i \neq k \neq \ell}^{d} \alpha_{ik\ell} x_{ni} x_{nk} x_{n\ell} + \ldots \end{aligned} \tag{4.32}$$

Thus, for each component of $h(x_n)$, the nonlinear function F generates a *linear combination* of all 2^d possible cross-products of the d binary inputs (i.e., pairs, triplets, $\ldots$, d-tuples). The coefficient α of each cross-product depends on the weight matrix $\overline{W}_1$ which will be adapted during the training so as to activate the relevant cross-products, i.e., those which are *typical of the training patterns* and *insensitive to the noise*. At the same time, $\overline{W}_2$ is also adapted in order to optimize the classifications on the basis of the generated cross-products.

In classification mode, a test input will activate some cross-products and the final decision will be based on this total information. If the test input contains errors, some cross-products will be affected. However, if the training and test conditions are consistent, it can be expected that the garbled cross-products will not be relevant. Consequently, a correct decision will still occur on the basis of discriminant (pattern characteristic and noise insensitive) pairs, triplets, ..., d-tuples.

The following two examples will illustrate these properties.

4.4.2 Phonemic Strings to Words

The first example presented here is the mapping of phonemes to words by a 2-layered perceptron: given the identity of all the phonemes that are in a word, without regard to time ordering or repetition, learn to predict the word. In this experiment, there was no noise on the training patterns (in the sense of variability, since a single fixed phonemic transcription was used for each word). The experiment, then, only tested the ability of the system to generate the proper function (in the sense of the relevant cross-products described in the previous section) on which it could base the word classification.

Letting n_w be the number of lexicon words and n_{pho} be the number of phonemes describing that lexicon, consider an MLP with $n_i = n_{pho}$ input units and $n_o = n_w$ output units. The components of the input pattern x_n at time n are then defined as: $x_{ni} = 1$ for all phonemes i in the phonetic transcription of the word presented at time n and $x_{nj} = 0$ for all phonemes j that are not in the transcription. Again, an input that is on tells nothing about where in the word the phoneme occurred, or whether or not it was repeated. During training, the associated desired output vector $d(x_n)$ is an index vector Δ_k if x_n is known to represent the phonetic transcription of word k.

In this example the training of the model parameters was performed by minimizing the error criterion (4.13) on the correct phonetic transcriptions only. Robustness of the resulting system was then tested on garbled phonetic transcriptions.

The experimental lexicon contained the 10 German digits ($n_w = 10$) described by a base of 20 phonemes ($n_{pho} = 20$). With $n_h < 3$, the minimum value of E was still quite high, and it was not possible to perfectly map (i.e., memorize) the training input patterns to their corresponding outputs. With $n_h = 3$, the error was decreased to zero and all the training phonemic strings were perfectly classified. This seems reasonable since the number of available parameters $[(n_i + 1)n_h + (n_h + 1)n_o = 103]$ is then slightly larger than the number of equations to be satisfied (10 training patterns $\times$ 10 outputs = 100). However, when garbled (but typical) phonemic strings were used, the correct classification of the associated word was not guaranteed. For more hidden units ($n_h = 10$), the classification capabilities significantly improved on garbled inputs. In this case, garbled phonemic sets or subsets which were sufficient for defining a lexicon word induced the correct output. This can be explained by the fact that, with more hidden units, the system was able to "memorize" a larger number of linear combinations of discriminant cross-products of the input units. This illustrates part of the explanation given in Section 4.4.1, i.e., the classification ability for garbled patterns on the basis of typical subsets of the trained ones. We also note that had this experiment been extended to much larger networks, this memorization may have led to overtraining (overfitting), a topic to be discussed later (see, for example, Chapter 12).

4.4.3 Acoustic Vectors to Words

In a second simple study, an MLP was used to recognize isolated words from their acoustic content (again disregarding time ordering and possible repetitions) and was trained on several pronunciations of each lexicon word. The training patterns for this problem can be viewed as noisy versions of utterances, and the system must determine the discriminant cross-products that are noise insensitive and typical of the patterns.

The acoustic front end analyzed the time signal and generated a 16-dimensional cepstral[17] vector (acoustic vector) for each 10 ms time slot. These vectors were then quantized to the closest of n_p prototype vectors.

Letting n_w represent the number of lexicon words, the MLP used $n_p = n_i$ binary input units and $n_w = n_0$ output units. As in the previous section, the components of the input pattern x_n were defined as: $x_{ni} = 1$ for all prototypes i present somewhere in the utterance (at least once), corresponding to an isolated word. Similarly, we set $x_{nj} = 0$ for all prototypes j that were not present in the utterance. During training, the desired output vector $d(x_n)$ was an index vector Δ_k when x_n is known to represent an utterance of word k.

Tests were performed on a database comprising 4 pronunciations of the 10 German isolated digits. After training of the MLP on the first two pronunciations of these isolated digits with $n_i = 60$, and $n_h = 10$,

[17] The cepstrum is the Fourier Transform of the log spectrum. For some front ends the cepstral values are actually estimated from a recursion starting with LPC coefficients.

there were no classification errors on the 40 utterances of the isolated digits (20 trained and 20 untrained).

As noted earlier, in both experimental examples the repetitions and time ordering of the prototype vectors inside each word are overlooked. This simple approach would not work very well for a more difficult speech classification problem, but does illustrate that the network can learn some degree of noise insensitivity in the sense of input variability due to quantization. As suggested in Section 4.4.1, in a sense what is happening is that the network is learning which cross-products of the training patterns are helpful for classification in the presence of this quantization error.

4.5 MSE and Discriminant Distance

The MSE functions E in (4.4), (4.9) and (4.13) can all be rewritten as:

$$E = \sum_{k=1}^{K} \sum_{x_n \in \omega_k} d_k(x_n) \tag{4.33}$$

leading to the definition of a *discriminant distance* between a vector x_n and a class ω_k, $k = 1, \ldots, K$, as follows:

$$d_k(x_n) = \| g(x_n) - \Delta_k \|^2 = \sum_{\ell=1}^{K} (g_\ell(x_n) - \delta_{k\ell})^2 \tag{4.34}$$

where $\delta_{k\ell}$ is the usual Kronecker delta function, which is only nonzero (and equal to 1) when $k = \ell$.

These discriminant distances could be used as local contributions in a dynamic programming procedure for pattern matching and speech recognition [Bourlard & Wellekens, 1986]. In this case, an unknown vector x_n is assigned to class ω_k if $d_k(x_n) \leq d_i(x_n)$, $\forall i \neq k$. A principal advantage of this approach is that the discriminant distance uses information from all the classes to classify x_n. By expanding the square in (4.34), we have:

$$d_k(x_n) = \sum_{\ell=1}^{K} g_\ell^2(x_n) + 1 - 2g_k(x_n) \tag{4.35}$$

The discriminant distance $d_k(x_n)$ is thus proportional to $-g_k(x_n)$ within an additive constant independent of the class k that may be dropped during classification.

Equation (4.33) can be easily compared with the Viterbi criterion (see Chapter 3) which determines the parameters of the emission probabilities $p(x_n|\omega_k)$ associated with each state q_k (or class ω_k) such that:

$$\prod_{k=1}^{K} \prod_{x_n \in \omega_k} p(x_n|\omega_k) \tag{4.36}$$

is maximized, or, equivalently such that:

$$\sum_{k=1}^{K} \sum_{x_n \in \omega_k} -\log(p(x_n|\omega_k)) \tag{4.37}$$

is minimized.

Minimization of (4.37) in a Viterbi training could then be replaced by minimization of (4.33), in which the optimization of the parameters of a class ω_k (or an HMM state q_k) will depend on all the other classes. During recognition, the logarithm of probabilities could then be replaced by $-g_k(x_n)$. Again, as for standard HMM approaches, explicit discrimination is important only during training and appears only as an additive constant term $(\sum_{\ell=1}^{K} g_\ell(x_n) + 1)$ during recognition. Note that this term is not constant during training! This approach was initially presented in [Bourlard & Wellekens, 1986], and will be further discussed in Section 5.5.3.

4.6 Summary

In this chapter some fundamental properties of MLPs have been discussed. MLPs are particularly interesting for ASR because of their discriminant capabilities, and their ability to represent some of the statistical properties of data distributions in an automatic manner. The extension from simple perceptrons to multiple layers permits the construction of complex decision surfaces, which can facilitate classification for difficult problems in which we do not know how to extract features that are linearly separable. Although multi-layer algorithms existed as early as the 1960's, the more recent generation's use of the differentiable sigmoid function permits extensions to more layers, as well as providing a mathematical framework that is useful in understanding what is going on.

MLPs are capable of constructing combinations of features which characterize higher-order moments in the data distributions. Although we have only shown this for the special case of binary inputs, (for which a particularly straightforward representation is possible), it is also true for the more general case of continuous inputs.

In the next chapters, additional properties of these networks will be discussed, including theoretical relationships with standard statistical techniques. These relationships can help us to understand how to design ANN-based subsystems and to interface them properly with other approaches (in our case, HMMs) to improve the overall system.

Part II

HYBRID HMM/MLP SYSTEMS

Chapter 5

SPEECH RECOGNITION USING ANNs

Between two evils, I always pick the one I never tried before.
– Mae West –

5.1 Introduction

Given all the difficulties presented in Chapter 1, *Automatic Speech Recognition* (ASR) remains a challenging problem in pattern recognition. After half a century of research, the performance currently achieved by state of the art systems is not yet at the level of a mature technology. Over the years, many technological innovations have boosted the level of performance for more and more difficult tasks. Some of the most significant of these innovations include: (1) pattern matching approaches (e.g., DTW), (2) statistical pattern recognition (e.g., HMMs), (3) better use of a *priori* phonological knowledge, and (4) integration of syntactic constraints in *Continuous Speech Recognition* (CSR) algorithms. However, despite impressive improvements, performance on realistic (i.e., fairly unconstrained) tasks are still far too low for effective use. It seems likely that new technological breakthroughs will be required for the major performance improvement that will be required. Even if one assumes infinite computational power, an infinite storage and corresponding memory bandwidth, and an infinite amount of training data, it is still not certain that one could solve the ASR problem in a satisfactory way. It has also become clear that the use of higher level knowledge during the recognition process (or more generally, the efficient interaction between multiple knowledge sources) is required to overcome the limitations of current ASR systems.

An early approach to ASR emphasized the use of symbolic *Artificial Intelligence* (AI) techniques to try to model human reasoning. In this approach, rules incorporating formal logic were used to represent human expertise about speech. Using such an approach can enforce some consistency between multiple knowledge sources. However, purely symbolic approaches are insufficient to handle probabilistic or uncertain information; some relationships are simply not well-described by symbolic rules. Additionally, the techniques for learning rules are quite weak in comparison with the simple approaches available in statistical or connectionist systems. For the specific case of expert systems, interviewing an expert about his data analysis does not really determine the strategy employed, since the true underlying strategy may well be unknown to the expert.

For these reasons, speech recognition systems based purely on lists of rules have not made a significant impact on the field. However, this does not mean that expert or domain-specific knowledge is useless for speech recognition. On the contrary, even with techniques that are far more data-driven, such as those emphasized in this book, there are strong arguments for constraining the parameter search with some domain-specific knowledge. There are now a number of researchers who are attempting to incorporate many of the ultimate goals of AI without the restriction to formal symbolic systems.

More recently, many groups have attempted to use *Artificial Neural Networks* (ANNs), and *Multilayer Perceptrons* (MLPs) in particular, to perform ASR. Although examples of such research go back at least to Widrow's experiments in the early '60s, the resurgence of interest in ANN techniques in the 1980s has been reflected in an increasing amount of work in this application area. The reasons for using ANNs for ASR are numerous; although some of them are fallacious or not unique to ANNs (see Section 5.2), others are particularly attractive for ASR (and will be discussed in Section 5.4).

As with many technological pursuits, this is a fast-moving field, so it is difficult to describe without being out of date by the time of publication. Lippmann [1989] gives a very good review of the status of speech recognition by neural networks at that time. For a good review of neural networks for speech processing, see [Morgan & Scofield, 1991]. However, both works were written before the recent successes of the techniques described in this book.

5.2 Fallacious Reasons for Using ANNs

We begin this chapter by considering a number of popular fallacies about the use of ANNs in pattern recognition. With the slate clean of these errors, we can then discuss legitimate reasons to use ANNs. We begin with the most fundamental and widespread fallacy:[1]

Fallacy: *Humans understand speech through the use of biological neural networks; therefore Artificial Neural Networks, which are models of the real thing, should be able to recognize speech better than engineering systems that are less analogous to biology.*

Artificial neural networks are extremely crude models of biology, and in many common cases (such as error back-propagation), explicitly include engineering features that are generally believed to be biologically implausible. Even if the neural models were precise, unless one knew how to model interactions for large masses, one would be unsure of any ability to mimic human capabilities. Most fundamentally, however, engineering designs have different constraints (e.g., planar connectivity in silicon circuits) than those imposed by biology, so that the best engineering system could well be one that bore little resemblance to a biological system.

In defense of the spirit behind this fallacy, we note that biology can often offer useful insights for the design of practical engineering systems.

Fallacy: *Neural systems are inherently adaptive. Therefore an ANN will learn and generalize to new data, unlike classical pattern classification systems.*

Many new devotees of neural networks are not aware of the long history of trainable classifiers, including statistical approaches like

[1] We must give credit here to the wonderful book by John Hennessy and Dave Patterson, *Computer Architecture A Quantitative Approach* [Hennessy & Patterson, 1990], which gave us the idea for this section. In that work, the authors included a section on "Fallacies and Pitfalls" at the end of each chapter.

Hidden Markov Models. Most such systems can be adaptively trained, some with greater speed than common "neural" approaches.

Fallacy: *Classical pattern recognition systems require many "hacks", which purport to introduce application-specific knowledge, but which in fact constrain searches in a suboptimal (and frequently arbitrary) manner. Neural networks do not require such constraints, and so can objectively search to create the best overall system.*

It is true that adaptive systems such as ANNs do provide a mechanism for learning parameters. However, for sufficiently large problems, these systems also must be constrained, and application-specific knowledge can often be the best way to constrain them. Furthermore, any such ANN system will have some arbitrary parameters as well, such as adaptation step size, momentum, number of hidden units, range of initial random weights, etc. There is no free lunch; any recognition system begins with assumptions, and if these assumptions come from pre-existing knowledge, so much the better.

A related and interesting fallacy is the following:

Fallacy: *Traditional pattern recognition systems require the selection of arbitrary plausible features to be extracted from the original data. ANNs can do this automatically, eliminating this suboptimal step.*

Once again, it is true that optimal and automatic determination of parameters (features) is desirable. It is also true that in some limited cases it appears to be possible to automatically derive features from raw data, given significant application-specific constraints. This is the case for AT&T's handwritten zip code recognizer [Denker et al., 1989; le Cun et al., 1989] in which a simple convolutional method was used to extract important features such as lines and edges that are used for classification by an MLP. In early versions of this system, these features were designed by hand. In a later version, though, [le Cun et al., 1990], the AT&T group found that they could do as well with automatically learned features. Nonetheless, even this system incorporated much knowledge about the nature of the task. Particularly for speech, raw data frequently contains a significant amount of information that is irrelevant to the classification task, and some simple processing can often improve performance significantly. For example, many speech waveforms can sound quite similar while having entirely different morphology. Additionally, speech with very different power spectra can convey the same linguistic information. For instance, vowel spectrograms for adult males, adult females and children show major differences for sounds that are clearly identifiable as the same vowel. Of course, what can be learned and what must be pre-determined from speech knowledge is in general unknown, but it is likely that intelligent selection of at least a plausible superset of features will continue to be required for ANN-based speech recognition systems.

If these contentions are indeed fallacies, what remains as the potential of speech systems incorporating ANNs? An additional problem is that ANN systems are themselves poorly defined. Since biological analogies are strained for any ANN techniques, we adopt an operational definition:

An Artificial Neural Network = A Connectionist System.

This rids us of the biological baggage, but we still need a definition of a Connectionist System.

Connectionist System = System in which information is represented and processed in terms of the input pattern and strength of connections between units that do some simple processing on their input.

Associated with this simple definition is the collection of current techniques for choosing and adjusting the connection strengths. Given these elements, we can then reasonably claim some potential advantages in using a connectionist approach (at least for a speech recognition system).

5.3 Valid Reasons for Using ANNs

As we will show in the following chapters, a connectionist system can combine multiple constraints and sources of evidence using a simple criterion, but without explicit statistical assumptions (e.g., independence of evidential sources or a particular parametric form of their distributions). Note that this does not preclude hybrid systems with both connectionist and non-connectionist subsystems. As shown in [Bourlard & Wellekens, 1989a], ANNs can be used to provide statistical estimates for non-ANN systems.

Although we have suggested that the elimination of system heuristics is a naive hope, it is nonetheless a reasonable aim to partially supplant the arbitrary or semi-informed selection of key parameters with adaptive learning procedures.

Finally, MLPs have several advantages that make them particularly attractive for ASR, e.g.:

- Like HMMs, they can learn.
- They can provide discriminant-based learning; that is, models are trained to minimize the error rate while maximizing the distance between the correct model and its rivals.
- They can generate, in theory, any kind of nonlinear functions of the input [Lippmann, 1987; Cybenko, 1989; White, 1988].
- Because they are capable of incorporating multiple constraints and finding optimal combinations of constraints for classification, features do not need to be treated as independent. More generally, there is no need for strong assumptions about the statistical distributions of the input features (as is usually required in standard HMMs).
- They have a very flexible architecture which can easily accommodate contextual inputs and feedback.
- ANNs are typically highly parallel and regular structures, which makes them especially amenable to high-performance architectures and hardware implementations.

However, connectionist formalism is not tailored for time sequential input patterns (like speech). While ANNs have already proved useful in recognizing isolated speech units by using some of the techniques described below, they have only recently begun to make serious inroads into large vocabulary ASR systems. Indeed, nearly all ANN systems for ASR are just static pattern classifiers – given labeled and segmented training data, networks can be trained to recognize isolated speech segments. However, the most general form of ASR should accept continuous speech as input. Any such ASR system, whether it uses ANNs or not, must perform a dynamic recognition process, in which the input speech is segmented (perhaps implicitly) as well as being classified, so that the output is a succession of words which explain the acoustical input. Furthermore, linguistic constraints (both syntactic and semantic) are a component of most ASR systems. Because of these considerations, as well as the inherent difficulty of robust ASR, applying ANN methods to ASR is a challenging research area.

5.4 Neural Nets and Time Sequences

There are several problems related to sequence learning with ANNs (see [Hertz, Krogh & Palmer, 1991] for a good discussion of these):

1. Sequence recognition: Classification of an input string into a specific output class – this is the typical problem of isolated speech unit recognition.

2. Sequence completion and prediction (signal extrapolation): Given the beginning of a sequence, the ANN should be able to complete it or to predict the future events (see Chapter 13).

3. Temporal association: A particular output sequence must be produced in response to a specific input sequence – this, of course, includes (1) as a special case.

4. Sequence generation: Given an input code (called "plan" in [Jordan, 1989]), generate a specific output sequence.

The general formulation of the CSR problem (with HMMs or ANNs) is actually the following: how can an input time sequence be properly explained in terms of an output time sequence when the two sequences are not synchronous (since there are multiple acoustic vectors associated with each pronounced word or phoneme)? Although this formulation is related to problem (3) mentioned above, it is even more general since the two sequences are not synchronous. Also, ideally, the acoustic vectors should enter the network sequentially as they become available. However, connectionist formalism is not very well suited to solve such a problem; most previous applications of ANNs depended on severe simplifying assumptions (e.g., small vocabulary, known word or phoneme boundaries). In this section, several ANNs that have been used to solve sub-tasks of the ASR problem will be reviewed. However, none of these approaches addresses the general problem of CSR, including time alignment and segmentation (which is efficiently solved by dynamic time warping of HMMs). Neural networks that have been designed to implement some aspects of HMM functionality will be discussed in Section 5.5.

5.4.1 Static Networks with Buffered Input

The simplest way to perform sequence recognition is to turn the temporal pattern into a spatial pattern at the input of an MLP. The MLP can then be trained using the standard *Error Back-Propagation* (EBP) algorithm. In this case, the entire input sequence to be trained or recognized is stored in a buffer at the input of the MLP [Landauer et al., 1987; Peeling & Moore, 1988; Burr, 1986; Burr, 1987; Gold et al., 1987] and the possible speech units of the lexicon are associated with the output units of the MLP. In this case, MLPs are used to classify isolated words (primarily digits), phonemes, and vowels using presegmented speech tokens which are typically applied at once as a whole input pattern (typically spectrograms). Several variants of this approach have been studied, including: (1) the input buffer is large enough to accommodate the largest possible input pattern – in this case, the input sequence to be classified can simply be located at the middle of the input window or shifted across all possible positions to track the optimal output; (2) the input patterns are scaled to fit a predefined fixed window width – either linear [Krause & Hackbarth, 1989] or nonlinear scaling can be used, where dynamic programming is required for the latter case.

In some cases, MLPs were used in conjunction with conventional time alignment techniques (see Sections 5.5.3 and 5.6). The generic MLP architecture of this approach is given in Figure 5.1.

A similar approach using continuous-time filter delays (but no hidden units) has also been presented in [Tank & Hopfield, 1987; Unnikrishnan et al., 1988] in an analog input signal application with a specific probabilistic interpretation on the signal shape degradation through the filter delays. In this case, the processing may be viewed as a form of forward recurrence (as used to estimate full MLE in HMMs) [Bourlard & Wellekens, 1989b].

Although static networks appear to work at least as well as HMMs for some cases of isolated sequence recognition, there are several known drawbacks or limitations:

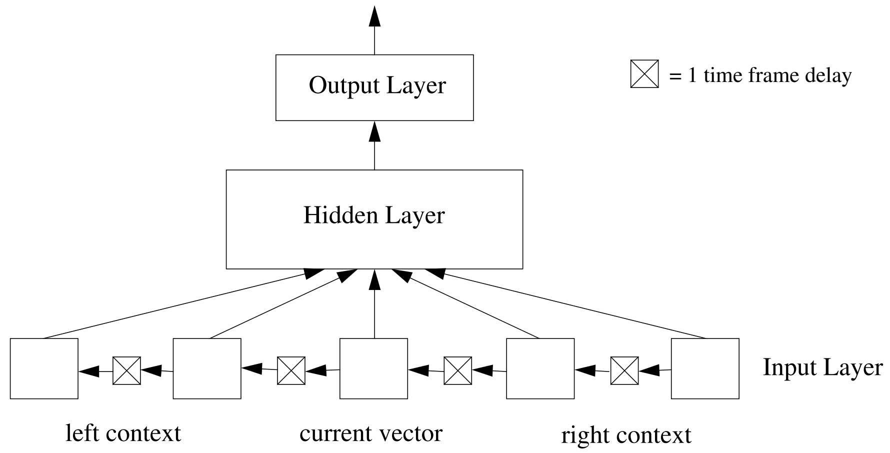

Figure 5.1: MLP with tapped delay lines. This can be used to classify isolated speech units (words or phonemes) if the input buffer is large enough to accommodate the longest possible sequence. This can also be used to generate context-dependent frame labeling in conjunction with conventional time alignment techniques.

- The input buffer must be large enough to accommodate the longest possible input sequence, which increases the number of parameters and, consequently, the number of required training examples.

- When used, linear scaling is too crude; nonlinear scaling is better but requires either prototypes or trained HMMs.

- The network is not automatically shift- and distortion-invariant; to have such a property it is often necessary to train it on a large number of utterances for each output class (words) and to shift them everywhere through the input layer.

- This approach does not seem to be appropriate to the recognition of connected speech units or, in other words, to find the best explanation of an input pattern in terms of a sequence of output classes.

One interesting application of this approach was NETtalk, in which an MLP was trained to pronounce English text [Sejnowski & Rosenberg, 1987]. The architecture of the network was similar to the one represented in Figure 5.1 in which the input of the network constituted seven consecutive characters from some written text, while the desired output was the phoneme associated with the letter at the center of the input window. Although the final quality of this system was much lower than is achieved in the best rule-based systems (such as DECtalk), NETtalk was interesting from the standpoint of being able to essentially learn the necessary rules automatically from examples. This approach can be of more practical interest for problems that are not understood as well, so that extensive rule sets have not been formulated. For problems in well-understood domains, rule-based systems can sometimes significantly outperform unconstrained neural network approaches. In practice, some combination of knowledge-based and "ignorance-based" approaches [Gevins and Morgan, 1984] will be optimal for any given task, since we will have explicit knowledge for some parts and very little for others. Since deep understanding is not yet available for phonetic classification, a NETtalk-like approach was tested for phonemic labeling of acoustic vectors, and will be described in Section 5.6.

5.4.2 Recurrent Networks

Ideally, ANNs used for speech recognition should accept input vectors sequentially. This requires some kind of recurrent internal state that would be a function of the current input and the previous internal state [Bridle, 1990b] (as is usually done with the state space equations in control theory [Jordan, 1989]). Various recurrent networks using time-step delayed recurrent loops on the hidden and/or output units of a feedforward network have been proposed and tried. For sequences of small maximum length N, we can turn these recurrent networks into equivalent feedforward networks (by "unfolding" them over the time period N) that can be trained by a slightly modified form of EBP, referred to as *EBP through time*, in which:

- All copies of the "unfolded" weights are constrained to be identical during training. In practice this is usually achieved by computing the correction terms separately for each weight and using their average for updates.

- The desired outputs are functions of time, and errors have to be computed (and back-propagated) for every copy of the output layer. This requires the selection of an appropriate time-dependent target function (which is currently ad hoc – even for training isolated units, there is no principled method for selecting this target function, usually chosen as a linear ramping function [Watrous & Shastri, 1987]). Another solution is to define the target function (and to back-propagate the error) only at certain times (i.e., on certain copies of the output units) corresponding, for instance, to the end of the words or phonemes.

This approach, also referred to as *unfolding of time,* was originally suggested in [Minsky & Papert, 1969] and combined with EBP in [Rumelhart et al., 1986a], where it was shown that this worked well for the task of learning to be a shift register and for a sequence completion task.

An implementation of EBP through time for speech recognition was proposed in [Watrous & Shastri, 1987], where sequential processing was performed with the "temporal flow model," and delayed selfloops were added to each hidden unit (a single layer of hidden units is considered) and to each output unit of an MLP. A training procedure based on EBP through time was presented in [Watrous & Shastri, 1986].

Apart from the arbitrary choice of the time-dependent target function, the other problem of EBP through time is the large computer resource requirement (memory and CPU) resulting from the duplication of the units. For long sequences, or for sequences of unknown length (which is the case in speech recognition), this approach quickly becomes impractical. In [Kuhn, 1987; Kuhn et al., 1990], it was shown that it is possible to avoid EBP through time, at the expense of a large extra set of partial derivatives which must be carried forward.

Pineda [1987, 1988], Almeida [1987, 1988], and Rohwer & Forrest [1987] showed that it was possible to generalize EBP to arbitrary recurrent networks without duplicating the units. The approach works in principle for networks without time-step delays on the recurrent loops, as long as they converge to stable states. See also [Williams & Zipser, 1989a,b; Robinson & Fallside, 1988; and Rohwer, 1990]. These more fully recurrent networks have not been used very much for speech recognition. Most of the time, partially recurrent networks with feedback of the hidden or output units to the input layer seem to suffice and are less costly to implement.

5.4.3 Partial Feedback of Context Units

A popular approach for time sequence classification (and sometimes generation [Jordan, 1989]) is to use feedforward networks that are complemented by a carefully chosen set of local feedback connections [Hertz, Krogh & Palmer, 1991] with one time-step delay. These networks are usually implemented by extending the input field with additional "feedback units" containing the hidden or output values generated by the preceding input. These feedback units will encode the past information that is required to generate the correct output (both for classification and sequence generation) given the current input. In theory, the current state of the whole network will nonlinearly depend on a combination of the previous state activation and of the current input. In this case, a simpler form of EBP through time must be used to train the weights of the feedback units, although the standard EBP algorithm is also sometimes used (restricting the dependency of the network to the previous time frame only).

Figures 5.2 and 5.3 show the two most common architectures that have been used. Elman [1988] suggested the architecture shown in Figure 5.2 in which the hidden unit values from the previous time step are fed

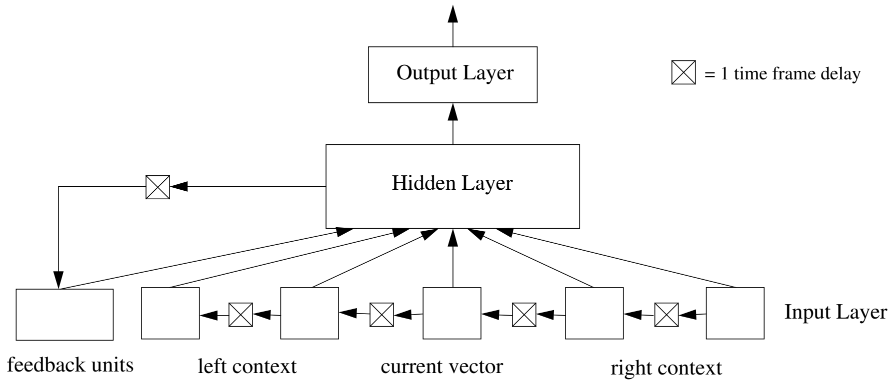

Figure 5.2: MLP with contextual inputs and hidden vector feedback.

back to the input field. The resulting system is then a more general implementation of the temporal flow network (without recurrency on the output layer). Indeed, in this case, on top of having self-loops on the hidden units, there are also recurrent connections between the different hidden units. This network was used successfully to classify short speech sequences like phonemes in [Robinson & Fallside, 1990]. In this work, the network was used to classify phonemes from input sequences of 10 msec acoustic vectors; the EBP through time algorithm (going up to 20 frames back into the past) was used to train the recurrent connections. In [Robinson & Fallside, 1990; Robinson & Fallside, 1991], it was shown that this approach was able to improve over state-of-the-art phoneme recognizers. Recently, this work was extended to integrate this network in an HMM/ANN hybrid similar in spirit to the approach that will be used in Chapters 6-8. This type of recurrent network has also been shown to be able to produce short continuations of known sequences [Elman, 1988]. In [Cleeremans et al., 1989] it was shown that the architecture was also appropriate to model finite state automata.

Another recurrent model, shown in Figure 5.3, was also proposed in [Jordan, 1986, 1989]. In this case, one time-step delayed output values were fed back to the input field. This approach was initially used as a production model, with a fixed input referred to as a "plan" in [Jordan, 1986]. The network was trained to generate specific sets of output

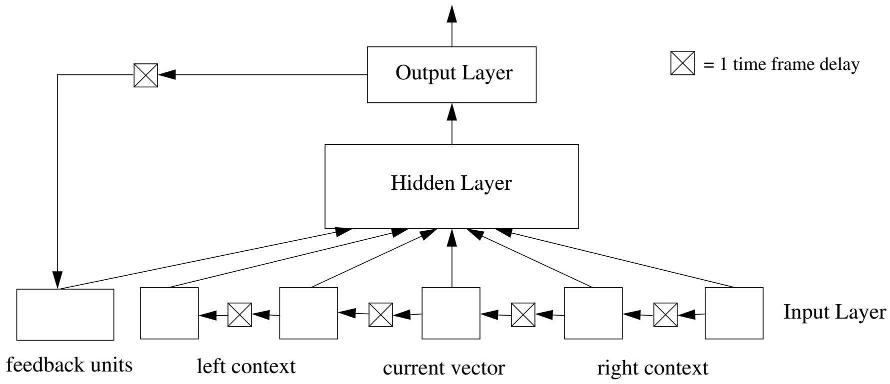

Figure 5.3: MLP with contextual inputs and output vector feedback.

sequences, each of them being associated with a specific input pattern. This architecture can also be used for the classification of sequential inputs [Bourlard & Wellekens, 1990]; its relation to HMMs will be discussed in Chapter 6. Again, the training of this network can be done with standard EBP if the memory of the system is limited to the previous time frame. Otherwise, a simplified version of the EBP through time algorithm can be used to take several previous time frames into account.

All of these networks are clearly recurrent and can be interpreted in terms of control-theoretic state space equations [Robinson & Fallside, 1987]. Let x_n, g_n and s_n be the input vector, the output vector, and the feedback or state vector, respectively, at time n. In this case we identify state vector s_n with either the previous hidden vector or output vector. Given these definitions, the most general formulation of the *state space equations* are:

$$\begin{aligned} g_n &= g(s_n, x_n) \\ s_{n+1} &= s(s_n, x_n) \end{aligned}$$

As noted earlier, an MLP can, in theory, approximate any kind of nonlinear input-output mappings [Lippmann, 1987; Cybenko, 1988; White, 1988]. Therefore, separate MLPs could be used to generate functions $g(\cdot)$ and $s(\cdot)$. However, in the recurrent MLPs presented above, there is just one MLP computing function $g(\cdot)$. It is further assumed that function $s(\cdot)$ is given by the same network, and corresponds either to the function computed on the hidden units or on the output units (which is also a function of the hidden vector). Similar assumptions are common in linear control theory, where the state vector is defined as [Jordan, 1989]:

$$s_{n+1} \;=\; \mu s_n + g_n$$

where μ is restricted to a scalar value. Consequently:

$$s_{n+1} \;=\; \sum_{\tau=1}^{n} \mu^{\tau} g_{n-\tau}$$

and, in this case, it is known that $0 \leq \mu < 1$ is a sufficient stability condition. As shown in [Jordan, 1986], this behavior can be modeled by adding extra input units to the MLP, called state nodes, receiving input from themselves and the output nodes. For linear units, these networks have a rational input-output transfer function and, in view of the analogy with digital filters, they may be called *"Infinite Impulse Response (IIR) Dynamic Nets"* [Robinson & Fallside, 1988].

Usually, no contextual information is explicitly used at the input of these networks, but it is clear that these architectures could easily accommodate contextual inputs as represented in Figures 5.2 and 5.3. The capabilities of recurrent networks are atill not completely understood. However, it is likely that such architectures will be useful for speech recognition, since they can potentially classify sequential inputs by incorporating both the recent past of both network state and observations (feature vectors). See [Poritz, 1988] for related comments about preferred forms for HMMs to be used for speech recognition.

5.4.4 Approximating Recurrent Networks by MLPs

A solution that is intermediate between the spatial input model and recurrent models has been proposed in [Makino et al., 1983; Waibel et al., 1988]. In this approach, recurrent networks are approximated over a finite time period (say D time slots) by a feedforward network in which the loops are replaced by the explicit use of several preceding activation values. As for static networks, this kind of network, usually referred to as *Time Delay Neural Network* (TDNN) [Waibel et al., 1988; Waibel et al., 1989; Lang et al., 1990], can be trained to recognize a sequence of predefined length (defined by the width of the input window) using standard EBP. In this case, the activations in a layer are computed from the current and multiple delayed values of the preceding layer, and the output units are activated only when a complete speech segment has been processed. The corresponding architecture (in the case of only one hidden layer) is shown in Figure 5.4. This approach has been successfully used to classify pre-segmented phonemes in [Waibel et al., 1988] and led to very good results. TDNNs have also been used for a variety of non-speech problems, such as the recognition of zip codes [le Cun et al., 1990].

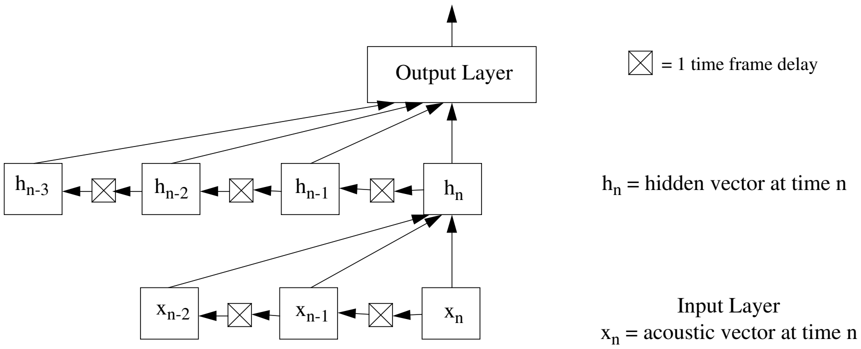

Figure 5.4: Time Delay Neural Network (TDNN).

By analogy with filter theory, this kind of network may be referred to as a *"Finite Impulse Response (FIR) Dynamic Net"*. In the linear case, IIR and FIR dynamic nets may be compared to IIR and FIR filters. The system response (z-transform of the unit sample response) of an IIR filter (Figure 5.2 and 5.3) is of the form

$$\frac{1}{1 - az^{-1}} \simeq \sum_{i=0}^{\infty} a^i z^{-i}$$

where a is related to the loop weight matrix, and can be approximated over D time slots by the response of an FIR filter (Figure 5.4)

$$\sum_{i=0}^{D} a_i z^{-i}$$

where the a_i's do not have to be equal to the successive powers of a in the IIR expansion. This is the same for a TDNN, where two possibilities have been considered: (1) tied values for the weights originating from the successive time-delayed copies of a same layer, in which case it is exactly equivalent to a finite time approximation of the feedback connections (of partially recurrent neural networks), and (2) the values of the weights originating from the time-delayed versions of the same units are not tied, which results in a system with more parameters, which is no longer equivalent to finite-time approximation of a recurrent network.

Although the TDNN by itself is not well suited to continuous speech or long word recognition, it can be used in a standard approach to better discriminate problematic (short) words like functional words.[2] These networks have been proved to perform well on different phoneme sets as well as on a larger set like 100 CV syllables in Japanese [Sawai, 1989].

5.4.5 Discussion

All of these models have been shown to yield good performance (sometimes better than HMMs) on short, isolated speech units. By their recurrency and their implicit or explicit temporal memory, they can perform some kind of integration over time. However, neural networks by themselves have not been shown to be effective for large scale recognition of continuous speech. There is at least one fundamental difficulty with supervised training of a connectionist network for continuous speech recognition: a target function must be defined, even though the training is done for connected speech units where the segmentation is generally unknown. As noted earlier, this is particularly a problem for recurrent networks. Target definition is not a problem for HMM-based training, which only requires the sequence of speech units and not their temporal segmentations. For recognition, HMMs not only tackle the variability of speech pronunciation, but are also efficient tools for connected speech recognition and segmentation. This property seems to be difficult to achieve using a connectionist architecture by itself.[3]

[2] Both feedforward and recurrent networks can be used as part of a continuous speech recognizer – in fact, this is the main point of this book. More on this later.

[3] We note that some successful work has recently been done on explicit segmentation prior to classification, with both tasks being performed by MLPs [Fanty & Cole, 1991; Cole et al., 1991].

5.5 ANN Models of HMMs

Motivated by the success of HMM algorithms for speech recognition problems, several neural network implementations of HMMs have been studied. These are different implementations of the same formalisms, and can help in understanding HMMs and ANNs, both in terms of their relationships and limitations.

In this section, three ANN paradigms are discussed which are closely related to the HMM formalism: a neural network implementation of the Viterbi algorithm (Section 5.5.1), a neural network implementation of the full MLE criterion (Section 5.5.2), and a first attempt to combine ANNs with HMMs (Section 5.5.3).

5.5.1 The Viterbi Network

The Viterbi Network [Lippmann & Gold, 1987; Lippmann, 1989] emulates the function of the Viterbi algorithm. In this case, each HMM is associated with a neural network in which each output unit corresponds to a HMM state. In the case of Gaussian HMMs, the first hidden layer computes the set of Gaussian outputs (implemented using the perceptron structure – see Section 4.2.3) for the input vectors that are presented sequentially to the network. Each output node is also complemented by time-delayed connections between the different output units to represent the topology of the underlying HMM followed by a comparator sub-network to compute the minimum of the activation values of output nodes at the previous time step; for this network these outputs are roughly equivalent to the negative logarithm of output probabilities for the original HMM.

This formulation is a neural network implementation of the *Dynamic Time Warping* (DTW) algorithm (see Section 3.3.5) that finds the best path through an HMM given a sequence of input vectors [i.e., ANN implementation of recurrence (3.24)]. This system does not implement the basic data movement required in a practical implementation of the Viterbi algorithm, which can be dominated by pointer bookkeeping. Also, it does not overcome the limitations of standard HMMs. However, it does show that a fundamental HMM algorithm can be mapped to a connectionist framework.

5.5.2 The Alpha-Net

In [Bridle, 1990b; Kehagias, 1989; Niles & Silverman, 1990], a form of recurrent neural network was introduced to emulate the formulation of HMMs using the MLE criterion (i.e., as opposed to the Viterbi, which only finds the best path through the HMM). In this case, the units in the recurrent loop are linear and the acoustic vectors enter the loop via a multiplication to simulate the operation of a standard HMM state using the full likelihood criterion. This simulates the forward recurrences (3.14), and explains why these networks were referred to as *Alpha-Nets* in [Bridle, 1990b]. It was also shown that the Alpha-Net training could be done by a kind of EBP through time that had the same form as the Forward-Backward algorithm for MLE training of HMMs (as briefly recalled in Section 3.4.1; remember that this algorithm can be viewed as a gradient technique [Levinson et al., 1983]).

Beyond this equivalence between HMMs and Alpha-Nets, some of the HMM constraints can be relaxed. For example, the constraint that probabilities sum to one and are positive could be dropped in the neural network implementation. However, in this case, the probabilistic interpretation of the model parameters will be sacrificed. Moreover, while this approach can also be generalized to other criteria (such as MMI, LMS, and relative entropy), the actual differences and potential advantages versus standard HMM approaches are not clear, and it remains to be seen if this approach can improve performance over conventional algorithms. Ultimately, the Alpha-Net also suffers from the same limitations as HMMs using the MLE criterion (e.g., the difficulty to recognize continuous speech). However, as with the Viterbi Net, this work shows that what has been thought of as an entirely non-connectionist algorithm in fact has a reasonable connectionist formulation.

5.5.3 Combining ANNs and Dynamic Time Warping

In this chapter, we have discussed several ANN paradigms that can handle (to some extent) the sequential aspect of the speech signal. However, training and recognition of continuous speech does not seem possible solely by training connectionist networks with supervised learning algorithms. In particular, a target function has to be defined, which is difficult if training is carried out on connected speech units where the segmentation is generally not known. Even for training isolated units, there is no principled method for selecting the target function.

In standard recognition algorithms (based on HMMs or not), though DTW is used to tackle the variability of speech pronunciation, it is also an efficient tool for connected speech recognition and segmentation. This latter property seems to be difficult to achieve with neural networks by themselves. Even assuming a perfect dynamic system taking the entire past into account, the ANN output values would still represent the scores for states (or output classes) at the current time and would not give any idea about the underlying segmentation (as the output values alone do not carry the information needed to recover the best interpretation).

However, all is not lost.[4] Recall that many of the networks presented in this chapter were formulated using variations of the EBP algorithm, mostly using a LMS criterion. It can be shown that, in this case, it is theoretically possible to combine these ANNs with a DTW procedure to perform embedded training or continuous speech recognition.

Let K denote the number of output classes of a particular neural network. Since the final goal is to classify speech patterns, the desired output vector during training is a K-dimensional index vector denoted Δ_k. As already shown in Section 4.5, whether the input vectors are real or binary, LMS criteria (4.4), (4.9) and (4.13) can be rewritten under the general form:

$$E = \sum_{n=1}^{N} \| g(x_n) - \Delta(x_n) \|^2 \tag{5.1}$$

where N represents the total number of training input patterns x_n, ($n = 1, \ldots, N$), $g(x_n)$ is a general (linear or nonlinear) function of the n-th input pattern x_n and $\Delta(x_n)$ is the index vector associated with the class to which it is supposed to belong. In Section 4.5, we have shown that this led to the definition of discriminant distances

$$d_k(x_n) = \| g(x_n) - \Delta_k \|^2 = \sum_{\ell=1}^{K} g_\ell^2(x_n) + 1 - 2g_k(x_n) \tag{5.2}$$

As already shown in Section 4.5, the two first terms of (5.2) are independent of the class q_k and the distance $d_k(x_n)$ between an input pattern x_n and a class q_k can be replaced by $-g_k(x_n)$, i.e., the activation value of the k-th output unit of the network.

[4] It better not be. There are a lot more pages left to this book.

However, minimization of (5.1) to determine the parameters of the discriminant functions requires supervised training. Since manual segmentation and labeling of speech databases is a tedious and expensive task, this is generally not applicable. Therefore some unsupervised learning procedure should be devised. A solution to this problem is to embed the LMS criterion described above in an iterative procedure alternatively combining a DTW and a MSE minimization [Bourlard & Wellekens, 1986]. In this case, as for the training of HMM, we do not require any frame labeling but only the class sequence associated with each training sentence (i.e., we do not need a segmentation of the training material). Starting from a rough segmentation (e.g., a linear segmentation) of the training material, a partition into classes results, and can be used to provide the network, (or the linear discriminant functions) with a target function. Minimization of the MSE (e.g., using EBP) yields a set of parameters $W^{(0)}$ corresponding to a score function value $E = E^{(0)}$.

This set of parameters can then be used to generate "discriminative distances" (5.2), or equivalently $-g_k(x_n)$, which can be used as local distances in a classical DTW procedure to find a new segmentation of the training sentences (by mapping the vector string of each training sentence on its corresponding phonemic transcription). In other words, a training utterance $X = \{x_1, x_2, \ldots, x_N\}$ will be segmented in L consecutive classes q_ℓ, $(\ell = 1, \ldots, L)$, associated with its phonemic transcription. Since we assume to know the class sequence associated with each acoustic vector string and since we assume that each possible partition must preserve the production order, the DTW recurrence is the following:

$$D_\ell(X_1^n) = d_\ell(x_n) + \min \left\{ \begin{array}{l} D_\ell(X_1^{n-1}) \\ D_{\ell-1}(X_1^{n-1}) \end{array} \right. \tag{5.3}$$

$\forall \ell = 1, \ldots, L$, $\forall n = 1, \ldots, N$, where $D_\ell(X_1^n)$ represents the minimum matching distance between the first n acoustic vectors of X and the first ℓ classes (phonemes). The backtracking of the optimal path provides the new vector partition. Note the similarity to the Viterbi recurrence given in (3.24), with sums of distances replacing products of probabilities (also there are no transition probabilities in the DTW case).

It can easily be shown [Bourlard & Wellekens, 1986] that the global score $D_L(X_1^N)$ (or the accumulation of these global scores for all the utterances in the training set) can be expressed exactly as E in (5.1) but for the new vector partition: this allows successive comparisons between the

DTW scores and MSE. The new score, denoted $E^{(1)}$, is obviously smaller than (or equal to) $E^{(0)}$ as it corresponds to the best path, i.e., a better vector partition. With this new vector partition, a new set of weights $W^{(1)}$ can be computed via MSE minimization resulting, for the path, in a score $E^{(2)}$ smaller than $E^{(1)}$. Successive DTW and MSE minimization will respectively generate improved vector partitions, better parameters W, and a sequence of continuously decreasing scores E, which guarantees the convergence of this iterative process [Bourlard & Wellekens, 1986]. When the segmentation points are stabilized, the process is stopped and yields optimal parameter values and optimal segmentation of the training data set.

Convergence of this process (of course, always to a local minimum!) was proved in [Bourlard & Wellekens, 1986] for linear discriminant functions. However, since this proof did not rely on any particular form of the discriminant functions, it remains valid for ANNs. As a consequence, it is thus possible to embed the determination of neural network parameters in a DTW process to iteratively improve the segmentation points.

This iterative training was tested with linear discriminant functions [Bourlard & Wellekens, 1986] and MLPs [Bourlard & Wellekens, 1987] but never led to good results (in this form), although the phonemic labeling at the frame level obtained from hand segmented training data was significantly better (see Section 5.6). This training, consisting of iterating the optimization of the network parameters and the segmentation by DTW, did indeed converge, as predicted by theory. However, the parameters of the linear discriminant functions or MLP appeared to minimize the sum-squared error measure by clustering all of the utterances in the class that occurred most frequently in the initial segmentation (usually the silence model). Similar results were also reported in [SPRINT, 1990]. The reasons for these problems will become clearer in the next chapter.

5.5.4 ANNs for Nonlinear Transformations

It is important to mention here that preliminary research has also been done with other forms of hybrid HMM/ANN approaches in which the outputs of a (recurrent) ANN constitute an observation sequence for the HMMs [Bridle & Dodd, 1991; Bengio et al., 1992]. As opposed to the approach presented in Section 5.5.3 (where the ANN is used to generate locally discriminant distances for use in DTW), the ANN is now used to optimize the output parameters via some (linear or nonlinear) transformation. This allows a global optimization of the input transformation (i.e., the parameters of the ANN) together with a global training of the HMMs according to an MLE or an MMI criterion.

5.5.5 ANNs for Preprocessing

ANNs have also been used to perform particular kinds of clustering of the acoustic vectors for use in (standard) discrete HMM systems. The best known example of this approach is the phonotopic or self-organized feature map, proposed by Kohonen [Kohonen, 1988a; Kohonen, 1988b]. In this approach, reference feature vectors are used to partition a two-dimensional space (referred to as a self-organized or topological map) that the n-dimensional feature vectors are mapped to while preserving the topology of the initial space.

A number of researchers have also experimented with a related method that directly incorporates supervisory information in the formation of the reference vectors. These approaches are generally called Learning Vector Quantization (LVQ), and there are a number of variants, most notably LVQ2 [McDermott & Katagiri, 1989; McDermott & Katagiri, 1991]. At least for small speech classification tasks such as phoneme recognition, these approaches appear to be competitive with back propagation.

5.6 Discrimination with Contextual MLPs

As briefly presented in Section 5.4.1, NETtalk [Sejnowski & Rosenberg, 1986, 1987] was an MLP that was trained to convert graphemes to phonemes. It is shown in this section that the same network architecture and learning algorithm can be useful for phonemic labeling. Moreover, the contextual aspects which are known to play an important role in speech recognition [Furui, 1986; Wellekens, 1987] and which are not easily implemented as HMMs will be explicitly taken into account. More precisely, a 2-layered network (i.e., with one hidden layer) will be trained for the mapping of acoustic vectors to phonemes.

Let q_k, with $k = 1, \ldots, K$, be the output units of the MLP associated with different classes (phonemic classes in our case). To make the MLP training simpler and faster, we decided to use discrete features.[5] In this

[5] Later experiments, using fast training hardware developed at ICSI, used contin-

case, each acoustic vector x_n of an utterance $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ is quantized and replaced by the closest prototype vector (see Section 3.3.3) from $\mathcal{Y} = \{y_1, \ldots, y_I\}$, the set of prototype vectors (I being the total number of prototype vectors). Sequence X is thus replaced by a prototype vector sequence $Y = \{y_{i_1}, \ldots, y_{i_n}, \ldots, y_{i_N}\}$, where $i_n \in [1, I]$ represents the label of the closest prototype vector associated with x_n. Typically, each prototype vector y_{i_n} of Y will be represented at the input of the MLP as a I-dimensional binary vector with all zero components but the i_n-th one equal to one, which will be referred to as the *index vector* of x_n and will be denoted v_{i_n} in the following. If no contextual information is used, the input pattern of the MLP at time n is simply v_{i_n}. In the case of the NETtalk architecture, contextual information is used, and the input of the MLP is then built up by concatenating the prototype vectors belonging to a given contextual window centered on a current y_{i_n}, as represented on Figure 5.1. Thus, if $2c + 1$ is the width of the input window (c frames of left context, the current frame, and c frames of right context), the MLP input at time n will be $\{v_{i_{n-c}}, \ldots, v_{i_n}, \ldots v_{i_{n+c}}\}$.

Acoustic contextual information was already used in the ERIS system [Marcus, 1981, 1985] where the classification of vector-pairs was based on discriminant principles (responsible "demons" for the recognition of one particular class and the rejection of all other stimuli as "non-words"). However, for larger contexts, this becomes impractical as the number of possible combinations grows exponentially, needing an excessive amount of training data and an excessive storage capacity. In this case, a NETtalk-like MLP becomes an interesting alternative. Figure 5.1 shows the schematic arrangement of this system, characterized by two layers of perceptrons (one layer of n_h hidden units and one layer of $n_o = K$ output units) computing the classification of the input field. The input field is constituted by several groups of units, each group representing a prototype vector. Thus, if $2c+1$ is the width of the contextual window, there are $n_i = (2c + 1)I$ input units and the number of possible input patterns is equal to I^{2c+1}, which makes it impossible to estimate the discrete emission probabilities by a simple counting as was done in [Marcus, 1981] for vector-pairs.

During training, the desired output of the network is an index vector Δ_k if the current input v_{i_n} is associated with phonemic class q_k. After quantization of the training utterances (replacement of each acoustic

uous input features; see Chapter 11 for a description of this system development.

vector by its nearest prototype), each quantized acoustic vector is presented within its context as a training pattern and the error between the generated output and the associated desired output is back-propagated for adjusting the weights. The prototype vectors are stepped through the contextual window time slot by time slot and, at each step, the MLP parameters are adjusted by the classical (on-line) EBP algorithm.

This algorithm was used for training 26 pseudo-phonemes of a *speaker-dependent* German database containing four pronunciations of the digits and 100 pronunciations of 7-digit strings spoken continuously. The MLP architecture was characterized by $2c + 1 = 11$ (five acoustic vectors for the left and right contexts), $I = 60$ (i.e., 60 prototype vectors), $n_i = (2c+1)I = 660$, $n_h = 50$ and $n_o = K = 26$. The training was performed on the first two pronunciations of each isolated digits. The *a priori* segmentation used for the training was obtained by the method described in [Aubert, 1987]. Starting from a speech signal sampled at 8kHz, the acoustic analysis computed a 16-dimensional cepstral vector from a sliding window of 30 ms shifted every 10ms.

Figure 5.5 shows the time signal and the 26 output levels $g_k(v_{i_{n-5}}, \ldots, v_{i_n}, \ldots, v_{i_{n+5}})$ (corresponding to the 26 pseudo-phonemes) of the trained network for each time frame n of a third pronunciation of the word "eins" [AI-j-N-S], which has not been seen by the system during the training. Label "si" stands for the "silence phoneme". The result of the automatic segmentation is also shown. It can clearly be observed (Figure 5.5) that the correct phonemes were strongly discriminated.

The next step was to use these output values in a DTW process [Bourlard et al., 1985] to perform word recognition. As shown in Section 5.5.3, the output values $-g_k(\cdot)$ of the MLP can then be used in a one-pass DTW algorithm [Ney, 1984; Bourlard et al., 1985] to perform the recognition at the word level. In this test, word models were built from the concatenation of phonemic models. Forty utterances of isolated digits, among which 20 have been used as training set, were recognized without any errors (recognition score = 100 %). In a second test, 50 strings of 7 connected digits (without any pause between digits) were recognized with 3 insertions, 22 substitutions and 1 deletion, i.e., with a score of 92.5 %. To compare these results with those obtained with standard HMMs, equivalent 1-state phonemic HMMs were used in the same conditions: discrete probability density functions for the emissions based on the same 60 prototype vectors are determined from the automatically segmented 20 isolated digits and no phonemic duration modeling is in-

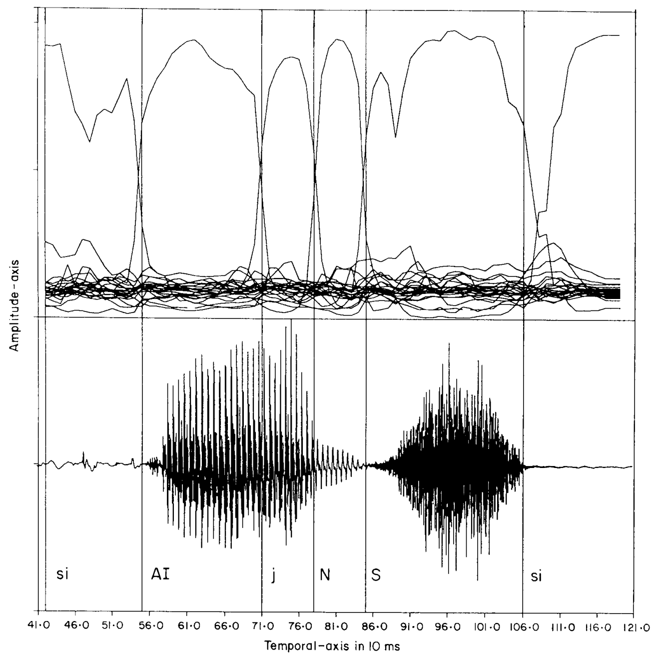

Figure 5.5: Time signal and phonemic output activations of the contextual MLP on a particular test word. Phonetic segmentation is given for information but is not used during labeling.

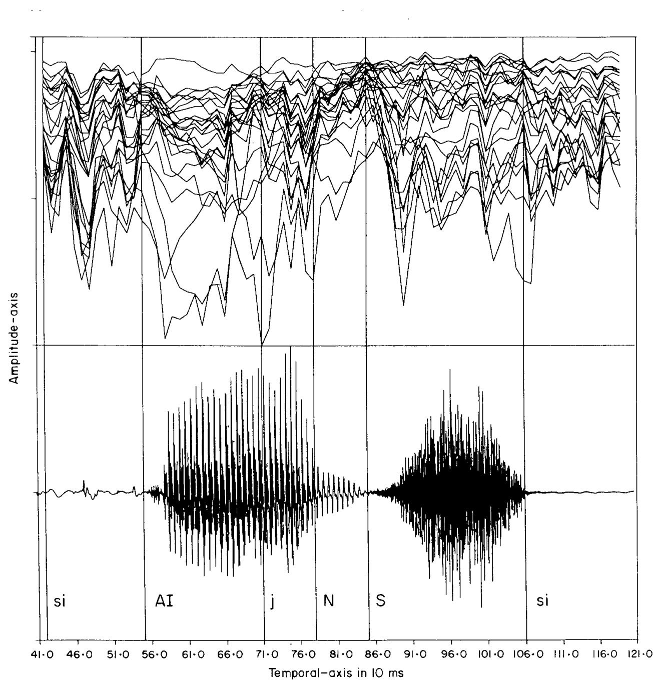

Figure 5.6: Time signal and phonemic Gaussian emission probabilities for a particular test word. Phonetic segmentation given for information.

corporated. The results on the 40 isolated words were: 6 insertions, 2 substitutions and no deletions, i.e a score of 80 % while the score drops to 70 % for the 50 connected digit strings (32 insertions, 69 substitutions and 2 deletions). Thus, a strong improvement was observed when contextual MLPs are used. This can be explained by the fact that during the training phase more information is stored by the model; more precisely, the acoustic contextual information is used and, as already mentioned in Section 4.4.1, cross-products of the inputs are memorized on the hidden units.

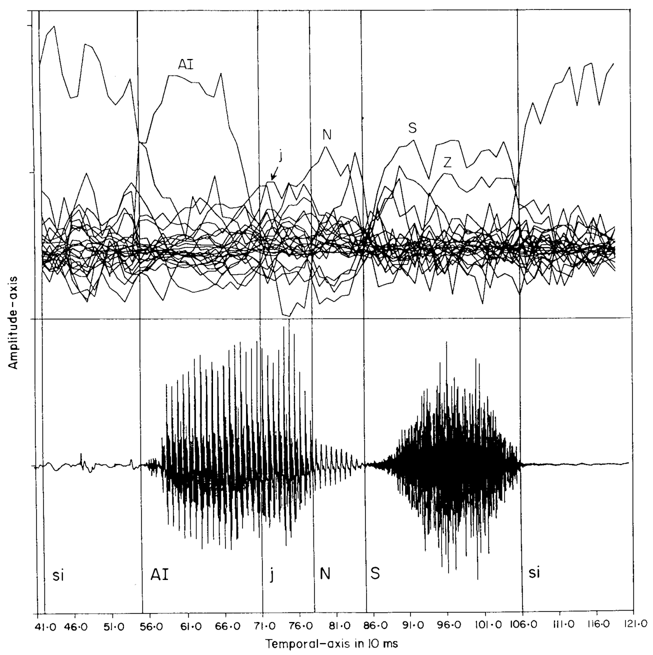

Figure 5.7: Time signal and output values of phonemic linear discriminant functions for a particular test word. Phonetic segmentation given for information.

Another test was made using the same 1-state phonemic HMMs on the same test and training data but with continuous Gaussian probability

	40 isolated words	50 strings of 7 digits	# errors
1-state discrete HMM	6I, 2S, 0D	32I, 69S, 2D	111
1-state Gaussian HMM	0I, 0S, 0D	0I, 31S, 2D	33
MLP	0I, 0S, 0D	3I, 22S, 1D	26

Table 5.1: Comparison of the recognition error rates (I=insertions, S=substitutions, D=deletions) obtained with 1-state phonemic HMMs with discrete emission probabilities, Gaussian emission probabilities, and outputs of a contextual MLP.

density functions. This test led to better results than for the discrete HMMs: no errors occur on the isolated words and no insertions, 31 substitutions and 2 deletions are observed on the digit strings. This score of 90 % correct (33 errors) was still not as good as the score for the MLP.

The recognition results are summarized in Table 5.1, where I, S and D stand for the number of insertions, substitutions and deletions.[6]

In all cases, the weak scores of HMM are due to the extreme simplicity of the models and the low amount of training material. Of course, better recognition results can be obtained with more sophisticated HMMs (several classes per phoneme, phonemic duration modeling, larger training set and embedded training). The purpose of these early tests was only to emphasize the advantages of the contextual MLPs for speech recognition in the case of very limited training material.

Figures 5.5, 5.6 and 5.7 compare the discriminant capabilities of the MLP approach (Figure 5.5) with other methods like the Gaussian classifier (Figure 5.6) used in continuous HMMs and the linear discriminant functions (Figure 5.7) as used in [Bourlard & Wellekens, 1986] and recalled in Section 4.2.2. In all these examples, the model parameters have been determined by the same set of automatically excised phonemes from two pronunciations of the 10 German digits. No iterative refinement of that segmentation has been done (because of the problem mentioned

[6]These preliminary tests have been performed around 1987 and were using extremely crude HMM models. They are certainly no longer up-to-date today but had the advantages to shown the potential (but still unclear) benefit of using MLPs for speech recognition.

in Section 5.5.3) and the resulting parameters were tested on unseen utterances (the third pronunciation of the word "eins" [AI-j-N-S]).

As described previously, Figure 5.5 represents the evolution of the 10-ms MLP output activations for binary input patterns using contextual information ($2c + 1 = 11$). In this case, it is clear that discrimination is very good (which is probably due to the nonlinear discriminant capabilities of MLPs together with contextual information – contextual information has not been used for Figures 5.6 and 5.7).

Figure 5.6 plots, for each acoustic vector x_n corresponding to 10 msec of speech signal (with no contextual information), the logarithm of the emission probabilities $p(x_n|q_k)$ [computed by (2.5)] for each of the 26 phonemic classes. The word is globally correctly recognized when the negative logarithm of the probabilities is used in a DTW procedure, despite the fact that the local discrimination is weak. Indeed, a 10 msec local labeling is based on the maximum of the 26 output values. In this simple example, most of the labels are correct, but with a very low reliability margin.

In Figure 5.7, the same representation for *linear* discriminant function outputs is used. The parameters of these functions have been obtained (with simple linear algebra) by minimization of MSE as explained in Section 4.2.2 [equation (4.5)]. In this case, the continuous acoustic vectors were used without context to reduce the size of the matrix inversion. It can be observed that the discrimination is reinforced (as one might expect) and the local labeling is improved. However, the capabilities of linear functions are still limited when compared with the nonlinear functions generated by the MLP (Figure 5.5).

In [Bourlard & Wellekens, 1989a], additional tests were reported on a speaker-dependent, continuous speech recognition database called SPICOS [Ney & Noll, 1988] (further used in Sections 6.6.1 and 7.5). In this case, the vocabulary contained about a thousand words described by a set of 50 phonemes. The training data set consisted of two sessions of 100 German sentences which were representative of the phoneme distribution in the German language. The SPICOS test set consisted of 188 sentences with 1292 words and about 7300 phonemes. The recognition vocabulary contained 918 words, and the overlap between training and recognition vocabulary was only 51 words, which were for the main part articles, prepositions and other structural words. In [Bourlard & Wellekens, 1989a; Bourlard & Morgan, 1991] a significant improvement of the phonemic labeling (at the frame as well as at the phoneme level) was reported when the proposed approach was compared with standard HMMs. However, much to our surprise, we were unable (at the time of this experiment) to extend this improvement to the word level.[7] This problem will be explained further in the following chapters. It will be shown that the MLP outputs are in fact estimates of posterior probabilities, which are proportional to the implicit estimates of the a priori probabilities of the phonemic classes (via relative frequencies in the training). In the case of Spicos, since most of the words used in the training set are different from the words used in the test set, these priors are also different (which was not the case with the isolated digits). However, this is only one problem among a number that we had when attempting to perform word recognition with an MLP in a DTW procedure (see following chapters).

5.7 Summary

For speech recognition problems, the major weakness of the connectionist formalism is its difficulty in mapping time-varying input patterns to asynchronous output patterns. In this chapter we have reviewed several ANN architectures that attempt to deal with sequential signals. These models have exhibited good performance (sometimes better than HMMs) on short, isolated speech units (like E-set). By their implicit memory, they include some kind of integration over time. However, it was still not clear from the results what these models can actually compute and how continuous speech recognition and training could be handled by neural networks only. ¿From a recognition point of view, when DTW is used for HMM decoding, it not only tackles the variability of speech pronunciation, but is also an efficient tool for connected speech recognition and segmentation. This latter property seems to be difficult to achieve with a completely connectionist algorithm. Even assuming a perfect dynamic system taking the entire past into account, the output values would still represent the scores for states (or output classes) at the current time, but would not give any idea about the underlying segmentation (as the output values alone do not carry the information needed to recover the best interpretation). For example, in the HMM approach, even if we were able to build a high order Markov model, some kind of time warping

[7] This was in late 1987. After that we scratched our heads for a while, and figured it out.

process would still be necessary.

Thus, avoiding explicit DTW for a continuous speech recognition task remains an open problem. On the other hand, it is also questionable whether it is even desirable to replace the DTW algorithm, since this process is very effective (although limited by several assumptions), and also has efficient hardware implementations [Murveit & Brodersen, 1986]. It was also shown in Sections 5.5.3 and 5.6 that the output values of the neural networks could be used as local distance in DTW. The convergence of a training process minimizing a MSE by iteratively improving the parameters of the models (discriminant functions) and the segmentation obtained from DTW has been shown in Section 5.5.3. Initial attempts to use such an iterative embedded approach was unsuccessful, and led to absurd results. However, the capabilities of MLPs to generate a better local discrimination was experimentally demonstrated in Section 5.6 on pre-segmented speech data (because of the problems reported in Section 5.5.3).

In the next chapters, we will describe our work in hybrid systems using neural networks as specific modules of HMM systems. In these cases, time alignment (using a Viterbi procedure) will still be required.

Chapter 6

STATISTICAL INFERENCE IN MLPs

One never goes so far as when one doesn't know where one is going.
– Johann Wolfgang von Goethe –

6.1 Introduction

In Chapter 3, we showed that HMMs were stochastic models that dealt efficiently with the statistical and sequential character of the speech signal, but which also suffer from several limiting assumptions that are required for tractable solutions. In Chapter 4, we discussed ANNs and showed that they had their own attractive properties; in particular, they appear to rely on fewer basic assumptions. Chapter 5 briefly reviewed the most popular ANN approaches currently used for sequence processing in general and speech recognition in particular. We concluded that none of these were able to solve CSR properly using ANNs by themselves. Given these tradeoffs, we have been interested in using ANNs to overcome some HMM drawbacks while staying within the latter's formalism. This kind of hybrid is frequently not straightforward, however; for instance, it is difficult to optimally incorporate rule-based speech knowledge in an HMM-based ASR system.[1]

A good step in integrating HMMs and ANNs is to determine a common language for the technologies. Our best guess for this unifying theme has been a probabilistic description of ANNs. We have tried, then, to get a better understanding of the kind of statistical values that ANNs can compute, and how they relate to the parameters of an HMM. In this chapter, it is shown that the output values of an MLP are estimates of *Maximum a Posteriori* (MAP) probabilities (or something more general if there is feedback from higher layers to the input field). Using an MLP with this feedback and extending the input field to include context, output probabilities can be generated that depend on a fixed temporal window on both states and observations.

This perspective showed us an important link between MLPs and HMMs, and much of the rest of this book will exploit this relationship. As with the earlier material, our emphasis is on speech recognition; also as with the earlier material, the statistical relationships are quite general, and should apply to other application areas.

In this chapter, we will assume that pre-segmented (i.e pre-classified) data are available for training the MLP, which is generally not the case for speech. However, we will see in the next chapter (Section 7.6) that the statistical interpretation developed below will allow us to derive a Viterbi-based algorithm that will do automatic segmentation of the speech data for use in the MLP.

6.2 ANNs and Statistical Inference

In this section we will prove in two different ways that, when used for classification, MLPs can generate good estimates of posterior probabilities of the output classes conditioned on the input pattern. For clarity's sake, we will consider two different cases: discrete input and continuous input MLPs. These will be examined in the context of discrete and continuous HMMs (respectively). The main idea will be to use the probability density functions (pdfs) generated by the MLP as pdfs for the possible HMM-states. Accordingly, the output classes of the MLP

[1] Of course, systems do incorporate such knowledge, but it is difficult to do this in an integrated way; most commonly, all but the coarsest knowledge about the language or the world is built into a post-processor that is loosely-coupled with the acoustic recognizer.

will be denoted by q_k and will be uniquely associated with HMM-states $q_k \in \mathcal{Q} = \{q_1, \ldots, q_K\}$.

6.2.1 Discrete Case

The acoustic vector sequence $X = \{x_1, \ldots, x_n, \ldots, x_N\}$ can be quantized (for a discrete HMM) by a front-end processor in which each acoustic vector x_n, $n = 1, \ldots, N$, is replaced by the closest (typically in the sense of Euclidian distance) prototype vector y_i selected in a predetermined finite set $\mathcal{Y} = \{y_1, \ldots, y_i, \ldots, y_I\}$ of cardinality I. The prototype vector set is usually determined independently by clustering, e.g., K-means or binary splitting on a large number of acoustic vectors. In this case, (as in Section 3.3.3) the acoustic vector sequence X is replaced by a (prototype) vector sequence $Y = \{y_{i_1}, \ldots, y_{i_n}, \ldots, y_{i_N}\}$, where $i_n \in [1, I]$ represents the label of the closest prototype vector associated with x_n.

Let q_k, with $k = 1, \ldots, K$, be the output units of an MLP associated with different classes (or HMM states). Typically, as already done in Section 5.6, each prototype vector y_{i_n} of Y will be represented at the input of the MLP as an I-dimensional binary vector with all components equal to zero except for the i_n-th one which is equal to one. This vector will be referred to as the *index vector* of x_n in the following. At time n, the input pattern of the MLP is thus the index vector of x_n. Since the MLP training needs supervision, we suppose that the class q_k associated with each input pattern is known. The training of the MLP parameters is then commonly based on the minimization of the following MSE over all the training patterns Y:

$$E = \sum_{n=1}^{N} \sum_{k=1}^{K} [g_k(y_{i_n}) - d_k(y_{i_n})]^2 \qquad (6.1)$$

where $g_k(y_{i_n})$ and $d_k(y_{i_n})$ respectively represent the observed output value and the targeted output value for the k-th output unit (associated with q_k) given the index vector of x_n at the input. When using an MLP, $g_k(\cdot)$ is calculated by the forward equation (4.11). In classification mode, the goal is to associate each input vector with a single class, and $d_k(y_{i_n})$ will be equal to 1 if the input is known to belong to class q_k and 0 $\forall k \neq \ell$. Consequently, if a particular input $y_{i_n} \in q_k$, we have $d_\ell(y_{i_n}) = \delta_{k\ell}$, for $\ell \in [1, \ldots, K]$. Expanding summations to collect all terms depending on the same y_i, (6.1) can be rewritten as:

$$E = \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{\ell=1}^{K} n_{ik} \left[g_\ell(y_i) - d_\ell(y_i)\right]^2 \tag{6.2}$$

where n_{ik} is the number of times the prototype vector y_i has been observed in class (or state) q_k. Thus, whatever the MLP topology may be,[2] i.e., the number of its hidden layers and of units per layer, the optimal output values $g_\ell^{opt}(y_i)$ are obtained by canceling the partial derivative of E with respect to $g_\ell(y_i)$:

$$\frac{\partial E}{\partial g_\ell(y_i)} = 2\sum_{k=1}^{K} n_{ik} \left[g_\ell(y_i) - d_\ell(y_i)\right] = 0$$

The optimal values of the outputs are then

$$g_\ell^{opt}(y_i) = \frac{\sum_{k=1}^{K} n_{ik} d_\ell(y_i)}{\sum_{k=1}^{K} n_{ik}}$$

In classification mode, we have $d_\ell(y_i) = \delta_{k\ell}$, and consequently:

$$g_\ell^{opt}(y_i) = \frac{\sum_{k=1}^{K} n_{ik}\delta_{k\ell}}{\sum_{k=1}^{K} n_{ik}} = \frac{n_{i\ell}}{\sum_{k=1}^{K} n_{ik}}, \forall \ell \in [1, \ldots, K] \tag{6.3}$$

This global minimum is only related to the error criterion and is independent of the MLP topology, i.e., the number of its hidden layers and of units per layer. The optimum $g_\ell(y_i)$'s obtained from the minimization of the MLP criterion are thus estimates of *a posteriori* (Bayes) probabilities [Fukunaga, 1972]:

$$g_\ell^{opt}(y_i) = \hat{p}[q_\ell|y_i], \forall \ell \in [1, \ldots, K] \tag{6.4}$$

and, by definition, the constraint

$$\sum_{k=1}^{K} g_k^{opt}(y_i) = \hat{p}[q_k|y_i] = 1$$

is automatically verified. However, these optimal values can only be reached if the MLP has enough parameters, does not get stuck in a local

[2] One must be a bit careful here. If the network is recurrent, then the output depends not only on the current input but on previous inputs, so that these equations require a bit more detail. Assume for now that we speak only of feedforward networks.

minimum during the training (see Section 6.4.1), and is trained long enough to reach the global minimum.

Replacing in (6.2) the target outputs $d_\ell(y_i)$ by the optimal values (6.3) provides a new criterion where the target outputs depend on the current vector and the possible outputs:

$$E^* = \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{\ell=1}^{K} n_{ik} \left[g_\ell(y_i) - \frac{n_{i\ell}}{\sum_{k=1}^{K} n_{ik}} \right]^2 \tag{6.5}$$

and it is clear [by canceling the partial derivative of E^* versus $g_\ell(y_i)$] that the lower bound for E^* is reached for the same optimal outputs as (6.3). The minimum value of the criterion is now equal to zero, thus providing a very useful control parameter during the training phase. It should be noted that (6.5) is consistent with the suggestion that "training of an adaptive machine is best done *not* with the *correct* output (1 or 0) being supplied, but with an expert's estimate of the (posterior) probabilities of the possible output states" [MacKay, 1987].

As these results follow directly from the minimized criterion and *not from the topology of the network*, it is interesting to notice that the same optimal values (6.3) may also be obtained from other criteria as, for instance, the entropy [Hinton, 1987] or relative entropy[3] [Solla et al., 1988] of the targets with respect to the outputs. For instance, in the case of relative entropy, criterion (6.1) becomes:

$$E_e = \sum_{n=1}^{N} \sum_{k=1}^{K} \left[d_k(y_{i_n}) \ln \frac{d_k(y_{i_n})}{g_k(y_{i_n})} + (1 - d_k(y_{i_n})) \ln \left(\frac{1 - d_k(y_{i_n})}{1 - g_k(y_{i_n})} \right) \right] \tag{6.6}$$

and canceling its partial derivative versus $g_\ell(y_i)$ yields the optimal values (6.3) again. However, in this case, the optimal outputs actually correspond to $E_{e,min} = 0$.

Since these results are independent of the topology of the trained system, they also remain valid for the linear discriminant functions described in Chapter 4. However, since we are then limited to a small number of parameters, it is quite likely that the optimal values (6.3) will not be reached, even though the local error minimum is also the global one for this case.

[3] Also referred to as Kullback-Leibler divergence (or distance) [Kullback & Leibler, 1951; Kullback, 1959]

Compared with linear discriminant functions, the advantages of using MLPs to estimate Bayes probabilities are the following:

1. It is easy to control (i.e., increase or decrease) the number of parameters.
2. MLPs use nonlinear discriminant functions.
3. The sigmoid function usually used at the output of the MLP is an approximation of a decision threshold. As a consequence, MLPs will both estimate Bayes probabilities (due to the minimized criterion) and minimize the classification error rate.

6.2.2 Continuous Case

In the previous section it has been shown that, in the case of discrete inputs, the output values of an MLP will be estimates of the *a posteriori* Bayes probabilities. Although the same proof could be applied to continuous inputs, another proof (initially presented in [Richard & Lippmann, 1991]) will be given here which is more general and which allows us to better understand the behavior of such systems.

We first reformulate the cost function (6.2), which was explicitly written for the discrete case, in terms of continuous inputs. The MSE in (6.2) can be expressed as follows:

$$E = \sum_i \frac{n_i}{N} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \frac{n_{ik}}{n_i} \; [g_\ell(y_i) - d_\ell(y_i)]^2 \tag{6.7}$$

where $d_\ell(y_i) = \delta_{k\ell}$ if $y_i \in q_k$. In the case of continuous inputs, the sequence of acoustic vectors is no longer quantized and (6.7) becomes:

$$E = \int p(x) \sum_{k=1}^{K} \sum_{\ell=1}^{K} p(q_k|x) \, [g_\ell(x) - d_\ell(x)]^2 \; dx \tag{6.8}$$

where $d_\ell(x) = \delta_{k\ell}$ if $x \in q_k$. Since $p(x) = \sum_{i=1}^{K} p(q_i, x)$, we have:

$$E = \int \sum_{i=1}^{K} \left[\sum_{k=1}^{K} \sum_{\ell=1}^{K} [g_\ell(x) - d_\ell(x)]^2 \, p(q_k|x) \right] p(q_i, x) \, dx$$

After a little more algebra, adding and subtracting $p^2(q_\ell|x)$ in the previous equation leads to:

$$\begin{aligned}
E &= \int \sum_{i=1}^{K} \left[\sum_{\ell=1}^{K} \left(g_\ell^2(x) - 2g_\ell(x)p(q_\ell|x) + p^2(q_\ell|x) \right) \right] p(q_i, x)\, dx \\
&\quad + \int \sum_{i=1}^{K} \left[\sum_{\ell=1}^{K} \left(p(q_\ell|x) - p^2(q_\ell|x) \right) \right] p(q_i, x)\, dx \\
&= \int \sum_{i=1}^{K} \left[\sum_{\ell=1}^{K} (g_\ell(x) - p(q_\ell|x))^2 \right] p(q_i, x)\, dx \qquad (6.9) \\
&\quad + \int \sum_{i=1}^{K} \left[\sum_{\ell=1}^{K} (p(q_\ell|x)(1 - p(q_\ell|x))) \right] p(q_i, x)\, dx
\end{aligned}$$

Since the second term in the previous equation is independent of the network outputs, minimization of the squared-error cost function is achieved by choosing network parameters to minimize the first expectation term. However, the first expectation term is simply the mean squared-error between the network output $g_\ell(x)$ and the posterior probability $p(q_\ell|x)$. Minimization of (6.8) is thus equivalent to minimization of the first term of (6.9), i.e., estimation of $p(q_\ell|x)$ at the output of the MLP. This generalizes the proof given in Section 6.2.1. It also shows that the discriminant functions obtained by minimizing the MSE retains the essential property of being the best approximation to the Bayes probabilities *in the sense of mean square error.*

A similar proof was given in [Richard & Lippmann, 1991] for the entropy cost function.

6.3 Recurrent MLP with Output Feedback

In the preceding section, each input pattern (acoustic vector) was classified independently of the preceding classifications. Consequently, the sequential character of the speech signal is not taken into account by the MLP. The system has no short-term memory from one classification to the next one, and there are no constraints on the successive classifications. This is not the case in HMMs where, given the topology of the Markov models, only some class sequences (at the state level) and some phoneme sequences (at the word level) are allowed by the topology of the HMM. This problem may be circumvented by supplying some information about the preceding classification to the input field of the MLP. This leads to a form of recurrent network in which some representation of the output vector at time n is used in additional feedback units at the input to classify the input at time $n+1$. As we will see in Chapter 7, this model has an interesting relationship with HMMs.

In the following, we will only consider the case of discrete features. However, as with the feedforward networks, similar conclusions remain valid for continuous features.

Sequential classification must rely on the previous decisions, but our goal remains the association of the current input vectors with their own classes. For each current binary input vector y_i, an MLP achieving this task will generate, for each output unit $\ell \in [1, \ldots, K]$, a value $g_\ell(y_{i_n}, q_k^-)$, $(k \in [1, \ldots, K])$ depending not only on the input pattern y_{i_n} at time n, but also on the class q_k in which the preceding input vector was classified. In this case, criterion (6.1) becomes:

$$E = \sum_{n=1}^{N} \sum_{\ell=1}^{K} \left[g_\ell(y_{i_n}, q_k^-) - d_\ell(y_{i_n}) \right]^2 \tag{6.10}$$

A convenient way to generate the $g_\ell(y_i, q_k^-)$ is to modify its input by extending it with an extra input vector containing the delayed output values (or a binary representation of them) taken at time $n-1$ (as already represented in Figure 5.3). Since output information is fed back in the input field, such an MLP has a recurrent topology. However, compared with other recurrent models proposed in speech processing (see Chapter 5), the main advantage of this topology is the ability to supervise the feedback information during training. Indeed, since the training data consist of consecutive labeled speech frames, the correct sequence of output states is known and the training can be supervised by providing the correct information.

Since the goal is still to associate each input vector with a single class, the target outputs associated with an input pattern $y_{i_n} \in q_k$ are $d_\ell(y_{i_n}) = \delta_{k\ell}$, for $\ell \in [1, \ldots, K]$. The target outputs $d_\ell(y_{i_n})$ only depend on the current input vector y_{i_n} and the corresponding output unit, and not on the classification of the previous one. The difference between criterion (6.10) and that of a memoryless machine is the addition of a variable q_k^- to the mapping function g to take account of the previous decision. Collecting all terms depending on the same prototypes, (6.10) can be rewritten as:

$$E = \sum_{i=1}^{I} \sum_{k=1}^{K} \sum_{\ell=1}^{K} \sum_{m=1}^{K} n_{ik\ell} \left[g_m(y_i, q_k^-) - d_m(y_i) \right]^2 \tag{6.11}$$

where $n_{ik\ell}$ represents the number of times the particular input y_i has been assigned to q_ℓ, while the previous vector was known to belong to class q_k. As was done in Section 6.2.1, it is easy to show that the optimal output values $g_m^{opt}(y_i, q_k^-)$ obtained by canceling the partial derivative of E with respect to $g_m(y_i, q_k^-)$ are:

$$g_m^{opt}(y_i, q_k^-) = \frac{n_{ikm}}{\sum_{\ell=1}^{K} n_{ik\ell}} = \hat{p}(q_m | q_\ell^-, y_i), \forall \ell, m \in [1, \ldots, K] \tag{6.12}$$

which is nothing but the discriminant local probabilities defined in (3.45). Thus, the optimal $g_m(y_i, q_k^-)$'s obtained from the minimization of the MLP criterion are again estimates of posterior probabilities, but now taking the previous class into account. By definition, we then have

$$\sum_{m=1}^{K} g_m^{opt}(y_i, q_k^-) = 1$$

In the case of continuous input x_n, this kind of recurrent MLP will approximate probabilities like $p(q_\ell^n | q_k^{n-1}, x_n)$ since the proof given in Section 6.2.2 still holds.

However, this conclusion is only valid if:

1. The conditions listed in Section 6.2 are verified (i.e., MLP trained in classification mode, enough parameters, no local minimum).

2. During classification the correct output (i.e., the correct previous class) is provided to the feedback units (i.e., a binary vector simply coding the previous class as given by the training set); of course, in this case the network is not truly recurrent. Another solution is to provide the actual value of the observed outputs at the feedback units, in which case we estimate something different than (6.12); this will be discussed further in Section 7.2.4.

As discussed above for training, two approaches are also possible for the feedback units during recognition:

1. The feedback units are provided with the different possibilities for the previous classes (as forced by an underlying HMM topology – see Chapter 7). In this case, of course, the network will have to be computed several times, i.e., one time per possible previous state. A possible way to save CPU time is to use table-driven approaches to incorporate partial results.[4]

2. The feedback units are simply provided with the actual output values at the previous time frame – this will be discussed further in Section 7.2.4.

However, these MLPs are going to generate probabilities that are not the ones used in standard HMMs, which use likelihoods (see Chapter 3). In Chapter 7, it will be shown that it is possible to reformulate the HMM approach in terms of such probabilities. We repeat here for the sake of emphasis: these optimal values can only be reached if the MLP has enough parameters, and if local minima are avoided.

Finally, the same kind of recurrent MLP could also be used to estimate higher order local probabilities in which the MLP output values can be dependent on several preceding states, not only the previous one. This is easily implemented by extending the input field to include representations of these preceding classifications.

The theoretical results shown above follow directly from the minimized criterion, not from the topology of the model.[5] The same optimal values may be reached using other criteria, such as the entropy or relative entropy of the targets with respect to the output [Bourlard & Wellekens, 1990; Bourlard et al., 1990].

[4] This kind of approach is discussed in Section 9.4 for the case of efficient implementations of MLPs that generate estimates of posteriors dependent on phonetic context, e.g., triphones.

[5] As noted previously, if the model topology implements new dependencies, (for instance, a dependency on past values of the input), then this dependency must appear in the expression for the conditional probability that the net generates. However, given a particular dependency, the network will generate some estimate of the corresponding conditional probability regardless of the number of hidden layers, units, nonlinear functions, etc. How good the estimate is, is another question.

6.4 Practical Implications

6.4.1 Local Minima

The optimal solutions (6.3) and (6.12), obtained by canceling the partial derivative of the error criterion (LMS or entropy) with respect to the output vector $g(x_n)$, are the optimal sets of values that could be reached by the EBP algorithm. As noted previously, in this procedure a gradient estimate is used to cancel the partial derivatives of the error versus the weight parameters W:

$$\frac{\partial E}{\partial w_{ij}} = (\nabla_g E)^T \frac{\partial g(x_n)}{\partial w_{ij}}, \forall i, j \tag{6.13}$$

where T represents the transpose operation. Thus, a minimum in the output space ($\nabla_g E = 0$) is also a minimum in the parameter space ($\partial E / \partial w_{ij} = 0$, $\forall i, j$). However, it is also clear that $\partial E / \partial w_{ij} = 0$, $\forall i, j$, does not necessarily lead to $\nabla_g E = 0$; if the latter is not true, the network has converged to a local minimum of the error function.

6.4.2 Network Outputs Sum to One

Network outputs should sum to one for each input value if outputs accurately estimate posterior probabilities. However, if the network converges to a local minimum, it is no longer guaranteed that the network outputs estimate Bayesian probabilities. For the MLP network, the value of each output will in any case remain between zero and one because of the sigmoidal functions typically used. However, the criterion used for training did not require the outputs to sum to one. An elegant way to circumvent this problem is to replace the classical sigmoidal function applied at the output units by a "softmax" function [Bridle, 1990a] defined as:

$$g_k(x_n) = \frac{\exp\left(f_k(x_n)\right)}{\sum_{\ell=1}^{K} \exp\left(f_\ell(x_n)\right)} \tag{6.14}$$

where $f_k(x_n)$ is the value of output unit q_k prior to the nonlinearity for an input vector x_n. This function generalizes the sigmoid and has a nice relationship with the Gibbs distribution [Bridle, 1990a].

Nevertheless, as shown in [Richard & Lippmann, 1991] and in the experiments reported in Section 6.6.4, the outputs of the trained MLP network should sum to a value that is close to one. As such, normalization techniques proposed to ensure that the outputs of an MLP are true probabilities [Bridle, 1990a; El-Jaroudi & Makhoul, 1990; Gish, 1990] may be unnecessary. However, in some of our experiments, we have observed a small performance advantage to using a softmax function at the output layer.

6.4.3 Prior Class Probabilities and Likelihoods

Since the network outputs approximate Bayesian probabilities, $g_k(x_n)$ is an estimate of

$$p(q_k|x_n) = \frac{p(x_n|q_k)p(q_k)}{p(x_n)}$$

which implicitly contains the *a priori* class probability $p(q_k)$. It is thus possible to vary *a priori* class probabilities during classification without retraining, since these probabilities occur only as multiplicative terms in producing the network outputs. As a result, class probabilities can be adjusted during use of a classifier to compensate for training data with class probabilities that are not representative of actual use or test conditions [Richard & Lippmann, 1991].

Also, (scaled) likelihoods $p(x_n|q_k)$ can be obtained by dividing the network outputs $g_k(x_n)$ by the relative frequency of class q_k in the training set, which gives us an estimate of:

$$\frac{p(x_n|q_k)}{p(x_n)}$$

During recognition, the scaling factor $p(x_n)$ is a constant for all classes and will not change the classification. There are several reasons why this somewhat indirect method for estimating (scaled) likelihoods is attractive:

1. MLPs are well matched to discriminative objective functions and scaled likelihoods are thus derived in a discriminative fashion.

2. Although an MLP is a parametric model, a large network defines an extremely flexible set of functions. Thus only weak assumptions are made about the input statistics.

3. A maximum likelihood estimation of HMM parameters requires the assumption of conditional independence of outputs. As shown in Section 6.5, MLPs can model correlations across an input window of adjacent frames

It could be argued that, when dividing by the priors, we are using a scaled likelihood which is no longer a discriminant criterion. However, this is not true since discrimination is important only during training. During recognition the discriminant term (see, for example, Sections 3.6 and 4.5) becomes independent of the class and can be dropped.

6.4.4 Priors and MLP Output Biases

Given the previous conclusions, and using a sigmoid transfer function at the output of the MLP, the activation of the output unit q_k given the input x_n before the sigmoid is

$$h_k(x_n) = \log \frac{p(q_k|x_n)}{1 - p(q_k|x_n)}$$

which is usually referred to as *log odds* in statistics. It is tempting to identify the weighted sum of hidden units outputs as the data part and the bias as the prior part – or, more precisely, the log odds of the priors, i.e., $\log p(q_k)/(1 - p(q_k))$. A similar observation is also valid for the softmax function, in which case we would like to associate the biases with the priors of the classes. However, this relation is too facile. Even in the case of linear discriminant functions, the priors not only contain information about the priors, but also about the mean of each class (see Section 4.2.3).

However, in all our experiments, observation of the output biases of a trained network did indeed show a correlation with the log (odd) priors (relative frequencies) of the classes. As already mentioned in Section 4.3.5 this has been systematically used later to improve convergence of the EBP algorithm by initializing biases to the log (odds) of priors.

6.4.5 Conclusion

The previous conclusions have been obtained for MLPs used for classification with "one-from-K coding," i.e., one output for each class, with all targets equal to zero except for the correct class where it is unity. If there are enough parameters in the system and if the training does not get stuck at a local minimum, the output values of the MLP will approximate posterior probabilities. Section 6.6 will show some empirical evidence for this assertion.

The outputs of the MLP thus approximate MAP probabilities, which are known to lead to the optimal classification; they are discriminant by nature and minimize the classification error rate. However, these probabilities do not match the HMM formalism that requires likelihoods; a new HMM model that can accommodate these probabilities will be presented in Chapter 7. These probabilities are also related, by Bayes law, to the emission probabilities of standard HMMs. If they are divided by the prior probabilities of the classes observed on the training set, they may be used as emission probabilities in standard HMMs. In the following sections and chapters, we will show that this approach has several advantages over standard HMMs.

6.5 MLPs with Contextual Inputs

The theoretical results presented in Section 6.2 and 6.3 remain valid if the input field of the MLP is provided not only with the current acoustic vector x_n (or its associated index vector in the case of discrete features) but also with a contextual window of width $2c+1$ centered around x_n. Contextual inputs were used in the ERIS system [Marcus, 1981; Marcus, 1985] where the classification of acoustic vectors was based on discriminant principles (responsible "demons" for the recognition of one particular class and the rejection of all other stimuli as "nonwords") applied on state-pair vectors. However, this approach was difficult to extend to larger contexts. In this case, a NETtalk-like MLP[6] has also been used to classify 10-ms acoustic vector strings into phoneme strings, incorporating the context from the surrounding vectors [Bourlard & Wellekens, 1987; Bourlard & Wellekens, 1989c].

Figure 5.1 shows the schematic arrangement of this system, characterized by two layers of perceptrons (hidden and output layers) computing the classification of the input field. The input consists of $2c+1$ groups of units representing the sequence $X_{n-c}^{n+c} = \{x_{n-c}, \ldots, x_n, \ldots, x_{n+c}\}$ of acoustic vectors. During training, the desired output of the network is the correct phoneme associated with the center or "current" acoustic vector in a particular left and right context. The acoustic vectors are stepped through the contextual window time slot by time slot, and the matrix of weight parameters is adjusted by the error back-propagation algorithm (for more details about the training procedure, see Section 6.6.2).

[6]NETtalk was initially described in [Sejnowski & Rosenberg, 1987], and was an MLP trained to map written text into phoneme strings.

Since the output values of the MLP are estimates of the posterior probabilities of the output classes conditioned on the input, this kind of network will thus estimate, on each output unit q_k, the probability

$$p(q_k|X_{n-c}^{n+c}) \tag{6.15}$$

which is usually difficult to estimate by standard statistics given the large number of parameters. In Section 6.6, this kind of network will be used and tested, and will be shown to lead to better performance than some standard statistical approaches. Because ANNs are capable of incorporating multiple constraints and finding optimal combinations for classification, input features do not need to be treated as independent. More generally, there is no need for strong assumptions about statistical distributions and independence of the input features.

The results presented in Section 6.3 for recurrent MLPs with output feedback also still hold with a modified input taking the context into account. The proposed architecture represented in Figure 5.3 is a NETtalk-like MLP as in Figure 5.1, augmented by a direct feedback of the outputs associated with the previous input frames. The feedback is implemented by adding extra input units that reflect the delayed output values. This architecture was originally proposed by Jordan in [Jordan, 1986]. Since the training data consist of consecutive labeled speech frames, the correct sequence of output states is known and training with the correct feedback values is possible. According to the theory of Sections 6.2 and 6.3, the network outputs will estimate the following Bayesian probabilities at the end of the training procedure:

$$p(q_k^n|X_{n-c}^{n+c}, q_\ell^{n-1}) \tag{6.16}$$

which will be shown in Chapter 7 (Section 7.2) to be the basis of "Discriminant HMMs." However, in the experiments described in this book, this recurrent architecture will not be used.

Of course, the number of weights increases with the width of the contextual window, and thus a large amount of training data is required for training. However, the generalization properties of the MLP already discussed in Section 4.4 (in the form of the generation of cross-products by the nonlinearities of the hidden units) play the role of the interpolation required by HMMs. In particular, as opposed to standard discrete HMMs, an MLP will not generate a zero at the output when it is excited by an input vector never observed in the training set. Moreover, it has been shown that MLPs appear to be able to achieve better statistical pattern recognition performance than standard classifiers when trained on undersampled pattern spaces [Niles at al., 1989].

6.6 Classification of Acoustic Vectors

The goal of this section is to show empirically on a real task (phonetic classification of vector-quantized acoustic vectors from a larger database of speaker-dependent continuous speech) that an MLP probability estimator with contextual inputs can provide better frame level classification than a standard statistical approach. In this framework, different issues related to MLP training, quality of the probability estimates and advantages of this approach will be discussed. Because of problems that will be discussed in Sections 7.2.5 and 7.3, most of the MLPs considered in the rest of this book will not have feedback, and only probabilities like (6.15) will be estimated.

6.6.1 Experimental Approach

In the experiments described below, done in 1988, we used a continuous speech, speaker-dependent German database called SPICOS [Ney & Noll, 1988]. The speech signal was sampled at a rate of 16 kHz, and 30 points of smoothed, "mel-scaled" logarithmic spectra (over bands from 200 to 6400 Hz) were calculated every 10-ms from a 512-point FFT over a 25-ms window. The mel spectrum and the energy were vector-quantized to pointers into a single speaker-dependent table of prototypes.

Two independent sets of vocabularies for training and test were used. The training data-set consisted of two sessions of 100 sentences per speaker. These data were segmented into regions corresponding to 50 phonemic classes. However, as the phonetic segmentation of the test set was not available, only the first session of the training set was used for training the MLP, while the other one was used for testing the generalization capabilities and also as the stopping criterion (cross-validation). The test set consisted of one session of 200 sentences per speaker. The recognition vocabulary contained 918 words, including the "silence" word. The overlap between training and recognition was 51 words, which were mostly articles, prepositions and other structural words. The acoustic vectors were coded using an alphabet of 132 prototype-vector labels.

These prototype vectors were calculated from the training data by using a standard cluster-analysis technique (K-means).

In Table 6.1, results obtained with Maximum Likelihood Estimates (MLE) and Maximum a Posteriori (MAP) probabilities are given. In these cases, the parameters have been obtained by standard methods for estimating discrete probabilities (i.e., simply by counting). Since we know the phonemic transcription and segmentation of the training set, we can count the frequencies n_{ik} of observation of label y_i, $i = 1, \ldots, 132$ within a state q_k. In our case, each phoneme is associated with a single state and, consequently, $k = 1, \ldots, 50$. The estimate of the likelihood (MLE in Table 6.1) of state q_k is then given by:

$$p(y_i|q_k) = \frac{n_{ik}}{n_k} \simeq \frac{n_{ik} + \epsilon}{I\epsilon + n_k} \tag{6.17}$$

where n_k is the overall frequency of phoneme k, I the total number of prototype vectors y_i, and ϵ a smoothing constant to avoid estimated values of $\mu_k = 0$. The estimate of the MAP probability is then given by:

$$p(q_k|y_i) = \frac{n_{ik}}{n_i} \simeq \frac{n_{ik} + \epsilon}{K\epsilon + n_i} \tag{6.18}$$

where n_i is the overall frequency of prototype y_i in the training set, K the total number of states q_k, and ϵ the smoothing constant. For comparison, results obtained with a Gaussian classifier described by a full covariance matrix for each class are also given in Table 6.1 ("Full Gaussian"). In this case the results were much better, perhaps because the continuous mel-spectra were classified directly without losing any information through the vector quantization process.

6.6.2 MLP Approach, Training and Cross-validation

As shown in Sections 6.2 and 6.3, the MLP can at best approximate Bayes (MAP) probabilities. The MLP is potentially preferable to counting as in (6.17) and (6.18), because it generates interpolated estimates when there is insufficient training data for the input space, e.g., when the input is highly dimensioned through the use of multiple frames as contextual input. This fact is clearly illustrated in the following experiments.

Vector-quantized mel spectra were used as binary input to a hidden layer. Multiple input frames provided context to the network. While the

	training set (26767 patterns)	**test set** (27702 patterns)
Full Gaussian	65.1	64.9
MLE	45.9	44.8
MAP	53.8	53.0

Table 6.1: Phonetic classification rates at the frame level obtained by standard approaches. "Full Gaussian" refers to the case of one Gaussian with full covariance matrix per phoneme, "MLE" refers to the case of one discrete likelihood density per phoneme estimated by counting, and "MAP" refers to the case of one discrete posterior probability density estimated by counting.

size of the output layer was kept fixed at 50 units, corresponding to the 50 phonemes to be recognized, the width of the contextual input and the number of hidden units were varied. The acoustic vectors were coded as one of 132 prototype vectors by a simple binary vector with only one bit "on," so the input field contained $132 \times a$ bits where a represents the number of frames in the input field. In this case, the total number of possible inputs was equal to 132^a. There were 26767 training patterns and 26702 independent test patterns. Of course, in the case of contextual inputs, this represented only a small fraction of the possible inputs, so that generalization was potentially difficult.

Training was done by the EBP algorithm [Werbos, 1974; Rumelhart et al., 1986a], first minimizing an entropy criterion [Hinton, 1987; Solla et al., 1988] and then finally tuned by least-mean-square error. In each iteration, the complete training set was presented, and the parameters were updated after each training pattern. To avoid overtraining of the MLP, improvement on the test set was checked after each iteration. If the classification rate on the test set was decreasing, the adaptation parameter of the gradient procedure was decreased; otherwise it was kept constant. After several reductions of learning rate, performance on the test set ceased to improve and training was stopped. In another experiment, this approach was checked by splitting the data in three parts: one for the training, one for the above cross-validation, and a third one absolutely independent of the training procedure for final testing.

No significant difference was observed between classification rates for the cross-validation and test data. The important idea in this procedure was that we stopped iterating by any one particular criterion when that criterion was leading to no new cross-validation set performance. This appeared to ameliorate the effects of over-fitting that had been observed in our earlier experiments, and greatly improved classification for frames of continuous speech.[7]

6.6.3 MLP Results

Results obtained for different MLP architectures are given in Table 6.2; here "MLPa×b-c-d" stands for an MLP with a blocs of b (binary) input units, c hidden units and d output units. The size of the output layer was kept fixed at 50 units, corresponding to the 50 phonemes to be recognized. For the binary input case, b is the number of prototype vectors (equal to 132 in our case). If c is missing, there are no hidden units. Results reported in Table 6.2 clearly show that it is possible to improve the discrete-input classification rates (at the frame level) over those obtained by a classical approach (e.g., MLE). This was done by providing context to the network, which is a potential advantage of the MLP. For simple relative frequency (counting) methods, it is not possible to use contextual information, because the number of parameters to be learned would be too large. Therefore, in Table 6.1 and MLE in Table 6.2, the input field was restricted to a single frame. This restriction explains why the Bayes classifier (MAP, in Table 6.1), which is inherently optimal for a given pattern classification problem, is shown in Table 6.2 yielding a lower performance than the potentially suboptimal MLPs. Frame performance is also shown for the cases where the MLP outputs were divided by the respective *a priori* class probabilities (see "outputs/priors" in Table 6.2). While this generally degraded frame classification performance, we believed that it might lead to improved word recognition. This was later verified, as described in Chapter 7.

6.6.4 Assessing Bayesian Properties of MLPs

For some simple examples, we have shown empirically that the MLP is estimating MAP probabilities. In Table 6.3 we report the results obtained with a fixed contextual input window (9 frames) for a hidden

[7] See Chapter 12 for further discussion of cross-validation.

	training set (26767 patterns)	**test set** (27702 patterns)
MLE	45.9	44.8
MLP5×132-20-50	65.5	59.0
outputs/priors	60.2	51.7
MLP9×132-5-50	62.8	54.2
outputs/priors	61.5	51.9
MLP9×132-20-50	75.7	62.7
outputs/priors	72.1	57.5
MLP9×132-50-50	86.4	61.4
MLP9×132-200-50	86.9	59.4
MLP9×132-50	76.9	65.0
outputs/priors	67.7	54.5
MLP15×132-50-50	83.6	64.2
outputs/priors	86.8	64.9
MLP21×132-20-50	93.0	64.0
outputs/priors	89.7	59.1
MLP21×132-50-50	95.0	67.7
outputs/priors	95.4	66.1
MLP21×132-50	92.6	68.6
outputs/priors	87.8	62.7
MLP25×132-20-50	92.8	62.7

Table 6.2: Phonetic classification rates at the frame level obtained from different MLPs, compared with MLE. "MLPa×b-c-d" stands for an MLP with a blocs (width of context) of b (binary) input units, c hidden units and d output units. The size of the output layer was kept fixed at 50 units, corresponding to the 50 phonemes to be recognized.

layer which varied from 5 to 200 units. The numbers in parentheses give the average sum, over all the training or test patterns, of the MLP output values. Since these outputs should approximate MAP probabilities, their sum should be approximately 1. The average error between the output values obtained with the MLP with no contextual input (MLP1×132-50-50) and the actual MAPs, which can be obtained by counting in the case of no contextual inputs, is also reported: for the training and the test sets, this is equal to 2.78×10^{-4} and 2.93×10^{-4}, respectively, and the standard deviation is 1.15×10^{-2} in both cases, which leads to the confidence interval:

$$p(|g_k(y_i) - p(q_k|y_i)| > 0.04) < 0.001$$

using the standard assumption of normality. In this particular case (the only one we can compare with), it can also be observed that the MLP solution converges to the optimal MAP performance (53.5% and 53.8% for the training set and 52.7% and 53.0% for the test set). All these results clearly suggest that the training did not get stuck in a very suboptimal local minimum (since the optimal global minimum can be proven to correspond to Bayes probabilities at the output of the MLP). Therefore, we infer that an MLP can be useful in estimating Bayes probabilities associated with acoustic vectors in a temporal context, which is too large for the training of a classical HMM.

As a final check of the relationship of the MLP outputs to Bayes probabilities, Figure 6.1 shows the histogram of MLP outputs to the fraction of frames correct for each output that is in a given bin range. In fact, each data point in Figure 6.1 represents the average probability of being correct over all the (training or cross-validation) input patterns for the MLP outputs in each of the possible activation bins. This fraction (i.e., the probability of being correct) ideally corresponds to the Bayes probability [Duda & Hart, 1973]. For a perfect Bayes classifier, the Bayes probability should be equal to the probability of being correct (i.e., the diagonal in Figure 6.1).

It is evident from the figure that the higher MLP outputs are extremely good matches to the desired probability. However, the lower values are not as good an estimate. This is due to the fact that, by equation (6.9), the MLP outputs approximate posterior probabilities according to a mean square criterion implicitly using the squared error between the observed output and the actual posterior probabilities. In this case, it is clear that the high probabilities will contribute more to the error function and, as a consequence, will be better estimated than the lower probabilities. It can be shown that this conclusion remains valid for the entropy or relative entropy criterion.

Figure 6.1 also show that Bayes probabilities are underestimated for low values of MLP outputs, while high values of the MLP are overestimates of these probabilities. Were the MLP a perfect Bayes estimator, points above the diagonal would suggest that the MLP will perform better than what can be achieved from the optimal Bayes classifier. In the case of a practical measurement, points above the diagonal simply indicate that the MLP outputs should have been higher and, ideally, equal to the output probability given by the point of the diagonal associated with the observed "fraction correct".

For the training set, the direction of this mismatch can perhaps be explained by the fact that the MLP outputs tend (in the limit of infinite parameters and limited training data) towards 1's and 0's, since the input space is highly-dimensioned and continuous, and therefore highly probably to be non-overlapping for the limited training data. Therefore, the high values will tend to be too high and the low values will tend to be too low. In the limit, these values would be the true Bayes probabilities and there would be no mismatch. However, with finite training data, the MLP outputs do not actually minimize the error rate on the test data. This is the reason why cross-validation is important during training. Accordingly, Figure 6.1 shows that the match to the ideal diagonal is better at high values for the training set than for the cross-validation set, while the opposite is true for low MLP outputs.

6.6.5 Effect of Cross-Validation

In Table 6.3, it is also interesting to notice that large values for the parameterization ratio (i.e., number of parameters divided by the number of training measurements) only corresponds to a slight degradation of generalization performance (3.3% over a factor of 10 in number of parameters). This is due to the fact that cross-validation was used during training of the MLP. In this case the iterative estimation process was stopped when generalization degraded for an independent data set (cross-validation) [Morgan & Bourlard, 1990a], which explains the insensitivity of test set classification scores to the net size. This is further discussed in Chapter 12, where it is shown experimentally that the use

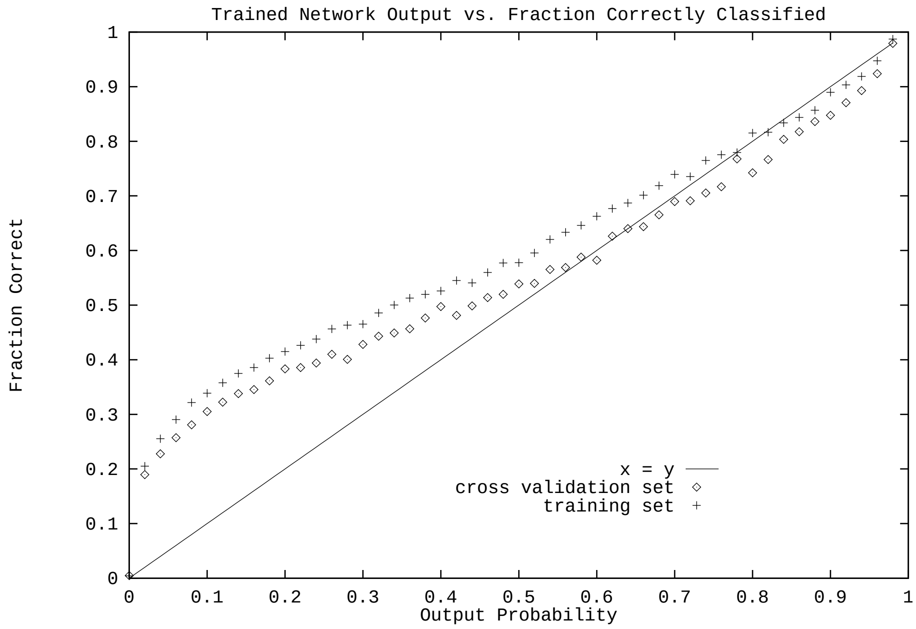

Figure 6.1: Histogram showing relationship between MLP outputs and percentage correct for each bin. This was generated by collecting statistics from a net with 9 frames of 26 continuous input parameters for a total of 234 inputs, and a 500-unit hidden layer, over the patterns from 1750 Resource Management speaker-independent training sentences and 500 cross-validation sentences.

# hidden units	R	training	test
MLP9×132-5-50	.23	62.8 (1.010)	54.2 (1.012)
MLP9×132-20-50	.93	75.7 (1.030)	62.7 (1.035)
MLP9×132-50-50	2.31	86.4 (1.018)	61.4 (1.000)
MLP9×132-200-50	9.3	86.9 (1.053)	59.4 (0.995)
MLE	.25	45.9	44.8
MAP	.25	53.8	53.0
MLP1×132-50-50	.34	53.5 (1.011)	52.7 (1.012)

Table 6.3: Phonetic classification rates at the frame level obtained from contextual MLPs, compared with standard likelihoods (MLE) and *a posteriori* probabilities (MAP). R represents the parametrization ratio, i.e., the number of parameters divided by the number of training patterns.

of a cross-validation technique is useful for MLP training (as well as for any other regression methods) to avoid the effects of overparametrization: poor generalization and sensitivity to overtraining in the presence of noise.[8]

6.6.6 Output Sigmoid Function

It can be observed in Table 6.2 that the best results are sometimes obtained with no hidden layer. Therefore, we also wished to learn if the sigmoid function at the output was useful or not. Without this nonlinearity the MLP would reduce to simple linear discriminant functions. We wanted to observe the effect of reducing the strong discrimination due to the sigmoid function, which approximates a logical decision. Accordingly, we trained one of the best MLPs (with 9 contextual input frames) with a linear function at the output. Two results are reported in Table 6.4: "LMLP9×132-50" stands for the MLP with no hidden units and linear outputs, "LCMLP9×132-20-50" stands for the MLP with 20 hidden units with linear outputs and the desired outputs that correspond

[8] A number of other approaches have been proposed or used to reduce the effects of overfitting, such as weight decay, regularization terms, weight pruning, and constructive networks. However, whatever the training scheme, very little is lost (either in training data or speed of computation) to test out a method on an independent data sample, which is ultimately what you wish to perform well on.

	training set 26767 patterns	test set 27702 patterns
LMLP9×132-50	57.2	52.3
MLP9×132-50	76.9	65.0
LCMLP9×132-20-50	54.2	50.5
MLP9×132-20-50	75.7	62.7

Table 6.4: Phonetic classification rates at the frame level on SPICOS obtained from MLPs with linear and nonlinear outputs.

to the confusion between classes (e.g., 0.9 for the correct class, 0.6 for the classes which are close to the good one and 0.1 for the others). For comparison, Table 6.4 also shows some results obtained with standard MLPs of Table 6.2: MLP9×132-50 and MLP9×132-20-50.

It can be observed in Table 6.4 that, for these experiments, the classification results at the frame level are worse than for the nonlinear case. This is probably because we no longer approximate the perceptron when the sigmoidal function is removed from the output; i.e., we no longer minimize the number of errors but simply a standard least square criterion. This can be seen by comparing "MLP9×132-50" and "LMLP9×132-50", where the only difference is the presence or absence of the output sigmoid. It is also important to notice that the result reported in "LMLP9×132-50" is probably a good approximation to the optimal linear discriminant since we are minimizing a standard quadratic function that has no local minima. The training is also faster.

6.6.7 Feature Dependence

An MLP can sometimes be useful without any contextual input. For example, in the discrete HMM instance of the SRI speech recognizer (DECIPHER) [Murveit & Weintraub, 1988], a state-of-the-art large vocabulary, speaker-independent, continuous speech recognition system, each acoustic vector at time n was described by 4 features, the mel-cepstrum (denoted y_{n1}), the delta mel-cepstrum (y_{n2}), the energy (y_{n3}) and the delta energy (y_{n4}). These features were independently quantized and described by 256, 256, 25 and 25 prototypes, respectively. Even without contextual information from the input field, it is impossible to directly estimate the probability of observing a set of 4 features given a class (or a state) q_k without an independence assumption (as there are $256 \times 256 \times 25 \times 25$ or 4×10^7 possible inputs).[9] Therefore, assuming independence, the joint probability estimate is

$$p(y_{n1}, y_{n2}, y_{n3}, y_{n4}|q_k) = \prod_{i=1}^{4} p(y_{ni}|q_k) \tag{6.19}$$

Using Bayes' rule, the MAP estimate can then be calculated:

$$\hat{p}(q_k|y_{n1}, y_{n2}, y_{n3}, y_{n4}) = \frac{\prod_{i=1}^{4} p(y_{ni}|q_k)p(q_k)}{p(y_{n1}, y_{n2}, y_{n3}, y_{n4})} \tag{6.20}$$

If we now consider an MLP with four input groups, each of them coding a particular feature ($256+256+25+25 = 562$ input units), the $k-th$ output will approximate, in theory, the MAP probability $p(q_k|y_{n1}, y_{n2}, y_{n3}, y_{n4})$ without the independence assumption. In this way, the system has the potential to extract and make use of the input feature correlation to improve classification performance. However, as before, the training procedure is not guaranteed to reach the optimal solution.

6.7 Radial Basis Functions

In this chapter we have shown theoretically and experimentally how the outputs of an MLP used in classification mode could be considered as good estimates of posterior probabilities.

In the following two sections (6.7 and 6.8) we discuss other approaches that use MLPs to generate similar (Radial Basis Functions – Section 6.7) or different (Predictive Neural Networks – Section 6.8) statistical quantities.

6.7.1 General Approach

Radial Basis Functions (RBFs) were originally introduced as a means of function interpolation [Powell, 1985; Broomhead & Lowe, 1988]. A RBF is usually defined as a linear combination of predefined (or trained)

[9] Maybe you could do this with hundreds or thousands of hours of speech. By contrast, the Resource Management (RM) training set (which was SRI's principal task at the time) was about 3 hours of speech.

nonlinear functions.[10] While standard MLPs combine hyperplanes as intermediate classification surfaces to approximate complex nonlinear decision boundaries, RBF networks aim at using more powerful (and more complex) functions to approximate the final decision surface. Usually these functions are simply second order functions (like Gaussians) or high order polynomials. In this case, a RBF network (i.e., an MLP using RBFs) will just compute the value of these different nonlinear functions on their hidden unit layer and, for each output class, will linearly combine them for each output class to optimize the trained criterion (e.g., LMS). In theory, if these nonlinear functions match the data space better, it can be expected that the same classification performance could be reached with fewer hidden units (and, consequently, fewer parameters). For instance, if the input data were normally distributed, one would need only one Gaussian-like distribution instead of at least two sigmoid functions.

For each class q_k ($k = 1, \ldots, K$) the approximating function $f_k(x_n)$ is then defined as a linear combination of J (nonlinear) basis functions $\phi_j(x_n)$:

$$f_k(x_n) \;=\; \sum_{j=1}^{J} c_{kj}\;\phi_j(x_n) \tag{6.21}$$

In terms of feedforward neural networks, this means that J functions ϕ are computed on the (first) hidden layer and that the standard parameters (weights) of the first layer are replaced by the parameters contained in ϕ.[11] These functions are then linearly combined to give the K activation values of output classes q_k, $k = 1, \ldots, K$. As a consequence, standard RBF networks generally do not use a nonlinear (sigmoid) function at their output.

Initially, the RBFs were fixed and *a priori* chosen to fit the overall data space. If some knowledge about the data is available, one can also reflect this information in the choice of functions. The advantage of using fixed RBFs is that once these have been chosen, all that is left to determine for each class q_k are their coefficients c_{kj}. In the case where

[10]Special thanks to Steve Renals for many discussions, experiments, and co-written sections that were used here.

[11]Although, in most of the cases, this can also be interpreted in terms of an inner product of the input vector by a weight (parameter) matrix followed by a fixed nonlinear function (identical for every hidden unit but different than a sigmoid – see Section 4.2.3 for the case where ϕ is Gaussian.

there is no sigmoid output function at the output, this is a standard problem of linear algebra that can be solved by the technique presented in Section 4.2.2, and does not require EBP. In effect, the RBFs just expand the input space into a higher-dimensional space where the data is more likely to be linearly separable.

Recently, RBFs were extended to accommodate the update of the basis function parameters, which then requires gradient descent training (like EBP).

6.7.2 RBFs and Tied Mixtures

The parameters of the hidden units in a RBF network may also be determined using maximum likelihood algorithms such as k-means or the EM algorithm [Dempster et al., 1977] to define a set of Gaussian distributions with means μ_j and covariance matrices Σ_j:

$$\phi_j(x_n) = R \exp\left(-\frac{1}{2}(x_n - \mu_j)^T \Sigma_j^{-1}(x_n - \mu_j)\right) \tag{6.22}$$

If the diagonal elements of the covariance matrix are represented by σ_{ij}^2, and we assume the covariance matrix is diagonal:

$$\phi_j(x_n) = R \exp\left(-\sum_{i=1}^{I} \frac{(x_{ni} - \mu_{ji})^2}{2\sigma_{ij}^2}\right), \quad 1 \leq j \leq J \tag{6.23}$$

where I is the number of input units. R is a constant term; here we shall not consider any dependency of R on the covariance matrix.

In this case, functions $f_k(x_n)$ in (6.21) are usually referred to as *tied Gaussian mixtures* and are of the form (2.6). These *probability density functions* (pdfs) have proven to be powerful tools for the estimation of emission probabilities $p(x_n|q_k)$ in HMM speech recognition systems [Renals, 1990; Bellegarda & Nahamoo, 1990]. The resulting systems are also known as *semi-continuous* HMMs. Tied mixture density estimation may be regarded as an interpolation between discrete and continuous density modeling. Essentially, tied mixture modeling uses a single "codebook" of Gaussians shared by all output pdfs. Each of these pdfs has its own set of mixture coefficients c_{kj} to combine the individual Gaussians. Alternatively, this may be regarded as "fuzzy" vector quantization [Ruspini, 1970, Bezdek, 1980; Pao, 1989]. In semi-continuous HMMs, the parameters of the RBFs (i.e., the mixture weights c_{kj} and, sometimes the means and variances of the Gaussian densities) are typically optimized according to a maximum likelihood criterion, using the EM algorithm [Dempster et al., 1977] or a modified version of the Baum-Welch algorithm [Huang & Jack, 1989].

With the renewal of ANNs, these RBFs have been used in the framework of MLPs in which the RBF parameters are also optimized according to a gradient procedure and a LMS criterion. To actually minimize the classification error rate (which is not achieved by a pure LMS), a sigmoid-like function is sometimes added on the output units; this is discussed in the next section.

6.7.3 RBFs for MAP Estimation

RBFs are sometimes used in MLPs, where the common sigmoid on the hidden units is replaced by Gaussian-like functions (6.22). As with sigmoidal MLPs, the networks are typically trained to minimize the MSE

$$E = \frac{1}{2}\sum_{k=1}^{K}(g_k(x_n) - d_k(x_n))^2, \tag{6.24}$$

or a relative entropy criterion

$$E = \sum_{k=1}^{K} d_k(x_n)\log\frac{d_k(x_n)}{g_k(x_n)} + (1 - d_k(x_n))\log\frac{(1 - d_k(x_n))}{(1 - g_k(x_n))} \tag{6.25}$$

at the output of sigmoidal output units; as before, $d_k(x_n)$ representing the desired network output and $g_k(x_n)$ the actual output for class q_k. If the parameters of the basis functions are kept fixed and if the output units are linear, training can be performed simply by linear algebra. If there is a sigmoid function at the output and/or if the parameters of the basis functions have to be updated, the optimization of such networks is generally performed by the EBP algorithm.

As shown in Section 6.1, in the "1-from-K" classification case, the outputs of a feedforward network trained according to either of these discriminative criteria may be interpreted as the Bayes posterior probabilities, $p(q_k|x_n)$, x_n being the input vector at time n (eventually built up by concatenating a few consecutive acoustic vectors into account).

However, RBF networks differ in several ways from standard MLPs. Standard RBF networks generally use linear outputs. This has several drawbacks:

- Since there is no approximation of the logical decision at the output, the network does not actually minimize the classification error rate but simply the MSE or the entropy (see discussion in Chapter 4).

- The posterior estimates obtained at the output are not constrained to be values between 0 and 1 (and to sum to one).

- If we view the weighted sum of Gaussians $f_k(x_n)$ as a likelihood, we also have a mismatch with the training criterion which approximates MAPs. Indeed, during training, the function transforming likelihoods into posterior probabilities is nothing else but Bayes' rule
$$p(q_k|x_n) = \frac{p(x_n|q_k)p(q_k)}{p(x_n)}$$
which is not a linear function. Although the prior probabilities $p(q_k)$ present in Bayes' rule can be included in the c_{kj} of (6.21), its denominator is not constant during training and depends on all the classes (as discussed in Chapter 3).

Although the use of a sigmoid (or softmax) function at the output (instead of the linear function) can solve the first two problems, it does not avoid the mismatch between likelihoods and posterior probabilities (since the function transforming likelihoods into MAPs is not a sigmoid!).

This may be resolved by using Bayes' rule at the output of the network to generate the posterior from the likelihood, instead of the standard linear or sigmoid function:

$$p(q_k|x_n) = \frac{p(x_n|q_k)p(q_k)}{\sum_{\ell=1}^{K} p(x_n|q_\ell)p(q_\ell)} \tag{6.26}$$

[where $p(x_n)$ is expanded in the denominator to explicitly show its dependence on all the class likelihoods]. Thus, we should define a transfer function for the output units that transforms the weighted sum of Gaussians $f_k(x_n)$ to posterior probabilities using Bayes' rule:

$$g_k(x_n) = \frac{f_k(x_n)p(q_k)}{\sum_i f_i(x_n)p(q_i)} \tag{6.27}$$

A final constraint is required. If $f_k(x_n)$ is to be a true approximation to the likelihood using tied mixture densities then the weights c_{kj} must be constrained:

$$\left.\begin{array}{l} c_{kj} \geq 0, \forall k, j \\ \\ \sum_{j=1}^{J} c_{kj} = 1 \end{array}\right\} \tag{6.28}$$

This may be achieved using a normalized exponential (softmax) transformation:

$$c_{kj} = \frac{\exp(w_{kj})}{\sum_h \exp(w_{kh})} \tag{6.29}$$

6.7.4 Lagrange Multipliers

It is thus necessary to reformulate the EBP algorithm to take the new transformations and additional constraints into account. To do this, it is better (and less error-prone!) to formulate the problem as a constrained minimization problem as explained in Section 4.3.4 instead of using the standard chain rule for differentiation.

In this section, we will only consider the LMS criterion and, in order not to pain the reader with an excess of mathematics, we will assume that the Gaussian parameters are fixed (not trained) and that only the mixing parameters c_{kj} are trained. Also, we will assume that the constraint (6.28) is not forced during training (since a mixture of Gaussians can also be defined with negative weights [Titterington, Smith & Makov, 1985]). Of course, this can easily be generalized to other criteria, taking all the above constraints into account. In [Renals et al., 1991], a complete derivation of the EBP algorithm for the relative entropy and including all the constraints [(6.26)-(6.28)] is given.

Introducing Lagrange multipliers for the constraints, we then must minimize the Lagrange function:

$$\begin{aligned} L &= \frac{1}{2}\sum_k (g_k(x_n) - d_k(x_n))^2 \\ &+ \sum_k \lambda_k \left[g_k(x_n) - \frac{f_k(x_n)p(q_k)}{\sum_{i=1}^K f_i(x_n)p(q_i)} \right] \\ &+ \sum_k \Lambda_k \left[f_k(x_n) - \sum_{j=1}^J c_{kj}\phi_j(x_n) \right] \end{aligned} \tag{6.30}$$

A constraint is met when the corresponding term in L is zero. It can be shown that:

$$\nabla L(f, g, c, \lambda, \Lambda) = 0 \tag{6.31}$$

corresponds to a minimum of E while meeting the constraints. We may split condition (6.31) into its constituent partials. It has been shown in Section 4.3.2 that optimizing with respect to the Lagrange multipliers gives the forward propagation equations. Optimizing with respect to the state variables gives the backward equations (the gradients). Optimizing with respect to the weights gives the weight update equation. We thus have:

$$\begin{aligned} \frac{\partial L}{\partial f_\alpha(x_n)} &= \frac{p(q_\alpha)\sum_{k=1}^K \lambda_k f_k(x_n)p(q_k)}{(\sum_{i=1}^K f_i(x_n)p(q_i))^2} - \frac{\lambda_\alpha p(q_\alpha)}{\sum_{i=1}^K f_i(x_n)p(q_i)} \; + \; \Lambda_\alpha \\ &= 0 \\ \frac{\partial L}{\partial g_\alpha(x_n)} &= (g_\alpha(x_n) - d_\alpha(x_n)) + \lambda_\alpha = 0 \end{aligned}$$

and, consequently:

$$\lambda_\alpha = d_\alpha(x_n) - g_\alpha(x_n)$$

and this is the back-propagated error at the output. Consequently:

$$-\Lambda_\alpha = \frac{p(q_\alpha)\sum_{k=1}^K \lambda_k f_k(x_n)p(q_k)}{(\sum_{i=1}^K f_i(x_n)p(q_i))^2} - \frac{\lambda_\alpha p(q_\alpha)}{\sum_{i=1}^K f_i(x_n)p(q_i)}$$

$$\frac{\partial L}{\partial c_{\alpha\beta}} = -\Lambda_\alpha \phi_\beta(x_n)$$

and this is proportional to the weight change of the connection matrix:

$$\nabla c_{\alpha\beta} = \phi_\beta(x_n)\left[\frac{p(q_\alpha)\sum_{k=1}^{K}\lambda_k f_k(x_n)p(q_k)}{(\sum_{i=1}^{K} f_i(x_n)p(q_i))^2} - \frac{\lambda_\alpha p(q_\alpha)}{\sum_{i=1}^{K} f_i(x_n)p(q_i)}\right]$$

It is thus possible to reformulate the RBF formalism to update all the parameters by the EBP algorithm while preserving the advantages of standard MLPs, i.e., estimation of MAPs and minimization of the classification error rate.

6.7.5 Discussion

Although this approach has yielded results slightly better than we observed with standard RBFs [Renals et al., 1991], results were still poorer than the results we obtained with standard MLPs. This could be due to the sensitivity of these networks to good initialization, but may also be due to "hidden" assumptions we make when using RBFs. Indeed, although any probability density function can be approximated from a mixture of a finite number of normal density functions, there are still some hypotheses underlying standard RBFs. For instance, it is usually assumed that the input components are not correlated, since normal densities with diagonal covariance matrices are ordinarily used. A solution to this problem is to use RBFs described in terms of normal densities with full covariance matrices. However, this results in a significant increase of the number of parameters.

6.8 MLPs for Autoregressive Modeling

Thus far, we have described MLPs as discriminant pattern classifiers that have been shown, in the case of "1-from-K" training, to be good MAP approximators. However, there are other ways to use neural networks and to relate them with stochastic quantities that can be used in other forms of hybrid HMM/MLP approaches. One of the most interesting alternatives consists of using MLPs as nonlinear predictors. In this case, the observations associated with each class are assumed to be drawn from a nonlinear *autoregressive* (AR) process. In this case, it can be shown that, under some assumptions, the predictor error can be considered as the logarithm of a local "emission" probability that can be used in a particular form of HMM (see Chapter 13 for further discussion).

Compared with the hybrid HMM/MLP system developed in this work, the predictive approach can potentially provide better models of speech signal dynamics. However, in current implementations, this is achieved at the cost of a weaker discrimination.

6.8.1 Linear Autoregressive Modeling

The idea of using MLPs as nonlinear predictors [Deng et al., 1991; Iso & Watanabe, 1990, 1991; Levin, 1990, 1993; Tebelskis & Waibel, 1990; Tebelskis et al., 1991; Tsuboka et al., 1990] is directly related to the theory usually referred to as "*autoregressive* (AR) HMMs" [Poritz, 1982; Rabiner, 1989; Juang & Rabiner, 1985], in which the pdfs describing the emission probabilities of standard HMMs (usually Gaussian or multi-Gaussian densities) are replaced by an AR function.

The basic idea of all these approaches is to assume that, for each class q_k (i.e., for each HMM-state), the observation vectors are drawn from an autoregressive process described by a function F_k with parameter set θ_k. During training, if the order of prediction is assumed to be p, the parameters θ_k of the functions F_k (each of these being associated with a particular q_k, $k = 1, \ldots, K$) are estimated to minimize, for all the acoustic vectors x_n observed on q_k, the MSE

$$
\begin{aligned}
E &= \sum_{k=1}^{K} \sum_{x_n \in q_k} \| x_n - F_k(X_{n-p}^{n-1}, \theta_k) \|^2 \\
&= \sum_{k=1}^{K} \sum_{x_n \in q_k} \| \epsilon_{n,k} \|^2 \qquad (6.32)
\end{aligned}
$$

which represents the sum over the whole training set of the squared errors between the actual observation x_n at time n and its predicted value $\hat{x_n}$ as given by the AR function associated with the correct class. $\epsilon_{n,k}$ is the *prediction error* for acoustic vector x_n and class q_k and is defined as:

$$\epsilon_{n,k} = x_n - F_k(X_{n-p}^{n-1}, \theta_k)$$

In standard AR modeling, function F_k is linear and, when used for speech processing, is usually applied at the sample level. This is the case for LPC analysis used for speech synthesis or in [Poritz, 1982] for speaker recognition and [Rabiner, 1989; Juang & Rabiner, 1985] for speech recognition. However, this method could also be applied after feature extraction on acoustic vectors, in which case $F_k(\cdot)$ becomes a matrix. In both cases, AR models are known to be better suited to dynamical systems.[12] When used directly at the sample level, another potential advantage of this approach is to avoid explicit feature extraction before classification. However, this is not that easy; it could be argued that *Autoregressive Conditional Heteroscedastic* (ARCH) models (see Chapter 13 and [Saerens & Bourlard, 1993] for further discussion) are better suited for such a task. It is also generally acknowledged that speech-like signals should be better modelled by *Autoregressive Moving Average* (ARMA) models than simple AR models [Priestley, 1991].[13]

6.8.2 Predictive Neural Networks

Thanks to ANNs, this approach has recently been generalized to nonlinear functions $F_k(\cdot)$ implemented by MLPs trained to minimize the MSE (6.32). This leads to a new family of hybrid HMM/MLP approaches usually referred to as *Predictive Neural Networks*. In this case, each class q_k is associated with a specific MLP and, during training, the input field of the MLP contains X_{n-p}^{n-1} (the p previous acoustic vectors) and the associated desired output is the current acoustic vector x_n. In this case, the MLP is no longer used for classification, but instead as a nonlinear function generator.[14] If the x_n's are real-valued vectors, the output units are simply linear (i.e., there is no sigmoid function at the output layer).

6.8.3 Statistical Interpretation

In both cases (linear and nonlinear AR models), it can be shown [Rabiner, 1989; Levin, 1990, 1993][15] that if the prediction errors $\epsilon_{n,k}$ are assumed to be Gaussian, *independent, identically distributed* (iid) random variables with zero mean and unity variance[16], minimization of E

[12] It seems reasonable to describe speech production in terms of a dynamical system. However, as our politicians have been noting lately, the Devil is in the details.

[13] Although it is not clear how ARMA models could be trained on speech signals; in particular, the choice of pole and zero model orders is difficult.

[14] Unfortunately, the discriminant properties of MLPs used for classification are lost.

[15] See Chapter 13 for further discussion about this.

[16] Variance can also be assumed to be equal to σ_k in the case of scalar inputs [Rabiner, 1989] or to a matrix Σ_k in the case of vectorial inputs. In ARCH models, it is also assumed that the parameters of the (linear) AR functions are themselves random variables for which we want to estimate the distribution (see Chapter 13 or [Saerens & Bourlard, 1993]).

in (6.32) is equivalent to maximization of

$$\prod_{k=1}^{K} \prod_{x_n \in q_k} p\left(x_n | q_k^n, X_{n-p}^{n-1}\right)$$
$$= \prod_{k=1}^{K} \prod_{x_n \in q_k} (2\pi)^{-d/2} \exp\left[-\frac{1}{2} \parallel x_n - F_k(X_{n-p}^{n-1}, \theta_k) \parallel^2\right] \quad (6.33)$$

if $x_n \in \mathcal{R}^d$, where q_k^n represents the (HMM) state q_k associated with the current acoustic vector x_n and where $p(x_n|q_k^n, X_{n-p}^{n-1})$ is the conditional likelihood taking the p previous feature vectors into account. This probability can be used in a kind of HMM, using emission probabilities similar to those defined in (3.9), where the observation-independence assumption is relaxed and emission on states is assumed. Error predictions obtained from linear or nonlinear autoregressive models can be considered as negative logarithms of these local probabilities and can be used as such in DTW of the Viterbi algorithm (3.17). If the variance of the error is not assumed to be unity, this will require the use of some kind of Mahalanobis distance during recognition. These are valid HMM emission probabilities in which the observation-independence assumption (H4) has been relaxed.

According to this approach, each class should have its own MLP predictor [Tebelskis & Waibel, 1990]. However, an interesting alternative approach was proposed in [Levin, 1990, 1992] in which only one MLP was required with additional "control" input units coding the state q_k being considered. This requires running the same network several times (i.e., once for each class considered) while the basic approach requires the estimation of one network for each class considered.

6.8.4 Another Approach

Using the main results presented in this chapter (i.e., that if trained in classification mode, the MLP's outputs are estimates of posterior probabilities), it is possible to generalize the predictive approach and to avoid the assumptions on the driving noise. It is indeed easy to prove (by using Bayes' rule with an additional conditional X_{n-p}^{n-1} everywhere) that:

$$p(x_n|q_k^n, X_{n-p}^{n-1}) = \frac{p(q_k^n|X_{n-p}^{n})p(x_n|X_{n-p}^{n-1})}{p(q_k^n|X_{n-p}^{n-1})} \quad (6.34)$$

As $p(x_n|X_{n-p}^{n-1})$ in (6.34) is independent of the classes q_k it can be overlooked in the dynamic programming recurrences (3.24). Based on what as been shown in Section 6.2, it is then interesting to observe that, without any assumption about mean and covariance of the driving noise, the same conditional likelihood $p(x_n|q_k^n, X_{n-p}^{n-1})$ (taking correlation of acoustic vectors into account) can be expressed as the ratio of the output values of two "standard" MLPs (as used in Section 6.2), respectively with X_{n-p}^{n-1} and X_{n-p}^{n} as input, and trained in classification mode according to a LMS or an entropy criterion.

This approach has thus two advantages: (1) no assumptions on the driving noise; (2) only two MLPs (with K output units, instead of K MLPs with one output unit) are required to estimate all the conditional likelihoods, which leads to a better sharing of the parameters and, consequently, to a better discrimination (although this advantage is already present in the approach proposed in [Levin, 1990, 1993]). However, a disadvantage is that the method requires division of two probabilities. This can result in very poor estimates when the denominator is small.

6.8.5 Discussion

Both approaches (presented in Section 6.8.3 and in Section 6.8.4) have been tested on the databases used in this book. Although (in our experience) the approach presented in 6.8.4 performed slightly better than the one originally proposed in [Levin 1990; Tebelskis & Waibel, 1990], results were still inferior to those achieved using MLPs trained in classification mode. One reason for this could be that the discriminant properties of MLPs are no longer used in predictive neural networks. A second reason could be related to a still open issue regarding the correct use of correlated emission probabilities in HMMs. Although the theory of autoregressive Markov models (including predictive neural networks) and time correlation modeling in HMMs is very attractive, as noted previously, we don't know of any successful attempt to use such information to improve performance [de La Noue et al., 1989; Wellekens, 1987]. Our own experiments seem to corroborate this. For the case of our own method, however, some of the difficulty may have come from the small-denominator problem described above.

6.9 Summary

In this chapter, it has been shown, both from a theoretical perspective and from empirical measurements, that the outputs of an MLP (when trained for pattern classification) approximate Bayesian *a posteriori* probabilities (MAP), i.e., the class probability conditioned on the input. When the estimation is accurate, network outputs can be treated as probabilities and sum to one (at least approximately). The experimental results that have been presented here show some of the improvement for MLPs over conventional classifiers, at least at the frame level. MLPs can make better frame level discriminations than simple statistical classifiers, because they can easily incorporate multiple sources of evidence (multiple frames, multiple features), which is difficult to do in standard classifiers without major simplifying assumptions. Finally, is the MLP simply accomplishing a nice interpolation for the joint density estimates we seek? Perhaps so, and other smoothing techniques (kernel estimators, for instance) may work as well.[17] Nonetheless, MLP approaches appear to offer a reasonable way of incorporating information from multiple sources of phonetic evidence into a continuous speech recognizer.

Other ways to use MLPs to approximate probabilities have also been reviewed and discussed. In the first case, radial basis function networks, viewed as a special case of MLPs, have also been shown to yield MAP estimates provided one implements the transformation function between the likelihoods estimated on the hidden units and the MAPs estimated at the output of the MLP. However, so far, this approach has not improved the performance obtained with sigmoidal MLPs. Another solution using MLPs as nonlinear autoregressive models has also been discussed. This has been shown to be equivalent to estimating likelihoods directly related to HMM emission probabilities, taking the correlation of the acoustic vectors into account. However, in this case, the discriminant properties of MLPs are lost and some assumptions about the distribution of the prediction error are necessary. A generalization that avoids

[17]Some recent experiments done in a collaboration with D. Specht of Lockheed seemed to indicate that a Parzen window approach was difficult to use effectively for this application, primarily because of storage and computational requirements. Results as of this writing were still significantly inferior to the MLP case, although this is probably not due to any fundamental limitation; it is quite possible that further experiments, particularly using hardware applicable to this kind of approach, might permit research that would result in a kernel-based estimator of similar performance to the MLP.

these distributional assumptions has been presented.

Given the good results obtained at the frame level with a standard MLP with contextual inputs and the clear statistical interpretation of its outputs, only this architecture will be explored in the following chapters. There we will investigate the means to integrate them into HMMs, with the aim to extend these good frame classification results to the level of word recognition in continuous speech.

Chapter 7

THE HYBRID HMM/MLP APPROACH

I can't understand why people are frightened of new ideas. I'm frightened of the old ones.
– John Cage –

7.1 Introduction

As described earlier, HMMs are now widely used for automatic speech recognition, and inherently incorporate the sequential and statistical character of the speech signal. However, notwithstanding their efficiency, standard HMM-based recognizers suffer from several weaknesses, mainly due to the many hypotheses required to make their optimization possible (see Chapter 3):

- Poor discrimination due to the training algorithm which maximizes likelihoods (MLE) instead of MAP probabilities (Chapter 3, hypothesis H2). Another algorithm based on the Maximum Mutual Information (MMI) criterion provides more discrimination but, in this case, the mathematics become more difficult, and many constraining assumptions have to be made.

- Assumption that the state sequences are first-order Markov chains (Chapter 3, hypothesis H3).

- No contextual information is taken into account, and thus the possible correlation of the successive acoustic vectors is overlooked (Chapter 3, hypothesis H4).

- *A priori* choice of model topology and statistical distributions, e.g., assuming that the emission probabilities $p(x_n|q_\ell)$ associated with each state q_ℓ can be described as a multivariate Gaussian density or, more recently, as a mixture of multivariate Gaussian densities (Chapter 3, hypotheses H1 and H6).

- For the sake of storage and computational efficiency, the need to treat as independent certain processes that are not truly independent (e.g., hypothesis H5).

On the other hand, ANNs (e.g., MLPs) developed for learning, feature extraction, and classification may be useful for signal processing. Their main advantages include:

- They are learning machines (as are HMMs).

- They provide discriminant-based learning (that is, models are trained to suppress incorrect classifications as well as to accurately model each class separately).

- When used in classification mode and trained with an LMS criterion or an entropy criterion, the network outputs will estimate posterior probabilities without requiring strong assumptions about the underlying probability density functions (see Chapter 6).

- Because ANNs are capable of incorporating multiple constraints and finding optimal combinations of constraints for classification, features do not need to be treated as independent. In other words, there is no need for strong assumptions about statistical distributions and independence of input features.

- Using the interpolative capabilities of the MLP, statistical pattern recognition can be performed over an undersampled pattern space [Niles et al., 1989] without many restrictive simplifying assumptions. However, in this case it may be useful to use cross-validation techniques (see Chapter 12) to avoid overfitting.

- ANNs are highly parallel structures, which makes them especially amenable to high-performance architectures and hardware implementations.

Unfortunately, as shown in Chapter 5, ANNs also have certain weaknesses for use in speech recognition. Most previous applications of neural networks to speech recognition have depended on severe simplifying assumptions (e.g., very small vocabularies, isolated words, and known word or phoneme boundaries). However, the major weakness of ANNs is their inability to deal with the time sequential nature of speech.

To circumvent those problems, a hybrid approach using an MLP to estimate local probabilities of HMMs has been developed [Bourlard et al., 1990; Bourlard & Morgan, 1990] and is presented here. As noted in Chapter 6, both theoretical and experimental results have shown that the MLP output values may be viewed as estimates of MAP probabilities for pattern classification. Therefore, these outputs, or other related quantities (such as the output normalized by the prior probability of the corresponding class) may be used in a Viterbi search to determine the best time-warped succession of states (representing speech sounds) to explain the observed speech measurements.

However, as shown in the following, it is not quite so simple to properly interface MLPs and HMMs to take advantage of what are believed to be the strong points of each of the two approaches while preserving the theory of HMMs.

One problem is that MLPs are generating posterior probabilities (e.g., $p(q_k|x_n)$ in the terminology of the previous chapters) while HMMs are designed to handle likelihoods $p(x_n|q_k)$. Consequently, a new kind of model called a "discriminant HMM", designed to handle posterior probabilities is presented and discussed in Section 7.2. However, it will be shown that we were not able to use this approach to achieve the word level performance we could expect from the improved phonetic frame labeling. Solutions to this problem and necessary modifications of the basic scheme are presented and discussed in Section 7.4, where it is shown that probabilities generated by MLPs can indeed be effectively used in HMMs to improve state-of-the-art speech recognizers.

7.2 Discriminant Markov Models

Although the actual criterion that should be used for HMMs is the posterior probability $P(M|X)$, where X is a sequence of acoustic vectors and M an HMM, it was shown in Chapter 3 that some necessary simplifying assumptions reduced this criterion to the estimation of the likelihood $P(X|M)$ at the cost of the discriminative capabilities of the models. In this section, we define discriminant Markov models that will estimate the optimal criterion $P(M|X)$. It is shown that their parameters are determined by conditional transition probabilities that can be generated in different ways by MLPs.

7.2.1 Formulation

As done in Chapter 3 (Section 3.3) for the global likelihood $P(X|M_i)$ and using the same notations, it is possible to estimate the global MAP probability $P(M_i|X)$ of a Markov model M_i given the acoustic vector sequence X as follows:

$$P(M_i|X) = \sum_{\ell_1=1}^{L} \cdots \sum_{\ell_N=1}^{L} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M_i|X) \tag{7.1}$$

$$= \sum_{\ell=1}^{L} P(q_\ell^n, M_i|X), \forall n \in [1, N] \tag{7.2}$$

However, without any simplifying assumption, we also have:

$$P(q_\ell^n, M_i, X) = P(q_\ell^n, M_i|X)P(X) = P(q_\ell^n, X|M_i)P(M_i) \tag{7.3}$$

or

$$P(q_\ell^n, M_i|X) = \frac{P(M_i)}{P(X)} P(q_\ell^n, X|M_i) \tag{7.4}$$

which is nothing other than the likelihood used in standard HMMs multiplied by a scaling factor. This kind of scaling was already used in [Devijver, 1985] and [Levinson et al., 1983] to avoid numerical problems (because of the product of probabilities), where it was shown that this led to the same forward and backward recurrences of the standard Baum-Welch algorithm (within a normalization factor).

Maximization of $P(X|M_i)$ and of $P(M_i|X)$ thus seems to lead to the same estimation formulas. Should we conclude from this that the discriminant approach does not change anything? Not at all, because we must not neglect the major constraint of the MAP approach, i.e.:

$$\sum_i P(M_i|X) = 1 \tag{7.5}$$

where the sum over i represents the sum over all possible Markov models. Here lies the difference between an MLE and an MAP criterion. Any modification of the parameters of a model M_i must be complemented by a modification of all the parameters of the other models so as to preserve this constraint, thus making the MAP procedure discriminant. Thus, even if the estimation formulas are the same, the re-estimation (maximization and update) formulas will have to be different to take the constraint (7.5) into account. In the following, we define the key parameters of such discriminant HMMs and we show which constraint they must meet to guarantee (7.5). It is shown that this constraint is automatically met when using MLPs to estimate these parameters.

7.2.2 Conditional Transition Probabilities

In order to apply the MAP principle properly, it is thus necessary to find a parametric expression of $P(M_i|X)$ that meets the constraint (7.5) for all possible parameter values.

As done in Section 3.3 with the MLE, a solution to this problem consists in splitting up the global MAP probability as follows:

$$P(M_i|X) = \sum_{\ell_1=1}^{L} \cdots \sum_{\ell_N=1}^{L} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M_i|X) \tag{7.6}$$

for all possible $\{q_{\ell_1}^1, \ldots, q_{\ell_N}^N\} \in \Gamma$, the set of all possible paths in M_i. The right hand side can be factored into

$$P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N, M_i|X) = P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X) P(M_i|X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N)$$

which suggests two separate steps for the recognition. The first factor represents the acoustic decoding, in which the acoustic vector sequence is converted into a sequence of states. The second factor represents a phonological and lexical step; once the sequence of states is known, the model M_i associated with X can be found from the state sequence without an explicit dependence on X so that

$$P(M_i|X, q_{\ell_1}^1, \ldots, q_{\ell_N}^N) = P(M_i|q_{\ell_1}^1, \ldots, q_{\ell_N}^N)$$

For example, if the states represent phonemes, this probability must be estimated from phonological knowledge of the vocabulary in a separate process without any reference to the input vector sequence. Neglecting this probability is equivalent to assuming that, given a sequence of states, it is possible to recover the model that generated it.

Probability $P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X)$ can be factored into discriminant local probabilities as

$$\begin{aligned} P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X) &= p(q_{\ell_1}^1|X)p(q_{\ell_2}^2|X, q_{\ell_1}^1)\ldots p(q_{\ell_N}^N|X, q_{\ell_1}^1, \ldots, q_{\ell_{N-1}}^{N-1}) \\ &= \prod_{n=1}^{N} p(q^n|Q_1^{n-1}, X) \end{aligned} \tag{7.7}$$

where q^n represents the state observed at time n and Q_1^N the state sequence associated with X_1^N. Probabilities $P(q_{\ell_1}^1, \ldots, q_{\ell_N}^N|X)$ can thus be calculated from "local" probabilities $p(q^n|Q_1^{n-1}, X_1^{n-1})$ that will be referred to as *conditional transition probabilities* in the following and are the key parameters of discriminant HMMs (see Section 7.2.5 for further discussion about these probabilities).

In (7.7), each factor may be simplified by relaxing the conditional constraints. For example, the factors of (7.7) can be assumed dependent on the previous state only and on a signal window of length $2c+1$ centered around the acoustic vector at time n (other kinds of assumptions on these conditional transition probabilities will be discussed in Section 7.2.5). We currently approximate these local contributions by

$$p(q_{\ell_n}^n|X, q_{\ell_1}^1, \ldots, q_{\ell_{n-1}}^{n-1}) = p(q_{\ell_n}^n|X_{n-c}^{n+c}, q_{\ell_{n-1}}^{n-1}) \tag{7.8}$$

where input contextual information is taken into account. As shown in Chapter 6, these probabilities can be estimated at the outputs of an MLP with contextual input and output feedback. Furthermore, if input contextual information is neglected ($c = 0$), (7.8) becomes:

$$p(q_{\ell_n}^n|x_n, q_{\ell_{n-1}}^{n-1}) \tag{7.9}$$

which is the discriminant local probability (3.45) for continuous features. This is the foundation of discriminant HMMs. Each observed acoustic vector is associated with a transition so that no splitting into transition and emission probabilities as in (3.11) or (3.39) and (3.40) is possible. Moreover, when comparing with the classical transition emitting HMM, the main difference lies in the normalization of the local contribution, which is performed on the set of states and not on the (prototype) vector space as was done, for example, in (3.40) and (3.41). We then have :

$$P(M_i|X) = \sum_{\ell_1=1}^{L} \cdots \sum_{\ell_N=1}^{L} \prod_{n=1}^{N} p(q_{\ell_n}^n|x_n, q_{\ell_{n-1}}^{n-1}) \tag{7.10}$$

and so it is possible to estimate the MAP probability $P(M_i|X)$ from the conditional transition probabilities $p(q_k^n|q_\ell^{n-1}, x_n)$.

It is also important to show that, in this case, if the constraint:

$$\sum_{k=1}^{K} p(q_k^n|x_n, q_\ell^{n-1}) = 1 \tag{7.11}$$

is met (which will be the case, at least approximately, with MLP outputs), the constraint (7.5) on the global MAP probabilities is also met. Indeed, if Γ denotes the set of all possible paths $\{q_{\ell_1}^1, \ldots, q_{\ell_N}^N\}$ in all possible Markov models M_i, we have:

$$\begin{aligned}
&\sum_i P(M_i|X) = \sum_{\Gamma} \prod_{n=1}^{N} p(q_{\ell_n}^n|x_n, q_{\ell_{n-1}}^{n-1}) = \\
&\sum_{q_{\ell_1}^1=1}^{K} p(q_{\ell_1}^1|x_1) \left(\sum_{q_{\ell_2}^2=1}^{K} p(q_{\ell_2}^2|x_2, q_{\ell_1}^1) \ldots \left(\sum_{q_{\ell_N}^N=1}^{K} p(q_{\ell_N}^N|x_N, q_{\ell_{N-1}}^{N-1}) \right) \ldots \right) \\
&= 1
\end{aligned}$$

Besides the advantage of forcing discrimination, numerical problems which plague the classical HMM are avoided when using discriminant models: namely, the lack of balance between the transition probability values which only depend on the topology of the model and the emission probability values which decrease with the length of the input features (if the components are assumed independent) or, in the discrete case, with the number of prototype vectors I. This effect worsens if context dependence is introduced by using $2c + 1$ appended consecutive vectors as features. Indeed, the number of possible prototype combinations (and thus the number of discrete emission probabilities for each state) grows as I^{2c+1}, thereby decreasing the values of the emission probabilities. Unfortunately, even with discriminant models, the exponential increase of the number of parameters with the width $2c + 1$ of the window is not avoided, resulting in a need for huge storage capacity and requiring an excessive amount of training data to obtain statistically significant parameters. Moreover, transitions unobserved in the training set will yield zero probabilities despite the fact that they may actually occur in a recognition phase. To cope with insufficiently large data sets, interpolation techniques can be used [Bahl et al., 1983] and a lower bound can be imposed on the local probabilities. Still, memory is wasted since many explicitly stored discriminant local probabilities reach the lower bound. Indeed, the likelihood of actually observing most of the I^{2c+1} possible input vectors is extremely low.

7.2.3 Maximum Likelihood Criterion

As done with equations (3.4) and (3.5) and Section 3.3.4, we can start from (7.1) and, using (7.4), we can derive equivalent Forward-Backward recurrences to compute the global MAP probabilities $P(M_i|X)$ [Devijver, 1985]. However, this can still be referred to as "Maximum Likelihood Criterion" since the only difference during calculation of $P(M_i|X)$ is the scaling factor present in (7.4). However, the re-estimation formulas briefly given in Section 3.4.1 are no longer valid since we have to take constraint (7.5) into account.

The goal of the next sections will be to use an MLP to estimate probabilities for use in HMMs. Since MLPs require supervised training, i.e., segmented speech data, and since the maximum likelihood criterion (or the MAP variant of it) does not provide us with the optimal state sequence (and, consequently, segmentation), this approach will not be investigated further in the following.

7.2.4 Viterbi Criterion

In the case of the Viterbi criterion, we approximate the MAP probability $P(M_i|X)$ by only considering the best state sequence. In this case, (7.6) is approximated as:

$$\overline{P}(M_i|X) \;=\; \max_{\ell_1,\ldots,\ell_N} P(q^1_{\ell_1},\ldots,q^N_{\ell_N},M_i|X) \qquad (7.12)$$

which yields the following dynamic programming recurrence:

$$\overline{P}(q_\ell|X_1^n) \;=\; \max_k \left(\overline{P}(q_k|X_1^{n-1})p(q_\ell|x_n,q_k)\right) \qquad (7.13)$$

where parameter k runs over all possible states preceding q_ℓ and $\overline{P}(q_\ell|X_1^n)$ denotes the cumulated best path probability of reaching state q_ℓ and having emitted the partial sequence X_1^n .

Thus, when using discriminant HMMs, it is possible to use the Viterbi decoding and training procedures described in Chapter 3 in which the local contributions $p(q_\ell^n|q_k^{n-1})p(x_n|q_\ell^n)$ are replaced by $p(q_\ell^n|x_n, q_k^{n-1})$.

7.2.5 MLPs for Discriminant HMMs

Varying the hypotheses (7.8) and (7.9), it is possible to define other conditional transition probabilities that can be used in discriminant HMMs. Based on the fact that MLPs are approximating posterior probabilities (i.e., probability of the output classes associated with the output units, conditioned on the pattern presented at the input of the MLP), different conditional probabilities for use in (7.7) can be estimated by MLPs.

Case 1: $p(q^n|Q_1^{n-1}, X) = p(q^n|q^{n-1}, x_n)$.

This is the simplest Markovian assumption, which has been described above. This probability can be approximated by a "recurrent" MLP with the current acoustic vector x_n and the previous state q^{n-1} at the input. However, this is not truly a recurrent network. During (Viterbi) training, the acoustic vectors are labeled and the state associated with the previous acoustic vector is known. We have thus all the required information to feed the network and estimate the transition probabilities. During recognition, q^{n-1} is given by the topology of the Markov model. However, since every state usually has several possible predecessors, it will be necessary to run the network several times (one run per possible predecessor) to estimate all the transition probabilities that are necessary to perform the DTW. Another solution consists in feeding back the whole output vector to the input (case 3).

Case 2: $p(q^n|Q_1^{n-1}, X) = p(q^n|Q_{n-k}^{n-1}, x_n)$.

In this case, the dependence is extended to the k preceding states and an MLP containing the current acoustic vector and the k preceding states could, in principle, estimate this probability. However, several problems plague this approach. This first problem is the excessive number of parameters which requires a huge amount of training data. Of course, the advantage of neural networks is that we can easily control this number of parameters by varying the number of hidden units. However, in this case, there is a risk of getting very poor estimates of the probabilities. Another possibility is to keep the number of parameters large and to use cross-validation techniques (see Chapter 12) to avoid overtraining and to get the maximum of the training data.

A second problem is related to the recognition process itself; it is not yet known how to use high-order probabilities in a Viterbi-based DTW. Indeed, not only will it be necessary to run the same net several times to get the contributions of all possible sets of previous state sequences (given by the HMM topology) but the DTW process will have to be modified to take all of these possible sequences into account. In this case, it can be shown that, to preserve DTW optimality, the memory and computational requirements will increase exponentially, making the search intractable.

Case 3: $p(q_k^n|Q_1^{n-1}, X) = g_k^n = p(q_k^n|g^{n-1}, x_n)$, where g_k^n represents the MLP output for class q_k at time t and $g^n = (g_1^n, \ldots, g_K^n)^T$ is the MLP output vector.

In this case, we assume that the conditional probabilities depend only on the current acoustic vector and on all the previous conditional probabilities (at time $n-1$). These probabilities can be generated by a truly recurrent MLP feeding back all the output values to the input field. This approach was already used in [Jordan, 1986] as a speech production model and has not been fully explored yet for speech recognition. Recently, an approach similar in spirit to this but feeding back hidden vectors instead of output vectors was successfully used in a hybrid HMM/MLP approach [Robinson & Fallside, 1990]. This makes sense since similar behaviors can probably be obtained whether the feedback is from the hidden or from the output units.[1] In this case, the local probabilities will depend on the probabilities of all the previous states. It is not clear, however, whether the optimality of DTW of the Viterbi algorithm is preserved.

Case 4: $p(q^n|Q_1^{n-1}, X) = p(q^n|X_{n-c}^{n+c})$.

[1] This is a great topic, and one which a number of us are examining now, but it falls outside of the scope of this book.

In this case the dependence on the previous state(s) is replaced by a dependence on the observations in a context of width $2c + 1$. These probabilities can be estimated by a feedforward MLP with a contextual input window containing, at time n, the set of acoustic vectors X_{n-c}^{n+c}. Although this is no longer strictly a Markov model, the same formalism applies and the probabilities estimated by the MLP can be used in standard Viterbi algorithms (training and decoding).

7.3 Problem

"Failure is an opportunity to learn" - In 1988, we tested the discriminant HMMs proposed above in the DECIPHER system [Cohen et al., 1990] developed at SRI International, one of the best speaker-independent, large vocabulary, continuous speech recognition systems, by simply replacing the local contributions by the discriminant probabilities obtained at the output of the MLPs discussed above. On the perplexity 1000 (no grammar) speaker-independent DARPA *Resource Management* (RM) task, a word recognition error rate of 130% was recorded [unsurprisingly unpublished].[2]

Given our obvious lack of basic understanding of some fundamental problems, we decided to focus on case 4 only, the easiest formalism. As shown in the following, this strategic retreat helped us to understand several basic principles of discriminant Markov models and their underlying problems. Now, using essentially the same approach, we have improved our accuracy to 70-80% on the same task (with no grammar) and are able to improve standard versions of DECIPHER significantly [Renals et al., 1992]. These results were only achieved after learning a number of things that were necessary to make this hybrid approach work. These will be discussed in the following sections.

[2]This was probably not due to any fundamental problem; we were most likely just ignorant of the practical details of how to make this approach work. Sometimes it is necessary to retreat to a simpler procedure to learn the basics, and this is what we did.

7.4 Methods for Recognition at Word Level

As shown by theoretical and experimental results in Chapter 6, MLP output values may be considered to be estimates of MAP probabilities for pattern classification. As shown in Section 7.2 these MLP outputs can also be used as local probabilities in discriminant HMMs. Either these or some other related quantity (such as the output normalized by the prior probability of the corresponding class) may be used in a Viterbi search to determine the best time-warped succession of states (speech sounds) to explain the observed speech measurements. This hybrid approach has the potential of exploiting the interpolating capabilities of MLP while using the DTW procedure to capture the dynamics of speech. In this way, most of the drawbacks of standard HMMs (i.e., lack of discrimination, *a priori* choice of probability density functions, poor contextual information) are tackled by the MLP while the temporal character is handled by the HMM formalism. For continuous speech training, an iterative process alternating MLP training and Viterbi matching can be used to improve initial segmentation points. In this case, Viterbi matching provides us with a segmentation of the training material and, consequently, with the target function needed for training the MLP. Convergence of this process has been proved in [Bourlard & Wellekens, 1990].

However, to achieve word level performance comparable to what we had achieved at the frame level, several modifications of this basic scheme were necessary [Bourlard & Morgan, 1990; Morgan & Bourlard, 1990b].

7.4.1 MLP Training Methods

Cross-validation

Our first major improvement came in the form of a fundamental change in our training strategy. The networks that were ultimately successful in the recognition task used hundreds of thousands of parameters. Originally we trained these networks by a criterion based on training set classification performance. This resulted in overtraining and very poor generalization performance on the test set.

A new cross-validation training algorithm was designed in which the stopping criterion was based on performance for an independent validation set [Morgan & Bourlard, 1990a]. In other words, training was stopped when performance on a second set of data began going down, and not when training error leveled off. This greatly improved generalization, which could be further tested on a third independent test set.

For more information about this cross-validation procedure, see Chapter 12.

Other Improvements

Among other improvements in our training method (including some developed more recently), were:

1. On-line training – instead of off-line training... of course! See Section 4.3.6.

2. Training criterion – using the relative entropy instead of LMS criterion. This was particularly helpful for avoiding local minima and speeding up convergence. The correction resulting from this criterion is always linear and does not saturate if we are on one of the tails of the sigmoid (where the correction is negligible)

3. Initialization of output biases – as mentioned in Section 6.4.4, consistent with some theoretical arguments, it was observed that output biases roughly encoded class priors. As a consequence, we initialized the output biases to the log (odds) of the *a priori* probabilities of the classes as they were represented in the training data. This appeared to significantly speed up the training process, as well as slightly improve the results.

4. Random pattern presentation – In earlier forms of our analysis we presented the data sequentially according to the speech signal. This can be slow, since it requires a very low learning rate in the case of on-line training. In another version, we scrambled the data to make sure that a different class was presented for every update of the network; this led to better results and faster convergence. More recently, we presented the speech vectors at random (while preserving the priors!), which further speeded up MLP training, and also slightly improved the results.

5. Learning heuristics – We found it helpful to reduce the learning rate when cross-validation indicated that a given rate was no longer useful. Additionally, we ultimately found that after the first reduction, only a single epoch at each rate was useful, so we no longer ran the

second epoch for any rate after the first. This heuristic almost cut down the number of epochs by two, and had essentially no effect on final performance. See Section 7.7.1 for further discussion.

6. Fast hardware – the RAP machine developed at ICSI (see Chapter 11). While not an algorithmic improvement, having fast hardware and flexible software permitted us to shift from discrete features (for which nets could be trained on our workstations) to continuous features, which gave us much higher performance.[3]

7.4.2 Posterior Probabilities and Likelihoods

In the original scheme [Bourlard & Wellekens, 1990] proposed in Section 7.2, it was shown that MLP outputs can be used in discriminant HMMs directly. However, while this helped frame performance, it hurt word performance (see tables in Chapter 6 and Table 7.1). We ultimately realized that this was (at least partly) due to a mismatch between the relative frequency of phonemes in the training sets and the priors imposed by the topology of the HMMs used during training. As discussed in Section 7.8, this is a typical problem of discriminant training. Division by the prior class probabilities as estimated from the training set removed this effect of the priors on the DTW. This led to a small decrease in frame classification performance, but a large (sometimes 10 - 20% in raw percentages) improvement in word recognition rates (see Table 7.1 and accompanying description). See Section 7.8 for further discussion about this problem.

7.4.3 Word Transition Costs

During word recognition we observed that to balance the number of insertions and deletions (as is usually done in any continuous speech recognition, or CSR, system) we consistently needed to increase the word transition cost for larger contextual windows. In phenomenon can be observed in Table 7.1 where the "optimal" word transition costs are reported for two different MLP contextual widths.

A reasonable value for this can be determined from recognition on a small number of sentences (e.g., 50), choosing a value which results in

[3] It is a cliche that algorithms can have a much bigger effect on run time than a faster computer. This is true, but try running an N-body problem on your hand-calculator sometime. N had better be 1.

insertions at most equal to the number of deletions. While this type of tuning is common to HMM systems, the systematic correlation to the width of the input context was something we had not realized prior to these experiments.

This can be explained in terms of the degree of correlation of the successive patterns presented at the input of the MLP. As noted earlier, MLPs can indeed make better frame level discriminations than simple statistical classifiers, because they can easily incorporate multiple sources of evidence (multiple frames, multiple features) without simplifying assumptions. However, when the input features within a contextual window are roughly independent, the Viterbi algorithm will already incorporate all of the context in choosing the best HMM state sequence explaining an utterance. If emission probabilities are estimated from the outputs of an MLP which has a $2c+1$ frame contextual input, the probability to observe a feature sequence $\{x_1, x_2, \ldots, x_N\}$ (where x_n represents the feature vector at time n) on a particular HMM state q_k is estimated as:

$$\prod_{n=1}^{N} p(x_{n-c}, \ldots, x_n, \ldots, x_{n+c} | q_k)$$

where Bayes' rule has already been used to convert the MLP outputs (which estimate MAP probabilities) into MLE probabilities. If independence is assumed, and if boundary effects (context extending before frame 1 or after frame N) are ignored (assume $(2c+1) \ll N$), this becomes:

$$\prod_{n=1}^{N} \prod_{j=-c}^{c} p(x_{n+j}|q_k) = \prod_{n=1}^{N} [p(x_n|q_k)]^{2c+1}$$

where the latter probability is just the classical Maximum Likelihood solution, raised to the power $2c+1$. Thus, if the features are independent over time, to keep the effect of transition costs the same as for the simple HMM, the log probabilities must be scaled down by the size of the contextual window. Note that, in the more realistic case where dependencies exist between frames, the optimal scaling factor will be less than $2c+1$, down to a minimum of 1 for the case in which frames are completely dependent (e.g., same within a constant factor). The scaling factor should thus reflect the time correlation of the input features. An equivalent effect is achieved by increasing the word entrance penalties, so that they are raised to a power that is comparable to the scaling of the log probabilities.

Thus, if the features are assumed independent over time, there is no advantage to be gained by using an MLP to extract contextual information for the estimation of emission probabilities for an HMM Viterbi decoding. In general, the relation between the MLP and MLE solutions will be more complex, because of interdependence over time of the input features. However, the above relation may give some insight as to the difficulty we have met in improving word recognition performance with a single discrete feature (despite large improvements at the frame level). Stated more positively, our results show that the probabilities estimated by MLPs can be used at least as effectively as conventional estimates, and that some advantage can be gained by incorporating more information for the estimation of these probabilities.

7.4.4 Segmentation of Training Data

Much as with HMM systems, an iterative procedure was required to time align the training labels in a manner that was statistically consistent with the recognition methods used. In one experiment, we segmented the data using an iterative Viterbi alignment starting from a segmentation based on average phoneme durations, and terminated at the segmentation which led to the best performance on an independent test set (see Section 7.6 for further discussion about this). This technique consistently improved our concomitant word recognition to be better than we could achieve either with the earlier HMM/MLP technique or the pure HMM technique (using, for both cases, HMMs with a single distribution for each phoneme, and a single vector-quantized feature).

7.4.5 Input Features

For the experiments reported in this chapter, simple vector-quantized inputs were used. Since the RAP (Chapter 11) was not available at the beginning of our work, this choice mainly aimed at reducing the CPU requirements for training. In this case, we had binary inputs and only additions (for the bits that were "on" at the input) between the input and hidden layer. However, this resulted in a very large input layer for which no hidden layer was necessary. It can be shown that the probability that binary vectors are orthogonal (and, consequently, linearly separable) tends quickly to 1 as the dimension of the space increases (see, e.g., [Kanerva, 1988]). However, this also explains why some of the reported results were low by current standards (although we compared them with equivalent HMMs).

We will see in the next chapter that incorporating good continuous input features improves performance significantly.

7.4.6 Better Speech Units and Phonological Rules

In our earlier developments we only used a small number of phoneme-like classes limited to about 60; no models of phonemic context were used. Additionally, we used a single-pronunciation lexicon of concatenated phoneme models. Later on, we extended our phoneme set somewhat, used context-dependent models (Chapter 9), and integrated our system in a full-scale recognizer that used phonological rules to generate multiple pronunciations.[4] Although this always improved our recognition performance, it is still interesting to develop a simple context-independent system. Recently [Robinson et al., 1993] we have seen that the performance we can achieve with hybrid HMM/MLP approach and context-independent phonemes is comparable to what has been achieved with much larger and complex systems. The hybrid approach, however, leads to a simpler system that contains many fewer parameters. It can be argued that a simple HMM described in terms of very powerful emission probabilities (ideally containing all the information about the input and output sequence correlation) may be more general and potentially better than a very complex HMM in which each state is modeled by "simple" pdfs.

7.5 Word Recognition Results

In the first set of experiments, we used the speaker-dependent German database (called SPICOS [Ney & Noll, 1988]) that was used in Chapter 6 to assess our frame performance with MLPs. The speech had been sampled at a rate of 16 kHz, and 30 points of smoothed, "mel-scaled" logarithmic spectra (over bands from 200 to 6400 Hz) were calculated every 10-ms from a 512-point FFT over a 25-ms window. For these experiments, the mel spectrum and the energy were vector-quantized to

[4] This was the SRI DECIPHER system, and the integration, as well as much further development, was largely done by Mike Cohen, Horacio Franco, and Victor Abrash at SRI.

pointers into a single speaker-dependent table of prototypes. The architecture and training procedure for the MLP were already described in Section 6.6.2. The output layer of an MLP (trained on automatically-labeled phonemes) was evaluated for each frame, and (after division by the prior probability of each phoneme) was used as the emission probability in a discrete HMM system. In this system, each phoneme k was modeled with a single conditional density associated with the particular MLP output class q_k, repeated $D/2$ times, where D was a prior estimate of the duration of the phoneme. Only selfloops and sequential transitions were permitted. The generic phonemic HMM and MLP models are respectively represented by Figure 7.1 and 7.2.

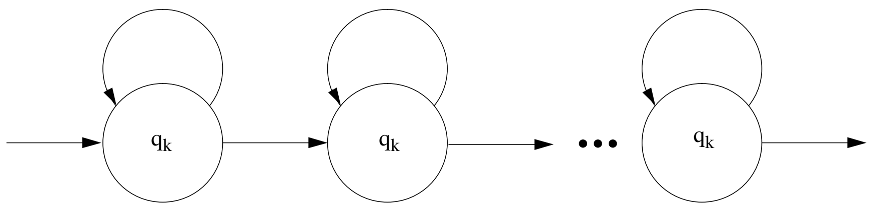

Figure 7.1: Generic phonemic HMM with a single conditional density estimated on the k-th MLP output unit associated with HMM state q_k, and a single state repeated $D/2$ times, where D is the average duration of the phoneme.

A Viterbi decoding was then used for word recognition, with words built up by concatenating phoneme models. We evaluated on the first hundred sentences of the test session (on which word entrance penalties were optimized), and our best results were validated by a further recognition on the second hundred sentences of the test set. Note that this same simplified HMM was used for both the MLE reference system (estimating probabilities directly from relative frequencies) and the MLP system, and that the same input features were used for both.

Table 7.1 shows the recognition rate (100% − error rate, where errors includes insertions, deletions, and substitutions) for the first 100

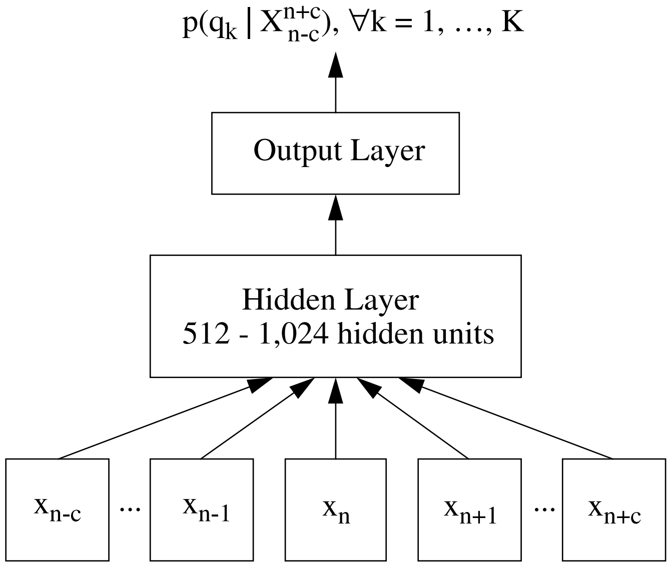

Figure 7.2: Generic MLP for probability estimation. Hidden layer is used for continuous (real-valued) input X, no hidden layer required for discrete (binary) X. In the simplest case, X is the feature vector from a single frame, but it can include features from surrounding frames as well. The q_k's are the classes (associated with HMM states) for which the MLP is trained as a classifier.

system method	size of context	word transition cost	% correct test	validation
MLP	1	10^{-7}	27.3	
MLP/priors	1	10^{-7}	49.7	
MLP	9	10^{-14}	40.9	
MLP/priors	9	10^{-14}	51.9	52.2
MLP/priors (no hidden)	9	10^{-14}	53.3	
MLE	1	10^{-7}	52.6	52.5

Table 7.1: Word recognition rate on SPICOS database (speaker m003) for different hybrid HMM/MLP approaches (MLP = no division of output values by priors, MLP/priors = division by priors) compared with standard HMMs trained with MLE criterion.

sentences of the test session. All runs except the last were done with the 20 hidden units in the MLP, as suggested by the results above. Note the significant positive effect of division of the MLP outputs, which are trained to approximate MAP probabilities, by estimates of the prior probabilities for each class (denoted "MLP/priors" in Table 7.1).

Not shown here are the earlier improvements required to reach this level of performance, which were primarily the modifications to the learning algorithm described above. Additionally, word transition probabilities were optimized for both the Maximum Likelihood and MLP style HMMs. This led to a word exit probability of 10^{-7} for the MLE and for 1-frame MLP's, and 10^{-14} for an MLP with 9 frames of context. These optimal word transition costs are also given in Table 7.1 where it can be observed that larger input windows require much smaller word transition costs. The reason for this is explained further in this section.

After these adjustments, performance was essentially the same for the two approaches. Performance on the last hundred sentences of the test session (shown in the last column of Table 7.1) validated that the two systems generalized equivalently despite these tunings.

An initial time alignment of the phonetic transcription with the data (for this speaker) had previously been calculated using a program incorporating speech-specific knowledge [Aubert, 1987]. This labeling had

method	context	test
MLP/priors (no hidden)	9	65.3
MLE	1	56.9

Table 7.2: Word recognition rate on SPICOS (speaker m003) using iterated Viterbi segmentation: MLP/priors = hybrid HMM/MLP approach with MLP output values divided by priors, MLE = HMMs using standard maximum likelihood estimate.

been used for the targets of the frame-based training described above. We then used this alignment as a "bootstrap" segmentation for an iterative Viterbi procedure, much as is done in conventional HMM systems (described further in the next section). As with the MLP training, the data was divided into a training and cross-validation set, and the segmentation corresponding to the best validation set frame classification rate was used for later training. For both cross-validation procedures, we switched to a training set of 150 sentences (two repetitions of 75 sentences) and a cross-validation set of 50 sentences (two repetitions of 25 each). Finally, since the best performance in Tables 6.2 and 7.1 were achieved using no hidden layer, we continued our experiments using this simpler network, which also required only a simple training procedure (entropy error criterion only). Table 7.2 gives the performance for the full 200 recognition sentences (test + validation sets), which shows a distinct improvement over the simpler MLE approach.[5]

7.6 Segmentation of training data

A problem in applying MLP methods to speech is the apparent requirement of hand-labeled frames for MLP training. To remove this obstacle, we have worked on embedding the MLP training in a Viterbi algorithm, iteratively improving an initial segmentation as suggested earlier

[5]When we first showed this result, a common comment was, "You are comparing apples and oranges. The MLE result uses 1 frame of input and the MLP uses 9." Precisely. The point here is the difference between the apple and the orange. The MLP provides a convenient and flexible way to incorporate the information from the 9 frames, while the MLE does not.

in [Bourlard & Wellekens, 1990]. Preliminary results show that we can generate a segmentation from a simple (e.g., linear) initialization, much as is done in conventional HMM systems.

As we still want to use cross-validation techniques for the MLP training and for the Viterbi matching in an unsupervised way, several modifications of the original scheme were necessary, and are briefly described here. Starting only with the phonetic transcription of the training and cross-validation sets, these two sets of sentences are linearly segmented, respectively providing the MLP output targets for the training set as well as for the cross-validation set. The MLP can then be trained (using cross-validation), which provides new weights and, consequently, new emission probabilities for the Viterbi matching. Using this newly trained MLP, Viterbi matching is performed on the training and cross-validation sets, providing us with new segmentations and, consequently, with new output targets for MLP training and cross-validation. This process is iterated until the score (product over all the optimal path probabilities) on the cross-validation set (and not on the training set) begins to decrease. Thus, two cross-validations take place in this process: one for the MLP itself, and one for the Viterbi matching. This new Viterbi training has been observed to converge to segmentations very close to the bootstrap segmentation. In particular, for 200 sentences, less than 4% of the frames were labeled differently by this automatic procedure than they had been by careful classification of the frames using the bootstrap procedure. This result, showing the effectiveness of embedding MLP training in a Viterbi segmentation, appears to have removed a major handicap of MLP use, the requirement for hand-labeled speech.

We duplicated our recognition tests for two other speakers from the same database. Since bootstrap segmentation data were not available for these speakers, we labeled each training set (from the original male plus a male and a female speaker) using a standard Viterbi iteration initialized from a time-alignment based on a simple estimate of average phoneme duration. This reduced all of the recognition scores, illustrating the effect of a good start point for the Viterbi iteration. However, as can be seen from the Table 7.3 results (measured over the full 200 recognition sentences), the MLP-based methods appear to do at least as well as the simpler estimation technique. In particular, the MLP system performed significantly better than the MLE ($p < 0.01$) for two out of three speakers, as well as for a multi-speaker comparison over the three speakers (in each case using a normal approximation to a binomial distribution for

speaker	MLE	MLP
m003	54.4	59.7
m001	47.4	51.9
w010	54.2	54.3

Table 7.3: Word recognition rates for 3 speakers on SPICOS, simple initialization of the Viterbi training.

the null hypothesis that the two systems were equivalent).

7.7 Resource Management (RM) task

7.7.1 Methods

For the DARPA RM data, initial development was done on a single speaker to confirm that the techniques we developed for the German data base were still applicable. Although we experimented slightly with this data, the ultimate system was substantially unchanged, with the exception of the program modifications required to use different VQ features (described below). Final reported scores are given for the 11 speakers which were left out in the development. For each speaker, we used 400 sentences for training, 100 for cross-validation, and a final 100 for recognition.[6] A transcription for each sentence was derived from the most likely pronunciations observed in a large speaker-independent database.[7] For each speaker, we initialized a Viterbi algorithm by assuming a segmentation obtained by assigning a length to each phoneme in the phonetic transcription which came from a table of average phoneme length (normalized to the length of the actual sentence). The labels were used to train an MLP on the 400 sentences for that person. Using probabilities generated by the MLP, the Viterbi was then used to generate phonemic frame labels. The training and Viterbi phases were then iterated until frame classification on the cross-validation set converged.

Input features used were based on the front end for SRI's DECIPHER system [Cohen et al., 1990], including vector quantized mel-cepstrum

[6] In later experiments, we trained on 500 sentences and used a separate evaluation set for recognition. Results were essentially the same.

[7] Supplied by M. Cohen of SRI.

(12 coefficients), vector quantized difference of mel-cepstrum, quantized energy, and quantized difference of energy. Both vector quantization codebooks contain 256 prototypes. Energy and delta energy were each quantized into 25 levels. A feature vector was calculated for each 10 ms of input speech. Each feature was represented by a simple binary input layer with only one bit "on." Nine frames of context were used as input to the network, allowing four frames of contextual information on each side of the frame to be classified. As we had found in our SPICOS experiments, a hidden layer was not useful when VQ features were used. The size of the output layer was kept fixed at 61 units, corresponding to the 61 phonemes to be recognized. The input field contained $9 \times 562 = 5,058$ units, and the total number of possible inputs was equal to 3×10^{68}. There were typically about 130,000 training patterns (from the 400 training sentences). Of course, this represented only a very small fraction of the possible inputs (or even of the inputs which are plausible for real speech), and generalization was thus potentially difficult. Training was done by the EBP algorithm using the relative entropy criterion. In each iteration, the complete training set was presented, and the parameters were updated after each training pattern. To avoid overtraining of the MLP, improvement on the cross-validation set was checked after each iteration. If the classification rate on the cross-validation set had not improved more than a small threshold, the adaptation parameter of the gradient procedure was decreased; otherwise it was kept constant. Training ended when improvement on the cross-validation set went below a second threshold. Performance was insensitive to the exact values of these thresholds. After some experiments with our development speaker (dtd05), we settled on an initial adaptation constant of .01 with no momentum term. The threshold for changing the learning constant was initially set at .5% improvement on the cross-validation set, but was then reset to an infinite value (i.e., forcing a change in the learning constant after every iteration) after the reduction from the first learning constant. This heuristic appeared to cut learning time roughly in half without adversely affecting performance. The learning constant was reduced by a factor of two for each change. The final stopping parameter was set at .5% improvement on the cross-validation set.

The output layer of the MLP was evaluated for each frame, and (after division by the prior probability of each phoneme) was used as the emission probability in a discrete HMM system. As with the SPICOS experiment, each phoneme was modeled with a single conditional density. A Viterbi decoding was then used for recognition of the first 30 sentences of the cross-validation set (on which word transition penalties were optimized). The trained system was then tested on the final 100 sentences for each speaker. Note that this same simplified HMM was used for both the Maximum Likelihood (ML) reference system (estimating probabilities directly from relative frequencies) and the MLP system, and that the same input features were used for both.

7.7.2 Results

Frame classification results, as reported in the previous chapter, showed statistically significant improvement for all RM speakers. Table 7.4 shows the word recognition rate (100% - error rate, where errors include insertions, deletions, and substitutions) for the 100 test sentences. For the MLE case, emission probabilities were computed by assuming independence between the four features and multiplying probabilities derived from relative frequency over the training set. The MLP, which incorporated 9 frames of context, provided significant improvement ($p < .001$) for every individual case, as well as for the pooled data. Thus, the MLP-based methods consistently show measurable improvement over the simpler estimation technique.

7.7.3 Discussion and Extensions

These results (all obtained with no language model, i.e., with a perplexity of 1,000 for a 1,000 word vocabulary) show some of the improvement for MLPs over conventional HMMs which one might expect from the improved frame level results. MLPs can sometimes make better frame level discriminations than simple statistical classifiers do, because they can easily incorporate multiple sources of evidence (multiple frames, multiple features), which is difficult to do in HMMs without major simplifying assumptions. In general, the relation between the MLP and MLE word recognition is more complex, because of interdependence over time of the input features. Part of the difficulty with good recognition in these experiments may have been due to our choice of discrete, vector-quantized features, for which no metric is defined over the prototype space. In the next chapter, it will be shown that it is possible to improve performance by using continuous features, e.g., mel cepstra or cepstra derived by *Perceptual Linear Predictive* (PLP) analysis [Hermansky, 1990a; Morgan et

speaker	MLE	MLP
jws04	48.2	62.3
bef03	39.3	56.7
cmr02	59.5	70.9
dtb03	49.8	61.2
das12	63.8	76.5
ers07	45.4	58.3
dms04	58.0	69.1
tab07	60.8	70.5
hxs06	60.9	76.3
rkm05	37.9	53.8
pgh01	50.4	63.6
mean	52.2	65.4

Table 7.4: Context-independent word recognition rate on the speaker-dependent DARPA Resource Management (RM) database, no grammar, discrete features.

al., 1991]. We have also applied the standard RM wordpair grammar (perplexity 60). Using these techniques, we were able to reduce word error to 4% for speaker-dependent RM test sets; more recently, using better systems, we have been able to do this well on speaker-independent recognition (to be described next chapter). It now appears that the probabilities estimated by MLPs may offer improved word recognition through the incorporation of context in the estimation of emission probabilities.[8] Furthermore, the results presented here show the effectiveness of Viterbi segmentation in labeling training data for an MLP. This result removed an apparent handicap of MLP use, the requirement for hand-labeled speech.

As noted previously, our best discrete HMM results were obtained using an MLP with no hidden layer. This suggests that, for the case of a single VQ feature, a single perceptron model is rich enough for the probabilistic estimation. This network can also be trained more easily than

[8]While the MLP case used a context input of 9 frames and the ML did not, we note that the MLE case did use delta features that were actually calculated over 9 frames.

networks with one or more hidden layers, particularly when an entropy criterion is used. However, MLPs using continuous inputs (see Chapter 8) required a large hidden layer (512-1,024 units) and, consequently, were impractical to train on standard workstations because of the excessive computation time requirement. Chapter 11 will discuss the development of computational capabilities to support these requirements.

7.8 Discriminative Training and Priors

As explained in the previous section, before using the output values of the MLP as probabilities for HMMs, it is necessary to divide them by the respective prior class probabilities $p(q_k)$ to get scaled likelihoods; these prior probabilities are estimated on the training data simply by counting the number of times each class q_k, $k = 1, \ldots, K$, appears in the training set.

While this generally degraded classification performance at the frame level, we observed it to be very important in order to get acceptable performance at the word level. Since it has been shown that, in theory, the training and recognition procedures used in standard HMMs remain valid for posterior probabilities one might wonder why, in practice, it is necessary to remove the effect of the priors for recognition.

This might be explained by a mismatch between the prior probabilities given by the training data and the prior probabilities imposed by the topology of the Markov models. In fact, once the HMM topology (including transition probabilities) is chosen, the prior probabilities are also fixed. For instance, if classes q_k represent phonemes, prior probabilities $p(q_k)$'s are fixed when word models are defined as particular sequences of phoneme models, i.e., particular HMM topologies. This discussion can be extended to different levels of processing: for instance, if q_k stands for sub-phonemic states and if recognition is driven by a language model, prior probabilities $p(q_k)$ are fixed by (and can be calculated from) the phonemic HMMs, the word HMMs and the language model. Of course, the ideal solution would be to infer the topology of all these models directly from the training data, by using a discriminant criterion which implicitly contains the prior probabilities. In this case, at least in theory, it would be possible to start from fully connected models and to determine their topology according to the priors observed on the training data. Unfortunately, this results in a huge amount of parameters that would require an unrealistic amount of training data to estimate them significantly. This problem was also raised in [Paul et al., 1991] in the framework of language models.

Since the ideal theoretical solution is not accessible in practice, it is usually better to get rid of the poor estimate of the prior probabilities as given by the actual training data and to replace our lack of training data by "*a priori*" phonological or syntactical knowledge. This conclusion applies to discriminant criteria in general and to neural networks in particular.

7.9 Summary

In this chapter we presented a new HMM formalism (discriminant HMMs) that can use discriminant local probabilities generated by MLPs. Several architectures with their respective advantages and problems have been discussed. Despite the simplicity of the theory, we learned that it was tricky to properly interface MLPs and HMMs. Although discriminant HMMs should theoretically lead to better performance, and has indeed been shown improving frame performance, it initially hurt word recognition. The reasons for this have been discussed, and an alternative approach has been presented that estimates scaled likelihoods from MLP outputs when the MLP is presented with a local acoustic context. Even in this case, several modifications of the basic scheme were necessary (i.e., cross-validation, increased word transition costs, and segmentation of training data) to get the expected improvements.

The experimental results that have been presented here show some of the improvement for MLPs over conventional HMMs that one might expect from the frame level results. MLPs can make better frame level discriminations than simple statistical classifiers, because they can easily incorporate multiple sources of evidence (multiple frames, multiple features), which is difficult to do in HMMs without major simplifying assumptions. However, the relation between the MLP and MLE word recognition is more complex, because of interdependence over time of the input features. Part of the difficulty with good recognition may also be due to our choice of discrete, vector-quantized features, for which no metric is defined over the prototype space. The features we have been using were chosen for their effectiveness in HMM systems, and different combinations may prove to be better for MLP inputs. In particular, we would expect that feature combinations that have not been vector-quantized should have more useful dependencies (both within-frame and over time) that the MLP may be able to learn and exploit. Despite these limitations, it now appears that the probabilities estimated by MLPs may offer improved word recognition through the incorporation of context in the estimation of emission probabilities.

As noted earlier, for this case of discrete inputs, our best results were obtained using an MLP with no hidden layer. This suggests that, for the case of a single VQ feature, a single perceptron model is rich enough for the probabilistic estimation. This network can also be trained more easily than networks with one or more hidden layers, particularly when an entropy criterion is used.

Furthermore, the effectiveness of Viterbi segmentation in labeling training data for an MLP has been shown. This result appears to remove a major handicap of MLP use, i.e., the requirement for hand-labeled speech, and allows us to handle more complex HMMs.

Chapter 8

EXPERIMENTAL SYSTEMS

Nothing is invented and perfected at the same time.
– Latin Proverb –

The work presented in this chapter (and indeed, some of the writing as well) has been done in collaboration with other members of the Realization group at ICSI (in Berkeley, CA), (and more particularly Chuck Wooters, Phil Kohn, and Steve Renals, currently at Cambridge), Hynek Hermansky of US West Advanced Technologies (Denver, CO), and more recently with Michael Cohen, Horacio Franco, and Victor Abrash of SRI (Stanford, CA). The results of this continuing work are presented here to show that the hybrid HMM/MLP approach can improve state-of-the-art large vocabulary, continuous speech recognition systems.

8.1 Introduction

In the previous chapter, we have described a phoneme based, speaker-dependent continuous speech recognition system embedding the Multilayer Perceptron (MLP) into a Hidden Markov Model (HMM). In this approach, the outputs of an MLP with contextual inputs are used as emission probabilities in an HMM recognizer.

Layered networks such as MLPs can be used as joint probability density estimators, providing a smoothed interpolation over multiple inputs. A possible advantage of the MLP over more conventional density estimates is that the former assumes neither a particular parametric form of the probability density, nor independence of the sources of evidence about the speech class. As described in the previous chapter, our early experiments with this technique were restricted to speaker-dependent tasks, and used multiple frames of vector-quantized mel cepstral features as inputs to the net. We have shown that MLP probability estimates yield better recognition performance than discrete likelihoods derived from the counts of vector indices for each phone label. When continuous input features are used, we expect more significant dependencies between features than in the discrete case; if this is true, an MLP advantage over conventional density estimates should be yet more obvious.

Further, when the MLP uses several neighboring analysis vectors as its input, it might incorporate temporal information about the given speech segment better than the current standard of vector quantized delta cepstrum. However, capturing the temporal aspect of speech by using a multi-frame MLP input has its drawbacks too: it increases the size of the net and therefore also the number of parameters the MLP needs to estimate from finite training data. Thus, given a fixed amount of training data, the power of the MLP to utilize the temporal information from the multi-frame input is limited. If we could find a succinct input speech representation which sufficiently describes the temporal information for a given speech segment, the MLP could be made smaller and computationally more efficient.

In Section 8.2, we extend the hybrid recognition system in several ways:

1. Continuous input features rather than discrete vector-quantized ones are used.

2. A speech representation which has been shown to be effective in preserving linguistic information while suppressing speaker-dependent variations, the *Perceptual Linear Prediction* (PLP) analysis technique [Hermansky, 1990a], is applied.

3. The use of dynamic spectral features [Furui, 1986] as additional inputs for our neural network estimators is investigated. The dynamic spectral features are estimated as a weighted combination of cepstral coefficients from several neighboring frames. When the

MLP is presented with several analysis frames at a time, it could in principle come up with a similar or even better weighting scheme. Therefore, we are also comparing performance of the network using the estimated dynamic features to the performance of the net with multiple-frame input of static features.

4. Besides the speaker-dependent experiments with the DARPA *Resource Management* (RM) database, we also experiment with TIMIT (a speaker-independent continuous speech database).

The systems into which we have previously integrated connectionist probability estimators were very simple: context-independent phone models, single density models (with duration modeling) and single pronunciations of each vocabulary item. In Section 8.3 we describe the integration of connectionist probability estimators into a large speaker-independent HMM continuous speech recognition system, SRI's DECIPHER [Cohen et al., 1990]. DECIPHER is a much richer system than the previous baseline systems we have used. It includes multiple probabilistic word pronunciations, cross-word phonological and acoustic modeling, context-dependent phone models, and models with multiple densities. We will also quickly describe the recent forms of our own recognizer, Y0,[1] a simpler system that used a single pronunciation per word and a single density per phone.

8.2 Experiments on RM and TIMIT

8.2.1 Methods

As in the experiments reported in the previous chapter, we did our development work using the training sentences from the *speaker-dependent* portion of the RM database. Initially, we used 400 sentences for training, 100 for cross-validation [Morgan et al., 1990a] to determine the right amount of the MLP training, and a final 100 for recognition. A transcription for each sentence was derived from the most likely pronunciations. In our discrete-feature experiments we used SRI's vector-quantized features [Murveit & Weintraub, 1988] and initialized a Viterbi algorithm by a segmentation obtained from an average length of each phoneme in

[1] Pronounced "Why nought."

the phonetic transcription. Relative frequency of co-occurrences of codebook indices and phonetic labels was then used to estimate the emission probabilities. The Viterbi search was then used iteratively to generate a frame re-labeling. This is the same segmentation procedure as used in the previous chapter.

For the *speaker-independent* TIMIT experiments, we used a subset of the TIMIT database consisting of 190 speakers, uttering 5 (SX) sentences each. These were further divided into a training set (152 speakers), and a test set (38 speakers).[2] Each group was roughly balanced to match the TIMIT overall balance for gender and for regional dialects. The hand-marked TIMIT segments (rather than the averaged phoneme lengths used in the speaker-dependent experiment) were used for the initial segmentation.

Besides the vector quantized mel-cepstral features, the PLP cepstral vectors were used as input features. Essentially, PLP features are the cepstral coefficients of the autoregressive all-pole model of the auditory-like spectrum, where the latter is a critical band integrated power spectrum with an equal-loudness pre-emphasis and a cubic root nonlinearity to simulate the auditory intensity-loudness relation.[3] These values were calculated from a Hamming-windowed 20 msec frame calculated every 10 msec. In the following experiments, we chose the PLP features because they are a succinct representation of the linguistically relevant portion of the speech signal [Hermansky, 1990a]. Typically we use the first 6 cepstral coefficients of the 5th order PLP model (log gain of the model included) for the speaker-independent experiments, and we use the first 13 cepstral coefficients of the 12th order PLP model in the speaker-dependent experiments. Such small vectors permit us to use large hidden layers without excessive requirements on the overall size of the MLP (and on the number of its free parameters).[4]

For training of the network, each of the PLP cepstral coefficients was normalized for zero mean and unity variance across all classes. The nor-

[2] Given the high perplexity of this speaker-independent data set (about 2000 for the TIMIT segment used), this was undoubtedly an insufficient training set.

[3] The main difference between the mel cepstrum and the PLP cepstrum is in the method of spectral smoothing, which is done by cepstral truncation in the mel case and by autoregressive modeling in PLP.

[4] In later experiments with more training data and fairly uniform speakers and recording conditions, we found that we could benefit from PLP orders ranging from 8 to 12, even in the speaker-independent case.

malization constants are currently saved for use in recognition; however, our real-time (on-line) recognizers typically require some additional kind of normalization.[5]

In addition to static PLP features, we tested the utility of several dynamic features. In particular, we tried estimates of the coefficient's slope and curvature [Furui, 1986; Hanson & Applebaum, 1990] (1st and 2nd temporal derivatives). Following [Furui, 1986], the temporal derivative was estimated as a linear regression line through the 9 neighboring frames (a 5 frame window also yielded almost identical results), which is equivalent to a summation of those frames under a triangular-like odd-symmetric butterfly window. The 2nd temporal derivative estimate is then equivalent to a summation of neighboring frames under a parabolic even-symmetric window.

After some initial experimentation, we used a 1024 unit hidden layer MLP in all experiments with the continuous PLP features. In the case of the VQ discrete features, we have used an MLP with no hidden layer, which yielded the best results in our previous discrete-feature experiments. In all cases, the outputs correspond to DARPA phones (61 for RM, 64 for TIMIT). Tests were run with a single frame's feature vector as input to the network, and subsequently with a multi-frame input of 9 feature vectors from the temporal context of the current frame. As before, training was done by an error back-propagation algorithm using an entropy criterion and a cross-validation approach to adaptively set the learning constant and to determine when to stop training. Performance was insensitive to the exact values of the initial learning constant for these experiments.

The trained systems were then tested on the final 100 sentences of the RM database (speaker dtd), or on 190 test sentences from our TIMIT subset (38 male and female speakers). In each case the recognizer used the Viterbi algorithm to determine the sequence of single-density phoneme models for which the observed features were most likely. Within-word transitions were restricted to self-loops and moves to the next state in a phone or word model. The latter was just a concatenation of phone models; no state-skipping was permitted. Transition probabilities between states were assumed to be equally likely for all within-word cases. A new-word transition probability that performed best for the

---

[5]At ICSI, we use the RASTA filtering approach, which does a kind of on-line normalization of the features [Hermansky et al, 1991].

TYPE	mel cep VQ (4 codebooks)	PLP	PLP +slope	PLP +slope +curvature
FR 1	51.2	51.5	66.3	69.1
FR 9	67.9	69.8	72.4	72.6
WR 1	47.6	50.1	63.0	67.1
WR 9	63.5	65.2	70.7	70.1

Table 8.1: Phonetic classification rate at the frame level and word recognition accuracies of the HMM/MLP approach on speaker dtd05 of the speaker-dependent RM database. FR and WR respectively stand for frame and word classification rate while the next digit (1 or 9) stands for the number of frames used at the input of the MLP.

first 30 sentences of the cross-validation set was used.

For comparison in the case of TIMIT, a Maximum Likelihood classifier was trained using a full-covariance Gaussian for each phoneme model.

8.2.2 Recognition Results

The MLP results are tabulated in Tables 8.1 and 8.2. These tables should be examined with some knowledge of the significance of small variations in frame or word accuracy. Results were tabulated from test sets of 37,901 and 62,693 frames for RM and TIMIT, respectively. The word recognition scores were based on 914 and 1457 words for the two cases. Using a normal approximation to a binomial distribution to represent random variation in classification scores, a reasonable criterion of "difference" between methods can be shown to be about .5% at the frame level and about 3% at the word level. Thus, results much closer than these should be considered roughly equivalent, and results further apart reflect a significant difference between methods.

For the cases shown, there is a clear upward progression in recognition accuracy from the Mel cepstral VQ discrete features, through the static continuous PLP features to the full continuous PLP feature set, which includes all the static features, the slope and the curvature.

Adding one level of the dynamic features (e.g., going from only the static features to the features which include the slope, or adding the curvature to the set of static and slope features) never helps as much as using the multi-frame input of the lower-level features. However, the one-frame data of the full set of the static, slope and curvature features yields essentially the same performance as the multi-frame input of the static features. This can be viewed in 2 ways: 1) given first and second derivatives, no more information is needed from the 9 frames of context, or 2) from the static data alone, the network can compute its own functions that are just as good as the derivatives. Still, the best performance is obtained from the multi-frame input of the full set of all static and dynamic features.

Since the experiment described above, we have applied the standard RM wordpair grammar (perplexity 60). Additionally, we have improved our training somewhat by using random pattern presentation, pre-setting of unit biases to negative values (log priors for the output units), and a softmax function at the output layer.

Using these techniques, we have achieved an average score of 4.0% error for the wordpair grammar case, and 18.3% error without a grammar. These scores, shown in Table 8.2, were achieved for 1200 RM development set sentences (100 per speaker) that were taken from the 1990 NIST CD, and using a single arbitrary choice for the word transition penalty (10^{-6}). Following a small amount of optimization of this parameter, we tested on the 300 Feb 1991 speaker-dependent evaluation sentences (25 per speaker) and got a score of 4.1% error for the wordpair grammar case, and 19.1% error without a grammar. These results are comparable to those of the sites that reported on this test set in the Feb 1991 DARPA evaluation.

For the single-frame cases of TIMIT, Table 8.3 also shows in brackets the results of the conventional statistical classifiers (maximum likelihood Gaussian with a full covariance matrix for the word recognition, and incorporating prior phoneme probabilities for the frame level recognition). At the frame level, the MLP is always significantly more accurate than the Gaussian, although the difference is small for the static features only. At the word level, this is also true for feature vectors that include the derivatives. These results indicate the non-Gaussian character of the augmented feature vectors. As expected, the feature sets which do better in the conventional classifiers also do better in the MLP classification. This supports the intuition that feature selection is not only important for conventional pattern classification, but for MLPs as well.

The results of Table 8.3 were produced at an early stage of the devel-

speaker	Perplexity = 998 MLP	Perplexity = 60 MLP
jws04	16.1	4.1
bef03	29.0	7.6
cmr02	18.6	4.1
dtb03	17.7	3.2
das12	10.5	1.8
ers07	25.0	4.9
dms04	11.9	1.6
tab07	13.4	2.3
hxs06	15.1	3.0
rkm05	30.8	8.6
pgh01	14.7	2.6
dtd05	17.0	3.8
mean	18.3	4.0

Table 8.2: Context-independent word error rates, RM, continuous PLP features.

TYPE	mel cep VQ (4 codebooks)	PLP	PLP +slope	PLP +slope +curvature
F 1	39.5	34.6 (33.6)	44.3 (39.2)	49.1 (43.3)
F 9	48.0	50.9	53.8	54.8
W 1	7.3	12.1 (10.9)	22.2 (16.5)	25.1 (18.6)
W 9	14.0	26.0	28.1	27.9

Table 8.3: Recognition accuracies of the HMM/MLP-based classification for the speaker-independent TIMIT database. F and W respectively stand for frame and word classification while the next digit (1 or 9) stands for the number of frames used at the input of the MLP. Accuracies of the Gaussian classifier, when available, are given in brackets.

opment of our system, particularly the speaker-independent tasks. They were all obtained with no explicit language model, i.e., a perplexity of 2200 for our subset of TIMIT. For the speaker-dependent RM case, we achieved a performance that was significantly better than our best discrete input score, even when the curvature is not used. A preliminary experiment with the continuous mel-cepstral 12 features, (plus power, and the slopes for these 12 features) on the TIMIT subset gave a score of 52.7% at the frame level (using a multi-frame input), which is slightly lower than the score shown in the table for PLP-5 plus slopes (53.8%). At the word level, these features yielded an accuracy of 28.3%, which is not significantly better than the result for the PLP-5 plus slopes case (28.1%). Both of these cases, as well as the VQ case, require the estimation of a fairly similar number of parameters (170,000 - 300,000), although the PLP case is near the bottom of this range. These results suggest that the improvements over mel cepstral VQ results are probably due to the use of continuous inputs for the MLP. They also show that, for the speaker-independent TIMIT subset, the PLP-5 is at least as good as the larger mel cepstral vector. A similar result for the RM speaker yielded frame and word scores of 72.5% and 67.9%, respectively. The latter score is somewhat smaller than the corresponding PLP + slopes case.

For both tasks, we see the benefit of the continuous PLP features and their temporal derivatives. For the TIMIT case, though, we have no clear comparison to previous work at other sites. However, for the speaker-independent RM task, Lee [Lee, 1989] reported 25.8% word accuracy on his baseline system and 40.1% accuracy on a later version of his system, which included the cepstral derivative and frequency-axis warping in the analysis, but which still used only context-independent phone models (as in our system). Given that his training set was more than 5 times larger than ours and that his task had less than half the perplexity of our experiment, we considered our 28% word recognition rate quite encouraging. We then applied these techniques to a speaker-independent task with more widespread evaluation on standard test sets, the Resource Management task (see Section 8.3).

8.2.3 Discussion

We may make a number of qualitative conclusions based on the quantitative results of these pilot experiments:

- The dynamic feature that estimates the instantaneous temporal derivative of the cepstral coefficients (slope) improves both word and phoneme level performance significantly. The curvature, while providing less improvement than the slope, clearly improves the performance further for the 1-frame case. These effects can be seen for both MLP and Gaussian cases, so that the result is not restricted to a particular classifier type.

- Given the whole 9-frame context, the net always performs better than when given only 1-frame input of the same features.

- Given the whole 9-frame context, the net always performs better than when given only 1-frame input of the same features appended with the vector of the one step higher-level temporal dynamic features (e.g., 9 frames of PLP are better than 1 frame of PLP plus slope).

- However, the 1-frame combination of the static PLP, the slope and the curvature features performs roughly as well as the multi-frame input of the static features. This finding can be used with advantage in reducing the MLP computational requirements.

- The very best performance is achieved using a 9-frame context window of static and all dynamic features. Further experiments are needed here to determine whether this is merely due to the effectively larger analysis window used in the computation of the 9 frames of derivatives.

- The MLP performs better than a Gaussian classifier that uses a full covariance matrix per phone, particularly when dynamic features augment the static ones. This indicates that the extended feature vector has a non-Gaussian multivariate distribution.

- The continuous PLP features give a better performance than discrete (vector-quantized) mel cepstral features; in fact, Tables 8.1 and 8.2 show that PLP with no temporal derivatives does at least as well as vector-quantized mel cepstrum including the derivatives. Initial tests suggest that this is largely due to the lack of quantization in the continuous features, although the pilot study discussed in the previous section suggests that the PLP results are somewhat better than the mel cepstral ones for the frame level on TIMIT, and the word level for the RM speaker. Further experiments, in particular with model order, are required to explore these differences more fully.

These results were achieved at a cost of over 100 TeraFLOPs. Such a computation would have taken several years on a SparcStation 1+, and was possible as a series of sporadic overnight runs because of a fast, parallel network training machine, the RAP [Morgan et al., 1990b; Morgan et al., 1992] (see Chapter 11).

8.3 Integrating the MLP into DECIPHER

The speaker-independent recognition experiments of the previous section were instructive, but did not employ a language model or have enough training data to yield good performance. Additionally, the systems into which we had previously integrated connectionist probability estimators were very simple: context-independent phone models, single density models (with duration modeling) and single pronunciations of each vocabulary item. Our next step was to integrate connectionist probability estimators into a large HMM continuous speech recognition system, SRI's DECIPHER [Cohen et al., 1990]. DECIPHER is a much richer system than the previous baseline systems we had used. It includes multiple probabilistic word pronunciations, cross-word phonological and acoustic modeling, context-dependent phone models, and models with multiple densities.

Word models are represented as probabilistic networks of phone models, specifying multiple pronunciations. These networks are generated by the application of phonological rules to baseform pronunciations for each word. In order to limit the number of parameters that must be estimated, phonological rules are chosen based on measures of coverage and overcoverage of a database of pronunciations. This results in networks which maximize the coverage of observed pronunciations while minimizing network size. Probabilities of pronunciations are estimated by the forward-backward algorithm, after tying together instances of the same phonological process in different words. Phonological rules can be specified to apply across words, adding initial or final arcs which are constrained to connect only to arcs fulfilling the context of the rule [Cohen, 1989; Cohen et al., 1990].

Context-dependent phone models include word-specific phone, triphone, generalized triphone, cross-word triphone (constrained to connect to appropriate contexts), and left and right biphone (and generalized biphone). All these models are smoothed together, along with context independent models, using the deleted interpolation algorithm. Most phone models have three states, each state having a self transition and a transition to the following state. A small number of phone models have two states, to allow for short realizations.

In addition to the DECIPHER integration, we have also been gradually improving our in-house system, Y0. This uses a much simpler model, typically incorporating a single density per phone and a single pronunciation per word.

8.3.1 Coming Full Circle: RM Experiments

As noted earlier, our initial failure to integrate HMM and MLP approaches was noted in experiments with the speaker-independent DARPA RM database. For several years we restricted our efforts to speaker-dependent recognition while we learned the "tricks of the trade." By 1991 we were ready to return to the original task. The RM database used a vocabulary of 998 words and no grammar (perplexity = 998) or a word pair grammar (perplexity = 60). We also collaborated with SRI International on this work, in particular by incorporating features from their HMM recognizer as described below.

For these experiments, a 12th order mel cepstrum front end was used,[6] producing 26 coefficients per frame: energy, 12 cepstral coefficients and derivatives of each static feature computed over a 4-frame window. The inputs to the MLP consisted of the current frame with 4 frames each of left and right context, totaling a feature vector length of 234. The MLPs that we used contained 512 hidden units (a number determined by empirical experiments, trading off representational power with computation) and 69 output units (corresponding to 69 monophone categories), giving a total of around 150,000 weights.[7] Stochastic gradient descent training typically required about 10 passes through the

[6] Despite success with PLP, mel cepstra were used here for comparison with other results from the SRI team.

[7] Our more recent experiments have all been done with 1000 hidden units, resulting in around 300,000 weights. SRI's context-dependent networks have had up to 1.5 million weights.

training database of 1.3 million frames.[8] This required less than 24 hours computation time, using a 5-board RAP.

To train an MLP we required a bootstrap model to produce time-aligned phonetic labels. In this case we used the context-independent DECIPHER system to perform the forced alignment between the training data and word sequence.

The baseline DECIPHER system modeled the output distributions using tied Gaussian mixtures. Training used the forward-backward algorithm to optimize a maximum likelihood criterion.

We used two sets of test sentences for evaluation. A 300 sentence development set (the June 1988 RM speaker-independent test set) was used to tune the HMM recognition parameters, such as the word transition penalty. The results reported here were obtained from a 600 sentence test set (the February 1989 and October 1989 RM speaker-independent test sets); no tuning of parameters was performed using this set.

8.3.2 Context-independent Models

We first experimented using context-independent models. A baseline context-independent DECIPHER system incorporated multiple pronunciations, cross-word phonological modeling, etc., but had only 69 of the two- or three-state phone models (200 distributions in all).

The baseline connectionist system (Y0) had 69 single distribution phone models; the lexicon consisted of a single pronunciation for each word. Each phone model was a left-to-right model (with self-loops) with N/2 states, where N was the average duration of the phone. Transition probabilities were all tied to be 0.5. The connectionist probability estimator was integrated into DECIPHER in two ways:

- DECIPHER alignments were used for training, and the single-best pronunciations from DECIPHER were used in recognition, but otherwise the simplified HMM described above was used with no further DECIPHER integration.
- The MLP probability estimator was used for a simplified form of DECIPHER. In this approach, the usual 2,3-state DECIPHER models were used, but each model had only a single output distribution (from the MLP). Thus the 2 or 3 states in a model shared

[8] Recent improvements in our training have reduced this to 5 passes while actually reducing the recognition errors.

		% error	
		Perplexity	
	Parameters	998	60
Y0 Baseline	155,717	36.1	12.8
CI-DECIPHER	125,769	44.7	14.0
CI-MLP-DECIPHER	155,717	30.1	7.8

Table 8.4: Results using 69 context-independent phone models. The baseline MLP system Y0 uses 69 single distribution models with a single pronunciation for each word in the vocabulary. The DECIPHER system also uses 69 phone models, each with two or three states 200 independent distributions in total. The MLP-DECIPHER system uses DECIPHER's multiple pronunciation and cross-word modeling.

a distribution. The maximum likelihood transition probabilities (which basically encoded duration information) were retained. Additional potentially significant contributions to the recognition process were multiple word pronunciations and cross-word phonological models.

Results for these context-independent systems are shown in Table 8.4. There are several notable aspects to these results:

- The MLP system using simple pronunciations and single distribution phone models has a lower error rate than the context-independent DECIPHER system, which uses multiple pronunciations and cross-word phonological modeling.
- Incorporating the MLP estimator into the context-independent DECIPHER system results in still better performance, lowering the error rate substantially from 12.8% to 7.8%.[9]

Since this experiment, further work with SRI [Cohen et al., 1992] has led to significantly improved scores on this task, incorporating improved

[9] Recent improvements to Y0 also improved the 12.8% to 7.1%, despite a reliance on single pronunciations and a single density per phone. This recent work was done as part of a collaboration between the authors and Cambridge University researchers Tony Robinson, Steve Renals, and Mike Hochberg. The biggest source of improvement was getting new people to find our old bugs.

training and larger hidden layers (typically 20-30% fewer errors, relatively speaking, than we describe in the table). Additional experiments in which the MLP probability estimates were smoothed with the probabilities from the more complex context-dependent version of DECIPHER showed significant improvements over either system alone. In one recent DARPA evaluation, this combination actually was the best-performing system of any that had been tested on the February 1991 test set (by a small margin).

As with the earlier experiments, this work showed that context-independent MLP probability estimators could perform surprisingly well. However, what remained to be answered is whether more detailed phonetic models (e.g., incorporating context) could still benefit from this approach. The next chapter will address the theoretical issues of this problem, and will report some recent work with SRI that confirms the viability of the general approach.

8.4 Summary

In this chapter, we have extended and tested our basic hybrid HMM/MLP approach in two ways:

1. Using continuous acoustic vectors and dynamic features (Section 8.2). As for standard HMMs, this significantly improves performance. Moreover, the resulting HMM/MLP performance has been better than what was achieved with the equivalent HMM system.

2. Integrating our approach in DECIPHER (using phoneme models only), one of the best large vocabulary, speaker independent, continuous speech recognition systems currently available (Section 8.3). In the first case, we simply replaced the DECIPHER probabilities by the MLP probabilities and we showed that this led to better performance. However, since our approach was limited to phoneme models, the performance was not quite up to that of state-of-the-art HMM systems using context-dependent models – however, it has recently been shown to be quite close, despite the use of many fewer parameters, and virtually no intelligent application of speech knowledge. As noted above, in a second experiment, context-dependent HMM probabilities were mixed with context-independent MLP probabilities, resulting in performance that was

actually better than that of a comparable state-of-the-art system.

Since the major weakness of the HMM/MLP approach seems to be the limitation to context-independent phoneme models, it is shown in the next chapter how to circumvent this problem and to use an MLP to estimate probabilities of thousands of context-dependent models.

Chapter 9

CONTEXT-DEPENDENT MLPs

It's a poor sort of memory that only works backwards.
– Lewis Carroll –

9.1 Introduction

Chapters 7 and 8 have shown the ability of Multilayer Perceptrons (MLPs) to estimate emission probabilities for Hidden Markov Models (HMM). In these chapters, we have shown that these estimates led to improved performance over standard estimation techniques when a fairly simple HMM was used. However, current state-of-the-art continuous speech recognizers require HMMs with greater complexity, e.g., multiple densities per phone and/or context-dependent phone models. Will the consistent improvement we have seen in these tests be washed out in systems with more detailed models?

One difficulty with more complex models is that many more parameters must be estimated with the same limited amount of data. Brute-force application of our earlier techniques would result in an output layer with many thousands of units, and a network with many millions of connections. This network would be impractical to train, both in terms of computation and learnability, using current-sized public databases. In each of our earlier studies, a simple context-independent trained network used a single output unit for each phone. As described in the previous chapter, for our recent Resource Management tests, we used 69 of these units. Were one to consider the coarticulatory effects from the right only, this number would expand out to 69^2, or over 4000. Considering both right and left context, we would require 69^3 units, or about 328,000. With a typical hidden layer of 500 units, we would have over 10^8 connections, which is far too many for a practical system.

Of course, HMM researchers have had a similar consideration in reducing the number of parameters in their VQ-based or tied-mixture-based systems. The solution has been, in one form or another, to use a reduced number of context-dependent models (typically a few thousand). However, this is still a large number. For instance, with 12,000 outputs and 1,000 hidden units, a network would still have over 12 million connections, which makes good generalization difficult for training sets of a million frames. Even if enough training data were available, networks with millions of parameters can be expected to take impractical amounts of time to train using back-propagation approaches, even with fast special-purpose machines such as the RAP (see Chapter 11).

In the approach reported here, we are able to estimate, without any simplifying assumptions, likelihoods for context-dependent phonetic models with nets that are not substantially larger than our context-independent MLPs, and that require only a small increase in computation.

9.2 CDNN: A Context-Dependent Neural Network

As described above, with a few assumptions an MLP may be viewed as estimating the probability $p(q_k|x_n)$ where q_k is a speech class or an HMM state $\in \mathcal{Q} = \{q_1, \ldots, q_K\}$, the set of all possible HMM states from which phoneme models are built up, and x_n is the input data (speech features) for frame n. If there are K such classes, then K outputs are required in the MLP. This probability may be considered "context-independent" in the sense that the left-hand side of the conditional probability contains no term involving the neighboring phones.

For a context-dependent model, we may wish to estimate the joint probability of a current HMM state with a particular neighboring phone. Using $\mathcal{C}$ to represent the set of possible contexts, we wish to estimate

$p(q_k, c_j|x_n)$, where $c_j \in \mathcal{C} = \{c_1, \ldots, c_J\}$. If there are J context classes, this will require $K \times J$ output units for an MLP estimator. However, if we use the definition of conditional probability, the desired expression can be broken down as follows [Morgan & Bourlard, 1992]:

$$p(q_k, c_j|x_n) \;=\; p(q_k|x_n)p(c_j|q_k, x_n) \tag{9.1}$$

Thus, the desired probability is the product of the monophone posterior probability and a new conditional. The former can be realized with the usual monophone network. Viewing an MLP as an estimator of the probability of the left side of a conditional given the right side as input, the second term can be estimated by an MLP trained to generate the correct context class given inputs of the current class and the speech input frame. The latter network only has as many outputs as there are context classes.

Equivalently,

$$p(q_k, c_j|x_n) \;=\; p(c_j|x_n)p(q_k|c_j, x_n) \tag{9.2}$$

where the two factors can be interpreted similarly.

Let us now consider the problem of triphone modeling in an HMM-based system. In that case, the main problem lies in the estimation of probabilities like $p(x_n|q_k, c_j^\ell, c_\ell^r)$ where c_j^ℓ and c_ℓ^r respectively represent the left and right phonemic contexts of state q_k. We show here how to get good estimates of these probabilities with MLPs, taking advantage of their discriminant and generalization properties. In this way, we will extend the advantages of our hybrid HMM/MLP approach to triphone models, as well as to other context-dependent models such as "generalized" triphones [Lee, 1989; Lee, 1990].

If one wants to model triphones, a straightforward approach could consist in having $K \times J \times J$ output units to model the $K \times J \times J$ possible contextual state probabilities. However, as noted above, such a solution is generally not reasonable given the excessive number of parameters to estimate in comparison with the (limited) amount of available training data. This is also a well known problem in standard HMM approaches.

9.3 Theoretical Issue

From simple statistical rules (the definition of conditional probability in terms of joint probability) and without any simplifying assumptions, the following relations hold:

$$p(q_k, c_j^\ell, c_\ell^r|x_n) = p(c_j^\ell|q_k, c_\ell^r, x_n)p(c_\ell^r|q_k, x_n)p(q_k|x_n) \tag{9.3}$$

and

$$p(q_k, c_j^\ell, c_\ell^r) = p(c_j^\ell|q_k, c_\ell^r)p(c_\ell^r|q_k)p(q_k) \tag{9.4}$$

During recognition, using Bayes's rule, $p(x_n|q_k, c_j^\ell, c_\ell^r)$ can then be estimated as:

$$\begin{aligned} p(x_n|q_k, c_j^\ell, c_\ell^r) &= \frac{p(q_k, c_j^\ell, c_\ell^r|x_n)p(x_n)}{p(q_k, c_j^\ell, c_\ell^r)} \\ &= \frac{p(c_j^\ell|q_k, c_\ell^r, x_n)p(c_\ell^r|q_k, x_n)p(q_k|x_n)p(x_n)}{p(c_j^\ell|q_k, c_\ell^r)p(c_\ell^r|q_k).p(q_k)} \end{aligned} \tag{9.5}$$

If we can estimate all the factors appearing in this expression, it will be possible to estimate the probability $p(x_n|q_k, c_j^\ell, c_\ell^r)$ needed for HMMs without any particular simplifying assumptions. Now, exploiting the conclusions we derived from the theory of our hybrid HMM/MLP approach for phoneme models (i.e., in classification mode, the output values of the MLP are estimates of the posterior probabilities of the output classes conditioned on the input), it can be shown that all the probabilities appearing in (9.5) can be estimated by a neural network (although $p(x_n)$ is ignored during recognition). Indeed:

- $p(c_j^\ell|q_k, c_\ell^r, x_n)$ can be estimated by the MLP represented in Figure 9.1 (referred to as MLP1) in which the output units are associated with the left phonemes of the triphones and in which the input field is constituted by the current acoustic vector x_n (possibly extended to its left and right contexts), the current state and the right phonetic contexts in the triphones (which are known during training).

- $p(c_\ell^r|q_k, x_n)$ can be estimated by the MLP represented in Figure 9.2 (referred to as MLP2) in which the output units are associated with the right phonemes and in which the input field is constituted by the current acoustic vector x_n (possibly extended to its left and right contexts) and the current state (associated with x_n).

- $p(q_k|x_n)$ is estimated by the same MLP as the one used for modeling context-independent phonemes (as done in the previous chapters and referred to as MLP3) where the input field contains the

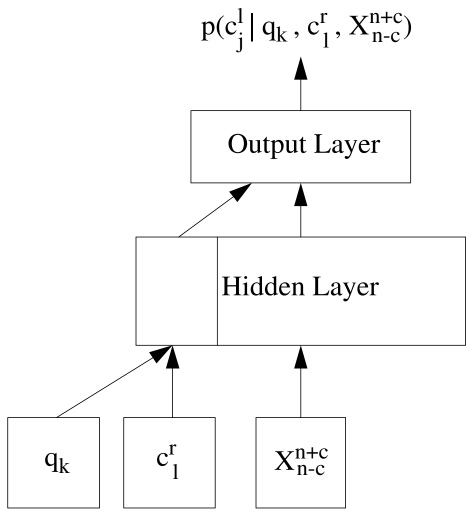

Figure 9.1: MLP estimator for left phonetic context, given input, current state, and right phonetic context.

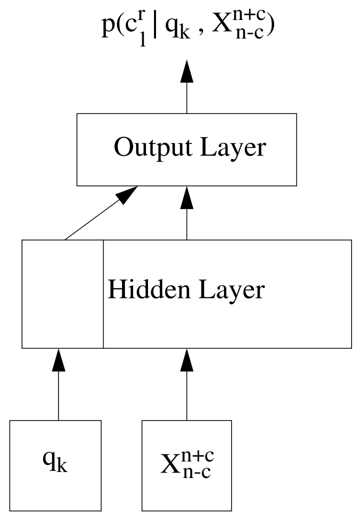

Figure 9.2: MLP estimator for right phonetic context, given input, and current state.

current acoustic vector only (possibly extended to its left and right contexts) and the output units are associated with the current labels.

- $p(c_j^\ell|q_k, c_\ell^r)$ can be estimated by an MLP in which the output units are associated with the left phonemes of the triphones and where the input field represents the current state and the right phonemes. This provides us with the *a priori* probability of observing a particular phoneme in the left part of a triphone given particular current state and right phonetic context.

- $p(c_\ell^r|q_k)$ can be estimated by an MLP in which the output units are associated with the right phonemes of the triphones and where the input field represents the current state. This provides us with the *a priori* probability of observing a particular phoneme on the right side of a particular state. Given the limited number of parameters in this model (i.e., $K \times J$), this probability can also be estimated by counting (i.e., this does not require a network).

- $p(q_k)$ is the *a priori* probability of a phoneme as also used in the standard hybrid HMM/MLP phonetic approach, and is simply estimated by counting on the training set (i.e., this also does not require a network).

The neural networks represented in Figures 9.1 and 9.2 are nothing else but standard MLPs with a simple restriction on the hidden layer topology. The reason for this restriction will be explained later on in Section 9.4.

By training different neural networks and using their output activation values in (9.5), it is possible to estimate, without any particular assumptions, the probability $p(x_n|q_k, c_j^\ell, c_\ell^r)$ that is required for modeling triphone probabilities to be used in HMMs. This generalizes to triphones the approach developed in the previous chapters and which was restricted to phoneme models. For limited training sets, these estimates may still need to be smoothed with monophone models, as is done in conventional HMM systems. Also, training improvements presented previously (such as the use of cross-validation to improve generalization performance) remain valid in this new approach. Additionally, if c^ℓ and c^r represent broad phonetic classes or clusters rather than phonemes, the above results apply to the estimation of "generalized triphones," such as are defined in [Lee, 1989]. As presented in the previous chapters, the input field containing the acoustic data (e.g., x_n) may also be supplied with contextual information. In this case, the x_n appearing in all the above (and subsequent) probabilities have to be replaced by X_{n-c}^{n+c}. This leads to the estimation of triphone probabilities given acoustic contextual information, which is even more important in the case of triphone models.

The procedure presented above reduces the training of a single network with $K \times J \times J$ outputs to the training of three smaller networks respectively with K, J and J outputs, and represents a generic way of splitting large MLPs used in classification mode into several smaller ones. It has the potential, however, of requiring much greater computation during the recognition phase. Indeed, if one implements this method naively, the second network must be computed K times for each frame during recognition, since the output probabilities depend on an assumption of the current class (corresponding to a monophone model in a hypothesized word sequence at that point in the dynamic programming). The next section will describe how this expense can largely be circumvented.

9.4 Implementation Issue

While the previous section gives a theoretical solution to triphone modeling with neural networks, an important implementation issue still has to be taken into account.

We have shown, in principle, how to transform, without any particular assumptions or simplifications, the huge neural network (which would result from the brute force application of our earlier hybrid HMM/MLP approach for phoneme modeling to triphones) to several smaller ones. Indeed, instead of having a single MLP estimating $p(q_k|x_n)$, we need to estimate

$$p(q_k, c_j^\ell, c_\ell^r|x_n) \tag{9.6}$$

which requires an MLP with $K \times J \times J$ output units. In the previous section, we have shown that it was possible to estimate the same probability with three smaller MLPs respectively estimating

$$p(c_j^\ell|q_k, c_\ell^r, x_n) \tag{9.7}$$

$$p(c_\ell^r|q_k, x_n) \tag{9.8}$$

and

$$p(q_k|x_n) \tag{9.9}$$

However, while this strongly reduces the memory requirement and the number of parameters, a naive implementation of these smaller networks would require much more computation.

In the case of phonetic modeling, a single MLP provided with the current acoustic vector x_n (possibly with its contextual acoustic vectors) as input can estimate $p(q_k|x_n)$ for all possible classes q_k on the associated output units. This remains valid for triphone modeling if we use the huge network (9.6) with x_n at its input and $K \times J \times J$ output units, each output unit being associated with a particular triphone. However, when this huge network is decomposed into smaller networks (9.7), (9.8), and (9.9), the first two networks must have input values depending on the current class [in (9.7) and (9.8)] and on the phonetic contexts [in (9.7)] constituting the triphones. For example, the input field of the network estimating (9.7) is constituted by the concatenation of the current acoustic vector x_n (possibly with its contextual acoustic vectors) and the middle (i.e., the current state) and right phonetic contexts in the triphones. Since the MLP training is supervised, i.e., we know exactly which triphone is associated with a particular acoustic vector, this is not a problem during training. However, this is no longer the case during recognition, where we do not know in advance which triphone is associated with x_n.

Therefore, in principle one would have to compute network activations at each frame for each possible phonetic context. This would amount to $J \times J$ times the monophone network computation, and would generally be prohibitive. Fortunately, a simple restriction on the network topology permits the pre-calculation of contextual phonetic contributions to the output; this computation can be done at the end of the training phase, prior to the recognition of any speech. By simply partitioning the net so that no hidden unit receives input from both phonetic context units and data input units, we can pre-calculate the contribution to the output units (prior to the output nonlinearity) for all possible combinations of left and right contexts, and form a table of these contributions. During recognition, the pre-sigmoid output values resulting from data vectors can be computed by a forward pass on the net for each frame. For each hypothetical triphone model, these contributions from the data inputs can be added to the corresponding context contributions from the table. The major new computation (in comparison with the monophone case) then is simply the cost of some look-ups, both for the contextual contributions, and for the final sigmoidal nonlinearity, which must now be re-computed for each hypothesized triphone (as opposed to once per frame, as in the monophone case). In practice this only doubles or triples the computation time, a reasonable cost for triphone models.

Let us consider the case of $p(c_j^\ell|q_k, c_\ell^r, x_n)$ computed by the neural network represented on Figure 9.1. More formally, letting $Y_j(q_k, c_\ell^r)$ be the contribution to the pre-sigmoid output for state q_j for the phonetic context-dependent partition of the net, and letting $Z_j(x_n)$ be the contribution to the pre-sigmoid output for state q_j for the data vector input. Then

$$p(c_j^\ell|q_k, c_\ell^r, x_n) = F(Y_j + Z_j) \tag{9.10}$$

where $F(x)$ is the standard sigmoid function.

A $(K \times J \times J)$-dimensional table Y is computed after network training by running the phonetic-context-dependent partition of the network (which has no inputs from the data vector) $K \times J \times J$ times, i.e., for all possible output units and for all possible combinations of phonetic contexts, with no output sigmoid computation. This table loading is a negligible amount of computation compared to the training of the network. During recognition, for each acoustic vector x_n, it is then enough to run the three networks (MLP1, MLP2, & MLP3) only once to get the contribution Z_j of the data inputs for each output unit q_j. For each hypothetical triphone model, this contribution Z_j just has to be added to the corresponding context contribution Y_j obtained by a simple look-up in table Y. In fact, this is equivalent to considering Y_j as an added bias (of output unit q_j) that depends on the phonetic context. Of course, the same method can be applied to (9.8) to compute $p(c_\ell^r|q_k, x_n)$. Also, for $p(c_j^\ell|q_k, c_\ell^r)$ and $p(c_\ell^r|q_k)$ it is sufficient to compute look-up tables at the end of the training phase for use in (9.5).

9.5 Discussion and Results

9.5.1 The Unrestricted Split Net

In equation (9.1), when splitting the original MLP with $K \times J$ output units into two smaller networks with K and J outputs respectively, the number of parameters is drastically reduced, which could affect the quality of the conditional distributions' estimation. However, parameter reduction is exactly the aim of the proposed approach, both to reduce computation and to improve generalization. As it was done for $p(q_k|x_n)$ it will be necessary to find (e.g., by using cross-validation techniques) the number of hidden units (and hence the number of parameters) leading to the best estimate of $p(c_j|q_k, x_n)$. The desired probabilities can in principle be estimated without any statistical assumptions (e.g., independence). Of course, this is only guaranteed if the training does not get stuck in a local minimum and if there are enough parameters.

9.5.2 The Topologically Restricted Net

As shown above, while reducing the number of parameters, the splitting of the network into two smaller networks results in much greater computation in the contextual network. To avoid this problem it is proposed to restrict the topology of the second network so that no hidden unit shares input from both q_k and x_n. Consequently, the q_k input only changes the output thresholds. This means that we have severely limited the ability to represent correlations between the inputs that have no hidden units in common.[1] However, a recent experiment with frame classification for continuous speech (trained using 160,000 patterns from 500 sentences uttered by a speaker in the Resource Management continuous speech recognition corpus) suggested that this did not affect the correct estimation of $p(c_j|q_k, x_n)$. In this example, the network with a split hidden layer predicted (for a test set of 32,000 patterns from 100 sentences) the correct right context 63.6% of the time, while a network with a unified hidden layer predicted the context 63.5% of the time, an equivalent figure.

9.5.3 Preliminary Results and Conclusion

Prior to experimenting with the CDNN for continuous speech recognition using biphone and triphone models (to be reported at a later date), we wanted to check experimentally that the split MLP was equivalent to the original one. We compared biphone probabilities generated by the original and split MLP for the speaker-independent Resource Management database. The number of hidden units in each MLP was chosen such that

[1] In later experiments with SRI, we have experimented with context-dependent networks that do develop representations of these correlations.

the number of parameters was approximately the same in both cases. After having trained both cases on 4,000 sentences, biphone probabilities were computed on a test set of 100 sentences pronounced by 4 different speakers, yielding a total of 17,012,088 probabilities. To compare both sets of probabilities we computed the correlation coefficient to be 0.65, and the mean absolute difference that was equal to 0.0017. Thus, the two sets of probabilities are significantly correlated. This suggests that CDNN may be a good way to compute context-dependent probabilities with nets that have a limited number of parameters and that require an acceptably small increase in computation over the context-independent case. Looking at the differences between the probabilities (since the correlation coefficient is far less than 1), it appears that the factored net outputs somewhat smoothed values. That is, second or third-ranked outputs do not appear to be as suppressed in the factored case. This is not necessarily a bad result, as overly sharp discrimination at the frame level can sometimes hurt word level performance.

In recent collaborative work with SRI International [Cohen et al., 1992], this class of approach was tested for left and right generalized biphone context. Significantly, it was shown in [Cohen et al., 1992] how smoothing between context-independent and context-dependent probabilities can be done using cross-validation training initialized with context-independent weights. Additionally, they experimented with an alternate architecture in which the output layer was replicated for each of 8 possible broad phonetic contexts (8 for the left and 8 for the right, each associated with one state of a 3-state model). These experiments showed significant improvements over the simpler context-independent approach, eliminating roughly one-fourth of the errors in a speaker-independent Resource Management test set. Thus, statistical factorization of MLP probabilistic estimators appears to have practical significance. Ongoing research is exploring the use of different architectures and more detailed phonetic models.

9.6 Related Prior Work

A number of other researchers [Franzini et al., 1990; Haffner et al., 1991; Robinson & Fallside, 1990] have also recently reported experiments with MLPs to estimate local costs (sometimes expressed as log probabilities) for speech recognition, and some of this work has used continuous speech.

All of these systems have exclusively used context-independent phonetic models, in the sense that the probabilities or costs are estimated for simple speech units such as phonemes or words, rather than biphones or triphones. Numerous conventional HMM systems have been reported that use triphone or triphone-like context-dependent models [Lee, 1990; Murveit et al., 1989; Cohen et al., 1990; Paul, 1989; Paul, 1991; Schwartz et al., 1985]. In one recently reported case [Austin et al., 1991], the outputs of a context-dependent MLP were used to help choose the best sentence from the N best sentences as determined by a context-dependent HMM system. Recently, those authors have estimated simple context dependencies using an MLP, but they did so by training J MLPs for J different context classes. As noted earlier, our studies have thus far shown the advantage to using a relatively assumption-free discriminant probability estimator like the MLP. Other considerations related to memory bandwidth and computational requirements are discussed in the next chapter.

9.7 Summary

The early form of our hybrid HMM/MLP approach focused on HMMs that are independent of phonetic context, and for these models the MLP approaches have consistently provided significant improvements (once we learned how to use them). In this chapter, we extended this approach to context-dependent models, which has been shown in standard HMMs to be an important feature for robust recognizers. However, a brute force application of our initial approach would result in an MLP with a huge number of output units and, consequently, an excessive number of parameters to estimate in comparison with the (limited) amount of available training data. To overcome this problem, a general approach to split any big nets used for pattern classification ("1-from-K" classification) into smaller nets has been presented. However, while this appears to solve the problem of context-dependent modeling, it is at the cost of much more computation since each elementary net has to be run several times. Fortunately, it is shown that a simple restriction on the network topology allows us to save this extra computation. Preliminary experiments have been reported where it is shown that this is indeed an effective approach to estimate context-dependent probabilities and that the restriction on the network topology is actually not a problem. Later

SRI experiments showed that related techniques can give significant improvements in recognition performance [Cohen et al., 1992].

Chapter 10

SYSTEM TRADEOFFS

Entities should not be multiplied unnecessarily.
– William of Occam –

10.1 Introduction

The results described in the preceding chapters suggest the utility of connectionist approaches for probabilistic estimation in speech recognition. However, word accuracy is only one measure of a practical speech recognition technique. Any computational method requires resources in the form of storage and communication (memory) bandwidth, as well as the ability to do the required arithmetic. The number of parameters used for a particular technique also has consequences for training. A particular design choice implies some tradeoff between these requirements. Additionally, trained systems such as those considered here may require entirely different resources for training and recognition modes, and these will be traded off in different techniques.

We will briefly examine the resource requirements for HMMs with and without MLP probability estimation. We will consider two cases: a discrete HMM system, and a continuous density HMM. In both cases, we will assume a context-dependent system that uses triphones, and will ignore costs related to transition probabilities; the emphasis will be on local distance computation (i.e., HMM emission probability estimation) for dynamic programming (Viterbi decoding) in an HMM. While costs for dynamic programming will tend to dominate when no pruning is used, local distances can be an extremely important cost for a practical system.

10.2 Discrete HMM

A conventional HMM using vector-quantized (VQ) inputs requires a table look-up architecture. Density estimates are computed for each sound unit (e.g., monophone or triphone), and are smoothed together with weights determined by deleted interpolation [Bahl et al., 1983]. All of this training can be one or two orders of magnitude less computation than is required for the corresponding MLP system. However, for the discrete HMM, neither approach requires a prohibitive amount of computation, and can usually be done on an engineering workstation. Here, we only consider major tradeoffs for the recognition phase.

Storage - During recognition, each hypothesized state has a corresponding index to a sub-table of probabilities that can be read in from a random access memory and buffered in fast memory for each new VQ input. Letting N be the number of distinct densities, and M the total number of codebook entries (totaling over n features, e.g., 3 or 4), we require storage of NM probabilities. For a large system that would have a smoothed probability for each of 200,000 triphones, with around 500 VQ codes (actually split into several tables), this would mean 100,000,000 probabilities. Current requirements are somewhat less than this, and one could easily require far more for a system that included word models. Additionally, for each frame, a fast buffer is required for N probability values, and a significant amount of fast memory for relevant pointers.

For the MLP case, smoothed triphone densities are estimated by running three networks (as described in Chapter 9), each of which would be of moderate size, with a weak dependence on N. Storage is typically dominated by the weights. For a discrete input system, we have found that a hidden layer is unnecessary, so that the number of weights would be $3N^{1/3}MW$, where W is the input context window size in frames. For 9 frames of context and $N >> 125$, the storage is much less than what is required for the pure HMM. For example, for $N = 200{,}000$, there are less than 10^6 weights, requiring two orders of magnitude less storage than the pure HMM. This reduction in storage also corresponds to a reduction in the number of parameters that must be estimated, which is important for limited training sets.

Memory bandwidth - Using the assumptions from above, and a frame rate of 100 frames/second, the table look-up recognizer would require $100N$ accesses/second from main memory, For N of 200,000, this is feasible for page mode access on DRAM. For the MLP, memory bandwidth is dominated by the forward processing of the network. As noted above, a hidden layer is unnecessary for this case, so that a 4-feature input would require only a few thousand reads per frame. This has the same weak dependence on N as was noted for the storage, but has a dependence on n and W that the pure HMM does not. Specifically, the memory bandwidth required for the weights is $300N^{1/3}nW$ reads/second from the weight memory. For the typical value of N used here, and for $n = 4$ features, this is over an order of magnitude lower data rate than is required by the pure HMM.

For an existing hardware system that has the capacity for recognition of vocabularies of up to 60,000 words [Stoelzle et al., 1991], the fast buffer bandwidth requirements are dominated by the operand acquisition for dynamic programming, which would be the same for both systems.

Computation - Both systems require the same computations for the dynamic programming. The MLP system requires the additional computation of the forward propagation through the network. For the simple VQ input case, however, this only requires $300N^{1/3}nW$ additions/second. For the numbers used above, this is less than 10^6 additions/second, which is typically negligible compared to the additions required for the dynamic programming. Practically speaking, the computational requirements are essentially equivalent for the two approaches.

Summary - Table 10.1 summarizes the order-of-magnitude resource requirements for the two cases discussed here. Once again, only the costs for local distance computation are considered. N can range from 10^3 to 10^6, M is typically 10^2 to 10^3, n is taken to be 4, and W is commonly about 9. The table also gives "typical" values for an idealized system with $N = 10^4$ and $M = 10^3$.

For $N >> 1000$, storage and bandwidth requirements are considerably better for the MLP case. The local distance computation is always more complex for the network than for a simple look-up, but is negligible

	Storage	Memory Accesses/sec	Computation/sec
H	NM	$100N$	0
typ	10^7	10^6	0
M	$3N^{1/3}MW$	$300N^{1/3}nW$	$300N^{1/3}nW$
typ	10^6	10^5	10^5

Table 10.1: Comparison of resource requirements for local distance computation in the discrete case: hybrid HMM/MLP and pure HMM.Typical order-of-magnitude values are given for systems with 10^4 triphone or generalized triphone densities

in comparison to the core dynamic programming steps (adds, compares, and address calculations).

10.3 Continuous-density HMM

An HMM-based system with continuous input features typically uses tied mixtures of Gaussian probability densities [Paul, 1991]. For such a system, density values are computed during recognition, much as in the MLP approach, but in contrast with the table-driven discrete HMM. Training, which once again includes a deleted interpolation algorithm, generates a vector of weights for each sound unit (e.g., triphone) to linear combine outputs from the common pool of Gaussians. In this case, the training phase for the MLPs require a significant computational resource. Current trainings for triphone MLPs using the 4000-sentence Resource Management speaker-independent corpus require roughly 10^{14} floating point operations, which would take months on a SUN SparcStation 2 workstation. To support this training ICSI (International Computer Science Institute, Berkeley, CA) constructed a multi-DSP processor called the RAP [Morgan et al., 1990b], which reduces the time to an overnight run. This power is not universally available, so the increased training time is a clear disadvantage to the MLP-based approach.

As before, the detailed tradeoffs will be considered for the recognition phase.

Storage - For the pure HMM case, means and variances for each feature and each Gaussian must be stored. Additionally, mixture weights must be stored for each hypothesized state. Let N be the number of distinct blended densities, M the number of tied Gaussians, and L the number of continuous features per Gaussian. We require $2ML$ parameters for the pooled distributions, and kN parameters for the mixtures, where k is the average number of Gaussians used for a state density. Taking k to be 10, N to be 200,000 (as with the discrete HMM example), M to be 250, and L to be 20, total storage would be 10,000 + 2,000,000 locations, which is modest in comparison with the table-driven discrete HMM.

Our experiments have shown us that a hidden layer is required for continuous-input MLPs used to estimate emission probabilities. Each hidden unit plays a similar role to a Gaussian from the tied mixture pool, and can even take the same form when radial basis functions are used in this layer [Renals, 1990]. For M the number of hidden units, L, W and N defined as before, the number of weights in the three MLPs used for triphone estimation is $3(WL + N^{1/3})M$. Storage is again dominated by these values. For typical cases, this is an order of magnitude less storage than is required for the pure HMM. In a large vocabulary system, storage for both cases would be dominated by tables of pointers used in the dynamic programming.

Memory bandwidth - Using the assumptions from above, and a frame rate of 100 frames/second, the pure HMM recognizer would require $200ML$ reads/second to generate the Gaussians, and $100kN$ reads/second to get the mixture weights (assuming no pruning). For the sizes assumed above, this would be 10^6 reads/second for the means and variances, and 2×10^8 read/second for the mixture weights. For the MLP, the memory bandwidth is again dominated by the forward processing of the network, which would be $300(WL + N^{1/3})M$, or typically about 2×10^7 reads/second. This is an order of magnitude less than the pure HMM requirements, and again has a weak dependence on N, so that larger systems would scale better for the MLP case. Once again, fast buffer requirements for dynamic programming could be very high, and would be equivalent for both systems.

	Storage	Memory Accesses/sec	Computation/sec
H	kN	$100kN$	$100kN$
typ	10^5	10^7	10^7
M	$3(WL + N^{1/3})M$	$300(WL + N^{1/3})M$	$300(WL + N^{1/3})M$
typ	10^6	10^8	10^8

Table 10.2: Comparison of resource requirements for local distance computation in the continuous case: hybrid HMM/MLP (M) and pure HMM (H).

Computation - Both systems require the same computations for the dynamic programming. For the pure HMM, local distance computation is dominated by the $100kN$ multiply-accumulates per second to combine the Gaussians for each state. Once again, the network size (and thus the computation) is weakly dependent on N, and $300(WL + N^{1/3})M$ multiply-accumulates/second are calculated. For typical values of N (10^5), the MLP requires about an order of magnitude less computation.

Summary - Table 10.2 compares order-of-magnitude resource requirements for the continuous density case. Once again, only the costs for local distance computation are considered. N can range from 10^3 to 10^6, while M is typically 10^2 to 10^3. The table also gives "typical" values for an idealized system with $N = 10^4$ and $M = 10^3$, $k = 10$, and $L = 20$.

For $N = 10,000$, storage, bandwidth, and computational requirements are all somewhat better for the HMM case. When k is much larger, or when the mixtures are not tied, the figures are all comparable. Finally, for large N and k, as might be seen in systems trained with large databases, storage, bandwidth, and computational requirements are all somewhat better for the MLP case. For the pure HMM, this computation can be comparable to the dynamic programming costs, while the latter costs will generally dominate the MLP costs in a recognizer without pruning.

10.4 Summary

For discrete HMMs, MLP probability estimation provides a direct trade-off of computation for memory. This can actually be preferable for some systems, since fast additions or dot-products can often be easier to provide than fast random access from large memories.

Continuous density HMMs using tied Gaussian mixtures are much more similar to MLPs, and both approaches require computationally oriented architectures. However, for both discrete and continuous HMMs, context-dependent posterior densities can be factored into several functions that can each be generated by networks that require the training of a relatively small number of parameters. This suggests a potential for improvement in parameter estimation with finite training samples. This advantage must be balanced against a large increase in training time, which requires the use of fast hardware in the training environment.

Chapter 11

TRAINING HARDWARE AND SOFTWARE

Training is everything ... a cauliflower is nothing but a cabbage with a college education.
– Mark Twain –

11.1 Introduction

The previous chapter showed that the MLP probability estimation approach appears to be conservative in system resource requirements for the recognition process. In particular, a speech recognizer using MLP-based approaches scales well with more phonetic categories for requirements of storage, memory bandwidth, and numerical computation. This is particularly true when one takes into account the parameter-sharing that occurs in the hidden layer(s) of a continuous input system.

However, these apparent advantages in recognition come with a price; whereas HMM recognizers can be trained with a moderate amount of computation (typically an overnight on a SUN SparcStation 2 workstation), HMM/MLP hybrids require training runs that could take months on the same workstation. This is a strong deterrent against research in this area, one that tends to force much work to be in the arena of small or "toy" problems. On the other hand, computational capabilities of computers in general and workstations in particular are increasing so rapidly (recently by a factor of 1.6 each year) that this handicap is likely to be overcome for many researchers within a few years.

In the meanwhile, however, anyone contemplating experiments with MLP training for large speech recognition tasks needs access to computers that are significantly more powerful than current workstations. This power cannot simply be expressed in terms of MIPS and MFLOPS, since neural network applications require a significant memory bandwidth (between getting weights and activations to the arithmetic elements). This differs from the requirements of general purpose computing only in that data caches are typically of little use for large networks, since there is little reuse - each weight is only used once for each input pattern in a forward connection, and the reuse for updating cannot be done until errors have propagated back from the entire net. Thus, unless a data cache is large enough to hold the entire network, it is not of great use. This means that the computational rates in a typical workstation will be limited by DRAM access times, which have not scaled up with the peak computing numbers in engineering workstations.[1]

In order to provide the large memory bandwidth required for these applications, many research and commercial sites have designed and built specialized machines for connectionist computations. There are a large number of these projects, and a survey is beyond the scope of this book. However, many of the results described in chapters 8 and 9 were made possible by a machine that was built at ICSI for this purpose, the *Ring Array Processor* (RAP) [Morgan et al, 1992]. This chapter will summarize some key motivations for this project, and will briefly describe some of the key points of this architecture (though a detailed description of the design is beyond the scope of this book). The chapter will conclude with a brief discussion of successor projects.

11.2 Motivations

As noted above, complete connectionist speech recognition systems have so many parameters to optimize that training time is the main impediment to progress, often forcing the researcher to make suboptimal decisions. While such research is computationally intensive, the required power could be provided by one of the many large conventional computers being built (such as the Intel Paragon series). Researchers can often

[1]This situation may be changing, as a number of fast DRAM alternatives are being developed, such as a packet-based DRAM interface, e.g., the Rambus approach.

do very well with a large, fast uniprocessor machine, such as a vector supercomputer. However, the computations/dollar for these machines is quite poor in comparison to what can be achieved with a more custom architecture. One of the reasons for this is that connectionist computation can be done with moderate precision (commonly 8 bits or less for activations and 16 bits for weights). This not only reduces the size of arithmetic units, but more importantly reduces memory and memory bandwidth requirements significantly. Memory interfaces, operating systems, and many other aspects of the system become much simpler. In fact, recent experiments at Berkeley showed ICSI's RAP machine to be significantly more effective than a single-head Cray X-MP for a variety of connectionist computations, but the RAP was less than 1% of the cost of the CRAY.[2]

Many research centers will be able to acquire specialized connectionist supercomputers, while only centers that can afford multi-million dollar machines (and correspondingly large maintenance costs) will be able to get a conventional supercomputer. Even in the latter case, such installations usually have a large number of users (to justify the expense), so that the turn-around time and total available CPU operations is frequently disappointing. Additionally, the smaller physical size, power, and cooling requirements of the more specialized machine makes it a better candidate for embedded applications. Some of these applications may also include a requirement for a soft real-time connection to external sensors and actuators, something that is also more straightforward for a machine that has been designed with this requirement in mind.

General-purpose supercomputers may not satisfy the needs for most connectionist researchers. However, as noted previously, workstation performance is improving by roughly a factor of 60% per year. They are also two orders of magnitude less expensive than the smaller supercomputers referred to above, and are currently only one order of magnitude less powerful (for instance, using 1992 Linpack benchmarks for the Digital Equipment Alpha 3000/500). However, parallel computers using similar technology and relying on moderate wordlength assumptions can significantly out-perform commercial uniprocessor workstations. This has been shown to be true for different projects over the last decade,

[2]A uniprocessor Y-MP C90, introduced in 1992, runs about 3-4 times faster than the X-MP on standard Linpack benchmarks, and so would most likely be a bit faster than a RAP on our problems. It is still two orders of magnitude more expensive.

and will probably continue to be true unless the workstation manufacturers decide that there is some advantage for them to pursue fast, efficient low-precision arithmetic—but if so they will hopefully "profit" from our experience. Even if specific academic projects result in implementations that are not faster than commercial computers, they often teach us much that can strongly affect the entire field of computing. An example at U.C. Berkeley was the RISC development. While the RISC I and II chips were not faster than computers that one could buy at that time, they showed the strengths of this style of computing quite convincingly. RISC is now ubiquitous.

Thus, it appears that at least at this time there are significant advantages to be gained by the development of fast machines (and accompanying software) that are designed to be efficient for connectionist algorithms. We now also have a significant amount of experience with the first machine developed in our community for this purpose, the RAP.[3]

11.3 A Basic Neurocomputer Design - the RAP

11.3.1 The ICSI Ring Array Processor

In 1989 researchers at ICSI designed and implemented the RAP for fast execution of the hybrid HMM/MLP continuous speech recognition training algorithms that are described in this book. The RAP uses multiple floating point DSP chips (the TI TMS320C30). It also uses a low-latency ring interconnection scheme using programmable gate array technology and a significant amount of local memory per node (16 MBytes of dynamic memory and 1 MB of fast static RAM) (figure 11.1).

Theoretical peak performance is 128 MFlops/board, with sustained performance of 30-90% for back-propagation problems of interest to current users. Each board contains four processor nodes. Systems with up to 40 nodes have been tested, for which throughputs of up to 574 Million Connections Per Second (MCPS) have been measured, as well as learn-

[3] We will not go into great detail about hardware and software system design here, and therefore we must neglect mention of a great many similar neurocomputer design projects around the world. We briefly mention here the CNAPS machine from Adaptive Solutions, which has less weight memory than we need for our work but has higher computational throughput for smaller nets; and the SYNAPSE machine from Siemens, which is a systolic array with very high performance that has recently been completed.

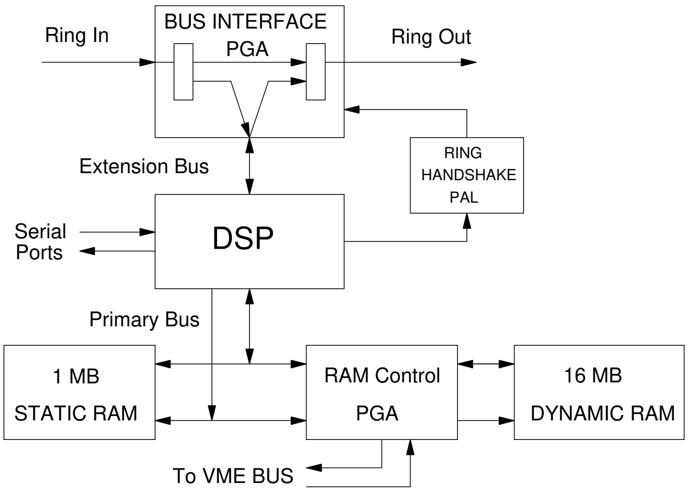

Figure 11.1: RAP node.

ing rates of up to 106 Million Connection Updates Per Second (MCUPS) for training. While the system is tuned to these algorithms, it is also a fully programmable computer, and users code in C++, C, and assembly language. Additionally, an object-oriented library of simulation classes called CLONES [Kohn, 1991] is used for the construction of network architectures and algorithms.

The DSP chips were originally chosen for their efficiency at multiply-accumulate operations, as well as a high memory bandwidth. However, their capability for general-purpose computing and the availability of a C compiler made further development as a real computing tool a viable option. While key connectionist operations were coded in assembly language (in the form of matrix-vector libraries) for efficiency, general code to "glue" these operations together into working programs could be written in C or C++. We believe that this versatility made the difference between having a little-used "stunt box " and having a working computer that scientists got their work done with.

11.3.2 Current Developments and Conclusions

SPERT (Synthetic PERceptron Testbed) is a fully programmable single chip microprocessor under development at ICSI for efficient execution of artificial neural network algorithms [Asanović et al, 1991]. The first implementation will use a 1.2 micron CMOS technology and is projected to have a peak 50MHz clock rate. A prototype system is being designed to occupy a double SBus slot within a Sun SparcStation. Compared with a RAP multiprocessor of similar performance, SPERT represents over an order of magnitude reduction in cost for problems where fixed-point arithmetic is satisfactory. This will be useful for replicating RAP-range capabilities within sister laboratories for at least some of the tasks of interest.

The main components are a JTAG[4] interface and control unit, an instruction fetch unit with an instruction cache, a scalar 32b integer datapath and register set, a SIMD array containing multiple 32b fixed point datapaths each with an associated register file, and a 128b wide external memory interface. In the absence of instruction cache misses and host memory accesses, one instruction can be completed every cycle. The scalar unit is a RISC processor, and is used for general scalar computation and to support the SIMD array by providing address generation and loop control.

The most significant innovation in the SPERT architecture is the tight coupling between a general-purpose RISC engine and a specialized array of DSP-like engines. The latter is also designed to have a very high memory bandwidth so that peak computational speeds can be supported without data reuse. Additionally, the use of custom VLSI design techniques (which are often used for general-purpose processors designs but are rarely used for special-purpose DSP chips) provided significant area/power/speed advantages over standard cell approaches. Much of the effort during the previous grant period has been the development of a consistent and testable design path for the kinds of macrocells that we required.

An important goal in the SPERT design has been to prototype ideas for a parallel processing node that will be used in a future, scalable, MIMD multiprocessor system. Such a system would target large, irregular network structures. The larger project, a joint project between U.C. Berkeley (principally Professor John Wawrzynek) and ICSI is named

[4]The Joint Test Action Group formulated the IEEE1149.1 boundary scan standard.

the Connectionist Network Supercomputer (CNS-1), and officially began May 1992.

The more recent designs differ from the RAP in several significant ways:

1. Price/performance: these systems pack a significantly larger amount of computational power per dollar or per unit volume than the RAP. The primary reason for the difference is the design of specialized processors and communication paths that take advantage of shorter wordlength fixed point representations. Additionally, the use of custom circuits means that all interfaces can be simplified and chip count (and variety) can be reduced.

2. Application goals: particularly for the CNS-1, the aim is efficient operation for a range of connectionist tasks that greatly exceeds the goals for the RAP. In particular, the machine should be useful for sparsely connected networks and networks with shared or tied weights.

3. Difficulty: the SPERT and CNS-1 designs are significantly more difficult than the RAP since the latter required no VLSI development.

Despite these differences, these new systems are being developed in the same spirit as the RAP; they are intended as specialized but fully programmable computing systems that mere mortal speech researchers can use.

11.4 Summary

While MLP estimators are conservative in system resource requirements for recognition, they can require formidable resources for training. For difficult problems in continuous speech recognition, 1993 workstation technology is insufficient for network learning.

These considerations led to the design of a series of hardware and software projects at ICSI to support the speech research described in this book (as well as for architectural research in programmable connectionist systems). The first was a system composed of commercial floating-point DSP chips connected via a ring of programmable gate arrays. This system produced most of the results described in this book. The second and third systems, briefly described in this chapter, are designs in progress, and incorporate a number of the ideas that we developed given the experience of the first system. We are continuing to develop these systems in parallel with the speech recognition research so that we can consider databases and algorithms that might otherwise be out of reach.

Part III

ADDITIONAL TOPICS

Chapter 12

CROSS-VALIDATION IN MLP TRAINING

We should be careful to get out of an experience only the wisdom that is in it - and stop there; lest we be like the cat that sits down on a hot stove-lid. She will never sit down on a hot stove-lid again - and that is well; but also she will never sit down on a cold one anymore.
– Mark Twain –

12.1 Introduction

It is well known that system models which have too many parameters (with respect to the number of measurements) do not generalize well to new measurements. For instance, an autoregressive (AR) model can be derived which will represent the training data with no error by using as many parameters as there are data points. This would generally be of no value, as it would only represent the training data. Criteria such as the *Akaike Information Criterion* (AIC) [Akaike, 1974, 1986] can be used to penalize both the complexity of AR models and their training error variance. In feedforward nets, we do not currently have such a measure. In fact, given the aim of building systems which are biologically plausible, there is a temptation to assume the usefulness of indefinitely large adaptive networks. In contrast to our best guess at Nature's tricks, man-made systems for pattern recognition seem to require nasty amounts of data for training. In short, the design of massively parallel systems is limited by the number of parameters that can be learned with available training data. It is likely that the only way truly massive systems can be built is with the help of prior information, e.g., connection topology and weights that need not be learned [Feldman et al., 1988]. Learning theory [Valiant, 1984; Pearl, 1978] has begun to establish what is possible for trained systems. Order-of-magnitude lower bounds have been established for the number of required measurements to train a desired size feedforward net [Baum & Haussler, 1988]. Rules of thumb suggesting the number of samples required for specific distributions could be useful for practical problems. Widrow has suggested having a training sample size that is 10 times the number of weights in a network ("Uncle Bernie's Rule") [Widrow, 1987].

In 1989 we conducted an empirical study of the relation of the number of parameters in a feedforward net (e.g., hidden units, connections, feature dimension) to generalization performance for data sets with known discrimination complexity and *Signal-to-Noise Ratio* (SNR). In this experiment, we used simulated data sets with controlled parameters, such as the number of clusters of continuous-valued data. In a related practical example, we have trained a feedforward network on vector quantized mel cepstra from real speech samples. In each case, we used the back-propagation algorithm [Rumelhart et al., 1986a] to train the feedforward net to discriminate in a multiple class pattern classification problem. Our results confirmed that estimating more parameters than there are training samples can degrade generalization. However, the peak in generalization performance (for the difficult pattern recognition problems tested here) can be quite broad if the networks are not trained too long, suggesting that previous guidelines for network size may have been conservative. Furthermore, cross-validation techniques, which have also proved quite useful for autoregressive model order determination, appear to improve generalization when used as a stopping criterion for iteration, and thus preventing overtraining.

All of these led to a cross-validation based training procedure that we have used throughout our recognition research. As reported in the earlier chapters, this idea appears to have been instrumental in our achievement of good recognition results with an HMM/MLP hybrid structure.

12.2 Random Vector Problem

12.2.1 Methods

Studies based on synthesized data sets will generally show behavior that is different from that seen with a real data set. Nonetheless, such studies are useful because of the ease with which variables of interest may be altered. In this case, the object was to manufacture a difficult pattern recognition problem with statistically regular variability between the training and test sets. This is actually no easy trick; if the problem is too easy, then even very small nets will be sufficient, and we would not be modeling the problem of doing hard pattern classification with small amounts of training data. If the problem is too hard, then variations in performance will be lost in the statistical variations inherent to methods like back-propagation, which use random initial weight values. Random points in a 4-dimensional hyperrectangle (drawn from a uniform probability distribution) were classified arbitrarily into one of 16 classes. This group of points will be referred to as a cluster. This process is repeated for 1-4 non overlapping hyperrectangles. A total of 64 points were chosen, 4 for each class. All points were then randomly perturbed with noise of uniform density and range specified by a desired signal-to-noise ratio (SNR). The noise was added twice to create 2 data sets, one to be used for training, and the other for test.

Intuitively, one might expect that 16-64 hidden units would be required to transform the training space for classification by the output layer. However, the variation between training and test and the relatively small amount of data (256 numbers) suggest that for large numbers of parameters (over 256) there should be a significant degrading of generalization. Another issue was how performance in such a situation would vary over large numbers of iterations. Simulations were run on this data using MLPs with 4 continuous-valued inputs, 16 outputs, and a hidden layer of sizes ranging from 4 to 128. Nets were run for an SNR of 1.0 and 2.0, where the SNR is defined as the ratio of the range of the original cluster points to the range of the added random values. Error back-propagation without momentum was used, with an adaptation constant of .25. For each case, the 64 training patterns were used 10,000 times, and the resulting network was tested on the second data set every 100 iterations so that generalization could be observed during the learning. Blocks of ten scores were averaged to stabilize the generalization estimate. After this smoothing, the standard deviation of error (using the normal approximation to the binomial distribution) was roughly 1%. Therefore, differences of 3% in generalization performance are significant at a level of .001. Roughly a trillion floating point operations were required for the study.

12.2.2 Results

Table 12.1 shows the test performance for a single cluster and a SNR of 1.0. The chart shows the variation over a range of iterations and network size (specified both as number of hidden units, and as ratio of number of weights to number of measurements, or "weight ratio"). Note that the percentages can have finer gradation than 1/64, due to the averaging, and that the performance on the training set is given in parentheses. Test performance is best for this case for 8 hidden units (24.7%), or a weight ratio of .62 (after 2,000 iterations), and for 16 units (21.9%), or a weight ratio of 1.25 (after 10,000 iterations). For larger networks, the performance degrades, presumably because of the added noise. At 2,000 iterations, the degradation is statistically significant, even in going from 8 to 16 hidden units. There is further degradation out to the 128-unit case. The surprising thing is that, while this degradation is quite noticeable, it is quite graceful considering the order of magnitude range in net sizes. An even stronger effect is the loss of generalization power when the larger nets are more fully trained. All of the nets generalized better when they were trained to a relatively poor degree, especially the larger ones.

Table 12.2 shows the results for the same 1-cluster problem, but with higher SNR data (2.0). In this case, a higher level of test performance was reached, and it was reached for a larger net with more iterations (40.8% for 64 hidden units after 5,000 iterations). At this point in the iterations, no real degradation was seen for up to 10 times the number of weights as data samples. However, some signs of performance loss for the largest nets was evident after 10,000 iterations. Note that after 5,000 iterations, the networks were only half-trained (roughly 50% error on the training set). When they were 80-90% trained, the larger nets lost considerable ground. For instance, the net with 128 hidden units (containing 10× more parameters than training patterns) lost performance from 40.5% to 28.1% during these iterations. It appears that the higher signal-to-noise of this example permitted performance gains for

# weights/ # inputs	% Test (Train) Correct after N Iterations 1,000	2,000	5,000	10,000
.31	9.2 (4.4)	21.7 (15.6)	12.0 (25.9)	15.6 (34.4)
.62	11.4 (5.2)	24.7 (17.0)	20.6 (29.8)	21.4 (63.9)
1.25	13.6 (6.9)	21.1 (18.4)	18.3 (37.2)	21.9 (73.4)
2.50	12.8 (6.4)	18.4 (18.3)	17.8 (41.7)	13.0 (80.8)
5.0	13.6 (7.7)	18.3 (20.8)	19.7 (34.4)	18.0 (79.2)
10.0	11.6 (6.7)	17.7 (19.1)	12.2 (34.7)	15.6 (75.6)

Table 12.1: Test (and training) scores: 1 cluster, SNR = 1.0.

# weights/ # inputs	% Test (Train) Correct after N Iterations 1,000	2,000	5,000	10,000
.31	18.1 (8.4)	25.6 (29.1)	32.2 (29.8)	26.9 (29.2)
.62	22.5 (12.8)	31.1 (34.7)	34.5 (44.5)	33.3 (62.2)
1.25	22.0 (11.6)	33.4 (32.8)	33.6 (57.2)	29.4 (78.3)
2.50	25.6 (13.3)	33.4 (35.2)	39.4 (51.1)	34.2 (87.0)
5.0	26.4 (13.9)	36.1 (35.0)	40.8 (45.2)	33.6 (86.9)
10.0	26.9 (12.0)	34.5 (34.5)	40.5 (47.2)	28.1 (91.1)

Table 12.2: Test (and training) scores: 1 cluster, SNR = 2.0.

even higher overparametrization factors, but that the result was even more sensitive to training for too many iterations.

Table 12.3 shows the performance for a 4-cluster case, with SNR = 1.0. Small nets are omitted here, because earlier experiments showed this problem to be too hard. The best performance (21.1%) is for one of the larger nets at 2,000 iterations, so that the degradation effect is not clearly visible for the undertrained case. At 10,000 iterations, however, the larger nets do poorly.

Figure 12.1 illustrates this graphically. The "undertrained" case is relatively insensitive to the network size, as well as having the highest raw score.

# weights/ # inputs	% Test (Train) Correct after N Iterations			
	1,000	2,000	5,000	10,000
2.5	13.8 (12.7)	18.3 (23.6)	15.8 (38.8)	9.4 (71.4)
5.0	13.6 (12.7)	18.4 (23.6)	14.7 (42.7)	18.8 (71.6)
7.5	15.3 (13.0)	21.1 (24.7)	15.9 (45.5)	16.3 (78.1)
10.0	15.2 (13/1)	19.1 (23.8)	17.5 (40.5)	10.5 (10.9)

Table 12.3: Test (and training) scores: 4 clusters, SNR = 1.0.

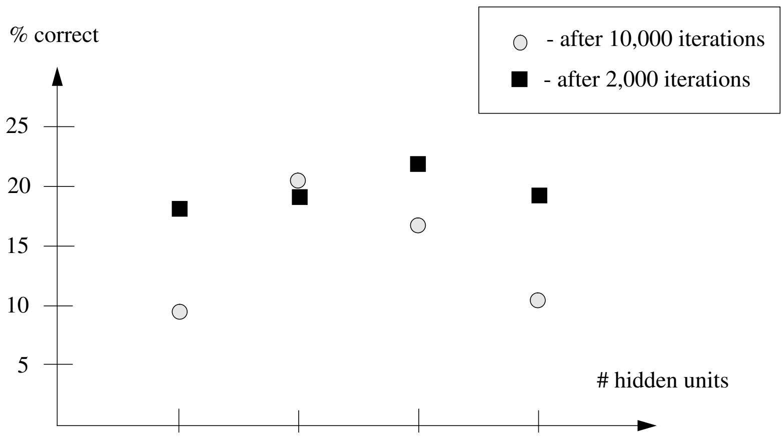

Figure 12.1: Sensitivity of MLP training to net size.

12.3 Speech Recognition

12.3.1 Methods

A German language database[1] consisting of 100 training and 100 test sentences (both from the same speaker) were used for training of a multi-layer-perceptron (MLP) for recognition of phones at the frame level, as well as to estimate probabilities for use in the dynamic programming algorithm for a discrete Hidden Markov Model (HMM) [Bourlard & Wellekens, 1989c; Bourlard et al., 1990]. Vector-quantized mel cepstra were used as binary input to a hidden layer. Multiple frames were used as input to provide context to the network. While the size of the output layer was kept fixed at 50 units, corresponding to the 50 phonemes to be recognized, the hidden layer was varied from 20 to 200 units, and the input context was kept fixed at 9 frames of speech. As the acoustic vectors were coded on the basis of 132 prototype vectors by a simple binary vector with a single bit 'on,' the input field contained $9 \times 132 = 1188$ units, and the total number of possible inputs was thus equal to 132^9. There were 26,767 training patterns and 26,702,, independent test patterns. Of course, this represented only a very small fraction of the possible inputs, and generalization was thus potentially difficult. Training was done by the classical "error-back propagation" algorithm, starting by minimizing an entropy criterion [Solla et al., 1988] and then an MSE. In each iteration, the complete training set was presented, and the parameters were updated each training pattern. To avoid overtraining of the MLP, (as was later demonstrated by the random vector experiment described above), improvement on the test set was checked after each iteration. If the classification rate on the test set was decreasing, the adaptation parameter of the gradient procedure was decreased; otherwise it was kept constant. In another experiment this approach was systematized by splitting the data in three parts: one for the training, one for the test and a third one absolutely independent of the training procedure for validation. No significant difference was observed between classification rates for the test and validation data. Other than the obvious difference with the previous study (this used real data), it is important to note another significant point: in this case, we stopped iterating (by any one particular criterion) when that criterion was leading to no new test set performance improvement. While we had not yet done the simulations

[1] SPICOS; see earlier text (Section 6.6.1).

# hidden # units	# parameters / # training patterns	training	test
5	.23	62.8	54.2
20	.93	75.7	62.7
50	2.31	73.7	60.6
200	9.3	86.7	59.6
MLE	.25	45.9	44.8
MAP	.25	53.8	53.0

Table 12.4: Test Run: Correct (phonemic) frame classification rate for training and test sets.

described above, we had observed the necessity for such an approach over the course of our speech research. We expected this to ameliorate the effects of overparameterization.

12.3.2 Results

Table 12.4 shows the variation in performance for 5, 20, 50, and 200 hidden units. The peak at 20 hidden units for test set performance, in contrast to the continued improvement in training set performance, can be clearly seen. However, the effect is certainly a mild one given the wide range in network size; using 10 times the number of weights as in the "peak" case only causes a degradation of 3.1%. Note, however, that for this experiment, the more sophisticated training procedure was used which halted training when generalization started to degrade.

For comparison with classical approaches, results obtained with Maximum Likelihood (ML) and Bayes estimates are also given. In those cases, it is not possible to use contextual information, because the number of parameters to be learned would be 50×132^9 for the 9 frames of context. Therefore, the input field was restricted to a single frame. The number of parameters for these two last classifiers was then $50 \times 132 = 6,600$, or a parameter/measurement ratio of .25. This restriction explains why the Bayes classifier, which is inherently optimal for a given pattern classification problem, is shown here as yielding a lower performance than the potentially suboptimal MLP.

12.4 Summary

While both studies show the expected effects of overparameterization, (poor generalization, sensitivity to overtraining in the presence of noise), perhaps the most significant result is that it was possible to greatly reduce the sensitivity to the choice of network size by directly observing the network performance on an independent test set during the course of learning (cross-validation). If iterations are not continued past this point, fewer measurements are required. This only makes sense because of the interdependence of the learned parameters, particularly for the undertrained case. In any event, though, it is clear that adding parameters over the number required for discrimination is wasteful of resources. Networks which require many more parameters than there are measurements will certainly reach lower levels of peak performance than simpler systems. For at least the examples described here, it is clear that both the size of the MLP and the degree to which it should be trained are parameters which must be learned from experimentation with the data set. Further study might, perhaps, yield enough results to permit some rule of thumb dependent on properties of the data, but our current thinking is that these parameters should be determined dynamically by testing on an independent test set.

Chapter 13

HMM/MLP AND PREDICTIVE MODELS

It's hard to predict - especially the future.
– Niels Bohr –

13.1 Introduction

Hidden Markov models are widely used for speech recognition. However, as shown in Chapter 3, strong assumptions have to be made to render the model computationally tractable. One of these assumptions is the observation independence of the acoustic vectors. Indeed, it is usually assumed that the probability that a particular acoustic vector is emitted at a given time only depends on the current state and the current acoustic vector. As a consequence, this model does not take account of the dynamic nature of the speech signal.[1]

Many authors have tried to take account of the dynamics of the speech signal. For instance, in [Furui 1986, 1991; Gurgen et al., 1990] features such as the time-derivative of the acoustic vectors have been introduced.[2] In [Deng, 1992a,b], the temporal evolution of the acoustic

[1]At least directly. In fact, the correlation of the acoustic vectors is implicitly (but only partly) taken into account via the constraint on the possible HMM state sequences. However, the time correlation of the acoustic vectors emitted on a single HMM state is not used.

[2]This has led to significant improvements in recognition rates, although the theo-

feature inside a state is modeled by a given function of time, i.e., a polynomial trend function of time t spent in the state. In [Saerens, 1992a], the acoustic vectors observed on each state are assumed to be generated by a continuous-time Markov model or a stochastic differential equation [Saerens, 1992b]; reestimation formulae are derived for the Viterbi algorithm. In [Wellekens, 1987], the emission probability is explicitly assumed dependent not only on the current vector but also on the last observed vector. In the case of a correlated Gaussian probability distribution function, it is then proved that the emission probabilities depend on the prediction error of a first order linear predictor.

In order to introduce time correlation between successive acoustic vectors, some authors have considered the time series of observations (at the sample level or at the acoustic vector level) to be generated by a hidden linear autoregressive model. In this case, standard emission probability density functions associated with HMM states are replaced by linear autoregressive functions, which results in the introduction of the prediction errors of these autoregressive functions in the likelihoods (i.e., emission probabilities). Along this line, Poritz (1982), Juang (1984) and Juang & Rabiner (1985) (see also [Kenny, et al., 1990; Tishby, 1991; Woodland, 1992]) use Gaussian autoregressive densities per state to model the speech dynamics and the Baum-Welch algorithm is applied for the reestimation of the parameters. In this case, it has been shown that the emission probabilities depend on the prediction error of a linear predictor.

More recently, some authors have considered the possibility of using nonlinear prediction models (essentially multilayer neural networks) for speech recognition with hidden Markov models [Tsuboka et al., 1990; Levin, 1990, 1993; Tebelskis & Waibel, 1990; Tebelskis et al., 1991; Petek et al., 1992; Iso & Watanabe, 1990, 1991; Deng et al., 1991]. In this case, the acoustic vectors are assumed to be generated at each frame by a discrete nonlinear process, different for every state, corrupted by an additive uncorrelated Gaussian noise.

This leads to a new family of hybrid HMM/MLP systems, usually referred to as predictive neural networks. Although these models have already been briefly discussed in Section 6.8 (in the framework of neural networks and statistical inference), we review here the basic ideas underlying these models and we discuss their advantages, drawbacks and

---

retical foundations of this approach could be questionable.

relationships with standard autoregressive models. Also, their use in estimating likelihoods or conditional likelihoods for use in HMMs will be discussed.

13.2 Autoregressive HMMs

In linear and nonlinear predictive (autoregressive) models, it is assumed that, for every HMM state $q_k \in \mathcal{Q} = \{q_1, q_2, \ldots, q_K\}$ (see Chapter 3 for notations), the acoustic vector x_n at time n has been generated by a linear or nonlinear process corrupted by additive noise, i.e.,

$$x_n \;=\; F_k(X_{n-p}^{n-1},\ \theta_k) + \epsilon_{n,k} \tag{13.1}$$

where $X_{n-p}^{n-1} = \{x_{n-p}, \ldots, x_{n-2}, x_{n-1}\}$ represents the sequence of the p previous acoustic vectors (if p is the prediction order), and $\epsilon_{n,k}$ is the prediction error on state q_k at time n and is assumed to be an *independent and identically distributed* (iid) random variable with probability density function $p_\epsilon(\epsilon|\lambda_k)$ with parameters λ_k and zero mean. $F_k(X_{n-p}^{n-1},\ \theta_k)$ is a deterministic function, associated with state $q_k \in \mathcal{Q}$, of the last p acoustic vectors and parameters θ_k, and can be linear [Makhoul, 1975] or nonlinear. Finally, the autoregressive process associated with state q_k is entirely described by two sets with parameters: θ_k, the parameters of the autoregressive function F_k and λ_k the parameters describing the distribution of the residual noise.

From equation (13.1), the conditional probability density of the observation $p_x(x_n|X_{n-p}^{n-1}, \Theta_k)$ ($\Theta_k = \{\lambda_k, \theta_k\}$ being the set of parameters appearing in the distribution) is as follows:

$$p_x(x_n|X_{n-p}^{n-1}, \Theta_k) \;=\; p_\epsilon(x_n - F_k(X_{n-p}^{n-1}, \theta_k)|\lambda_k) \tag{13.2}$$

$$\;=\; p_\epsilon(\epsilon_{n,k}|\lambda_k) \tag{13.3}$$

This is simply because, from equation (13.1), the conditional distribution of the stochastic variable x given the past is the same as the distribution of ϵ, but with a shift in the mean value.

As done in Chapter 3, given a sequence of acoustic vectors $X = X_1^N = \{x_1, x_2, \ldots, x_n, \ldots, x_N\}$ and a Markov model M, training and recognition are based on the MLE criterion, i.e., $P(X|M)$. If q^n represents the HMM state ($\in \mathcal{Q}$) visited at time n, we can define a path C of length N in the model as an ordered sequence

$Q_1^N = \{q^0 = q_I, q^1, q^2, \ldots, q^N, q^{N+1} = q_F\}$ of N states entering via the (non-emitting) initial state q_I and ending via the (non-emitting) final state q_F of M. As done in Section 3.1.1, if Γ denotes the set of all possible paths C in M, the actual calculation of $P(X|M)$ involves summing the probabilities of all possible paths $C \in \Gamma$, i.e.:

$$P(X|M) = \sum_{\Gamma} P(C, X|M) = \sum_{\Gamma} P(Q_1^N, X|M) \tag{13.4}$$

Keeping in mind that we are working in a particular Markov model M, we can omit M in the conditional and each term in (13.4) can be expressed as:

$$\begin{aligned} P(Q_1^N, X) &= P(X_1^N|Q_1^N)P(Q_1^N) \\ &= p(x_N|X_1^{N-1}, Q_1^N)p(x_{N-1}|X_1^{N-2}, Q_1^N)\ldots \\ &\quad \ldots p(x_2|x_1, Q_1^N)p(x_1|Q_1^N)p(Q_1^N) \end{aligned} \tag{13.5}$$

However, since we want to estimate each factor (local probability) of this expression by a (linear or nonlinear) predictive model of order p, we can only estimate probabilities like $p(x_n|X_{n-p}^{n-1})$ and the p first conditional probabilities ($n = 1, \ldots, p$) cannot be evaluated because of the problem of the initial conditions. As a consequence, the likelihood of an acoustic vector sequence along a particular state sequence is usually replaced by the *conditional likelihood* that is defined as:

$$\begin{aligned} P(X_{p+1}^N, Q_{p+1}^N|X_1^p, Q_1^p) &= p(x_N|X_1^{N-1}, Q_1^N)p(x_{N-1}|X_1^{N-2}, Q_1^N)\ldots \\ &\quad \ldots p(x_{p+1}|X_1^p, Q_1^N)p(Q_{p+1}^N|Q_1^p) \end{aligned} \tag{13.6}$$

Since the process (13.1) is of order p, each local probability of the total conditional likelihood only depends on the p previous vectors. Moreover, we also assume that the emission of acoustic vectors through the process defined by (13.1) only depends on the present state, so that

$$\begin{aligned} &P(X_{p+1}^N, Q_{p+1}^N|X_1^p, Q_1^p) \\ &\qquad = \prod_{n=p+1}^{N} p(x_n|X_{n-p}^{n-1}, q^n)p(Q_{p+1}^N|Q_1^p) \end{aligned} \tag{13.7}$$

Furthermore, since the state sequence is generally assumed to be generated by a first order Markov process, we can rewrite (13.7) as

$$P(X_{p+1}^N, Q_{p+1}^N | X_1^p, Q_1^p)$$
$$= \prod_{n=p+1}^{N} p(x_n | X_{n-p}^{n-1}, q^n) \prod_{n=p+1}^{N} p(q^n | q^{n-1}) \qquad (13.8)$$

where $p(q^n|q^{n-1})$ are the transition probabilities.

Given (13.2) it is then clear that the conditional likelihood for the whole utterance can now be expressed in terms of the prediction errors of the (linear or nonlinear) predictors associated with each of the HMM-states, i.e.,

$$P(X_{p+1}^N, Q_{p+1}^N | X_1^p, Q_1^p)$$
$$= \prod_{n=p+1}^{N} p_\epsilon(x_n - F_{k_n}(X_{n-p}^{n-1}, \theta_{k_n} | \lambda_{k_n})) \prod_{n=p+1}^{N} p(q^n | q^{n-1}) \quad (13.9)$$

where k_n represents the index of the state q_k at time n along C. There is another, more straightforward, way to obtain the same result. Indeed, the likelihood of the parameters, given the values of $(\epsilon^{p+1}, \epsilon^{p+2}, ..., \epsilon^N)$ (where ϵ^n represents the prediction error for the HMM-state visited at time n, given x_n) and conditioned on the sequence of states is

$$P(\epsilon^{p+1}, \epsilon^{p+2}, \ldots, \epsilon^N | q^{p+1}, q^{p+2}, \ldots, q^N) \qquad (13.10)$$

and, since the Jacobian of the transformation from the $\{x_n\}$ to the $\{\epsilon^n\}$ is unity, this joint probability function also represents the likelihood function of the parameters given $(x_{p+1}, x_{p+2}, \ldots, x_N)$. Equation (13.9) follows directly.

13.3 Full and Conditional Likelihoods

What has been presented in the previous section does not require any particular assumption about the predictor; this is valid for linear as well as nonlinear predictive functions provided that we restrict the likelihood of the sequence to the conditional likelihood. However, if there is a large amount of data, this can have a negligible effect.

It is however interesting to observe that it is possible to compute the exact total likelihood in the linear autoregressive case: see [Box &

Jenkins, 1976]; in the context of speech recognition, see [Juang, 1984]. However, this is no longer valid for nonlinear predictors (and, consequently, neural networks), in which case it is important to keep in mind that we will always be restricted to conditional likelihoods.

These methods have been used for both likelihood and Viterbi training and using linear or nonlinear (neural network) predictors. Also, the modeling can be done at the feature vector level [Levin, 1990] or at the sample level [Poritz, 1982].

13.4 Gaussian Additive Noise

13.4.1 Training

As already shown in Chapter 6, the parameters θ_k of the AR functions are usually trained by minimizing the MSE

$$E = \sum_{k=1}^{K} \sum_{x_n \in q_k} \| x_n - F_k(X_{n-p}^{n-1}, \theta_k) \|^2$$

in the parameter space of θ_k.

The training procedures will of course depend on the kind of predictor and on the assumptions about the noise distribution. Of course, the most studied case is the one for which the noise is Gaussian distributed with zero mean and covariance Σ_k, in which case (13.3) takes the form:

$$p_\epsilon(\epsilon_{n,k}|\lambda_k) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_k|}} \exp\left(-\frac{\epsilon_{n,k}^T (\Sigma_k)^{-1} \epsilon_{n,k}}{2} \right) \tag{13.11}$$

During Viterbi training [Levin, 1990], one has to minimize the negative log conditional likelihood within a constant term:

$$\begin{aligned} -\log[P(X_{p+1}^N, Q_{p+1}^N)|X_1^p, Q_1^p] \\ = \frac{1}{2} \sum_{n=p+1}^{N} \epsilon_{n,k}^T (\Sigma_{k_n})^{-1} \epsilon_{n,k} + \frac{1}{2} \sum_{n=p+1}^{N} \log |\Sigma_{k_n}| \\ - \sum_{n=p+1}^{N} p(q^n|q^{n-1}) \end{aligned} \tag{13.12}$$

By introducing the usual constraint that the transition probabilities coming from the same state sum to one, one can easily derive reestimation formulae for the variance-covariance matrix and the transition probabilities. Most of the authors assume, however, the variance-covariance to be the identity matrix (as was done in Chapter 6).[3]

For what concerns the predictor parameters θ_k, for $k = 1, \ldots, K$, the training method will, of course, be different for linear and nonlinear predictors. In the linear case, both standard techniques for the estimation of the parameters of autoregressive processes (for Viterbi algorithm), and Forward-Backward algorithm [Poritz, 1982; Juang, 1984; Juang & Rabiner, 1985; Kenny et al., 1990; Tishby, 1991; Woodland, 1992] can be used. In the nonlinear case, gradient descent is used, both for Viterbi algorithm [Levin, 1990, 1993; Tebelskis & Waibel, 1990; Tebelskis et al., 1991; Petek et al., 1992; Iso & Watanabe, 1990, 1991] and Forward-Backward algorithm [Tsuboka et al., 1990].

13.4.2 Recognition

During recognition, if the Viterbi criterion is used, the best state sequence Q_1^N given the observation sequence X_1^N is the one that maximizes

$$P(Q_1^N|X_1^N) = \frac{P(X_1^N|Q_1^N)P(Q_1^N)}{P(X_1^N)} \tag{13.13}$$

Maximization of this last expression (within an initialization constraint related to the first p acoustic vectors) is equivalent to minimization of (13.12), which can be done by standard dynamic programming in which the local distance associated with state q_k and acoustic vector x_n is nothing else than the prediction error

$$\| x_n - F_k(X_{n-p}^{n-1}, \theta_k) \|^2$$

if the prediction error is assumed to be Gaussian with zero mean and unity variance, and the normalized prediction error

$$[x_n - F_k(X_{n-p}^{n-1}, \theta_k)]^T (\Sigma_k)^{-1} [x_n - F_k(X_{n-p}^{n-1}, \theta_k)]$$

if the prediction error is assumed to be Gaussian with zero mean and covariance matrix Σ_k.

[3] Although this can be expected to hurt recognition performance.

13.4.3 Discussion

As already discussed in Section 6.8, these predictive approaches have been tested on speech recognition problems but, to our knowledge, never led to better performance. According to our experience with predictive networks (and related approaches like the one presented in Section 6.8.4), although very attractive in theory, these approaches always led to significantly poorer performance than the one achieved with MLPs trained in classification mode. This also seems to corroborate results obtained with standard (linear) autoregressive models [de La Noue et al., 1989; Wellekens, 1987].

There are several potential reasons for this problem:

- As opposed to the main approach developed in this book (i.e., MLPs trained in classification mode), AR (predictive) approaches are no longer discriminant. They are (potentially) better in modeling the time dynamics of the sequence but each class (e.g., each HMM state) is associated with an AR model or a neural network that is trained independently of the others.
- In most of the work using predictive neural networks, the covariance matrix describing the error pdf was generally assumed to be unity, which obviously can hurt performance.[4]
- Linear models are too smooth to represent speech dynamics and nonlinear models are difficult to estimate properly – see next section.

13.5 Linear or Nonlinear AR Models?

Under the usual assumptions on the error prediction (i.e., Gaussian iid random variable), standard linear AR models [Makhoul, 1975; Juang & Rabiner, 1985], as well as nonlinear (MLP) AR models will result in Gaussian like emission probabilities. However, it is clear that when using an MLP, $F_k(X_{n-p}^{n-1})$ will be nonlinear functions and can approximate the dynamic of the system better. However, although the theory of AR models (linear and nonlinear) is particularly appealing to speech recognition, it has not yet led to any significant improvements over state-of-the-art approaches. In cases where direct comparisons were done,

[4]This is usually not overlooked in linear AR models.

many researchers have observed lower performance using some variant of this approach, despite the potential advantage suggested by theory. This is discussed, for example, in [de La Noue et al, 1989] where it is shown that linear AR models do not work well for speech recognition, although they do for artificial AR speech.[5]

There could be several reasons for this, including:

1. Any (linear) autoregressive models assume some kind of "smooth" dynamic, which is obviously wrong in speech (e.g., plosives). Even for nonlinear AR processes, most underlying dynamics are smooth. However, there are some nonlinear AR processes (even with low order) that can generate "speech-like behaviors," and which are no longer really smooth [Priestley, 1991].

2. The speech dynamic is highly nonlinear and, in this case, it does not help to increase the order of linear AR models to improve performance. A critical issue is the determination of the order of the underlying nonlinear predictor. Viewing speech as a chaotic time series, it has been shown that the underlying fractal dimension (usually referred to as "correlation dimension" or "embedded dimension") of speech signals is quite low, and is typically estimated to be between 3 and 5 [Tishby, 1990; Townshend, 1991]. A theorem by Takens [1981] proves that if D is the correlation dimension, the maximum dimension of the optimal nonlinear predictor is equal to $2D + 1$. Even if the model order were precisely known, however, we would still need to know the nonlinear function itself. The need to learn this function is a good match for the training capabilities of a neural network.

13.6 ARCH Models

As already mentioned before, the modeling can be done at the sample level or at the feature vector level. However, we have to keep in mind that, very roughly, there are two kinds of variability in the speech signal: (1) real noise that affects the speech signal itself and (2) inter- and intra-speaker variability that results in different vocal tract shapes and articulator positions, and therefore different transfer functions, related

[5] Wouldn't it be nice to be able to manufacture the data to match our models? Nature can be so messy ...

to the same phonetic unit. In [Saerens & Bourlard, 1993], a model that takes account of the two sources of variability is introduced. The main idea is to assume that, in the linear case, the autoregressive coefficients θ_k in (13.1) are also random variables, subject to fluctuations. In fact, this is exactly what we are doing when extracting LPC coefficients and clustering them with Gaussian distributions. The advantage here is that we directly introduce the variability at the sample level. This leads to processes that are known as *Autoregressive Conditional Heteroscedastic* (ARCH) processes [Engle, 1992; Gourieroux, 1992] with nonconstant variances conditional on the past.

13.7 Summary

In this chapter, we reviewed the basic ideas underlying the use of predictive models for speech recognition with hidden Markov models. These models assume that the acoustic vectors are generated by a linear or nonlinear process corrupted by additive noise. Both linear and nonlinear models can be used for prediction, and both forward-backward algorithm and Viterbi algorithm can be used for training. If neural networks are used to estimate nonlinear functions F_k, this leads to a new kind of hybrid HMM/MLP approach. However, although this approach is able to model the time dynamics of the sequence (and the correlation of the successive acoustic vectors), it loses the discriminant character that was one of the main properties of the general method presented in this book.

Chapter 14

FEATURE EXTRACTION BY MLP

When the only tool you have is a hammer, everything begins to look like a nail.
– Lotfi Zadeh –

14.1 Introduction

In the preceding chapters, emphasis was put on the use of MLPs as discriminant pattern classifiers for speech recognition applications. Although pattern classification plays a crucial role, it is only part of the vast speech recognition task. In spite of the spectacular progress made over the last decade, unrestricted speech recognition is still out of reach, and it is suspected that part of the difficulty lies in the use of inappropriate features for recognizing speech. *A priori* phonetic knowledge seems of little practical use in this respect. The elementary sounds composing speech can indeed be described by place and manner of articulation for instance, but it seems difficult to translate this knowledge to a precise characterization at the signal level. On the other hand, one can consider that the hidden units of an MLP develop an internal representation of the input signal which is the most appropriate for the classification task. From this point of view, the MLP performs some type of feature extraction which is given by the activity levels of the hidden units. This view of an MLP as a trainable feature extractor for speech processing was described in [Rumelhart et al., 1986a], was systematically investigated in [Elman & Zipser, 1988], and was more generally the original perspective in some of the work of Rosenblatt and his students.

In most MLP architectures used for feature extraction, the number of units on the hidden layer is smaller than on the input layer. Consequently, the hidden layer acts as a narrow-band channel and thus performs some form of dimensionality reduction. Again, if the learning procedure of the MLP is successful, one could expect that this reduction extracts the most salient features in the signal. In view of this observation, the MLP can be considered as an attractive alternative for efficient speech coding and image compression as examined in [Elman & Zipser, 1988] and [Cottrell et al., 1988].

Feature extraction and dimensionality reduction can be learned in many ways but the most efficient one is to use teaching signals which are identical to the input since this avoids explicit segmentation and labeling of the signal and thus allows unsupervised training of the MLP. For this particular mode of operation, known as auto-association or identity mapping, the output layer generally does not contain any nonlinear function (at least for real valued inputs) since the output target is identical to the input pattern.

Of course, there are other techniques by which data compression and feature extraction can be achieved. Most important among these is the Karhunen-Loève or principal components transform, which is a purely *linear* method, in contrast with the nonlinear operation mode of the MLP, due to the sigmoidal function at the hidden units. In spite of this opposition, it was already anticipated in [Cottrell et al., 1988] that the auto-associative MLP should somehow be related to more classical techniques, the more so that a linear version of it produced results which were compatible with the nonlinear version. At this point, however, the exact nature of this relationship was not fully understood.

The purpose of this chapter is to show on a rigorous basis that an auto-associative MLP with linear output units is nothing but an indirect way of performing data compression by a Karhunen-Loève transform [Bourlard & Kamp, 1988] (at best). More precisely, it will be shown that the optimal weight values can be derived by standard linear algebra, consisting in singular value decomposition (SVD) thus making the nonlinear functions at the hidden layer unnecessary. The advantages are obvious: the solution is obtained explicitly in terms of the training data, whereas the EBP algorithm generally used for training MLPs proceeds iteratively and may well miss the optimum solution since it relies on a gradient technique and can get trapped in local minima. The analysis presented below offers the additional benefit that the optimal parameters are given a meaningful interpretation in terms of reconstruction of the average value and covariance of the input patterns.

14.2 MLP and Auto-Association

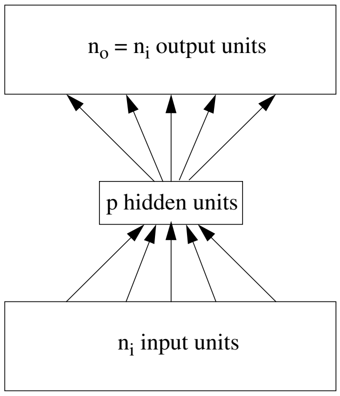

Figure 14.1: MLP with one hidden layer for auto-association.

Consider an MLP with a single hidden layer as represented in Figure 14.1 where p is the number of hidden units. When using this type of network to achieve dimensionality reduction by auto-association, it is desired that the input units communicate their values to the output units through a hidden layer acting as a limited capacity bottleneck which must optimally encode the input vectors. Thus, for this particular application, $n_i = n_0 = n$ and $p < n$. When entering an n-dimensional real input vector x_k $(k = 1, 2, \ldots, N)$, the output values of the hidden units form a p-vector given by

$$h_k = F(W_1 x_k + w_1), \qquad k = 1, 2, \ldots, N \tag{14.1}$$

where W_1 is the (input-to-hidden) $p \times n$ weight matrix, w_1 is a p-vector of biases and the nonlinear (typically sigmoid) function F is operated component wise. For most applications of MLPs, e.g., for classification, the values in the output layer are obtained in a similar way. However, in the case of auto-association, the output values should approximate the inputs as closely as possible. Consequently, in the case of real valued inputs, the non-linearity at the output must be removed and the output values form an n-vector given by

$$y_k = W_2\, h_k + w_2 \qquad (k = 1, 2, \ldots, N) \tag{14.2}$$

where W_2 is the (hidden-to-output) $n \times p$ weight matrix and w_2 is an n-vector of biases. The problem is to find optimal weight matrices W_1, W_2 and bias vectors w_1, w_2 minimizing the mean-square error $E = \sum_{k=1}^{N} \| x_k - y_k \|^2$, which corresponds to the standard optimization criterion used for MLP training.

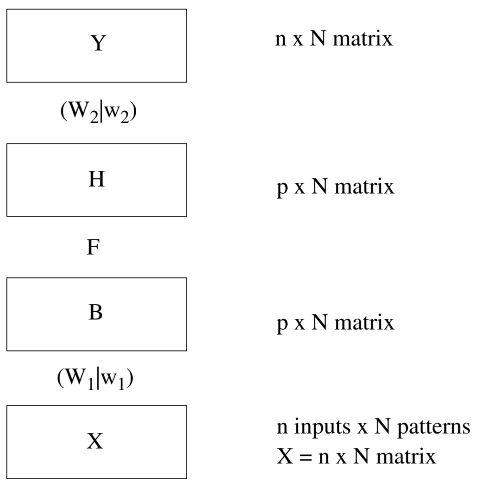

Figure 14.2: Sequence of operations in the auto-associative MLP.

Let $X = [x_1, x_2, \ldots, x_N]$ be the $n \times N$ real matrix formed by the N input vectors of the training set and let $H = [h_1, h_2, \ldots, h_N]$ and $Y = [y_1, y_2, \ldots, y_N]$ be the $p \times N$ and $n \times N$ matrices formed by the corresponding vectors of the hidden and output units respectively. Given

(14.1) and (14.2), the output matrix Y of the auto-associative MLP is obtained from the input matrix X as the result of the following sequence of operations illustrated by Figure 14.2:

$$B = W_1 X + w_1 u^T \tag{14.3}$$

$$H = F(B) , \tag{14.4}$$

$$Y = W_2 H + w_2 u^T \tag{14.5}$$

where B is a $p \times N$ real matrix and u is an N-vector of ones. With this notation, the squared error norm E can be rewritten as

$$E = \| X - Y \|^2 \tag{14.6}$$

where $\| \cdot \|$ now denotes the Euclidean matrix-norm (or Frobenius norm). The training problem is to minimize E with respect to the parameter set W_1, W_2, w_1, w_2.

14.3 Explicit and Optimal Solution

Using (14.5) the squared error norm can be rewritten as

$$E = \| X - W_2 H - w_2 u^T \|^2 \tag{14.7}$$

and, in view of $\| A \|^2 = tr(AA^T)$, one easily verifies that minimization of E with respect to w_2 yields

$$\widehat{w}_2 = \frac{1}{N}(X - W_2 H)u \tag{14.8}$$

Substituting (14.8) in (14.7) one obtains for the squared error norm:

$$E = \| X' - W_2 H' \|^2 \tag{14.9}$$

where $X' = X(I - uu^T/N)$ and $H' = H(I - uu^T/N)$. In view of the fact that W_2 normally has rank $p < n$, expression (14.9) shows that the product $W_2 H'$ minimizing E is the best rank p approximation of X' in Euclidean norm. This is a standard problem and can be solved as follows. Consider the SVD of X' [Golub & Van Loan, 1983; Stewart, 1973]:

$$X' = U_n \, \Sigma_n \, V_n^T \tag{14.10}$$

where $U_n(V_n)$ is an $n \times n$ $(N \times n)$ matrix formed by the normalized eigenvectors of $X'X'^T$ (X'^TX') associated with the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ and where Σ_n= diag$[\sigma_1, \sigma_2, \ldots, \sigma_n]$ is a diagonal matrix with $\sigma_i = \sqrt{\lambda_i}$. For simplicity we will assume that X has full row rank $(\sigma_n > 0)$. It is known [Golub, 1968; Stewart, 1973] that the best rank p approximation of X' is given by

$$\widehat{W}_2 \, \widehat{H}' = U_p \, \Sigma_p \, V_p^T \tag{14.11}$$

with Σ_p = diag $[\sigma_1, \sigma_2, \ldots, \sigma_p]$ and where $U_p(V_p)$ is formed by the first p columns in $U_n(V_n)$. Consequently

$$\widehat{W}_2 = U_p \, T^{-1} \, , \qquad \widehat{H}' = T \, \Sigma_p \, V_p^T \tag{14.12}$$

where T is an arbitrary non-singular $p \times p$ matrix which will subsequently play an important role as a scaling matrix.

Let us pause here to comment on the results derived so far and to point out a few interesting properties of the optimally trained auto-associative MLP.

Let μ_X denote the average of the training input vectors $x_1, x_2, \ldots, x_N$ i.e., $\mu_X = \frac{1}{N} \, X \, u$ and let similarly $\mu_Y = \frac{1}{N} \, Y \, u$ be the average of the MLP output vectors. Taking (14.5) and (14.8) into account, it follows that the optimal bias vector $\widehat{w}_2$ insures

$$\mu_Y = \mu_X \tag{14.13}$$

or, in other words, that the average input and output vectors are equal. Observe also that, in the very special case where all training vectors are identical, i.e., $X = \mu_X \, u^T$, this vector is exactly reproduced at the output $(Y = \mu_Y \, u^T)$ since then $X' = 0$ and hence $\widehat{W}_2' \widehat{H}' = 0$ by (14.11).

If $\mu_H = \frac{1}{N} \, H u$ denotes the average of the vectors at the output of the hidden units, then the definitions of X' and H' can be rewritten as $X' = X - \mu_X u^T$ and $H' = H - \mu_H u^T$ which means that they represent respectively the input and hidden unit vectors after subtraction of their average value. Consequently, the computational effect of the bias vector $\widehat{w}_2$ is thus to reduce the training problem (14.9) to zero-average patterns.

Finally, one can show that the covariance of the output vectors $\{y_1, \, y_2, \ldots, y_N\}$ is the best rank p approximation of the covariance of the input vectors $\{x_1, \, x_2, \ldots, x_N\}$ and, in this sense, the auto-associative MLP is nothing but an indirect way of performing data compression by a

Karhunen-Loève transform on zero-average data [Ahmed & Rao, 1975]. Indeed, owing to (14.10),

$$C_X = X' X'^T = U_n \Sigma_n^2 U_n^T \tag{14.14}$$

On the other hand, the output covariance matrix defined as

$$C_Y = (Y - \mu_Y u^T)(Y^T - u \mu_Y^T)$$

can, in view of (14.13), (14.5), (14.11) and orthogonality properties, be rewritten as

$$C_Y = U_p \Sigma_p^2 U_p^T \tag{14.15}$$

and comparison of (14.15) with (14.14) terminates the proof.

It is a remarkable fact that the optimal expressions in (14.8) and (14.12), as well as the preceding properties, have been obtained completely independently of the way in which H' is produced by the MLP and, more specifically, independently of the particular nonlinear function used at the output of the hidden units. In the following section, we will first consider the case where this nonlinear function is absent which implies $H = B$. Next, we will show that this optimal situation can be approximated as closely as required even when a sigmoidal function is present at the output of the hidden units, as is usually the case in an MLP.

14.4 Linear Hidden Units

Since $B = H$, we have to prove that $\widehat{H}'$ as prescribed by (14.12) can be generated in accordance with equation (14.3) by an appropriate choice of W_1 and w_1. Multiplying both sides of (14.3) by $(I - uu^T/N)$ we have thus to solve the following equation for W_1 and w_1

$$T \Sigma_p V_p^T = W_1 X' + w_1 u^T (I - uu^T/N) \tag{14.16}$$

In view of $u^T u = N$, the second term on the right-hand side vanishes, showing that w_1 is arbitrary. Next, taking (14.10) into account, the left-hand side can be rewritten as $T U_p^T X'$ and (14.16) then becomes $T U_p^T X'$ $= W_1 X'$, so that

$$\widehat{W}_1 = T U_p^T \tag{14.17}$$

Finally, to find the optimal value of the bias vector w_2, it is sufficient to eliminate $H = B$ from (14.8), via equation (14.3) and to incorporate results (14.12) and (14.17). One finds

$$\widehat{w}_2 = (I - U_p\, U_p^T)\, \mu_X - U_p T^{-1} w_1 \tag{14.18}$$

Thus, for arbitrary w_1, vector $\widehat{w}_2$ should be adjusted according to (14.18) which, as observed before, insures $\mu_X = \mu_Y$. In summary, after SVD of X', equations (14.12), (14.17) and (14.18) give the optimal solutions for W_1, W_2, w_1 and w_2 of the "linear" MLP.

14.5 Nonlinear Hidden Units

Now consider the case where a nonlinear function F is present at the output of the hidden units. We will not need strong assumptions about the particular form of this function except that, for small values of its argument, it can be approximated as closely as desired by the linear part of its power series expansion, i.e.,

$$F(x) \sim \alpha_0 + \alpha_1 x \qquad \text{for } x \text{ small} \tag{14.19}$$

with nonzero α_1. For the asymmetric sigmoid, $F(x) = 1/(1 + e^{-x})$, this gives $\alpha_0 = 1/2$ and $\alpha_1 = 1/4$; whereas for the symmetrical sigmoid, $F(x) = (1 - e^{-x})/(1 + e^{-x})$, one has $\alpha_0 = 0$, $\alpha_1 = 1/2$.

We will now show that, within minor modifications, the optimal values obtained in the previous sections still produce the expression for $\widehat{H}'$ required by (14.12). If we take

$$\widehat{W}_1 = \alpha_1^{-1}\, T\, U_p^T \tag{14.20}$$

we obtain by (14.3),

$$\widehat{B} = \alpha_1^{-1}\, T\, U_p^T\, X + w_1\, u^T \tag{14.21}$$

Obviously, if we want to use approximation (14.19), then B should be made small by acting on w_1 and on the arbitrary scaling matrix T. This leaves still some freedom on $\widehat{w}_1$ which could e.g., be chosen equal to zero. Another interesting possibility is to force $\mu_B = \frac{1}{N}\, Bu$, the average vector of matrix B defined in (14.3), to be zero by selecting

$$\widehat{w}_1 = -\alpha_1^{-1}\, T\, U_p^T\, \mu_X \tag{14.22}$$

In both cases, $\| T \|$ should be sufficiently small but nonsingular. With $\widehat{w}_1$ as given in (14.22), one finally obtains

$$\widehat{B} = \alpha_1^{-1}\, T\, U_p^T\, X' = \alpha_1^{-1}\, T\, \Sigma_p\, V_p^T \tag{14.23}$$

and equation (14.4) yields $\widehat{H} = \alpha_0 u u^T + \alpha_1 \widehat{B}$, leading to

$$\widehat{H} = \alpha_0\, u u^T + T\, \Sigma_p\, V_p^T \tag{14.24}$$

Since $\widehat{H}'$ has been defined by $\widehat{H}' = H(I - uu^T/N)$, this gives, as desired, $\widehat{H}' = T\, \Sigma_p\, V_p^T$. As for the optimal bias w_2, it can easily be computed from (14.8), (14.12) and (14.24) as

$$\widehat{w}_2 = \mu_X - \alpha_0\, U_p\, T^{-1}\, u \tag{14.25}$$

Thus, in the case of a sigmoidal function at the hidden units, the optimal parameters of the MLP are given via the SVD of X' by expressions (14.12), (14.20), (14.22) and (14.25).

It is not difficult to see that essentially the same approach can be used in the case of multiple hidden layers. The key operation remains the SVD of X' and its rank p approximation where p is now given by the last hidden layer. The freedom in the choice of the weight matrices and bias vectors becomes then even wider.

Finally, when the units on the output layer contain nonlinear functions, then of course, the approach presented above breaks down. However, even in this case, some interesting results can still be derived by analytical ways and are shown to be closely connected with low rank realizations of prescribed sign matrices [Delsarte & Kamp, 1988].

14.6 Experiments

A simple training database was composed of 60 vectors in R^{16} (hence X is a 16×60 real matrix). These were cepstral vectors obtained from 10-ms frames of speech signal and corresponded to the mean vectors associated with the states of phonemic hidden Markov models [Bourlard et al., 1985]. In order to confirm the theoretical results, we determined by the SVD of X' and equations (14.12), (14.20), (14.22) and (14.25), the optimal weight matrices W_1, W_2 and biases w_1, w_2 for a rank 5 approximation (corresponding to 5 hidden units) and used these values as initialization of the EBP training algorithm. In that case, the EBP was unable to improve the parameters by reducing the MSE (14.6). Moreover, when starting the EBP training algorithm several times with random weights, it always got stuck in local minima, giving higher error values. This illustrated that the linear approach was preferable.

One could object that the MLP and the associated EBP algorithm allow on-line learning, which is an important advantage when the number of training patterns becomes large. However, the SVD algorithm also has a sequential version [Bunch & Nielsen, 1978], so this argument does not apply. Similarly, while the MLP can be implemented on fast parallel hardware, similar mappings can be made for SVD. Perhaps the only hardware-oriented argument that may favor the MLP approach is that MLP training can be done with lower precision (e.g., 16 bits for weights and 8 bits for activation), while SVD requires more precision (typically 32-64 bit floating point is used).

It is also important to remember that the theoretical developments presented in this chapter are only valid for the auto-associative MLP with linear outputs and linear or nonlinear hidden units, where the number of hidden units is smaller than the number of input (and output) units. In the case where the (bottleneck) hidden layer with $p < n$ hidden units is preceded and followed by at least one additional hidden layer with $q > n$ units, this network will perform a nonlinear expansion of the input space before doing SVD in that expanded space. In this case, an explicit solution by linear algebra is no longer possible. However, this kind of nonlinear preprocessing has been shown to lead to better classification performance on some speech recognition problems [Nakamura et al., 1991].

Some parts of the theory developed in this chapter can also be used to improve our understanding of hetero-associative MLPs used for classification and their relationships with discriminant analysis (see, e.g., [Webb & Lowe, 1989]). However, this will never enable us to find the optimal solution for all the weights of an MLP (as done here for auto-association) except in some very particular cases where all the hidden and output units have a linear transfer function. In this case, of course, more strict mathematical treatments about the absence of local minima, the presence of saddle points, learning properties and relationships with principal component analysis is possible (see, e.g., [Baldi & Hornik, 1989; Baldi & Hornik, 1991]).

14.7 Summary

While the previous chapters focused on the use of MLPs as a particular module of a complete CSR system (i.e., probability estimation of HMMs), we have investigated here the possibility to use MLP for another subtask, i.e., feature extraction, which is more related to the front-end processing of a speech recognizer. In this case, MLPs working in "auto-association" mode are usually used to extract relevant features from rough data. Such a network was studied here and it was shown that EBP can be avoided by analytically determining the optimal parameters of the network. It was proved that the optimal solution of the MLP was strictly equivalent to the standard singular value decomposition (SVD) approach and that, in this case, the nonlinearity in the hidden units is theoretically of no help. It was shown that the network actually projects the input onto the subspace spanned by the first p principal components of the input, where p is the number of hidden units.

Although this conclusion sounds a bit pessimistic, this approach has some merits: while allowing a better understanding of neural network processing and its relationships with standard signal processing techniques, it also provides us with an efficient parallel implementation of the SVD algorithm which can be integrated easily in a general neural network framework. Such a framework, as noted earlier, can be based on low or moderate precision hardware.

Part IV

FINALE

Chapter 15

FINAL SYSTEM OVERVIEW

We need to say something clever here.
– Hervé Morgan –

15.1 Introduction

In this book, several theoretical and experimental developments related to HMMs, neural networks and hybrid HMM/MLP systems have been presented. While one of our goals was to discuss relationships between neural networks, statistics and linear algebra, the main aim of this book was to present the theories, experiments and hardware that were required to develop our hybrid HMM/MLP approach to improving large vocabulary, continuous speech recognition systems.

For clarity's sake, this chapter will summarize the hybrid HMM/MLP approach (including training) that led to our basic system. This system is now the basis for extensions that are the subject of current research.

15.2 System Description

15.2.1 Network Specifications

Although we experimented with many systems, the ANN that we have ultimately adopted as our core system was a multilayer perceptron with the following characteristics:

- Feedforward MLP; i.e., without recurrence (feedback) – In Section 6.3 the theory was developed for an MLP with feedback from the output to the input field. There it was shown that, in theory, this network was able to generate "discriminant" probabilities that were a good fit to the framework of discriminant HMMs described in Section 7.2. However, because of the many problems we encountered on our way (due to our initial lack of understanding of basic hybrid HMM/MLP systems, we only focused on feedforward networks and standard HMMs (i.e., using discriminant training but likelihoods during recognition). Now that we believe that we understand these systems better, we are interested in investigating the recurrent approach again.

- Discrete or continuous input units representing 9 frames of 10-ms acoustic vectors – Comparisons on several databases have shown that the use of continuous inputs led to significantly better performance, but at the cost of more processing time during both training and recognition. On speech databases containing more than a million training input patterns and networks with a few thousand parameters, it was possible to train MLPs with discrete inputs on standard workstations (SUN SparcStation 2). However, MLPs with continuous inputs required the development of the RAP computer (see Chapter 11) to obtain practical training times.

- Hidden units – In the case of discrete inputs, the architecture with no hidden units had the best performance. This is probably due to the fact that high-dimensional binary vectors as used in this case (typically 5,058 binary input units) are most likely to be linearly separable. In the case of continuous inputs (typically 234 input units), it was shown experimentally that performance improved significantly for larger hidden layers up to 1,000 units wide. Two kinds of hidden units have been tested on our large databases for our state-of-the-art systems: the standard sigmoidal hidden units and RBF units. In spite of some improvements of the theoretical formulation of the RBF approach (resolving the mismatch between likelihoods and posterior probabilities, output values constrained to be between 0 and 1 and summing to one), we never succeeded in getting improvements from RBFs.

- Output units – As many as there are distinct HMM states (or observation densities). Since we have generally used single state phone models, we typically had 50-69 outputs (corresponding to the number of phone models in the lexicon). In the case of Resource Management, for instance, we used 69 output units. In some experiments we added more units for phone occurrences in function words, and sometimes got a small improvement. In other experiments, we tried two- or three-state phone models, and used up to 200 output units. In this case we didn't see any improvement, so the 69-output net remained our standard. In case of context-dependent phone models, we developed a theory to split the resulting large network into smaller ones without any major assumptions. Results of this approach have not been reported explicitly in this work. A related system developed at SRI provided significant improvements in recognition performance [Cohen et al., 1992]. Recent improvements to our HMM recognizer[1], done in collaboration with researchers at Cambridge University, UK, has improved our performance for the simple single-density-per-phone, context-independent system to be comparable to that of the more complex context-dependent multi-state-per-phone systems. However, since so far context dependence has always improved our performance, further work is still required to improve context dependent HMM/MLP approaches.

15.2.2 Training

The main characteristics of our training algorithm are the following:

- On-line training – Standard EBP algorithm with weight update after each input training pattern. This leads (as is well known) to much faster convergence than is observed in off-line gradient approaches.[2]

[1] Mainly correcting bugs!

[2] Speed-up techniques like conjugate gradient optimization can also greatly improve the speed of gradient search. We have not done a comparison, but we suspect that the resultant training time is probably close to what we achieve with our simpler technique. However, since we routinely train our nets now with only 5 passes through the data, it is unlikely that a conjugate gradient approach would be faster, and it requires more computation per pass.

- Random sampling of the training data – Since we were doing on-line adaptation, we initially sorted acoustic vectors so that they were presented to the network in sequence by class number (e.g., first a vector from class 0, then a vector from class 1, ...). This was done to avoid overtraining that could result from successive presentation of several vectors corresponding to the same class. This also speeded up the training. Later on, it was observed that a random presentation of the training patterns led to slightly better training time and classification performance.

- Cross-validation – This was an important factor. Since we use large networks (with hundred of thousands of parameters), overtraining was a real problem. As a consequence we developed a simple (but efficient) cross-validation technique. In early work, we split the training data into (roughly) 2/3 for the actual training and 1/3 for cross-validation.[3] After each cycle of MLP training (presentation of all the training patterns) we test the performance of the resulting system on the cross-validation data and continue the training only if the performance on the cross-validation set improves.

- Step-size adaptation – This was also important in combination with cross-validation. Starting with a gradient step-size such as .01 (for the continuous input data) each time the performance on the cross-validation data degraded we divided the gradient step-size by 2 for the next MLP iteration. This process was iterated until the step-size was so small that a pass through the data provided no significant improvement. This also significantly speeded up training time. In a later refinement, we did only one training pass at each new learning rate after the first, since we had observed that the second pass for each new rate was never helping. This change cut the convergence time in half, while preserving the performance.

- Output nonlinear function – The standard sigmoid function has been successfully used. This has the advantage of minimizing the classification error rate while forcing the output values to be between 0 and 1. It has been shown that using a linear output function degraded the classification performance (even at the frame

[3] Ultimately we found that the approach worked as well with much less cross-validation data, so we ended up using 7/8 for training.

level, i.e., without using the output values as probabilities yet). Later on, it was observed that the use of the softmax function (instead of a sigmoid) slightly improved convergence time and performance (but not always significantly). This has the additional advantage of forcing the output values to sum to one.

- Training criterion – It has been observed that the relative entropy criterion led to faster convergence, especially at the beginning of the training. This can be explained by the fact that applying EBP rules to this criterion with a sigmoid function at the output leads to a correction linearly proportional to the error, which is not the case with the standard LMS criterion that saturates (even if the observed output is 0 for a desired output of 1). However, we also sometimes observed that an LMS criterion provided slightly better classification performance at the end of the training. As a consequence, we have been using both criteria quite successfully: starting with a relative entropy criterion, cross-validation and step-size adaptation and shifting to an LMS criterion when it was impossible to get any further improvements with the relative entropy. More recently, we have used the relative entropy only and achieved comparable performance.[4]

- Better initialization of the output node biases – It has been shown that, since the optimal output values are estimates of posterior probabilities, the output node biases (before the output nonlinearity) should be of the same order of magnitude as the "log odds" of the class priors (in the case of sigmoidal output) or log priors (in the case of softmax output). Since we can compute the class priors on our training data, it was possible to initialize the output node biases to values very close to the optimal solution. This significantly reduced training time, and also gave a small performance boost.

- Viterbi iteration – As was done with standard HMM Viterbi training, estimation of the probability density function parameters has to be embedded in an iterative improvement of the segmentation

[4]Comparison is difficult since many aspects of the system have changed over the last few years. However, at the time we last tried dropping the LMS step, we did so because it no longer seemed to be improving performance after the entropy-based learning was completed.

of the training database. This has to be done also when using an MLP to estimate these probabilities. The general HMM/MLP Viterbi training scheme is then the following:

- Start with a linear segmentation (or any other better segmentation) of your training data. In practice, we have found that, when no other choice is available, a solution better than the linear segmentation was to start with a segmentation in which the length of the segments is proportional to the average length of the constituting phonemes. Alternatively, you may start with a pre-existing MLP trained up on a labeled database such as TIMIT, and use the MLP to phonetically label the new database by a Viterbi that is forced to follow the known word transcription.
- Given this segmentation, train the MLP according to the specifications given above.
- When cross-validation tells you to stop, use the MLP probabilities (after division by priors; see next section on recognition) to resegment the training data by Viterbi alignment.
- Retrain the MLP and iterate with the previous step until the segmentation does not change anymore.

15.2.3 Recognition

For recognition, the following scheme has been used:

- MLP is used to generate posterior probabilities – Each 10-ms of speech we present the current acoustic vector with 4 frames of left context and 4 frames of right context to the input of the network. Running the MLP once for every time frame n provides us with all the required posterior probabilities $p(q_k|x_n)$ for all HMM states q_k (x_n representing a sequence of 9 acoustic vectors).

- Division by priors – Since HMMs require likelihoods we need to divide each of the MLP outputs by the relative class priors to give us a scaled likelihood that can be used in HMMs for recognition.

- Optimization of word transition costs – When used for continuous speech, it was very important to optimize by test-and-trial (on a set independent of the test set) the word transition probabilities.

This is a very well know requirement, even with standard HMMs (e.g., when using time derivative features). This is due to the fact that (word) transition probabilities are nondimensional while the order of magnitude of the emission probabilities depends on the input space. When using MLPs this problem is even worse given the size of the input space (this has been discussed in Chapter 7).

15.3 New Perspectives

The previous sections have described the basic scheme we have developed. Work at our institutions, as well as at Cambridge University[5] and SRI International[6], is currently extending these approaches in continuing research. It is beyond the scope of this book to discuss this work in any detail. However, here is a short list of some of the topics currently under consideration:

- *Context Dependent Neural Network* (CDNN) – as discussed in Chapter 9, this approach can potentially provide more detailed phone models. Most of the recent work on this topic has been done at SRI, where systems with duplicated output layers (and up to 1.5 million parameters) and systems with binary units representing context have been tried.

- *Gender Dependent Neural Networks* (GDNN) – at both SRI and ICSI, these networks (obeying the same factorization principles as the CDNNs) have been found to consistently provide a small performance improvement. In this case, the side information provided to the net is a gender label for the speaker (known during training, hypothesized during recognition). Further work has been done to try other categories than gender, including self-organized classes.

- Unnones – work by Chuck Wooters at ICSI is currently focused on altering the Viterbi training to be independent of a phonemically based lexicon. Like the IBM fenones, this approach would develop sound categories in a self-organized procedure. Unlike fenones,

[5] This collaboration is part of the European ESPRIT project WERNICKE (Basic Research Project no. 6487), in which LHS and ICSI are collaborating with Cambridge and INESC of Portugal to improve the base system.

[6] Currently this collaboration is funded by ARPA contract MDA904-90-C-5253.

the approach uses an MLP for probability estimation, as has been explained in this book.

- Recurrence – as noted earlier, we are interested in returning to the original form of MLP with feedback from the output targets. We intend to pursue this. Additionally, T. Robinson of Cambridge University (UK) has developed for some years a system similar to ours based on feedback from the hidden layer. We are currently working to bring our systems in sync with one another so that we can understand the differences.[7]

- Adaptation – Speaker adaptation is generally viewed as the modification of a system to provide better performance for a single speaker given a small amount of new training data. This is an issue with our hybrid system. A related issue is how to create a new, more general system rather than a more specific one – how can we integrate the new information without forgetting the old? This is a current topic of research.

- Return to prediction – Although we have had difficulties with predictive networks in the past, we are still intrigued, and still plan to continue this research.

15.4 Summary

In this chapter, we have tried to summarize the major points that we have ended up with in our standard hybrid HMM/MLP system. Hopefully both we and our readers can use this as a starting point for the next stages of our research.

[7] This is being done in the framework of an Esprit project (WERNICKE) - see [Robinson et al., 1993]

Chapter 16

CONCLUSIONS

I stand by all the misstatements I've made.
– Dan Quayle –

16.1 Introduction

As of this writing (1993), it is still too early to describe the long-term impact of neural networks on future ASR systems. Many of us have been attracted to ANNs at least partly because they often are useful for problems in which we have little prior knowledge. However, for a problem that has been investigated for decades like speech recognition, it is quite difficult to improve state-of-the-art systems with such a simple approach. We have yet to develop specific instances where such a completely "ignorance-based" approach can be used to successfully solve difficult problems. However, we now have a number of examples (in addition to the one described in this book) in which neural network techniques are successfully applied to practical problems such as recognizing handwritten postal mail codes or predicting time series.[1] However, progress in any of these areas still requires an extensive knowledge of relevant fields; we cannot disregard what has been achieved by more

[1] As noted earlier in the book, students and colleagues of Frank Rosenblatt developed Perceptron-based procedures, including Multi-layered forms, that were used in the '60s and '70s for a number of practical pattern recognition tasks. A number of these, including aerospace and biomedical applications, are described in [Viglione, 1970].

traditional techniques.

In this book we have mainly been concerned with improving state-of-the-art continuous speech recognition (CSR) systems by using ANN formalism. We believe that today's technology does not permit a full (and satisfactory) ANN solution to this problem. However, we have used networks to improve the statistical pattern matching at the local (frames) and global (words and sentence) level. We have also examined the use of networks for feature extraction, which at least appeared to preserve performance. In this work we have tried to describe the difficulty of improving a system composed of several building blocks just by improving one of them using ANNs. As we learned how to make a successful system, we gained an increased understanding of the basic principles that we have attempted to share with the reader.

Attacking CSR in this way permits us to compare our results with state-of-the-art systems, while generally ensuring that we incrementally improve the overall system. The hope is that this could ultimately lead to a "pure" neural network formalism that would make further interactions between the different modules much easier. This is the approach we have advocated in this work. Our main topic was improving the estimation of local probabilities with an MLP trained for phonetic classification. Additionally, we also discussed the properties and limitations of using an auto-associative MLP for feature extraction. This latter technique was only one approach out of many in this field, but it is one that is particularly interesting, and which was conducive to analysis using a linear algebraic perspective. In this case, we have not improved the global performance of the system; however at least this approach adopted a compatible "neural" form.

Although this book focused on the speech recognition application, most of the concepts discussed in this work can be useful for many pattern recognition problems.[2]

16.2 Hybrid HMM/ANN Systems: Status

From Chapter 6 to Chapter 9 of this book, we have shown how MLPs could improve HMM-based CSR systems by overcoming some of their limiting assumptions (e.g., better discrimination, better estimation of the probability density functions and some modeling of the correlation

[2]We hope.

between acoustic vectors). Accordingly, we have shown that MLPs, when appropriately used, could improve standard classifiers at the local level (when used as standard static classifiers) as well as the global level (when used in conjunction with higher-level systems such as HMMs). However, in this latter case, many modifications to the initial scheme were necessary to get demonstrable and systematic improvements. This required that we greatly increase our understanding of some fundamental issues that are discussed in this book. Among these were: statistical interpretation of the MLP outputs both in terms of *a posteriori* probabilities and *likelihoods*, good generalization properties of large neural networks (overtraining), and correct interface with HMMs.

The proposed approach has been developed and tested on several reference databases used for medium vocabulary (around 1,000 words), speaker-dependent and speaker-independent, continuous speech recognition systems. Initial comparisons have been done with simple HMM systems that were equivalent (in terms of topology and complexity) to our hybrid HMM/MLP approach, and the latter consistently led to better performance. The hybrid approach has also been integrated into one of the full-scale state-of-the-art recognizers [Murveit et al., 1989], and significantly improved the global system. Also, we have begun to see a shrinking of the difference in performance between our simplest HMM/MLP hybrid and this full-scale state-of-the-art system [Renals et al., 1992]. Through some recent system improvements[3] the recognition performance for a simple hybrid system (with a context-independent, single-density-per-phone HMM) appears to be competitive with much more complex (state-of-the-art) systems [Robinson et al., 1993]. This is impressive[4] considering that the more complex HMM approaches have been optimized for more than 10 years, while we started implementing our techniques only 5 years ago.

Naturally, we hope that this is just the beginning; the collaborative research between the authors, as well as the work at other research sites, may open the door to many new directions of enquiry, using new research paradigms. In fact, if we don't try something new, we are abandoning the future to boring and incremental improvements that may never converge to any real solution.

[3]Primarily finding more bugs!

[4]At least we're impressed...

16.3 Future Research Issues

16.3.1 In Hybrid HMM/ANN Approaches for CSR

There are still many issues in the hybrid HMM/MLP approach that require further research. Among these are:

- It is now clear that hybrid HMM/MLP approaches using a feedforward network with contextual input can improve recognition performance; this is unsurprising since one knows that discrimination and contextual information are two important features for good phoneme recognition. Similar approaches using MLPs with partial (hidden) feedback (but no contextual inputs) also appear to be at least as good as the approach presented in this work. However, we feel that both approaches should be complementary, and that it is probably worth going back to the initial theory (presented in Chapter 7) dealing with HMMs and MLPs with feedback and contextual input (to take the recent past on both states and observations into account). This is one of the goals of a new collaborative project (a European ESPRIT project referred to as "Wernicke") between Cambridge University (UK), INESC (Portugal), ICSI (Berkeley, CA), & L&H Speech Products (Belgium) [Robinson et al., 1993].

- Although we believe we have gained some insight into the relationships between ANNs and conventional statistics, there are still several problems that are understood quite well in conventional parametric statistics, but that need to be investigated in ANNs. A few examples are:

  - How can we combine neural networks? This is a problem that is easy to solve for example in the case of Gaussian classifiers: Given two Gaussian pdfs (as defined by the mean and variance) trained on two different training sets, it is easy to combine them into a single (Gaussian) density if we just remember the number of patterns that were present in each of the training sets. Is it also possible to do it properly with ANNs? As an example, this is an important issue when one wants to do speaker adaptation or, better, to train speaker-independent systems that automatically improve when getting more data (without forgetting what they have seen in the past).

  - How should ANNs deal with undersampled training data?

- How can we optimize the acoustic features as an integral part of the whole recognition system?

- How can we extend this approach to truly dynamical systems, using different strategies (including different features and variable frame rates) over time according to the incoming data and performing some kind of "assumption-free" integration over time?

16.3.2 In General

We still have to improve our understanding of neural networks in several ways [Morgan & Scofield, 1991], some of which are:

- Development of the means of recognizing (and producing) sequences of temporal patterns; so far, we still do not have any clues about how the brain solves this problem. We have a few ideas about the functionality of different "processing centers," but we do not know at this time how to piece them together to complete the figure of human speech perception.

- Better understanding of why ANN algorithms perform so well on certain types of problems while they completely fail on others. This is probably an issue as important as the understanding of ANNs themselves as well as their training algorithms. This was advocated in [Minsky & Papert, 1969] and is still reasonable.

- Learning how to use modular networks composed of multiple interconnected "subnetworks" associated with different sources (and types) of knowledge (e.g., low-level and high-level) and performing specific tasks.[5] It can be assumed that this will be a must for any complex pattern recognition device. For example, at the low level of the human speech recognition process, it is known that the auditory pathway is composed of different "groups" of neurons performing different auditory functions.

[5]We have been surprised by the size of the networks that we were able to use for the research described in this book. This use does not mean that we oppose the use of modular ANN structures. However, in some cases the modules may be large.

- How to have an efficient interaction between these networks. Ideally, activation of these networks should be data driven. Also, these networks (composed of a number of sub-networks) should be able to modify automatically their behavior or their connectivity (by training or adaptation) to external stimuli.

- Defining input features and internal data representations better suited to specific tasks, and to develop networks that incorporate (some of) the features found in biological systems.

- Hybrid systems combining neural networks and conventional approaches of pattern recognition, control theory, information theory, etc.

- For speech recognition in particular, it is also important to improve our models to better take into account the dynamical properties of the speech process. In this framework, methods should be developed to use more contextual information for classification. Also, (nonlinear) dynamical systems should be investigated further in different ways, including: (1) addressing the problems of dimensionality reduction and temporal variability by using nonlinear dynamic methods [Tishby, 1990; Farmer & Sidorowitch, 1987], and (2) finding how to make predictive approaches discriminant.

16.4 Concluding Remark

In this book we have discussed a few techniques that can improve state-of-the-art CSR systems. These also have the potential advantage of leading to simpler systems (in terms of complexity of the underlying HMM topology, speech units, and number of parameters). As a consequence, these systems could also be easier to integrate into more complex systems to further improve their performance. While our system shows promise, it is still very primitive and does not "solve" speech recognition. Despite all efforts to the contrary, we have left some space for improvement. In fact, we have the nagging feeling that something fundamental is still escaping us...

Bibliography

[1] Ahmed, N. & Rao, K.R. (1975). *Orthogonal Transforms for Digital Signal Processing*, New York, Springer-Verlag.

[2] Akaike, H. (1974). A New Look at the Statistical Model Identification, *IEEE Trans. on Autom. Control*, vol. 10, pp. 667-674.

[3] Akaike, H. (1986). Use of Statistical Models for Time Series Analysis, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, vol. 4, pp. 3147-3155, Tokyo.

[4] Almeida, L.B. (1987). A Learning Rule for Asynchronous Perceptrons with Feedback in a Combinatorial Environment, *IEEE Proc. First Intl. Conf. on Neural Networks*, M. Caudill and C. Butler (eds.), vol. II, pp. 609-618, New York: IEEE.

[5] Almeida, L.B. (1988). Backpropagation in Perceptrons with Feedback, in *Neural Computers*, R. Eckmiller and C. von der Marslburg (eds.), pp. 199-208, Berlin: Springer-Verlag.

[6] Amari, S.I. (1967). A Theory of Adaptive Pattern Classifiers, *IEEE Trans. on Elec. Com.*, vol. EC16, pp. 279-307.

[7] Asanović, K., Beck, J., Kingsbury, B., Kohn, P., Morgan, N. & Wawrzynek, J. (1991). SPERT: A VLIW/SIMD Microprocessor for Artificial Neural Network Computations. *Technical Report TR-91-072*, Intl. Computer Science Institute.

[8] Aubert, X.L. (1987). Supervised Phonetic Segmentation with Application to Speech Recognition, *Proc. of the European Conf. on Speech Technology*, Edinburgh (GB), vol. 2, pp. 161-164.

[9] Austin, S., Makhoul, J., Schwartz, R. & Zavaliagos, G. (1991). Continuous Speech Recognition Using Segmental Neural Network, *Proc. of the Fourth DARPA Workshop on Speech and Natural Language.*

[10] Bahl, L.R., Jelinek, F. & Mercer R.L. (1983). A Maximum Likelihood Approach to Continuous Speech Recognition, *IEEE Trans. on PAMI*, vol. 5, no. 2, pp. 179-190.

[11] Bahl, L.R., Brown, P.F., de Souza P.V. & Mercer R.L. (1986). Maximum Mutual Information Estimation of Hidden Markov Model Parameters, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 49-52, Tokyo.

[12] Bahl, L.R., Brown, P.F., de Souza P.V. & Mercer R.L. (1988). A New Algorithm for the Estimation of Hidden Markov Model Parameters, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 36-39, New York.

[13] Baker, J.K. (1975a). The DRAGON System - An Overview, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 24-29.

[14] Baker, J.K. (1975b). Stochastic Modeling as a Means of Automatic Speech Recognition, *Ph.D. Thesis, Computer Science Department, Carnegie Mellon University.*

[15] Bakis, R. (1976). Continuous Speech Recognition via Centisecond Acoustic States, *91st Meeting of the Acoustical Society of America.*

[16] Baldi, P. & Hornik, K. (1989). Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima, *Neural Networks*, vol. 2, pp. 53-58.

[17] Baldi, P. & Hornik, K. (1991). Back-Propagation and Unsupervised Learning in Linear Networks, *Back-Propagation: Theory, Architectures and Applications*, Y. Chauvin and D.E. Rumelhart (eds.), Lawrence Erlbaum, NJ.

[18] Baum, L.E. & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains, *Annals of Mathematical Statistics*, vol. 37, pp. 1554-1563.

[19] Baum, R. & Eagon, J.A. (1967). 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*, vol. 73, pp. 360-363.

[20] Baum, L.E., Petrie, T., Soules, G. & Weiss, N. (1970). A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains, *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164-171.

[21] Baum, L.E. (1972). An Inequality and Associated Maximization Techniques in Statistical Estimation of Probabilistic Functions of Markov Processes, *Inequalities*, no. 3, pp. 1-8.

[22] Baum, E.B. & Haussler, D. (1988). What Size Net Gives Valid Generalization?, *Neural Computation.*

[23] Beale, R. & Jackson, T. (1990). *Neural Computing: An Introduction*, Adam Hilger.

[24] Becker, S. & le Cun, Y. (1988). Improving the Convergence of Back-Propagation Learning with Second Order Methods, *Proc. 1988 Connectionist Models Summer School*, Carnegie-Mellon University.

[25] Bellegarda, J.R. & Nahamoo, D. (1990). Tied Mixture Continuous Parameter Modelling for Speech Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 2033-2045.

[26] Bengio, Y., De Mori, R., Flammia, G. & Kompe, R. (1992). Global Optimization of a Neural Network-Hidden Markov Model Hybrid, *IEEE Trans, on Neural Networks*, vol. 3, no. 2, pp. 252-259.

[27] Bezdek, C.J. (1980). A Convergence Theorem for the Fuzzy ISODATA Clustering Algorithms, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 2, pp. 1-8.

[28] Bourlard, H., Kamp, Y., Ney, H. & Wellekens, C.J. (1985). Speaker-Dependent Connected Speech Recognition via Dynamic Programming and Statistical Methods, in *Speech and Speaker Recognition*, M.R.Schroeder (ed.), Karger.

[29] Bourlard, H. & Wellekens, C.J. (1986). Discriminant Functions for Connected Speech Recognition, *Proc. of the EUSIPCO-86*, The Hague, pp. 507-510, I.T.Young, J.Biemond, R.P.W.Duin and J.J.Gerbrands (eds.).

[30] Bourlard, H. & Wellekens, C.J. (1987). Multilayer Perceptrons and Automatic Speech Recognition, *Proc. of the First Intl. Conf. on Neural Networks*, vol. IV, pp. 407-416, San Diego, CA.

[31] Bourlard, H. & Kamp, Y. (1988). Auto-Association by Multilayer Perceptrons and Singular Value Decomposition, *Biological Cybernetics*, no. 59, pp. 291-294.

[32] Bourlard, H. & Wellekens, C.J. (1989a). Links between Markov Models and Multilayer Perceptrons, in *Advances in Neural Information Processing Systems 1*, D.J. Touretzky (ed.), San Mateo: Morgan Kaufmann, pp. 502-510.

[33] Bourlard, H. & Wellekens, C.J. (1989b). Speech Dynamics and Recurrent Neural Networks, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 33-36, Glasgow, Scotland.

[34] Bourlard, H. & Wellekens, C.J. (1989c). Speech Pattern Discrimination and Multilayer Perceptrons, *Computer, Speech and Language*, vol. 3, pp. 1-19,

[35] Bourlard, H., Morgan, N. & Wellekens, C.J. (1990). Statistical Inference in Multilayer Perceptrons and Hidden Markov Models with Applications in Continuous Speech Recognition, in *Neurocomputing: Algorithms, Architectures and Applications*, F. Fogelman Soulié and J. Hérault (eds.), NATO ASI Series, pp. 217-226.

[36] Bourlard, H. & Wellekens, C.J. (1990). Links Between Markov Models and Multilayer Perceptrons, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 12, pp. 1167-1178.

[37] Bourlard, H. & Morgan, N. (1990). A Continuous Speech Recognition System Embedding MLP into HMM, in *Advances in Neural Information Processing Systems 2*, D. Touretzky (ed.), San Mateo, CA: Morgan Kaufmann, pp. 186-193.

[38] Bourlard, H. & Morgan, N. (1991). Merging Multilayer Perceptrons and Hidden Markov Models: Some Experiments in Continuous Speech Recognition, in *Neural Networks: Advances and Applications*, pp. 215-239, E. Gelenbe (ed.), North-Holland.

[39] Box, G.E. & Jenkins, G.M. (1976). *Time Series Analysis, Forecasting and Control*, Revised Edition, Holden-Day.

[40] Bridle, J.S., Brown, M.D. & Chamberlain, R.M. (1982). An Algorithm for Connected Word Recognition, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 899-902, Paris.

[41] Bridle, J.S. (1990a). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition, in *Neurocomputing: Algorithms, Architectures and Applications*, F. Fogelman Soulié and J. Hérault (eds.), NATO ASI Series, pp. 227-236.

[42] Bridle, J.S. (1990b). Alpha-Nets: A Recurrent "Neural" Network Architecture with a Hidden Markov Model Interpretation, *Speech Communication*, vol. 9, no. 1, pp. 83-92.

[43] Bridle, J.S. & Dodd, L. (1991). An Alphanet Approach to Optimising Input Transformations for Continuous Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 277-280, Toronto, Canada.

[44] Broomhead, D.S. & Lowe, D. (1988). Multi-Variable Functional Interpolation and Adaptive Networks, *Complex Systems*, vol. 2, pp. 321-355.

[45] Brown, P. (1987). The Acoustic-Modeling Problem in Automatic Speech Recognition, *Ph.D. Thesis, Computer Science Department, Carnegie-Mellon University*.

[46] Bryson A.E. Jr. & Yu Chi Ho (1969). *Applied Optimal Control*, Blaisdel Publishing Company.

[47] Bunch, J.R. & Nielsen, C.P. (1978). Updating the singular value decomposition. *Num. Math.*, vol. 31, pp. 111-129.

[48] Burr, D.J. (1986). A Neural Network Digit Recognizer, *Proc. of the Intl. Conf. on Systems, Man, and Cybernetics*, IEEE.

[49] Burr, D.J. (1987). Speech Stabilization and Robust Front Ends for Neural Networks, in *Neural Information Processing Systems* (Denver 1987), D.Z. Anderson (ed.), New York: American Institute of Physics.

[50] Cleeremans, A., Servan-Schreiber, D. & McClelland, J.L. (1989). Finite State Automata and Simple Recurrent Networks, *Neural Computation*, vol. 1, pp. 372-381.

[51] Cohen, M. (1989). *Phonological Structures for Speech Recognition*, PhD Thesis, University of California at Berkeley.

[52] Cohen, M., Murveit, H., Bernstein, J., Price, P. & Weintraub, M. (1990). The DECIPHER Speech Recognition System, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 77-80, Albuquerque, New Mexico.

[53] Cohen, M., Franco, H., Morgan, N., Rumelhart, D. & Abrash, V. (1992). Hybrid Neural Network/Hidden Markov Model Continuous Speech Recognition, *Proc. of Intl. Conf. on Speech and Language Processing*, pp. 915-918, Banff, CANADA.

[54] Cole, R.A., Fanty, M., Gopalakrishnan, M. & Janssen, R.D.T. (1991). Speaker-Independent Name Retrieval from Spellings Using a Database of 50,000 Names, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 325- 328, Toronto, CANADA.

[55] Cottrell, G.W., Munro, P.W. & Zipser, D. (1988). Image Compression by Back-Propagation: A Demonstration of Extensional Programming, in *Advances in Cognitive Science* (vol. 2), N.E. Sharkey (ed.), Norwood, (NJ): Ablex.

[56] Cowan, J. (1967). A Mathematical Theory of Central Nervous Activity. Thesis, University of London.

[57] Cybenko, G. (1989). Approximation by Superpositions of a Sigmoid Function, *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303-314.

[58] DARPA (1991). Proc. of the Speech and Natural Language Workshop, Pacific Grove, CA, Morgan Kauffmann.

[59] Davis, K.H., Biddulph, R. & Balashek, S. (1952). Automatic Recognition of Spoken Digits, *Journal Ac. Society of America*, vol. 24, no. 6, pp. 637-642.

[60] de La Noue, P., Levinson, S. & Sondhi, M. (1989). Incorporating the Time Correlation Between Successive Observations in an Acoustic-Phonetic Hidden Markov Model for Continuous Speech Recognition, *AT&T Technical Memorandum No. 11226.*

[61] Deller, J.R., Proakis, J.G. & Hansen, J.H. (1993). Discrete-Time Processing of Speech Signals, MacMillan.

[62] Delsarte, P., Kamp, Y. (1988). Low Rank Matrices With a Given Sign Pattern, *SIAM Journal of Discrete Math.*, vol. 2, pp. 51-63, 1989.

[63] Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). Maximum Likelihood From Incomplete Data Via the EM Algorithm, *Journal of the Royal Statistical Society*, vol. 39, pp. 1-38.

[64] Deng, L., Hassanein, K. & Elmasry, M. (1991). Neural-Network Architecture for Linear and Nonlinear Predictive Hidden Markov Models: Application to Speech Recognition, in *IEEE Proc. Workshop on Neural Networks for Signal Processing*, H. Juang, S. Kung and C. Kamm (eds.), Princeton New Jersey, pp. 411-421.

[65] Deng, L. (1992a). A Generalized Hidden Markov Model with State-Conditioned Trend Functions of Time for the Speech Signal, *Signal Processing*, 27 (1), pp. 65-78.

[66] Deng, L. (1992b). A Class of Nonstationary-State Hidden Markov Models with Applications to Speech Modeling and Recognition, *Proc. of the Intl. Conf. on Signal Processing Applications and Technology*, Boston, pp. 1025-1034.

[67] Denker, J.S., Gardner, W.R., Graf, H.P., Henderson, D., Howard, R.E, Hubbard, W., Jackel, L.D., Baird, H.S. & Guyon, I. (1989). Neural Network Recognizer for Hand-Written Zip Code Digits, *Advances in Neural Information Processing Systems 1*, D.S. Touretzky (ed.), Morgan Kaufmann, San Mateo, CA, pp. 323-331.

[68] Devijver, P. (1985). Baum's Forward-Backward Algorithm Revisited, *Pattern Recognition Letters*, vol. 3, pp. 369-373, North-Holland.

[69] Devijver, P.A & Kittler, J. (1982). *Pattern Recognition – A Statistical Approach*, Prentice Hall.

[70] Duda, R.O. & Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, Wiley.

[71] El-Jaroudi, A. & Makhoul, J. (1990). A New Error Criterion for Posterior Probability Estimation with Neural Nets, *Proc. Intl. Joint Conf. on Neural Networks*, pp. III:185-192, IEEE, San Diego, CA.

[72] Elman, J. & Zipser, D. (1988), Learning the Hidden Structure of Speech, *Journal of the Acoustical Society of America*, vol. 83, pp. 615-626.

[73] Elman, J.L. (1988). Finding Structure in Time, *CRL Tech. Report 8801*, University of California at San Diego, Center for Research in Language, also in *Cognitive Science*, vol. 14, pp. 179-211.

[74] Engle, R. (1982). Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom inflation, *Econometrica*, 50 (4), pp. 987-1007.

[75] Fanty, M. & Cole, R.A. (1991). Spoken Letter Recognition, in *Advances in Neural Information Processing Systems 3*, R.P. Lippmann, J.E. Moody and D.S. Touretzky (eds.), San Mateo, CA: Morgan Kaufmann, pp. 220-226.

[76] Farmer, J.D. & Sidorowitch, J.J. (1988). Exploiting Chaos to Predict the Future and Reduce Noise, *Los Alamos Technical Report*.

[77] Feldman, J.A., Fanty, M.A. & Goddard, N. (1988). Computing with Structured Neural Networks, *Computer*, vol. 21, no. 3, pp. 91-104.

[78] Franzini, M.A., Lee, K.F. & Waibel, A. (1990). Connectionist Viterbi Training: A New Hybrid Method for Continuous Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 425-428, Albuquerque, NM.

[79] Fukunaga, K. (1972). *Introduction to Statistical Pattern Recognition*, Academic Press.

[80] Furui, S. (1986). Speaker Independent Isolated Word Recognizer Using Dynamic Features of Speech Spectrum, *IEEE Trans. on Acoustic, Speech, and Signal Processing*, vol. 34, no. 1, pp. 52-59.

[81] Furui, S. (1991). Recent Advances in Speech Recognition, *Proc. of EUROSPEECH*, Genova, pp. 3-10.

[82] Furui, S. (1993). Towards Robust Speech Recognition Under Adverse Conditions, Proc. of the ESCA Workshop on *Speech Processing and Adverse Conditions*, pp. 31-41, Cannes-Mandelieu, France.

[83] Gevins, A. & Morgan, N. (1984). Ignorance-Based Systems, *IEEE Proc. Intl. Conf. on Acoustics, Speech & Signal Processing*, pp. 39A.5.1-4, San Diego, CA.

[84] Gish, H. (1990). A Probabilistic Approach to the Understanding and Training of Neural Network Classifiers, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 1361-1364, Albuquerque, NM.

[85] Gold, B., Lippmann, R. & Malpass, M.L. (1987). Some Neural Net Recognition Results on Steady State Vowels, *First Intl. Conf. on Neural Network*, IEEE, San Diego, CA.

[86] Golub, G.H. (1968). Least squares, singular values and matrix approximations. *Applikace Matematiky*, vol. 13, pp. 44-51.

[87] Golub, G.H. & Van Loan, C.F. (1983). *Matrix computations*, Oxford: North Oxford Academic.

[88] Gourieroux, C. (1992). *Modeles ARCH et Applications Financières*, Economica, collection Economie et statistiques avancées.

[89] Greenberg, S. (1988). The Ear as a Speech Analyzer. *Journal of Phonetics*, vol. 16, pp. 139-149.

[90] Gurgen, F., Sagayama, S. & Furui, S. (1990). Line Spectrum Pair Frequency-Based Distance Measures for Speech Recognition *IEEE Proc. Intl. Conf. on Spoken Language Processing*, pp. 13.1.1-13.1.4, Kobe, Japan.

[91] Haffner, P., Franzini, M. & Waibel, A. (1991). Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 105-108, Toronto, Canada.

[92] Hanson, B. & Applebaum, T. (1990). Robust Speaker-Independent Word Recognition Using Static, Dynamic, and Acceleration Features: Experiments with Lombard and Noisy Speech, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 857-860, Albuquerque, NM.

[93] Hennessy, J.L. & Patterson, D.A. (1990). *Computer Architecture A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA.

[94] Hertz, J., Krogh, A. & Palmer, R.G. (1991). *Introduction to the Theory of Neural Networks*, Addison-Wesley.

[95] Hermansky, H. (1990a). Perceptual Linear Predictive (PLP) Analysis of Speech, *Journal of the Acoust. Soc. Am.*, vol. 87, no. 4.

[96] Hermansky, H. (1990b). Personal communication.

[97] Hermansky, H., Bayya, A., Morgan, N. & Kohn, P. (1991). Compensation for the effect of the communication channel in Perceptual Linear Predictive (PLP) analysis of speech, *Proc. of Eurospeech 1991*, pp. 1367-1370, Genova, Italy

[98] Hinton, G.E. (1987). Connectionist Learning Procedures, *Technical report CMU-CS-87-115*, Carnegie Mellon University.

[99] Huang, X.D. & Jack, M.A. (1989). Semi-Continuous Hidden Markov Models for Speech Signals, *Computer, Speech and Language*, vol. 3, pp. 239-251.

[100] Iso, K.I. & Watanabe, T. (1990). Speaker-Independent Word Recognition Using a Neural Prediction Model, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 441-444, Albuquerque, NM.

[101] Iso, K.I. & Watanabe, T. (1991). Large Vocabulary Speech Recognition Using Neural Prediction Model, *IEEE Proc. of Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 57-60, Toronto.

[102] Jacobs, R. (1988). Increased Rates of Convergence Through Learning Rate Adaptation, *Neural Networks*, vol. 1, no. 4.

[103] Jelinek, F. (1976). Continuous Recognition by Statistical Methods, *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532-555.

[104] Jelinek, F. (1990). Self-Organized Language Modelling for Speech recognition, in *Readings in Speech Recognition*, A. Waibel and K.F. Lee (eds.), pp. 450-503.

[105] Jordan, M. (1986). Attractor Dynamics and Parallelism in a Connectionist Sequential Machine, *Proc. of the Eighth Annual Conf. of the Cognitive Science Society* (Amherst 1986), pp. 531-546, Hillsdale: Erlbaum.

[106] Jordan, M. (1989). Serial Order: A Parallel Distributed Processing Approach, in *Advances in Connectionist Theory: Speech*, J.L. Elman and D.E. Rumelhart (eds.), Hillsade: Erlbaum.

[107] Juang, B.H. (1984). On the Hidden Markov Model and Dynamic Time Warping for Speech Recognition: A Unified View, *AT&T Laboratories Technical Journal*, 63 (7), pp. 1213-1243.

[108] Juang, B.H. & Rabiner, L.R. (1985). Mixture Autoregressive Hidden Markov Models for Speech Signals, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 33, no. 6, pp. 1404-14013.

[109] Juang, B.H., Levinson, S.E. & Sondhi, M.M. (1986). Maximum Likelihood Estimation for Multivariate Mixture Observations of Markov Chains, *IEEE Trans. on Information Theory*, vol. IT-32, no. 2, pp. 307-309.

[110] Kanerva, P. (1988). *Sparse Distributed Memory*, MIT Press, Cambridge, MA.

[111] Karayiannis, N.B. & Venetsanopoulos, A.N. (1993). *Artificial Neural Networks – Learning Algorithms, Performance Evaluation and Applications*, Kluwer Academic Publishers.

[112] Kehagias, A. (1989). Optimal Control for Training: The Missing Link Between Hidden Markov Models and Connectionist Networks, *Division of Applied Mathematics Technical Report*, Brown University, Providence, RI.

[113] Kenny, P., Lennig, M. & Mermelstein, P. (1990). A Linear Predictive HMM for Vector-Valued Observations with Applications to Speech Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-38 (2), pp. 220-225.

[114] Kesten, H. (1957). Accelerated Stochastic Approximation, *Annals of Mathematical Statistics*, vol. 29, pp. 41-59.

[115] Kohonen, T. (1988a). *Self-Organization and Associative Memory*, 2nd. edition, New York, Tokyo, Springer-Verlag.

[116] Kohonen, T. (1988b). The "Neural" Phonetic Typewriter, *IEEE Computer Magazine*, vol. 21, no. 3, pp. 11-22.

[117] Kohn, P. (1991). CLONES: Connectionist Layered Object-Oriented Network Simulator. *ICSI Technical Report TR-91-073*, Intl. Computer Science Institute, 1991.

[118] Kramer, A.H. & Sangiovanni–Vincentelli, A. (1989). Efficient Parallel Learning Algorithms for Neural Networks, in *Advances in Neural Information Processing Systems 1*, D.J. Touretzky (ed.), San Mateo: Morgan Kaufmann, pp. 40-48.

[119] Krause, A. and Hackbarth, H. (1989). Scaly Artificial Neural Networks for Speaker-Independent Recognition of Isolated Words. *IEEE Proc. Intl. Conf. on Acoustics, Speech & Signal Processing*, vol. 1, pp. 21-24, Glasgow.

[120] Kuhn, G. (1987). A First Look at Phonetic Discrimination Using a Connectionist Network with Recurrent Links, *SCIMP Working Paper 4/87*, Institute for Defence Analysis, Communication Research Division.

[121] Kuhn, G., Watrous, R.L. & Ladendorf, B. (1990). Connected Recognition with a Recurrent Network, *Speech Communication*, vol. 9, no. 1, pp. 41-48.

[122] Kullback, S. & Leibler, R.A. (1951). On Information and Sufficiency, *Ann. Math. Stat.*, vol. 22, pp. 79-86.

[123] Kullback, S. (1959). *Information Theory and Statistics*, Wiley, New York.

[124] Lang, K.J., Waibel, A.H. & Hinton, G.E. (1990). A Time-Delay Neural Network Architecture for Isolated Word Recognition, *Neural Networks*, vol. 3, no. 1, pp. 23-43.

[125] Landauer, T.K., Kamm, C.A. & Singhal, S. (1987). Learning a Minimally Structured Back Propagation Network to Recognize Speech, *Proc. of the Ninth Annual Conf. of the Cognitive Science Society*, pp. 531-536.

[126] le Cun, Y. (1988). A Theoretical Framework for Back-Propagation, *Proc. of the 1988 Connectionist Models Summer School*, pp. 21-28, CMU, Pittsburgh, PA, Morgan Kaufmann.

[127] le Cun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. & Jackel, L.D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation*, 1(4), pp. 541-551.

[128] le Cun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. & Jackel, L.D. (1990). Handwritten Digit Recognition With a Back-Propagation Network, *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (ed.), Morgan Kaufmann, San Mateo, CA, pp. 396-404.

[129] Lee, K.F. (1989). *Automatic Speech Recognition - The Development of the Sphinx System*, Kluwer Academic, Norwell Mass.

[130] Lee, K.F., Hon, H.W., Hwang, M.Y., Mahajan, S. & Reddy, R. (1989). The Sphinx Speech Recognition System, *IEEE Proc. Intl. Conf. on Acoustics, Speech & Signal Processing*, vol. 1, pp. 445-448, Glasgow.

[131] Lee, K.F. (1990). Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 38, no. 4, pp. 599-609.

[132] Lehman, A., Park, E., Liao, P., Marrakchi, A. & Ptael, J. (1988). Factors Influencing Learning in Backpropagation, *IEEE Proc. Intl. Conf. on Neural Networks*, vol. I, pp. 335-341, San Diego, CA.

[133] Levin, E. (1990). Speech Recognition Using Hidden Control Neural Network Architecture, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 433-436, Albuquerque, NM.

[134] Levin, E. (1993). Hidden Control Neural Architecture Modeling of Nonlinear Time Varying Systems and Its Applications, *IEEE Trans. on Neural Networks*, vol. 4, no. 1, pp. 109-116.

[135] Levinson, S.E., Rabiner, L.R. & Sondhi, M.M. (1983). An Introduction to the Application of the Theory of Probabilistic Functions on a Markov Process to Automatic Speech Recognition, *The Bell System Technical Journal*, vol. 64, no. 4, pp. 1035-1074.

[136] Levinson, S.E. (1985). A Unified Theory of Composite Pattern Analysis for Automatic Speech Recognition, in *Computer Speech Processing*, F. Fallside and W.A. Woods (eds.), Englewood Clifs, NJ: Prentice-Hall, pp. 243-272.

[137] Linde, Y., Buzo, A. & Gray, R.M. (1980). An algorithm for vector Quantizer Design, *IEEE Trans. on Communications*, vol. 28, no. 1, pp. 84-95.

[138] Liporace, L.A. (1982). Maximum Likelihood Estimation for Multivariate Observations of Markov Sources, *IEEE Trans. on Information Theory*, vol. IT-28, no. 5, pp. 729-734.

[139] Lippmann, R.P. (1987). An Introduction to Computing with Neural Nets, *IEEE Magazine on Acoustic, Speech, and Signal Processing*, vol. 4, no. 2, pp. 4-22.

[140] Lippmann, R.P. (1989). Review of Neural Networks for Speech Recognition, *Neural Computation*, vol. 1, no. 1, pp. 1-38.

[141] Lippmann, R.P. & Gold, B. (1987). Neural Classifiers Useful for Speech Recognition, *IEEE Proc. First Intl. Conf. on Neural Networks*, vol. IV, pp. 417-422, San Diego, CA.

[142] Lowerre, B.T. (1976). The HARPY Speech Recognition System, *PhD Thesis, Computer Science Department, Carnegie Mellon University.*

[143] MacKay, D.J. (1987). A Method of Increasing the Contextual Input to Adaptive Pattern Recognition Systems, *Technical Report no. RIPRREP/1000/14/87*, Royal Signals and Radar Establishment, Malvern, U.K.

[144] Makhoul, J. (1975). Linear Prediction: A Tutorial Review, *Proceedings of the IEEE*, vol. 63, pp. 561-580.

[145] Makhoul, J., Roucos, S. & Gish, H. (1985). Vector Quantization in Speech Coding, *Proceedings of the IEEE*, vol. 73, no. 11, pp. 1551-1588.

[146] Makino, S., Kawabata, T. & Kido, K. (1983). Recognition of Consonants Based on the Perceptron Model, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 738-741, Boston, MA.

[147] Makram-Ebeid, S., Sirat, J.A. & Viala, J.R. (1989). A Rationalized Back-Propagation Learning Algorithm, *Proc. Intl. Joint Conf. on Neural Networks*, vol. II, pp. 373-380, Washington.

[148] Marcus, S.M. (1981). ERIS-context sensitive coding in speech perception, *Journal of Phonetics*, vol. 9, pp. 197-220.

[149] Marcus, S.M. (1985). Associative Models and the Time Course of Speech, in *Speech and Speaker Recognition*, M.R.Schroeder (ed.), KARGER.

[150] Martin, E.A., Lippmann, R.P. & Paul, D.B. (1987). Two-Stage Discriminant Analysis for Improved Isolated Word Recognition, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 17.5.1-17.5.4, Dallas, TX.

[151] McDermott, E. & Katagiri, S. (1989). Shift-Invariant, Multi-Category Phoneme Recognition using Kohonen's LVQ2. *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 81-84, Glasgow.

[152] McDermott, E. & Katagiri, S. (1989). LVQ-Based Shift-Tolerant Phoneme Recognition, *IEEE Trans. on Signal Processing*, vol. 39, no. 6, pp. 1398-1411.

[153] Merialdo, B. (1988). Phonetic Recognition Using Hidden Markov Models and Maximum Mutual Information Training, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 111-114, New York.

[154] Minsky, M. & Papert, S. (1969). *Perceptrons*, Cambridge, MA: MIT Press.

[155] Minsky, M. & Papert, S. (1988). *Perceptrons*, Expended Edition, Cambridge, MA: MIT Press.

[156] Morgan, D.P. & Scofield, C.L. (1991). *Neural Networks and Speech Processing*, Kluwer Academic Publishers.

[157] Morgan, N. & Bourlard, H. (1990a). Generalization and Parameter Estimation in Feedforward Nets: Some Experiments, in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (ed.), San Mateo, CA: Morgan Kaufmann, pp. 630-637.

[158] Morgan, N. & Bourlard, H. (1990b). Continuous Speech Recognition Using Multilayer Perceptrons with Hidden Markov Models, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 413-416, Albuquerque, NM.

[159] Morgan, N., Wooters, C., Bourlard, H. & Cohen, M. (1990a). Continuous Speech Recognition on the Resource Management Database Using Connectionist Probability Estimation, *Proc. of Intl. Conf. on Speech and Language Processing*, pp. 1337-1340, Kobe, Japan.

[160] Morgan, N., Beck, J., Kohn, P., Bilmes, E., Allman, E. & Beer, J. (1990b). The RAP: A Ring Array Processor for Layered Network Calculations, *IEEE Proc. of the Intl. Conf. on Application Specific Array Processors*, pp. 296-308, IEEE Computer Society Press, Princeton, NJ.

[161] Morgan, N., Hermansky, H., Bourlard, H., Kohn, P. & Wooters, C. (1991). Continuous Speech Recognition Using PLP Analysis with Multilayer Perceptrons, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 49-52, Toronto, Canada.

[162] Morgan, N., Beck, J., Kohn, P., Bilmes, E., Allman, E. & Beer, J. (1992). The Ring Array Processor (RAP): A Multiprocessing Pe-

ripheral for Connectionist Applications, *Journal of Parallel and Distributed Precessing*, in press.

[163] Morgan, N. & Bourlard, H. (1992). Factoring Networks by a Statistical Method, *Neural Computation*, vol. 4, no. 6, pp. 835-838.

[164] Murveit, H. & Brodersen, R.W. (1988). An Integrated-Circuit Based Speech Recognition Systems, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 34, no. 6, pp. 1465-1472.

[165] Murveit, H. & Weinstraub, M. (1988). 1000-Word Speaker-Independent Continuous Speech Recognition System, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 115-118, New York.

[166] Murveit, H., Cohen, M., Price, P., Baldwin, G. & Weintraub, M. (1989). SRI's DECIPHER System, *Proc. of the DARPA Speech and Natural Language Workshop*, February.

[167] Nakamura, M., Tamura, S. & Sagayama, S. (1991). Phoneme Recognition by Phoneme Filter Neural Networks, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 85-88, Toronto.

[168] Newton, I. (1687). *Philosophiae Naturalis Principea Mathematica*, Royal Society Press, London.

[169] Ney, H. (1984). The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition, *IEEE Trans. on Acoustic, Speech, and Signal Processing*, vol. 32, pp. 263-271, 1984.

[170] Ney, H., Mergel, D., Noll, A. & Paeseler, A. (1987). A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 833-836, Dallas, TX.

[171] Ney, H. & Noll, A. (1988). Phoneme Modelling Using Continuous Mixture Densitites, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 437-440.

[172] Niles, L.T., Silverman, H., Tajcham, G. & Bush, M. (1989). How Limited Training Data Can Allow a Neural Network Classifier to

Outperform an "Optimal Statistical Classifier", *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 17-20, Glasgow, Scotland.

[173] Niles, L.T. & Silverman, H.F. (1990). Combining Hidden Markov Models and Neural Network Classifiers, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 417-420, Albuquerque, NM.

[174] Nilsson, N.J. (1965). *Learning Machines*, Mac Graw-Hill.

[175] Nilsson, N.J. (1980). *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto.

[176] Pao, Y.H. (1989). *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley.

[177] Parker, D.B. (1982). *Invention Report S81-64*, File 1, Office of Technology Licensing, Stanford University.

[178] Parker, D.B. (1985). Learning Logic, *Technical Report TR-47*, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA.

[179] Parker, D.B. (1987). Optimal Algorithms for Adaptive Networks: Second Order Back-Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning, *IEEE Proc. First Intl. Conf. on Neural Networks*, vol. 2, pp. 593- 600, San Diego, CA.

[180] Paul, D.B. (1989). The Lincoln Robust Continuous Speech Recognizer, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, Glasgow, Scotland.

[181] Paul, D.B. (1991). The Lincoln Tied-Mixture HMM Continuous Speech Recognizer, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 329-332, Toronto, Canada.

[182] Paul, D.B., Baker, J.K. & Baker, J.M. (1991). On the Interaction Between True Source, Training, and Testing Language Models, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 569-572, Toronto, Canada.

[183] Paul, D.B., Necioglu, B.F. (1993). The Lincoln Large-Vocabulary Stack-Decoder HMM CSR, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. II-660-663, Minneapolis, MN.

[184] Pearl, J. (1978). On the Connection Between the Complexity and the Credibility of Inferred Models, *Intl. Journal General Systems*, vol. 4, pp. 155-164.

[185] Peeling, S.M. & Moore, R.K. (1988). Isolated Digit Recognition Experiments Using the Multi-Layer Perceptron, *Speech Communication*, vol. 7, pp. 403-409.

[186] Petek, B., Waibel, A. & Tebelskis, M. (1992). Integrated Phoneme and Function Word Architecture of Hidden Control Neural Networks for Continuous Speech Recognition, *Speech Communication*, 11, pp. 273-282.

[187] Pineda, F.J. (1987). Generalization of back-propagation to recurrent neural networks, *Phys. Rev. Lett.*, vol. 18, pp. 2229-2232.

[188] Pineda, F.J. (1988). Dynamics and Architecture in Neural Computation, to appear in *Journal of Complexity*, special issue on Neural Networks.

[189] Poritz, A.B. (1982). Linear Predictive Hidden Markov Models and the Speech Signal, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 1291-1294, Paris.

[190] Poritz, A.B. (1988). Hidden Markov Models: A Guided Tour, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 7-13, New York.

[191] Powell, M.J.D. (1985). Radial Basis Functions for Multi-Variable Interpolation: A Review, Technical Report DAMPT/NA12, Dept. of Applied Mathematics and Theoretical Physics, University of Cambridge.

[192] Priestley, M.B. (1991). *Non-Linear and Non-Stationary Time Series Analysis*, Academic Press.

[193] Rabiner, L.R. & Schafer, R.W. (1978). *Digital Processing of Speech Signals*, Prentice-Hall, Inc., Englewood Cliffs, NJ.

[194] Rabiner, L.R. & Wilpon J.G. (1981). A Two-Pass Pattern-Recognition Approach to Isolated Word Recognition, *Bell System Technical Journal*, vol. 60, no. 5, pp. 739-766.

[195] Rabiner, L.R., Juang, B.H., Levinson, S.E. & Sondhi, M.M. (1985). Recognition of Isolated Digits Using Hidden Markov Models with Continuous Mixture Densities, *AT&T Technical Journal*, vol. 64, no. 6, pp. 1211-1233.

[196] Rabiner, L.R. & Juang, B. (1986). An Introduction to Hidden Markov Models, *IEEE Magazine on Acoustic Speech, and Signal Processing*, vol. 3, no. 1, pp. 4-16.

[197] Rabiner, L.R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-285.

[198] Renals, S. (1990). *Speech and Neural Network Dynamics*, PhD Thesis, Department of Physics, University of Edinburgh.

[199] Renals, S., Morgan, N., Bourlard, H. (1991). Probability Estimation by Feed-Forward Networks in Continuous Speech Recognition, in *IEEE Proc. Workshop on Neural Networks for Signal Processing*, pp. 309-318, B.H. Juang, S.Y. Kung and C.A. Kann (eds.), Princeton, NJ.

[200] Renals, S., Morgan, N., Cohen, M. & Franco, H. (1992). Connectionist Probability Estimation in the Decipher Speech Recognition System, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 601-604, San Francisco, CA.

[201] Richard, M.D. & Lippmann, R.P. (1991). Neural Network Classifiers Estimate Bayesian a Posteriori Probabilities, *Neural Computation*, vol. 3, no. 4, pp. 461-483.

[202] Richter, A.G. (1986). Modelling of Continuous Speech Observations, *Advances in Speech Processing Conf.*, IBM Europe Institute, July 1986.

[203] Robinson, A.J. & Fallside, F. (1987). The Utility Driven Dynamic Error Propagation Network, *Tech. Report CUED/F-INFENG/TR.1*, Cambridge University, U.K.,

[204] Robinson, A.J. & Fallside, F. (1988). Static and Dynamic Error Propagation Networks with Application to Speech Coding, in *Neural Information Processing Systems* (Denver 1987), D.Z. Anderson (ed.), pp. 632-642, New York: American Institute of Physics.

[205] Robinson, A.J. & Fallside, F. (1990). Phoneme Recognition from the TIMIT Database Using Recurrent Error Propagation Networks, *Technical Report CUED/F-INFENG/TR42*, Engineering Department, Cambridge University, UK.

[206] Robinson, A.J. & Fallside, F. (1991). A Recurrent Error Propagation Network Speech Recognition System, *Computer, Speech and Language*, vol. 5, pp. 257-286.

[207] Robinson, T., Almeida, L., Boite, J.M., Bourlard, H., Fallside, F., Hochberg, M., Kershaw, D., Kohn, P., Konig, Y., Morgan, N., Neto, J.P., Renals, S., Saerens, M., Wooters, C. (1993). A Neural Network Based, Speaker Independent, Large Vocabulary, Continuous Speech Recognition System: The WERNICKE Project, *Eurospeech'93*, Berlin, Germany.

[208] Rosenblatt, F. (1958). The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, *Psychological Review*, vol. 65, pp. 386-408.

[209] Rosenblatt, F. (1960). On the Convergence of Reinforcement Procedures in Simple Perceptrons, *Cornell Aeronautical Laboratory Report VG-1196-G-4*, Buffalo, NY.

[210] Rosenblatt, F. (1962). *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, Washington.

[211] Rohwer, R. & Forrest, B. (1987). Training Time-Dependence in Neural Networks, *IEEE Proc. First Intl. Conf. on Neural Networks*, M. Caudill and C. Butler (eds.), vol. 2, pp. 701-708, San Diego, CA.

[212] Rohwer, R. (1990). The "Moving Targets" Training Algorithm, in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (ed.), San Mateo, CA: Morgan Kaufmann, pp. 558-565.

[213] Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986a). Learning Internal Representations by Error Propagation, in *Parallel Distributed Processing. Exploration of the Microstructure of Cognition.*

vol. 1: Foundations, D.E.Rumelhart and J.L.McClelland (eds.), MIT Press.

[214] Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986b). *Parallel Distributed Processing. Exploration of the Microstructure of Cognition. vol. 1: Foundations*, D.E.Rumelhart and J.L.McClelland (eds.), MIT Press.

[215] Ruspini, E. (1970). Numerical Methods for Fuzzy Clustering, *Information Sciences*, vol. 2, pp. 319-350.

[216] Sankoff, D. & Kruskal, J.B. (1983). *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*, Reading, MA: Addison-Wesley.

[217] Saerens, M. (1992a). A Path Integral Formulation of Viterbi Algorithm for One-Gaussian-Per-State Hidden Markov Models, submitted for publication.

[218] Saerens, M. (1992b). Viterbi Algorithm for Acoustic Vectors Generated by a Linear Stochastic Differential Equation on Each State, submitted for publication.

[219] Saerens, M. & Bourlard, H. (1993). Linear and Nonlinear Prediction for Speech Recognition with Hidden Markov Models, *LHS Manuscript*, submitted for publication.

[220] Sawai, H., Waibel, A., Haffner, P., Miyatake, M. & Shikano, K. (1989). Parallelism, Hierarchy, Scaling in Time-Delay Neural Networks for Spotting Japanese Phonemes/CV-Syllables, *Proc. Intl. Joint Conf. on Neural Networks*, vol. II, pp. 81-88, Washington, D.C.

[221] Schwartz, R., Chow, Y., Kimball, O., Roucos S., Krasner M. & Makhoul J. (1985). Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 1205-1208, Tampa, FL.

[222] Sejnowski, T.J. & Rosenberg, C.R. (1986). NETtalk: A Parallel Network that Learns to Read Aloud, *Technical Report JHU/EECS-86/01*, Johns Hopkins University, Baltimore, MA.

[223] Sejnowski, T.J. & Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text. *Complex Systems*, vol. 1, pp. 145-168.

[224] Silva, F.M. & Almeida, L. (1990). Speeding Up Backpropagation, in *Advanced Neural Computers*, R. Eckmiller (ed.), pp. 151-158, North-Holland.

[225] Simpson, P.K. (1990). *Artificial Neural Networks: Foundations, Paradigms, Applications, and Implementations*, Pergamon Press.

[226] Solla, S.A, Levin, E. & Fleisher M. (1988). Accelerated Learning in Layered Neural Networks, *Complex Systems*, vol. 2, pp. 625-640.

[227] Sontag, E.D. & Sussman, H.J. (1989a). Backpropagation Separates When Perceptrons Do. *Proc. Intl. Joint Conf. on Neural Networks* vol. I, pp. 639-642, Washington, D.C.

[228] Sontag, E.D. & Sussman, H.J. (1989b). Backpropagation Can Give Rise to Spurious Local Minima Even for Networks without Hidden Layers, *Complex Systems*, vol. 3, pp. 91-106.

[229] SPRINT (1990). Speech Processing and Recognition Using Integrated Neurocomputing Techniques, *ESPRIT Project - BRA 3228.*

[230] Stewart, G.W. (1973). *Introduction to Matrix Computations.* New York, Academic Press.

[231] Stoelzle, A., Narayanaswamy, S., Schrupp, P., Richards, B., Yu, R., Rabaey, J. & Brodersen, R. (1986). A Flexible VLSI 60,000 Word Real-Time Continuous Speech Recognition System, in *VLSI Signal Processing IV*, H.S. Moscovitz, K. Yao and R. Jain (eds.), IEEE Press, New York, NY.

[232] Takens, F. (1981). *Dynamical Systems and Turbulence*, Lecture Notes in Math., Berlin, Springer-Verlag.

[233] Tank, D.W. & Hopfield, J.J. (1987). Concentrating Information in Time: Analog Neural Networks with Applications to Speech Recognition Problems, *Proc. of the First Intl. Conf. on Neural Networks*, vol. IV, pp. 455-468, San Diego, CA.

[234] Tebelskis, J. & Waibel, A. (1990). Large Vocabulary Recognition Using Linked Predictive Neural Networks, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 437-440, Albuquerque, NM.

[235] Tebelskis, J., Waibel, A., Petek, B. & Schmidbauer, O. (1991). Continuous Speech Recognition Using Linked Predictive Neural Networks, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 61-64, Toronto.

[236] Tishby, N. (1990). A Dynamical Systems Approach to Speech Processing, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 365-368, Albuquerque, NM.

[237] Tishby, N. (1991). On the Application of Mixture AR Hidden Markov Models to Text Independent Speaker Recognition, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-39 (3), pp. 563-570.

[238] Titterington, D.M., Smith, A,F. & Makov, U.E. (1985). *Statistical Analysis of Finite Mixture Distributions*, John Wiley & Sons.

[239] Townshend, B. (1991). Nonlinear Prediction of Speech, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 425-428, Toronto.

[240] Tsuboka, E., Takada, Y. & Wakita, H. (1990). Neural Predictive Hidden Markov Model, *Proc. Intl. Conf. on Spoken Language Processing*, pp. 31.2.1-31.2.4, Kobe, Japan.

[241] Unnikrishnan, K.P., Hopfield, J.J. & Tank, D.W. (1988). Learning Time-delayed Connections in a Speech Recognition Circuit, *Neural Network for Computing*, Snowbird, UT.

[242] Valiant, L.G. (1984). A Theory of the Learnable, *Communications of the ACM*, vol. 27, no. 11, pp. 1134-1142.

[243] Viglione, S.S. (1970). Applications of Pattern Recognition Technology in Adaptive Learning and Pattern Recognition Systems, in *Adaptive Learning and Pattern Recognition Systems*, J.M. Mendel and K.S Fu (eds.), New York, Academic Press, pp. 115-161.

[244] Viterbi, A.J. (1967). Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, *IEEE Trans. on Information Theory*, vol. 13, no. 2, pp. 260-269.

[245] Waibel, A., Hanazawa, T., Hinton, G.E., Shikano, K. & Lang, K.J. (1988). Phoneme Recognition: Neural Networks vs. Hidden Markov Models, *Proc. of the 1988 Intl. Conf. on Acoustics, Speech, and Signal Processing*, New York, vol. 1, pp. 107-110.

[246] Waibel, A., Hanazawa, T., Hinton, G.E., Shikano, K. & Lang, K.J. (1989). Phoneme Recognition Using Time-Delay Neural Networks, *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1888-1898.

[247] Watrous, R.L. & Shastri, L. (1986). Learning Phonetic Features Using Connectionist Networks: An Experiment in Speech Recognition, *Technical Report MS-CIS-86-78*, University of Pennsylvania, Philadelphia.

[248] Watrous, R. (1987). Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization, *IEEE Proc. First Intl. Conf. on Neural Networks*, vol. IV, pp. 389-396, San Diego, CA.

[249] Watrous, R.L. & Shastri, L. (1987). Learning Phonetic Features Using Connectionist Networks: An Experiment in Speech Recognition, *First Intl. Conf. on Neural Networks*, vol. 2, pp. 619-627, San Diego.

[250] Webb, A.R. & Lowe, D. (1989). Adaptive Feed-Forward Layered Networks as Pattern Classifiers: A Theorem Illuminating Their Success in Discriminant Analysis, *Neural Networks*.

[251] Weintraub, M., Murveit, H., Cohen, M., Price, P., Bernstein, J., Baldwin, G. & Bell, D. (1989). Linguistic Constraints in Hidden Markov Models Based Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, vol. 2, pp. 699-702, Glasgow, Scotland.

[252] Wellekens, C.J. (1986). Global Connected Digit Recognition Using Baum-Welch Algorithm, *IEEE Proc. Intl. Conf. on Acoustic Speech, and Signal Processing*, pp. 21.5.1-21.5.4, Tokyo.

[253] Wellekens, C.J. (1987). Explicit Time Correlation in Hidden Markov Models for Speech Recognition, *IEEE Proc. Intl. Conf. on Acoustic, Speech, and Signal Processing*, pp. 384-386, Dallas, TX.

[254] Werbos, P.J. (1974). Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, *PhD Thesis*, Harvard University, Cambridge, MA.

[255] White, H. (1988). Multilayer Feedforward Networks Can Learn Arbitrary Mappings: Connectionist Nonparametric Regression with Automatic and Semi-Automatic Determination of Network Complexity, *Discussion Paper, University of California*, San Diego, Department of Economics.

[256] Widrow, B. & Hoff, M.E. (1960). Adaptive Switching Circuits, *Technical Reports 1553-1*, Stanford University, Electron. Labs., Stanford, CA.

[257] Widrow, B. & Stearns, S. (1985). *Adaptive Signal Processing*, Englewood Cliffs: Prentice-Hall.

[258] Widrow, B. (1987). ADALINE and MADALINE, Plenary Speech, *IEEE Proc. First Intl. Conf. on Neural Networks*, vol. 1, pp. 143-158, San Diego, CA.

[259] Williams, R.J. & Zipser, D. (1989a). A Learning Algorithm for Continually Running Fully Recurrent Neural Networks, *Neural Computations*, vol. 1, pp. 270-280.

[260] Williams, R.J. & Zipser, D. (1989b). Experimental Analysis of the Real-Time Recurrent Learning Algorithm, *Connection Science*, vol. 1, pp. 87-111.

[261] Woodland, P. (1992). Hidden Markov Models Using Vector Linear Prediction and Discriminative Output Distributions, *IEEE Proc. Intl. Conf. on Acoustics, Speech, and Signal Processing*, pp. 509-512, San Francisco, CA.

[262] Zurada, J.M. (1992). *Introduction to Artificial Neural Systems*, West Publishing Company.

Index

Acronyms

AI: Artificial Intelligence

AR: Autoregressive (model)

ARCH: Autoregressive Conditional Heteroscedastic (model)

ARMA: Autoregressive Moving Average (model)

ANN: Artificial Neural Network

ASR: Automatic Speech Recognition

CDNN: Context-Dependent Neural Network

CSR: Continuous Speech Recognition

DP: Dynamic programming

DTW: Dynamic Time Warping

EBP: Error Back-Propagation

EM: Estimate-and-Maximize (algorithm)

FIR: Finite Impulse Response (filter)

GDNN: Gender-Dependent Neural Network

HMM: Hidden Markov Model

iid: independent, identically distributed (random variables)

IIR: Infinite Impulse Response (filter)

LMS: Least Mean Square (criterion)

MAP: Maximum A Posteriori (probability)

MLE: Maximum Likelihood Estimate

MLP: Multilayer Perceptron

MSE: Mean Square Error

MMI: Maximum Mutual Information

pdf: probability density function

RAP: Ring Array Processor

RBF: Radial Basis Function

RM : Resource Management (DARPA database)

TDNN: Time Delay Neural Network

Don't follow leaders; watch your parking meters.
– Bob Dylan –