Gernot A. Fink

Markov Models for Pattern Recognition

From Theory to Applications

Springer

Markov Models for Pattern Recognition

Gernot A. Fink

Markov Models for Pattern Recognition

From Theory to Applications

With 51 Figures

Springer

Gernot A. Fink
Department of Computer Science
University of Dortmund
Otto-Hahn-Str. 16
44221 Dortmund
Germany
Gernot.Fink@udo.edu

Library of Congress Control Number: 2007935304

Originally published in the German language by B.G. Teubner Verlag as “Gernot A. Fink: Mustererkennung mit Markov-Modellen”. © B.G. Teubner Verlag | GWV Fachverlage GmbH, Wiesbaden 2003

ISBN 978-3-540-71766-9 Springer Berlin Heidelberg New York

Typesetting: by the Author
Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover design: KünkelLopka Werbeagentur, Heidelberg

Printed on acid-free paper 33/3180/YL - 5 4 3 2 1 0

For my parents

Preface

The development of pattern recognition methods on the basis of so-called *Markov models* is tightly coupled to the technological progress in the field of automatic speech recognition. Today, however, Markov chain and hidden Markov models are also applied in many other fields where the task is the modeling and analysis of chronologically organized data, for example genetic sequences or handwritten texts. Nevertheless, in monographs, Markov models are almost exclusively treated in the context of automatic speech recognition and not as a general, widely applicable tool of statistical pattern recognition.

In contrast, this book puts the formalism of Markov chain and hidden Markov models at the center of its considerations. With the example of the three main application areas of this technology — namely automatic speech recognition, handwriting recognition, and the analysis of genetic sequences — this book demonstrates which adjustments to the respective application area are necessary and how these are realized in current pattern recognition systems. Besides the treatment of the theoretical foundations of the modeling, this book puts special emphasis on the presentation of algorithmic solutions, which are indispensable for the successful practical application of Markov model technology. Therefore, it addresses researchers and practitioners from the field of pattern recognition as well as graduate students with an appropriate major field of study, who want to devote themselves to speech or handwriting recognition, bioinformatics, or related problems and want to gain a deeper understanding of the application of statistical methods in these areas.

The origins of this book lie in the author's extensive research and development in the field of statistical pattern recognition, which initially led to a German book published by Teubner, Wiesbaden, in 2003. The present edition is basically a translation of the German version with several updates and modifications addressing an international audience. This book would not have been possible without the encouragement and support of my colleague Thomas Plötz, University of Dortmund, Germany, whom I would like to cordially thank for his efforts.

Dortmund, July 2007 *Gernot A. Fink*

Contents

1 Introduction . . . 1
- 1.1 Thematic Context . . . 3
- 1.2 Functional Principles of Markov Models . . . 3
- 1.3 Goal and Structure of the Book . . . 5

2 Application Areas . . . 7
- 2.1 Speech . . . 7
- 2.2 Writing . . . 14
- 2.3 Biological Sequences . . . 22
- 2.4 Outlook . . . 26

Part I Theory

3 Foundations of Mathematical Statistics . . . 33
- 3.1 Random Experiment, Event, and Probability . . . 33
- 3.2 Random Variables and Probability Distributions . . . 35
- 3.3 Parameters of Probability Distributions . . . 37
- 3.4 Normal Distributions and Mixture Models . . . 38
- 3.5 Stochastic Processes and Markov Chains . . . 40
- 3.6 Principles of Parameter Estimation . . . 41
  - 3.6.1 Maximum Likelihood Estimation . . . 41
  - 3.6.2 Maximum a posteriori Estimation . . . 43
- 3.7 Bibliographical Remarks . . . 44

4 Vector Quantization . . . 45
- 4.1 Definition . . . 46
- 4.2 Optimality . . . 47
  - *Nearest-Neighbor Condition* . . . 47
  - *Centroid Condition* . . . 48
- 4.3 Algorithms for Vector Quantizer Design . . . 50

Lloyd's Algorithm . . . 50
LBG Algorithm . . . 52
k-Means-Algorithm . . . 53
4.4 Estimation of Mixture Density Models . . . 55
EM algorithm . . . 56
4.5 Bibliographical Remarks . . . 59

5 Hidden Markov Models . . . 61
5.1 Definition . . . 61
5.2 Modeling Emissions . . . 63
5.3 Use Cases . . . 65
5.4 Notation . . . 67
5.5 Evaluation . . . 68
5.5.1 The Production Probability . . . 68
Forward Algorithm . . . 69
5.5.2 The "Optimal" Production Probability . . . 71
5.6 Decoding . . . 74
Viterbi Algorithm . . . 75
5.7 Parameter Estimation . . . 76
5.7.1 Foundations . . . 76
Forward-Backward Algorithm . . . 78
5.7.2 Training Methods . . . 79
Baum-Welch Algorithm . . . 80
Viterbi training . . . 86
Segmental k-Means . . . 88
5.7.3 Multiple Observation Sequences . . . 90
5.8 Model Variants . . . 91
5.8.1 Alternative Algorithms . . . 91
5.8.2 Alternative Model Architectures . . . 92
5.9 Bibliographical Remarks . . . 92

6 n-Gram Models . . . 95
6.1 Definition . . . 95
6.2 Use Cases . . . 96
6.3 Notation . . . 97
6.4 Evaluation . . . 98
6.5 Parameter Estimation . . . 100
6.5.1 Redistribution of Probability Mass . . . 101
Discounting . . . 101
6.5.2 Incorporation of More General Distributions . . . 104
Interpolation . . . 104
Backing Off . . . 106
6.5.3 Optimization of Generalized Distributions . . . 107
6.6 Model Variants . . . 109
6.6.1 Category-Based Models . . . 109

6.6.2 Longer Temporal Dependencies 111
6.7 Bibliographical Remarks 112

Part II Practice

7 Computations with Probabilities 119
7.1 Logarithmic Probability Representation 120
7.2 Lower Bounds for Probabilities 122
7.3 Codebook Evaluation for Semi-Continuous HMMs 123
7.4 Probability Ratios 124

8 Configuration of Hidden Markov Models 127
8.1 Model Topologies 127
8.2 Modularization 129
8.2.1 Context-Independent Sub-Word Units 129
8.2.2 Context-Dependent Sub-Word Units 130
8.3 Compound Models 131
8.4 Profile HMMs 133
8.5 Modeling Emissions 136

9 Robust Parameter Estimation 137
9.1 Feature Optimization 139
9.1.1 Decorrelation 140
Principal Component Analysis I 141
Whitening 144
9.1.2 Dimensionality Reduction 147
Principal Component Analysis II 147
Linear Discriminant Analysis 148
9.2 Tying 152
9.2.1 Model Subunits 153
9.2.2 State Tying 157
9.2.3 Tying in Mixture Models 161
9.3 Initialization of Parameters 163

10 Efficient Model Evaluation 165
10.1 Efficient Evaluation of Mixture Densities 166
10.2 Beam Search 167
10.3 Efficient Parameter Estimation 170
10.3.1 Forward-Backward Pruning 170
10.3.2 Segmental Baum-Welch Algorithm 171
10.3.3 Training of Model Hierarchies 173
10.4 Tree-like Model Organization 174
10.4.1 HMM Prefix Trees 174
10.4.2 Tree-like Representation for n-Gram Models 175

11 Model Adaptation 181
11.1 Basic Principles 181
11.2 Adaptation of Hidden Markov Models 182
Maximum-Likelihood Linear-Regression 184
11.3 Adaptation of n-Gram Models 186
11.3.1 Cache Models 187
11.3.2 Dialog-Step Dependent Models 187
11.3.3 Topic-Based Language Models 187

12 Integrated Search Methods 189
12.1 HMM Networks 192
12.2 Multi-Pass Search 193
12.3 Search Space Copies 194
12.3.1 Context-Based Search Space Copies 195
12.3.2 Time-Based Search Space Copies 196
12.3.3 Language-Model Look-Ahead 197
12.4 Time-Synchronous Parallel Model Decoding 198
12.4.1 Generation of Segment Hypotheses 199
12.4.2 Language Model-Based Search 200

Part III Systems

13 Speech Recognition 207
13.1 Recognition System of RWTH Aachen University 207
13.2 BBN Speech Recognizer BYBLOS 209
13.3 ESMERALDA 210

14 Character and Handwriting Recognition 215
14.1 OCR System by BBN 215
14.2 Duisburg Online Handwriting Recognition System 217
14.3 ESMERALDA Offline Recognition System 219

15 Analysis of Biological Sequences 221
15.1 HMMER 222
15.2 SAM 223
15.3 ESMERALDA 224

References 227

Index 245

1

Introduction

The invention of the first calculating machines and the development of the first universal computers was driven by the idea to liberate people from certain every-day tasks. At that time one really thought of help in computations only and by no means of helping hands in private homes. The computing machines developed thus should take over tasks, which of course could be carried out by humans, too. However, these tasks could be performed with substantially more perseverance by an automatic system and thus more reliably and consequently also more cheaply.

The rapid progress in the development of computer technology soon allowed researchers to dream of far more ambitious goals. In the endeavor of creating so-called "artificial intelligence" (AI), one tried to outperform the capabilities of humans in certain areas. In the early days of AI, primarily the solving of mathematical or otherwise formally described problems by symbolic methods was regarded as intelligence. Therefore, for a long time the prototypical area of research was the play of chess. The victory of the chess computer Deep Blue over the world champion Kasparov in 1997 arguably meant an important public relations activity for IBM. However, in the end it proved only that playing chess is probably not such a typical achievement of intelligence, as in this discipline even the best human expert can be defeated by rather brute computing power. In the field of understanding language, which is central for human capabilities, however, all symbolic and rule based methods, which originated from the roots of AI research, could achieve moderate success only.

Meanwhile, a radical change in paradigms has been completed. Typical human intelligence now is no longer considered to be manifest on the symbolic level, but rather in the capabilities for processing various sensory input data. Among these are the communication by spoken language, the interpretation of visual input, and the interaction with the physical environment by motion, touch, and grasping. Both in the area of automatic image and speech processing as well as in the field of robotics for many years first solutions were developed from an engineering background. Since it has been proven by the successful use of statistical methods, that automatically trained systems by far outperform their "hard-wired" rule-based counterparts with respect to flexibility and capabilities realized, the concept of learning receives special attention in these areas of research. In this respect the human example still is unsurpassed. Therefore, currently one has to content oneself with reproducing the respective human capabilities in strongly constrained settings within computing machines.

Central for all automatically learning methods is the availability of example data, from which the parameters of the models to be created can be derived. Therefore, no complex symbolic sets of rules are necessary to describe the typical properties of the data considered. Rather, these are automatically extracted by learning algorithms during the repeated presentation of training samples.

Probably the most well known class of learning methods are the so-called artificial neuronal networks. Their elementary building blocks correspond to largely simplified models of human nerve cells — the neurons. As a universal function approximator this formalism is very powerful, but also too general for some applications. Therefore, other statistical formalisms could establish themselves, which are especially well adapted to certain application areas. Especially for the statistical modeling of chronologically organized data *Markov models* are applied to a major extent.

The most common application area of this technology is the automatic recognition of speech. In the beginning of the respective research for quite a long time it competed with symbolic approaches. However, the availability of large sample sets of speech data heralded the triumph of statistical methods. Therefore, meanwhile *hidden Markov models* for describing acoustic events in combination with *Markov chain models* for the statistical modeling of word sequences on the symbolic level represent the standard technology for building successful automatic speech recognition systems.

Only in recent years these methods entered a both thematically and sensorily related application area. The automatic recognition of handwritten texts — in the same way as automatic speech recognition — can be considered a segmentation problem of chronologically organized sensor data. There the time axis either runs along the text line to be processed or along the line of writing itself. By this "trick" statistical modeling techniques known from the field of automatic speech recognition usually with minor changes only can be transferred to the problem of processing handwritten documents.

A third important application area of Markov models takes us beyond the field of man-machine interaction. Bioinformatics research is primarily concerned with cell-biological processes and their simulation and analysis by means of computer science techniques. Special attention currently lies on the analysis of the human genome. From the view of statistical pattern recognition this genetic information — and cell products like, e.g., RNA or proteins derived from it — essentially consists of linearly organized symbol sequences. Though for quite some years statistical techniques are used for the analysis of such biological sequences, the attention of bioinformatics research was only recently drawn to Markov models. The success of the respective methods in this application area was so convincing, that meanwhile several software packages for the application of Markov models as well as libraries of ready-made models for different analysis tasks are available.

1.1 Thematic Context

The thematic context for the treatment of Markov models is defined by the research area of *pattern recognition* (cf. [50, 170]). As *patterns* there primarily measurements of certain sensors are regarded, e.g., images or speech signals. However, pattern recognition methods can also be applied to different input data, e.g., the symbolically represented genetic information in DNA strands.

In order to separate relevant properties of the data from interfering or irrelevant ones, the patterns considered are transformed into a *feature representation*. In general this includes several preprocessing steps, which serve the purpose of "improving" the signals for future processing, e.g. by normalizing the illumination within an image or the loudness of a spoken utterance.

After feature extraction follows the *segmentation* of the data. For images here, e.g., regions of similar color or texture are determined. Those segments are subsequently mapped onto a certain pattern class by *classification*. On this level one thus obtains for the first time a symbolic representation of the data. However, not for all pattern recognition problems the tasks of segmentation and classification can be separated so clearly. In the processing of speech, e.g., it is not possible to generate a segmentation without knowing what was actually spoken, as word boundaries are not marked acoustically. Rather, not until the actual utterance is known, inferences can be drawn on the boundaries between the units involved[1]. For the solution of such pattern recognition tasks, therefore, integrated segmentation and classification methods are required. However, with respect to complexity such methods generally by far exceed techniques that can be applied in isolation.

The flat symbolic representation of patterns that is available after the classification step, is not sufficient for many pattern recognition applications, as no structural properties are represented yet. Generating these is the goal of *pattern analysis*, which tries to compute structured interpretations of patterns on the basis of the classification results. For images this could be, e.g., a description of the observed scene, which besides a classification of individual objects also specifies their relative positions with respect to each other and their composition to more complex structures. In the field of spoken language processing the analysis of an utterance usually consists in generating an internal representation of its meaning, which can serve as the basis for a man-machine dialog or the automatic translation into a different language.

1.2 Functional Principles of Markov Models

The simplest form of Markov models are the so-called *Markov chain models*, which can be used for the statistical description of symbol and state sequences. They were

[1] In the first commercial dictation systems by the companies IBM and Dragon Systems this dilemma was solved by a methodological trick. Users had to make small pauses between words while talking. Thus by detecting the pauses utterances first could be segmented into words and these could be classified subsequently.

developed by the Russian mathematician Andrej Andrejewitsch Markov (1856 – 1922), after whom they are also named. At the beginning of the past century he first applied them for the statistical analysis of the character sequences in the text of "Eugene Onegin", a novel in verse by Alexander Sergeyevich Pushkin [151].

The functional principle of this model variant can be explained well with the example of texts. The states of the model then are identical with the words of a certain lexicon, from which the word sequences investigated are formed. Markov chain models then indicate, how probable the occurrence of a word in a certain textual context is. By evaluating this probability for a sequence of words one obtains a total probability for the text section considered. By choosing a suitable model thus, e.g., plausible — i.e. highly probable — from implausible — i.e. less probable — sentences of a certain language can be discriminated. In contrast to a formal language definition this membership decision is not deterministic but probabilistic. If, e.g, multiple models for different text categories are available, the generation probability can be used as the basis for a classification decision. In the simplest case one decides for that text category, for which the associated model achieves the highest probability on a certain text section.

In so-called *hidden Markov models* the concept of a state sequence, which is modeled statistically, is augmented with state-specific outputs of the model. It is assumed, that only these so-called emissions can be observed. The underlying state sequence, however, is hidden, a fact, from which the name of this model variant is derived. For the statistical regularities, which underlie the generation of the state sequence and the emissions, strong limitations apply. In general a hidden Markov model can be regarded as a finite-state automaton with outputs, which is augmented statistically. Both the transitions between states and the generation of outputs occur depending on certain probability distributions.

In order to be able to apply such a generative model for the analysis of data, which is already available, a mental trick is necessary. First one assumes, that the data to be analyzed were generated by a natural process, which obeys similar statistical regularities. Then one tries to reproduce this process with the capabilities of hidden Markov models as closely as possible. If this attempt is successful, on the basis of the artificial model inferences can be drawn on the real process. On the one hand this may concern the probability for generating the available data. On the other hand the inference on the internal processes within the model is at least probabilistically possible. In particular one can determine the state sequence that generated a certain sequence of outputs with highest probability.

When associating the meaning of representing classes of patterns with complete models, the formalism can be used for classification. The by far more widely used procedure, however, is to identify parts of a larger total model — i.e. states or state groups — with meaningful segments of the data to be analyzed. By uncovering the state sequence then a segmentation of the data is possible with simultaneous classification into the chosen units.

In automatic speech recognition the outputs of the models correspond to the acoustic signal or its parametric feature representation, respectively. In contrast, the model states define elementary acoustic events, e.g., speech sounds of a certain language. Sequences of states then correspond to words and complete spoken utterances. If one is able to reconstruct the expected internal state sequence for a given speech signal, the — hopefully correct — sequence of words spoken can be associated with it and the segmentation and classification problem can thus be solved in an integrated manner.

This possibility, to treat segmentation *and* classification within an integrated formalism, constitutes the predominant strength of hidden Markov models. The dilemma pointed out in the beginning, that classification requires prior segmentation but segmentation often is possible only with knowledge about the classification result, thus can be circumvented elegantly. Because of this important property methods on the basis of hidden Markov models are also referred to as being *segmentation-free*.

Both Markov chain and hidden Markov models as opposed to symbolic approaches have the important advantage, that the model parameters required can be trained automatically from sample data. However, this possibility alone does not yet guarantee the success of this modeling technique. Statistical parameter estimation methods provide reliable results only, if sufficiently many training samples are available. Powerful Markov models can thus be created only, if sample sets of considerable size are available for the parameter training. Also, merely the parameters of the models and not their configuration — i.e. the structure and the number of free parameters — can be determined automatically by the training algorithms. For this purpose even in the framework of statistical methods the experience of experts and extensive experimental evaluations are required. Furthermore, almost all known estimation methods require the availability of an initial model, which is then optimized step by step. The choice of the stating point, therefore, can critically influence the performance of the final Markov model. Finally, Markov chain and hidden Markov models offer different modeling capabilities, so that they are often used in combination. In the technical implementation, however, this requires algorithmic solutions that are substantially more complex than the simple combination of probabilities.

The successful application of Markov model-based techniques for pattern recognition tasks, therefore, requires the solution of a number of methodological problems, which go far beyond a mere technical implementation of the underlying mathematical theory.

1.3 Goal and Structure of the Book

The extensive application and simultaneously the substantial further development of pattern recognition methods on the basis of Markov models took place in the field of automatic speech recognition. There today the combination of hidden Markov models for the acoustic analysis and Markov chain models for the restriction of potential word sequences is the predominant paradigm. This also explains the fact, that the treatment of these methods in monographs is almost always coupled to the subject of automatic speech recognition (cf. [81, 95, 97, 105, 181]).

In contrast, their use in different application areas such as, for example character or handwriting recognition or the analysis of biological sequences, becomes accessible only from the respective specialized technical literature. This is surprisingly true also for the presentation of Markov chain models, which are usually referred to as statistical language models. With the exception of the monograph by Huang and colleagues [97] the necessary foundations and algorithms are almost exclusively treated in tightly focused articles that appear in conference proceedings or scientific journals. The situation is the same for questions, which appear in combination with the practical application of Markov model technology. Among these are especially the successful configuration of the models, the treatment of efficient algorithms, methods for the adaptation of the model parameters to changed task conditions, and the combination of Markov chain and hidden Markov models in integrated search processes.

Therefore, this book pursues two goals. First, Markov models will be presented with respect to their nowadays extremely wide application context. Secondly, the treatment will not be concentrating on the theoretical core of the modeling only, but include all technological aspects that are relevant from today's view.

At the beginning of the book an overview over potential application areas of Markov model technology will be given in chapter 2. There as the prototypical application area first the automatic recognition of speech will be considered, before the two further main application areas character and handwriting recognition as well as the analysis of biological sequences will be presented. The chapter closes with an outlook onto some of the many further fields of application for Markov models.

Part I of this book provides the formal framework for the treatment of Markov models. It starts with a short introduction of relevant fundamental concepts of probability theory and mathematical statistics. Furthermore, basic methods for vector quantization and the estimation of mixture density models will be presented, which are applied for the modeling of high dimensional data. Afterwards the two representatives of Markov model technology are formally described, namely hidden Markov models and Markov chain models, which are frequently also referred to as n-gram models. There the focus is rather on presenting a sound general concept of the theoretical foundations than on covering all possible variants of the respective formalisms.

The subject of the second part are important aspects of the practical application of methods on the basis of Markov models. At the beginning the robust handling of probability quantities will be covered, which are omnipresent when dealing with these statistical methods. Chapter 8 presents methods for the configuration of hidden Markov models for certain application areas. Subsequently, the robust estimation of the necessary model parameters will be explained. Chapter 10 introduces the most important methods for the efficient processing of Markov models. The adaptation of the models to different tasks is the topic of chapter 11. Part II of the book concludes with the treatment of algorithms for the search in highly complex solution spaces, which results from the joint application of Markov chain and hidden Markov models.

In part III we will come back to the applications of Markov models. Here selected systems from the main application areas automatic speech recognition, character and handwriting recognition, and the analysis of biological sequences are presented with the focus on successfully realized combinations of different methods.

2 Application Areas

2.1 Speech

The interaction by spoken language is the dominant modality of communication between humans. By means of speech emotions can be conveyed, irony can be expressed, simply “small talk” can be made, or information can be transmitted. The last of these aspects is by far the most important for the automatic processing of speech, even though also approaches for recognizing emotions in spoken utterances are pursued. By means of spoken language information can be transmitted without hardly any effort — at least for healthy humans — and with a rather high “data rate” of up to 250 words per minute. Thus with respect to ease of use and efficiency this modality principally outperforms all other means of communication used by humans, as, e.g., gesture, handwriting, or typing on a keyboard. In the literature it is, therefore, often concluded that speech were also the best solution for the communication with technical systems. This may well be doubted, however, as an open-plan office, where all employees talk to their computers, or a coffee machine, that can be controlled by spoken language only and not by simply pushing a button, might not seem to be the best ideas.

There are, however, a number of scenarios, in which man-machine communication by spoken language makes sense — if necessary including additional modalities — and can be applied successfully. There the goal is either to control a certain device or to acquire information from an automatic system. Examples for the latter are information systems, from which time-table or event information can be queried over the telephone and also the respective train, cinema, or theater tickets can be ordered, if necessary. Among the control applications are the operation of mobile phones, which make the respective connection when the appropriate name or phone number is called, the operation of machinery in an industrial context, where the use of other modalities besides speech is not possible, and also the control of so-called non-safety relevant functions in vehicles, as, e.g., the car stereo or the air condition. As a very special case of device control the automatic transcription of texts by a dictation system can be viewed. Though automatic dictation did not make it the “killer

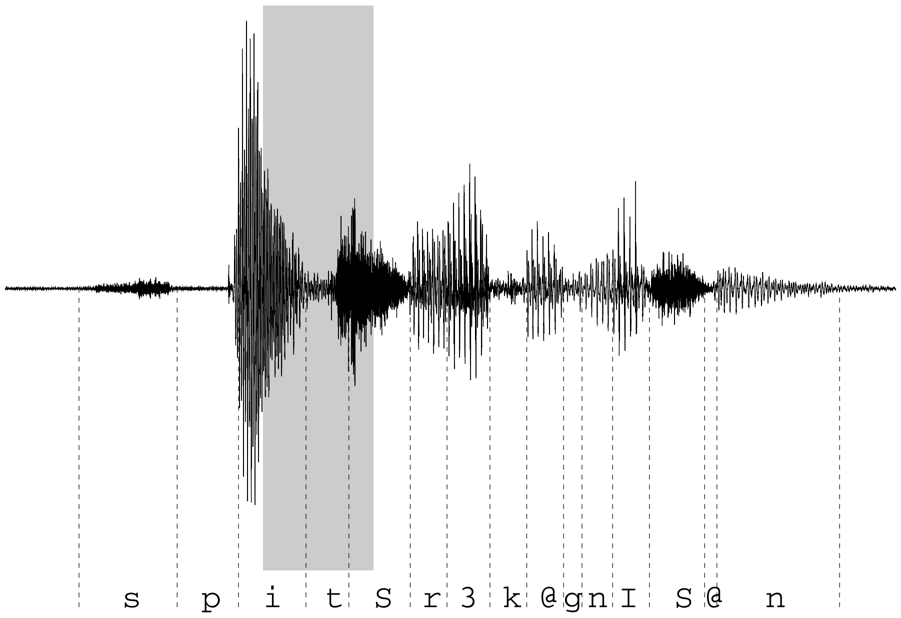


Fig. 2.1. Example of a digitized speech signal of the phrase "`speech recognition`" with manually marked phone segments.

application" of speech technology, it has had a crucial influence on the developments in the field.

In order to make spoken language man-machine communication possible, spoken utterances need to be mapped onto a suitable computer-internal symbolic representation, on the basis of which then the actions of the system are carried out. For this purpose first the physical correlate of speech — i.e. the minute changes in air pressure caused by the radiation of sound — needs to be represented digitally. By means of a microphone the sound pressure level is converted into a measurable electrical quantity, the temporal progression of which corresponds to the acoustic signal. In order to represent this signal with sufficient accuracy in digital form, it is sampled, i.e. the analog values are measured at certain regular time intervals, and subsequently quantized, i.e. the analog quantities are mapped onto a finite discrete domain of values. The combination of sampling and quantization is referred to as digitization. For speech one usually works with sampling rates of 11 to 16 kHz and stores the quantized measurements with a precision of 8 to 16 bits[1].

Figure 2.1 exemplarily shows a digitized speech signal of the utterance "`speech recognition`". For extremely simple applications of speech processing this infor-

[1] Representing speech sampled at 16 kHz with 16 bits per sample surely is not the best possible digital representation of speech signals. However, by and large it is sufficiently accurate for the automatic processing methods applied and has become standard in the field.

mation sometimes is already sufficient. Thus voice dialing in a mobile phone can be achieved by the direct comparison of the current speech signal with a small set of stored reference signals. In complex applications of spoken language processing, however, it is indispensable to first create a suitable intermediate symbolic representation, before an interpretation of the data in the context of the application is attempted.

Besides the realization as an acoustic signal for speech also the dual representation in writing exists. Though a number of characteristics of speech, as, e.g., loudness, speed, or timbre, can not be represented in written form, still the central information content can be specified orthographically. This "encoding" of the acoustic signal can also easily be represented in and manipulated by digital computers. Therefore, it is standard in more complex systems for spoken language processing, to first map speech signals onto a textual representation. This processing step is referred to as *speech recognition*. The process of *speech understanding* starts from the results of speech recognition, which, e.g., consist of a sequence of words, and tries to derive on this basis a representation of the meaning for the utterance considered. In information systems in this step the intention of the user is determined and the relevant parameters of his query are extracted. An automatic dialog system of an airline, e.g., needs to distinguish between a request for flight schedules and the actual order of a ticket. In both cases the airport of departure, the destination, and the desired travel time must be determined. For the syntactic-semantic analysis of so-called *natural language*, i.e. language input encoded in textual form, a multitude of different approaches were proposed in the literature (cf. e.g. [248]). Therefore, for the interpretation of utterances almost exclusively rule-based methods are applied, which either start directly from linguistic theories or are motivated by these.

The mapping of a speech signal onto its textual representation, as it is the goal of automatic speech recognition, however, can not be achieved with purely symbolic methods. The main reason for this is the large variability in the realization of principally identical spoken utterances by different speakers or in different environments. Furthermore, on the level of the speech signal boundaries between acoustic units are generally *not* marked at all.

The elementary unit for describing speech events it the so-called *phone*, which denotes a single speech sound. In contrast to a *phoneme*, — i.e. the smallest unit of speech used to distinguish meaning — phones define "building blocks" of spoken utterances that can be discriminated perceptually by listeners. The categories used were developed on the basis of the articulation of the respective speech units (cf. e.g. [7, 37]). The basis for this is a model of the speech production process, the principles of which will be described briefly in the following.

First a stream of air from the lungs is created — usually by exhalation —, which passes the phonation mechanism in the larynx formed by the vocal folds. If this so-called *glottis* is closed, a series of periodic impulses of air pressure is generated, while with the glottis opened the air passing by only causes something similar to white noise. This voiced or un-voiced *excitation signal* is then modified in its spectral content in the so-called *vocal tract* and a certain speech sound is formed. The vocal tract consists of the oral and nasal cavities and the pharynx. It can be modified in shape depending on the opening of the jaw and the position of the tongue and the soft palate. When putting it in simple terms, the two coarse classes of speech sounds *vowels* and *consonants* can be distinguished by the gross type of sound modification effected by the vocal tract. For vowels the excitation is always voiced and the vocal tract merely forms a resonant space. With, e.g., the largest possible opening one obtains a vowel as in the word "`hot`" (in phonetic transcription[2] `[hQt]`). In contrast, consonants result from a sort of constriction formed in the vocal tract being combined with either a voiced or un-voiced excitation signal. If, e.g., the tip of the tongue touches the back of the lower teeth, either a voiced or un-voiced S sound as in "*raise*" or "*race*" is generated, respectively (`[reIz]` vs. `[reIs]`).

Spoken language utterances always develop as a sequence of such elementary sounds. However, the units are not represented in isolation within these and, therefore, are by no means easy to segment. As the articulatory organs can not change their positions instantaneously from one sound to the next, this is achieved by continuous movements. Therefore, speech signals reflect the smooth transition between the characteristic features of subsequent sounds. In strong idealization of the real situation one may assume, that in deliberately articulated, slow speech the typical properties of a speech sound are expressed at the center of its total duration. The border regions of the respective signal segment, however, are influenced by the neighboring sounds. This mutual influencing among sounds in the speech current is referred to as coarticulation. In reality its effects can also extend across multiple neighboring sounds. In figure 2.1 the segmentation of an example signal is shown. However, as the segment boundaries can not even by experts be specified beyond any doubt, the discrimination between the individual phones is not uniquely defined in general.

The inherent continuity of speech makes a purely data-driven segmentation without any model knowledge practically impossible. Therefore, today exclusively so-called "segmentation-free" methods on the basis of hidden Markov models are applied for the purpose of automatic speech recognition. Though the digitized speech signal itself is already a linear sequence of samples, the statistical models of speech always start from a suitable feature representation. This aims at numerically describing the characteristic properties of speech units, which are mainly defined by the spectral composition of the signal. As no segmentation information is available on this level, the feature extraction needs to be performed on sections, where the properties in question vary as little as possible over time. Therefore, on the one hand these sections should be quite short. On the other hand they also need to be sufficiently long, in order to make the computation of useful spectral characteristics possible. Therefore, the speech signal is subdivided into sections of constant length of approximately 16 to 25 ms, which are called *frames*. In order to avoid loosing important information at the boundaries created by this elementary segmentation of the signal, the frames usually overlap. A *frame rate* of 10 ms has virtually become a standard

[2] If phonetic transcriptions of spoken utterances are given in this book, these use the symbol inventory defined by SAMPA, which was developed as a machine-readable version of the International Phonetic Alphabet (IPA) especially for the automated processing in digital computers [233].

in the field. With a frame length of 20 ms the signal sections would overlap by 50 percent. Figure 2.2 shows the subdivision of a speech signal into frames of 16 ms length for part of the example signal known from figure 2.1.

For every frame features are calculated. Thus one obtains a sequence of high-dimensional, continuous feature vectors, which are identified with the outputs of a hidden Markov model. All feature extraction methods have in common, that they use a measure for the signal energy and generate an abstract representation of the spectral composition of the respective section of the signal. Originally developed for the analysis of seismic data the so-called *cepstral*[3] analysis has become the standard feature extraction method in the field of automatic speech recognition ([20], cf. also [97, pp. 306–318]). The so-called model spectrum implicitly characterizes the shape of the vocal tract during the formation of speech and thus allows to draw inferences on the speech sound articulated[4]. The combination of subdividing a speech signal into frames and carrying out a local feature extraction is referred to as *short-time analysis*. In figure 2.2 results of such a procedure are exemplarily shown. There the computation of a model spectrum created by cepstral smoothing was used as the hypothetical feature extraction method.

By the training of hidden Markov models for acoustic units one tries to reproduce the statistical properties of the feature vector sequences that were generated by the short-term analysis. Usually, for this purpose a modular approach is applied for the description of complex structures of speech. On the basis of models for elementary units, as, e.g., phones, models for words are constructed by concatenation. An arbitrary sequence of models for words from a given lexicon then defines a model for spoken utterances from a certain application domain. The overall model is again a hidden Markov model. The segmentation of an utterance given by a sequence of feature vectors into the respective word sequence can be obtained by computing the optimal state sequence through the model. This passes through certain word models, which by construction are part of the utterance model, and the corresponding optimal textual representation can be given easily. However, in general this solution will only be an approximation of what was really said.

In order to avoid the consideration of arbitrary word sequences during the search for the solution, which might be quite implausible in the context of the respective application, in addition to the modeling on acoustic level statistical restrictions on a symbolic level can be introduced by means of Markov chain models. One then says to be using a so-called *language model*. Principally for this purpose also purely symbolic methods, as, e.g., formal grammars, can be used. However, the combination of two statistical techniques in general leads to more powerful integrated systems. Therefore, the combination of hidden Markov models for the acoustic modeling and Markov chain models for the language modeling has become the standard procedure within the field of automatic speech recognition.

[3] Terms as *cepstrum*, *saphe*, and also *alanysis* were artificially derived by the authors from their "equivalent" terms in the frequency domain, i.e. *spectrum*, *phase*, and *analysis*.

[4] A detailed explanation of different methods for feature extraction is, e.g., given in [97, Chap. 6 pp. 275–336]

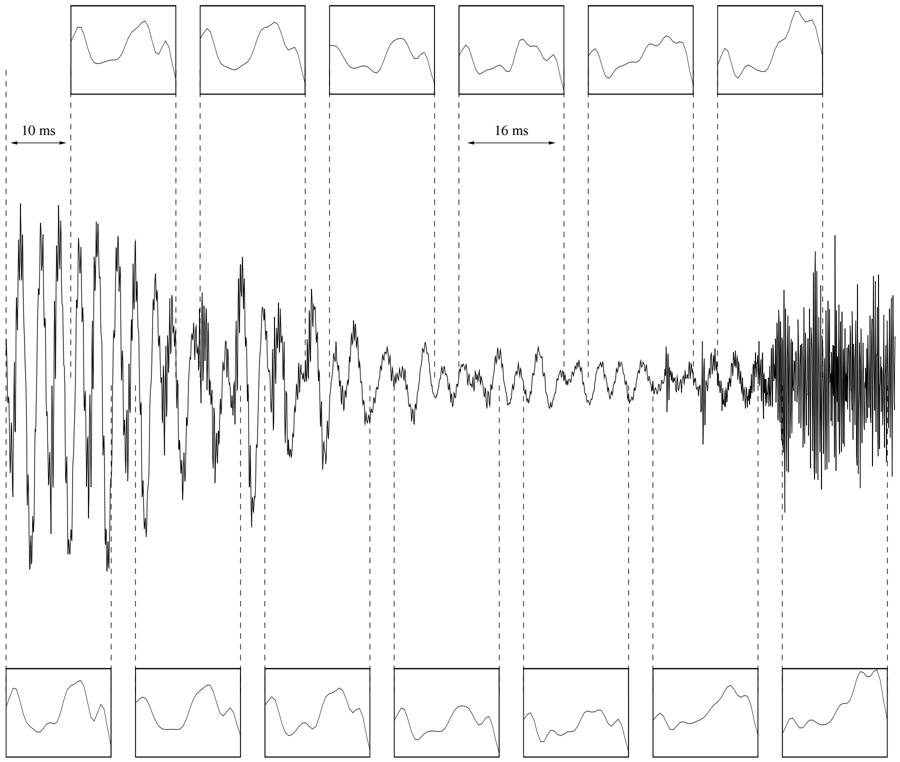


Fig. 2.2. Frame segmentation for the short example section marked in the speech signal known from figure 2.1, which represents the transition from the vowel `[i]` through the plosive `[t]` to the fricative `[S]`. As hypothetical feature representation the model spectrum is shown, which was created by cepstral smoothing.

The difficulty of an actual speech recognition task can be estimated from the restrictions, which apply for the spoken utterances to be expected. The more constrained, the simpler and the more diverse, the more difficult the necessary modeling will be. The problem is considerably simplified, if the actual speech data originates from only a single speaker. This is then said to be a *speaker dependent* recognition task. In contrast, systems are referred to as *speaker independent*, which are approximately capable of processing utterances from a wide range of different persons. The recognition problem can also be simplified by limiting the vocabulary considered. If only a simple set of command words are to be recognized, this can be achieved comparably easily and robustly. The decoding of a huge vocabulary of several 10 000 words, however, requires a dramatically increased effort. Therefore, off-the-shelf dictation systems work in speaker dependent mode, in order to achieve an acceptable recognition accuracy even for very large vocabularies. Information systems accessible over the telephone, in contrast, need to be speaker independent and generally use vocabularies, which are constrained to the actual task.

But not only the size of the lexicon influences the recognition performance achievable. Usually for large vocabularies one uses statistical models for the restriction of the potential or probable word sequences. Depending on how well these constraints can be brought to bear during model decoding, the recognition problem is simplified. This can be achieved especially well for utterances from clearly defined application areas, in which possibly even formalized language structures might be. Therefore, the first dictation systems that were commercially available were aimed at offices of attorneys and medical doctors.

In addition to the size of the space of potential solutions also the speaking style critically influences the quality of the recognition results. In experiments in the laboratory one often works with speech signals that were read from given text prompts and which, therefore, exhibit considerably less variability in their acoustic realization than can be observed in spontaneous speech. There in general the care taken in the articulation decreases, and coarticulation effects between neighboring speech sounds increase. In certain contexts individual phones or complete phone groups are potentially not realized at all. Furthermore, spontaneous speech effects as, e.g., hesitations or false starts may occur, which, of course, need to be taken into account when building the models required.

A further severe difficulty for speech recognition systems are changes in the environmental conditions, in which the signal data is captured. On the one hand these might effect the recording channel itself, which is defined by the technical solutions used for signal recording and the acoustic properties of the recording environment. Therefore, in off-the-shelf dictation systems often the special microphone required is sold together with the software package. Most systems also work in a quiet office environment only and not in a large exhibition hall. In such public spaces additionally interfering noises appear, which adversely affect the system performance in two respects. On the one hand they overlay the actual speech signal and thus influence the feature representations extracted. On the other hand also the speaker himself perceives the interfering noises and in general modifies his articulation in consequence. This phenomenon according to its discoverer is referred to as the *Lombard effect* [146]. Therefore, the robust recognition of utterances spoken in a vehicle, where severe noises occur, which are partly not predictable, is a considerable challenge for current speech recognition systems.

Over decades of speech recognition research by further and further refinements in the statistical methods as well as by special application specific techniques respectable achievements were made in counteracting the manifold difficulties of automatically recognizing speech input. Nevertheless, even today no system exists, that is able to recognize arbitrary utterances of an arbitrary person on an arbitrary subject in an arbitrary environment — but these requirements are not even met by the human example. But also for less ambitious goals as the building of speaker-dependent dictation systems the actual system performances often fall short of the promises made by the manufacturers [75]. The problem of automatic speech recognition, therefore, is not solved at all, and also in the future considerable research efforts and potentially radically new methods will be required for creating systems, that at least approximately achieve the capabilities of a human listener.

2.2 Writing

When thinking about writing, in the Western world first character alphabets come to mind and especially the widely used Roman alphabet, which was also used for printing this book. Alphabetic writing systems in general follow the phonographic principle, where the phonetic structure of the respective language is represented by the comparably few symbols of the alphabet[5]. Even though a Western reader, due to his cultural background, might consider this procedure to be the only one making sense, today still completely different approaches for writing down spoken language exist[6]. The most prominent example is the Chinese writing system, which mostly works according to the logograhic principle. There each of the complex symbols represents a certain word of the Chinese language. Therefore, at least a few thousand symbols are required, in order to write texts in everyday Mandarin, e.g. in newspapers. That such a form of writing, which in the eyes of Western people appears overly complex, even in the computer age is not replaced by an alphabetic writing system, is impressively demonstrated by the example of the Japanese writing, which as the writing system of a high-technology nation naturally is used also in the interaction with computers. Japanese texts are written in a mixture of Chinese symbols (so-called *kanji*) for representing word stems and two syllabic writing systems that were derived from it by simplification and which again follow the phonographic principle. *Hiragana* symbols are used for writing grammatical elements. Names and foreign words are written with the symbols of the *katagana* writing system.

Independently from the actual writing system written texts can either be produced by machine, i.e. printed, or by handwriting on paper by using a pen or brush. In machine print the form of the individual symbols or characters in principle is not constrained. In contrast, in handwriting one also aims at bringing these to paper as easily and as fluently as possible. Therefore, besides the symbol set used for machine printing texts from a certain writing system also a cursive version adapted for handwriting exists.

In contrast to the recognition of spoken language the automatic processing of writing is only partly carried out in the context of man-machine interaction. Rather, the origins of this technology to a major extent lie in industrially relevant applications in the field of automation technology. As this book does not aim at treating the automatic processing of writing thoroughly for all existing writing systems, in the following we will limit the considerations to the widely used alphabetic writing systems and there as a typical representative to the Roman alphabet. For a presentation of techniques, which are used for processing, e.g., Japanese, Chinese, or Arabic texts, the interested reader is referred to the respective specialized technical literature (cf. e.g. [30, Chap. 10 to 15]). With the exception of highly logographic writing systems, as, e.g., Chinese, the same principal approaches are applied. One only takes into ac-

[5] Due to the historic development of languages the relationship to the respective pronunciation is more or less obviously preserved in their current writing.

[6] A thorough overview over ancient writing systems and those still in use today together with their historic development and relationships amongst each other is given in [40].

count the respective special appearance of the image of the writing in the selection of methods and the combination of processing steps.

The classical application of automatic processing of writing is the so-called *optical character recognition* (OCR, cf. e.g. [160]). There the goal is to automatically "read" machine printed texts, which were captured optically and digitized afterwards, i.e. one aims at transforming the image of the writing into a computer-internal symbolic representation of the text. Thus the underlying data is images of document pages, as, e.g., the one at hand, which by means of a scanner are converted into a digital image with a typical resolution of 300 to 2400 dots per inch. Therefore, due to the very nature of the input data itself, methods for automatic image processing are predominant in the field of OCR.

Before the beginning of the actual text recognition in any case an analysis of the document layout is necessary. Here within the available page image text areas but also other elements of the document structure, as, e.g., headlines or graphics, are identified. Afterwards, the text areas can be segmented into paragraphs, individual lines, and in general also single characters due to the usually high precision in the production of machine-printed texts. As soon as the images of the written symbols are isolated, they can be mapped onto a symbolic representation by arbitrary techniques from pattern classification (cf. e.g. [30, 211]). The results of the classification are generally subject to one or more post-processing steps. There it is tried, to correct errors on the character level as far as possible by incorporating context restrictions, e.g. in the form of a lexicon (cf. e.g. [44]).

The complexity of the processing task, as in the field of automatic speech recognition, is defined by the variability of the input data to be expected. As the processing mainly is performed on the character level and the word and document context is only taken into account during post-processing, the size of the lexicon used is of subordinate importance. The variability of the data on the one hand results from differences in the respective image of the writing, as it is created by the printing process. On the other hand distortions in the optical capturing of the documents may severely affect the appearance of individual characters and text sections. In the printing process itself the type face (e.g. Times, Helvetica or `Courier`), the font family (e.g. regular, **bold**, or *italic*), and the character size may vary. Considerably increased difficulties for the automatic processing result, if the image of the writing could only be captured with poor quality. This can be due to, e.g., aging or a comparable degradation as a contamination of the original document. But also the repeated reproduction or transmission of a document, e.g. by fax or by means of copying machines, reduces the quality of the source data for automatic character recognition. In such cases a segmentation on character level is generally no longer possible with sufficient reliability. Therefore, segmentation-free methods on the basis of Markov models exhibit substantial advantages in the processing of such "degraded" documents as opposed to classical OCR methods (cf. e.g. [60]).

The problem of automatic character recognition also becomes considerably more challenging, if the texts considered were not printed by machine but written by hand. Especially the use of cursive writing, which for alphabetic writing systems in general links individual characters with connecting strokes, makes a reliable segmentation on character level practically impossible. Therefore, in this field today segmentation free methods on the basis of Markov models are predominant. However, the automatic reading of larger handwritten documents possibly due to the enormous difficulty of the task currently is not a relevant application. Even in the scientific area today only few respective research efforts exist. An example of a historic letter in cursive writing is shown in figure 2.3. In contrast, figure 2.4 shows a typical document, as it was used for building a large corpus of handwritten texts within a research initiative of the University of Bern, Switzerland (cf. [152]).

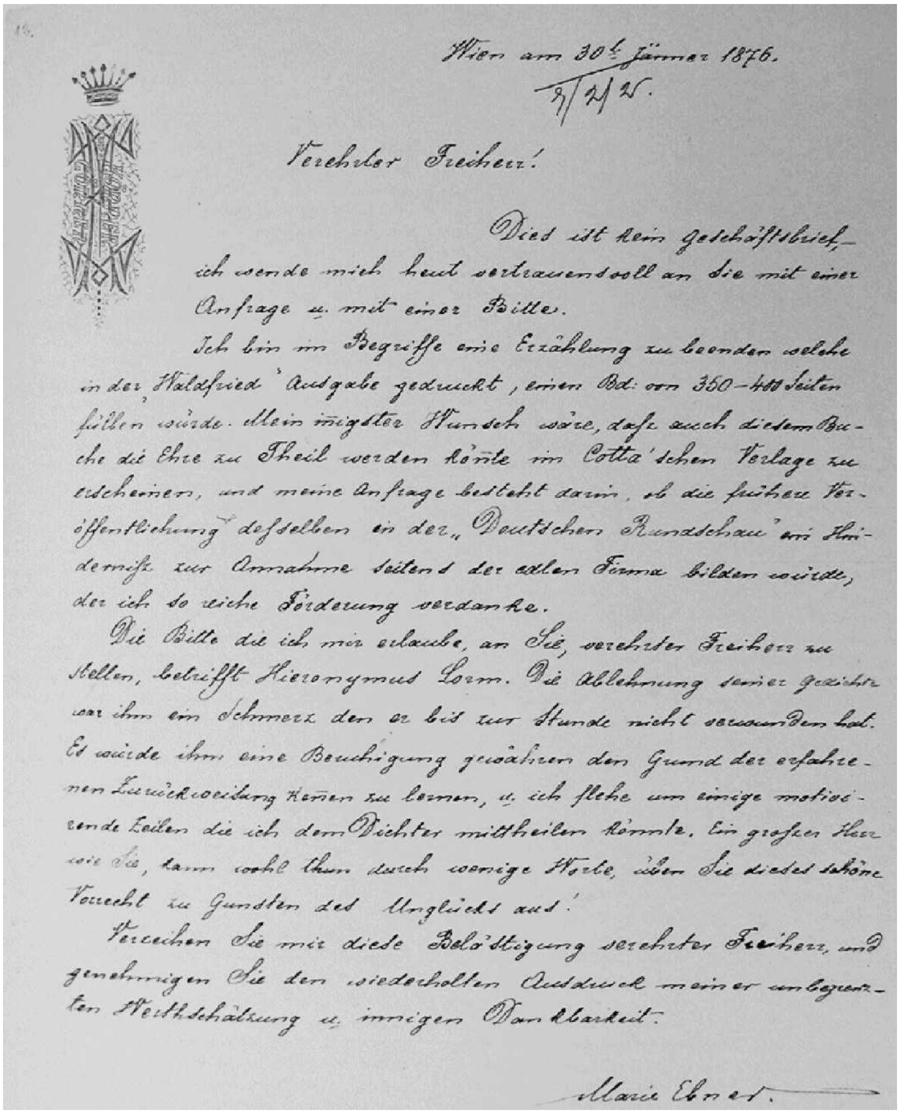
The complexity in the processing of handwritten documents results from the considerable larger variability in the image of the writing as opposed to machine-printed texts. Similarly to sounds in spoken language, the individual characters differ even when repeatedly realized by the same person, depending on their context, and the most severely between different writers. As without suitable restrictions on the potential character sequences no satisfactory results are achieved in the automatic recognition of handwritten texts, here also the vocabulary used and comparable contextual knowledge is of great importance.

An especially important application of the automatic recognition of writing is the automated reading of postal addresses in mail sorting machines. Here the essential difficulty results from the part of handwritten addresses, which is large even today. Therefore, in practice an automatic sorting of the mail pieces can be achieved only partially, as recognition errors by wrongly dispatched mail cause enormous costs. In order to be able to keep the error rate of such systems as low as possible, a quite large rejection rate has to be accepted. The rejected mail pieces then need to be sorted manually.

Powerful methods for the automatic analysis of addresses especially for machine-printed postal addresses are applied in practice for many years. There the quality of the results not only depends on the classification accuracy on character level being as high as possible. Additionally, it is extremely important to exploit relations in structure and content between the individual elements of a postal address — e.g. city name, zip-code, street name, and house number — in a clever way. After the introduction of a new automatic address reading system at the end of the last decade in the U.S. more than half of the handwritten addresses were analyzed automatically, while the estimated error rate was below 3 percent [39].

Methods for the automatic processing of forms are almost exclusively devoted to the processing of handwritten input. There the complexity of the recognition task can be considerably reduced by suitable technical measures for limiting the variability of the character images. Always special fields for writing individual words or even short phrases are used. Additionally, the writing style often is restricted to the exclusive use of hand-printed characters, which frequently also means that only capital characters may be used. The most restricted writing style is obtained, if additionally a single field for writing each character is given. Figure 2.5 shows this with the example of a form for bank transfers used in many countries sharing the Euro as currency in the European Union.

In the U.S. in contrast to Europe checks play an especially important role in financial transactions. There exists a multitude of approaches for automatically analyzing



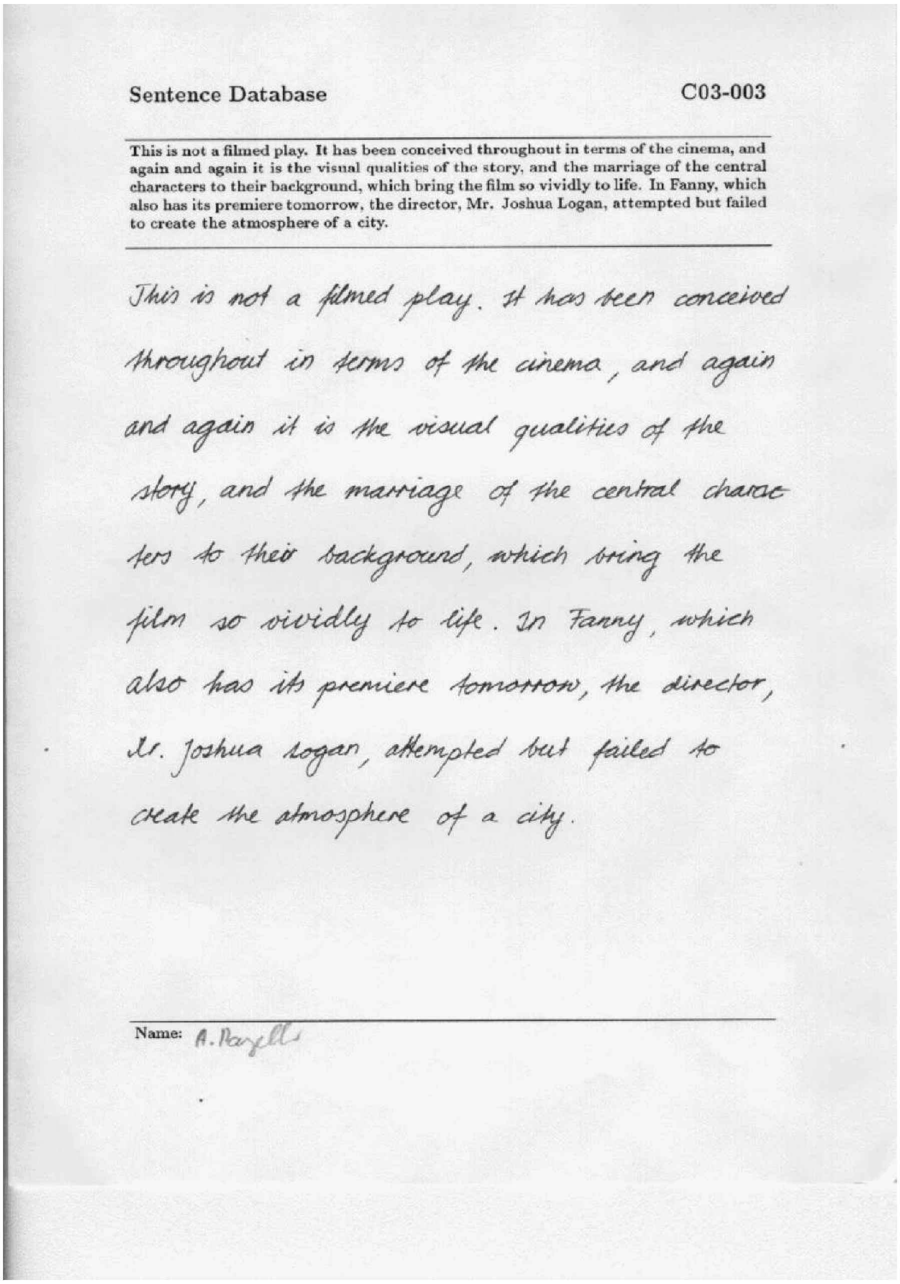
Wien am 30t. Jänner 1876.
3/2/76.

Verehrter Freiherr!

Dies ist kein Geschäftsbrief, –
ich wende mich heut vertrauensvoll an Sie mit einer
Anfrage u. mit einer Bitte.
Ich bin im Begriffe eine Erzählung zu beenden welche
in der „Waldfried" Ausgabe gedruckt, einen Bd: von 350–400 Seiten
füllen würde. Mein innigster Wunsch wäre, daſs auch diesem Bu-
che die Ehre zu Theil werden könnte im Cotta'schen Verlage zu
erscheinen, und meine Anfrage besteht darin, ob die frühere Ver-
öffentlichung desselben in der „Deutschen Rundschau" ein Hin-
derniſs zur Annahme seitens der edlen Firma bilden würde,
der ich so reiche Förderung verdanke.
Die Bitte die ich mir erlaube, an Sie, verehrter Freiherr zu
stellen, betrifft Hieronymus Lorm. Die Ablehnung seiner Gedichte
war ihm ein Schmerz den er bis zur Stunde nicht verwunden hat.
Es würde ihm eine Beruhigung gewähren den Grund der erfahre-
nen Zurückweisung kennen zu lernen, u. ich flehe um einige motivi-
rende Zeilen die ich dem Dichter mittheilen könnte. Ein groſser Herr
wie Sie, kann wohl thun durch wenige Worte, üben Sie dieses schöne
Vorrecht zu Gunsten des Unglücks aus!
Verzeihen Sie mir diese Belästigung verehrter Freiherr, und
genehmigen Sie den wiederholten Ausdruck meiner unbegrenz-
ten Werthschätzung u. innigen Dankbarkeit.

Marie Ebner.

Fig. 2.3. Example of a digitized handwritten document: Letter of Marie von Ebner-Eschenbach to Herrman von Reischach, dated Vienna, January 30, 1876 (*Brief der Marie von Ebner-Eschenbach an Herrman von Reischach, Wien, 30. Jänner 1876* [155, p. 111], reprinted with permission of Philipp Reclam Jun. Verlag, Stuttgart, Germany).



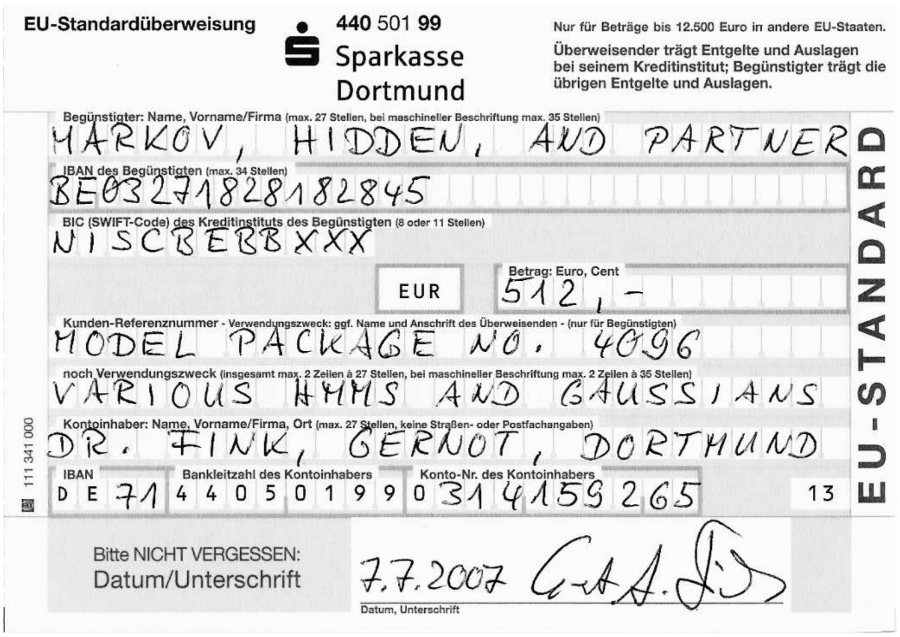
Sentence Database C03-003

This is not a filmed play. It has been conceived throughout in terms of the cinema, and again and again it is the visual qualities of the story, and the marriage of the central characters to their background, which bring the film so vividly to life. In Fanny, which also has its premiere tomorrow, the director, Mr. Joshua Logan, attempted but failed to create the atmosphere of a city.

This is not a filmed play. It has been conceived
throughout in terms of the cinema, and again
and again it is the visual qualities of the
story, and the marriage of the central charac-
ters to their background, which bring the
film so vividly to life. In Fanny, which
also has its premiere tomorrow, the director,
Mr. Joshua Logan, attempted but failed to
create the atmosphere of a city.

Name: [illegible]

Fig. 2.4. Example page from the corpus of handwritten documents created by the Institute for Informatics and Applied Mathematics, University of Bern, Switzerland [152]. In the upper part the text section to be written together with an identification code is printed. Below the respective handwritten version was filled in by one of the subjects (reprinted with permission).



EU-Standardüberweisung • 440 501 99 Sparkasse Dortmund

Nur für Beträge bis 12.500 Euro in andere EU-Staaten. Überweisender trägt Entgelte und Auslagen bei seinem Kreditinstitut; Begünstigter trägt die übrigen Entgelte und Auslagen.

Begünstigter: Name, Vorname/Firma (max. 27 Stellen, bei maschineller Beschriftung max. 35 Stellen)
MARKOV, HIDDEN, AND PARTNER

IBAN des Begünstigten (max. 34 Stellen)
BE03271828182845

BIC (SWIFT-Code) des Kreditinstituts des Begünstigten (8 oder 11 Stellen)
NISCBEBBXXX

EUR Betrag: Euro, Cent
512,-

Kunden-Referenznummer - Verwendungszweck: ggf. Name und Anschrift des Überweisenden - (nur für Begünstigten)
MODEL PACKAGE NO. 4096

noch Verwendungszweck (insgesamt max. 2 Zeilen à 27 Stellen, bei maschineller Beschriftung max. 2 Zeilen à 35 Stellen)
VARIOUS HMMS AND GAUSSIANS

Kontoinhaber: Name, Vorname/Firma, Ort (max. 27 Stellen, keine Straßen- oder Postfachangaben)
DR. FINK, GERNOT, DORTMUND

IBAN	Bankleitzahl des Kontoinhabers	Konto-Nr. des Kontoinhabers	
DE71	44050199	0314159265	13

EU-STANDARD

111 341 000

Bitte NICHT VERGESSEN: Datum/Unterschrift
7.7.2007 [illegible]
Datum, Unterschrift

Fig. 2.5. Example of a form for bank transfers in the European Union filled out manually. Handwritten capital characters need to be written into the appropriate fields given.

the handwritten legal amount, its numeric equivalent, the so-called courtesy amount, and the date of issue (cf. e.g. [133]). In this recognition task the lexicon is restricted to only a few entries as numerals and digits. However, from these basic units longer sequences may be constructed without appreciable sequencing restrictions. The writing style used also satisfies no constraints. Thus the use of hand-printed characters, cursive writing, and an arbitrary mixture of those styles is possible, and one arrives at so-called unconstrained handwriting.

All methods for processing writing presented so far have in common, that documents are captured digitally *after* they were completed and are then processed further independently from their generation process, which is referred to as *offline* handwriting or character recognition. In contrast to that, for the processing of handwritten documents also so-called *online* methods exist, where the motion of the pen is already captured during the process of writing. For this purpose special sensors, as, e.g., pressure sensitive tablets or LC displays are required and in general also the use of specialized tools for writing. As the digital representation of the written texts one obtains in essence a sequence of two-dimensional measurements of the pen position, which encode the trajectory of the pen during the writing process. In simple devices, as they are, e.g., used in so-called personal digital assistants (PDAs), position measurements are only obtained, if the pen touches the writing surface, resulting in so-called *pen-down* strokes. However, if the pen is lifted, only more sophisticated graphics tablets can still track the pen in so-called *pen-up* movements in the close vicinity

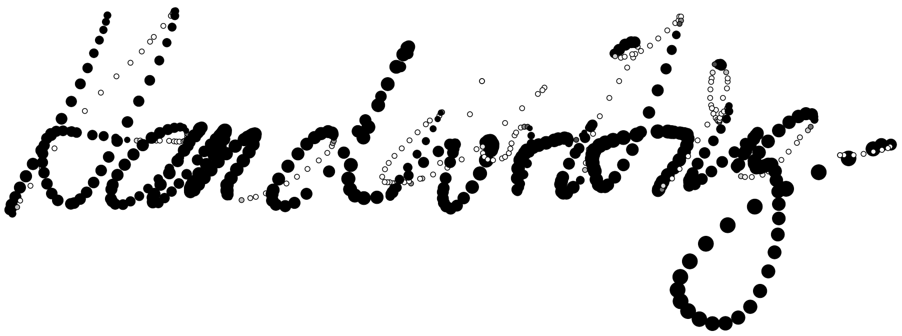


Fig. 2.6. Example of a pen trajectory of the handwritten word "*handwriting*" captured online. The size of the dots representing the pen positions encodes the pen pressure in pen-down strokes (black) and the shading represents the distance to the writing surface during pen-up movements.

of the writing surface by using inductive techniques and specialized pens. Such devices usually also provide the pen pressure and the inclination of the pen with respect to the writing surface in addition to the position measurements. Figure 2.6 shows a potential pen trajectory by the example of the word "`handwriting`". There the pen pressure and the distance of the pen tip to the writing surface are encoded in the size and shading, respectively, of the dots used to represent the pen positions.

Compared to the offline processing of handwriting, online methods have the advantage, that they can take into account during recognition the temporal organization of the writing process, which is additionally available. Thus it is, e.g., not possible that neighboring characters overlap in the image of the writing and, therefore, in the automatic segmentation can hardly be separated or not at all. The dynamic information is also essential for the verification of signatures. It represents a highly writer dependent peculiarity of the signature, which can not be forged even by experts on the basis of an available image of the signature.

However, the main application area for online handwriting recognition is man-machine interaction. Especially for operating extremely small, portable computing devices, which would not be reasonably possible with a keyboard, this form of text entry has become standard for device control. Usually the problem is simplified as much as possible, in order to achieve satisfactory results with the limited resources of a PDA, organizer, or smart phone while at the same time reaching sufficiently high reaction times. In the well known PalmPilot and its descendants only isolated characters are captured in special input fields. Additionally, a special writing style optimized for the purposes of automatic recognition needs to be used. On more powerful devices usually also the input of complete handwritten words is possible.

Inspired by the success of Markov model-based methods in the field of automatic speech recognition in recent years the principal approach of this technique was also transferred to problems of automatic character and handwriting recognition. As segmentation-free methods these techniques are mostly applied where the “classical” OCR approaches, that first segment on the level of characters and later classify, are either not reliably enough or fail completely. Therefore, Markov models are mainly used for the segmentation of handwritten documents in both online and offline mode and only rarely for the processing of machine-printed texts. Similar to automatic speech recognition hidden Markov models are applied for modeling the appearance in writing of individual characters or whole words, and Markov chain models are used for restricting potential sequences of elementary units on character or word level.

The fundamental prerequisite for the applicability of these methods is, that the signal data considered can be represented as a linear sequence. This is easily possible in the field of online handwriting recognition. By the temporal progress of the writing process itself a chronological order of the position measurements is given, which are provided by the respective sensors. The time axis of the signal thus virtually runs along the trajectory of the pen. As in the short-time analysis of speech signals, then local characteristic properties can be described by feature vectors. For this purpose mainly shape properties of the pen trajectory, as writing direction or curvature, are evaluated. In contrast, the writing speed is usually normalized in pure recognition systems, in order to avoid variations of the signal characteristics due to different writers.

To define a comparable procedure for the linearization of offline documents is considerably more difficult, as those are principally two-dimensional images. However, in general a segmentation of the document considered into individual text lines can be generated with sufficient reliability. When analyzing forms, which usually comprise only isolated words or phrases, the segmentation of the field contents is possible even more easily. Then a hypothetical time axis can be defined in parallel to the direction of the text. Along this axis it is then tried to describe properties of the image of the writing by feature vectors. For this purpose the text line usually is subdivided into a sequence of narrow overlapping image windows, which then in principle correspond to the frames known from automatic speech recognition. For each of these windows a feature representation is generated. However, in the field of offline handwriting or character recognition for this no generally accepted method exists. In many cases local structural properties are computed, as, e.g., the number of line ends or arcs, which lie within a certain frame of writing. The sequence of the feature vectors generated thus is then — similarly to automatic speech recognition — identified with the outputs of hidden Markov models for characters or words.

2.3 Biological Sequences

The genetic information of all living organisms, which influences their growth, their metabolism, and to a major extent also their behavior, is encoded in a symbolic sequence. In the majority of cases this is the macro molecule *deoxyribonucleic acid* (DNA), which consists of two strands intertwined in the form of a double helix. The strands are built as sequences of so-called *bases*. There exist four different types of bases (adenine, cytosine, guanine, and thymine), which are pairwise complementary and, therefore, in addition to the chemical bonds within a DNA strand also establish pair bonds to bases from the other strand. Thus one obtains the "ladder-type" structure of the double-stranded DNA. As the pairwise bonds are unique, already a single DNA strand contains the complete genetic information. Therefore, in the double strand it is encoded redundantly.

In higher developed organisms, as, e.g., mammals, the DNA is not available as a single complete sequence, but distributed across so-called *chromosomes*. Human cells contain 23 pairs of those, which represent the genetic information from maternal and paternal side, respectively. The entirety of the DNA strands in all chromosomes is referred to as the *genome*. Every cell of a living being contains an identical copy of this total genetic information. The size of the genome is coarsely connected to the complexity of the respective organism. While the genetic information of bacteria contains only a few million bases, the human genome comprises approximately 3 billion base pairs.

However, the majority of the DNA sequence has no cell-biological function or none that has been understood so far. In this additional "junk" material the elementary units of genetic information — the so-called *genes* — are embedded. Their relevant information, which codes the function of a gene — the so-called coding region — is generally split up into multiple *exons*, which are interrupted by *introns*. A few years ago it was still assumed, that the human genome contains approximately 30 000 to 40 000 genes [101, 225], while more recent estimates propose a total number of only approximately 25 000 coding regions [102].

For controlling most cell-biological functions genes are "transformed" into *proteins* in a process called *expression*. The proteins created then influence the metabolism and the growth of the cell and control its reproduction during cell division.

In order to express a certain gene, first the genetic information available on the double-stranded DNA is read and transformed into the equivalent representation of the single-stranded *ribonucleic acid* (RNA), which also represents a sequence of bases. This process, which is referred to as *transcription*, begins in a so-called *promotor* region before the actual DNA sequence, which contains the information of a specific gene. In the resulting raw version of the RNA the coding region of a gene in general is still interrupted by introns without known function. Therefore, the RNA is "cleaned" in a subsequent modification process, where the introns are discarded. The cleaned RNA is called *messenger* RNA or for short mRNA.

Finally, from mRNA in an additional transformation process, which is referred to as *translation*, a certain protein is generated, which realizes the functionality of the underlying gene. In contrast to DNA and RNA proteins consist of a sequence of

Amino Acid Sequence

```
MALSAEDRALVRALWKKLGSNVGVYTTEALERTFLAFPATKTYFSHLDLS
PGSSQVRAHGQKVADALSLAVERLDDLPHALSALSHLHACQLRVDPASFQ
LLGHCLLVTLARHYPGDFSPALQASLDKFLSHVISALVSEYR
```

DNA Sequence

```
atggcgctgt ccgcggagga ccgggcgctg gtgcgcgccc
tgtggaagaa gctgggcagc aacgtcggcg tctacacgac
agaggccctg gaaaggacct tcctggcttt ccccgccacg
aagacctact tctcccacct ggacctgagc cccggctcct
cacaagtcag agcccacggc cagaaggtgg cggacgcgct
gagcctcgcc gtggagcgcc tggacgacct accccacgcg
ctgtccgcgc tgagccacct gcacgcgtgc cagctgcgag
tggacccggc cagcttccag ctcctgggcc actgcctgct
ggtaaccctc gcccggcact accccggaga cttcagcccc
gcgctgcagg cgtcgctgga caagttcctg agccacgtta
tctcggcgct ggtttccgag taccgctga
```

Fig. 2.7. Part of the amino acid sequence and the underlying DNA sequence of the protein hemoglobin according the the SWISS-PROT database [9]. Individual amino acids are encoded by capital letters and bases by lower case letters.

20 different *amino acids*. Within the mRNA sequence a triple of bases — a so-called *codon* —- encodes a certain amino acid[7]. By special start and stop codons it is controlled, what area of the mRNA is covered by the translation process. After the generation of the amino acid sequence proteins form a characteristic three-dimensional structure by folding, which makes up a substantial part of their functionality. For the example of hemoglobin figure 2.7 shows a part of the amino acid sequence of a protein as well as its representation as a sequence of codons on the level of DNA.

In contrast to sensor data, which are always affected by measurement noise, the symbolic representations of DNA sequences or proteins can in principle be given exactly. Therefore, one might assume, that symbolic and rule-based methods are completely sufficient for genome analysis. However, the sequencing of a genome poses considerable difficulties in practice (cf. [61, Chap. 5]). Therefore, even after the completion of the "*Human Genome Project*" [99] the genetic information of the human cells investigated still can not be given completely and with absolute certainty. Furthermore, genetic information is not encoded uniquely in the sequence of base pairs and is subject to a wide range of random variations within a family of organisms and also within the same species. Therefore, for complex genomes the actual number and position of the individual genes can still be estimated only, even for the extensively studied human genome. For the proteins expressed it is additionally essential for the understanding of their function, to consider the so-called expression pattern, i.e. under what conditions they are created from the respective genes, and

[7] The relationship between codons and amino acids is not unique, as with 4 bases there exist $4^3 = 64$ potential triples.

to investigate the three-dimensional structure that is formed. The latter can result in functional equivalent form from different amino acid sequences.

The variations within biological sequences pointed out above, which — according to current scientific knowledge — are largely random, have helped statistical methods to become predominant for their investigation and modeling.

Depending on which data basis the analysis of genetic material starts from, different processing steps are relevant. When starting from so-called genomic DNA, i.e. virtually raw genetic information, first the coding regions of genes need to be found and the DNA sequences present there subsequently need to be cleaned from introns in the same way as in the creation of mRNA.

If for this purpose Markov model-based methods are applied, one creates individual HMMs for promotor regions as well as exons and introns. A segmentation of the DNA sequence considered then allows to localize genes and to extract their coding region in a cleaned-up representation (cf. [90, 128]). But also on the basis of Markov chain models, which define restrictions for the occurrence of individual bases within different genetic contexts, genes can be identified within DNA sequences (cf. [174, 202]).

When starting directly from mRNA this first processing step is not necessary, as only a single gene is transcribed at a time and the final mRNA was already cleaned. However, depending on the life cycle of a cell only a limited set of genes is expressed, so that the investigation of a complete genome is virtually impossible on the basis of mRNA only.

Often only the final product of the transcription and translation process itself is considered, namely the proteins. There the goal is not a segmentation, but the finding of similar sequences. The comparison of proteins is the most simple, if they are only considered pairwise. In order to be able to capture statistical variations of the sequences, long before the application of hidden Markov models probabilities for the mapping between amino acids at certain positions of the sequence as well as for their insertion and deletion were defined. Then an amino acid sequence can be associated with another one position by position and one obtains a so-called *alignment*. The logical positions within this mapping between two proteins are mostly directly connected to the three-dimensional structure formed.

It is considerably more demanding, to apply the sequence alignment to multiple proteins of a certain family. The results of such efforts are represented in the form of so-called *multiple alignments*. From pre-existing multiple alignments the statistical properties of the respective groups of similar sequences can be derived and then be described by hidden Markov models (cf. [52, 54, 55, 129]). These so-called *profiles* can then be used to search the respective databases automatically for further similar proteins. Figure 2.8 shows a multiple alignment created by an expert for the example of different goblins.

Of course the detection of new genes by the segmentation of a genome or the extension of a family of proteins with new members by means of statistical comparisons can not show the cell biological functions of the newly found structures. In the end this needs to be proved in biological experiments. However, from the structural comparison of biological sequences and the similarities found hypotheses about the

```
Helix                 AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBCCCCCCCCCCC
DDDDDDDEE
HBA_HUMAN  ---------VLSPADKTNVKAAWGKVGA--HAGEYGAEALERMFLSFPTTKTYFPHF-DLS-----HGSA
HBB_HUMAN  --------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNP
MYG_PHYCA  ---------VLSEGEWQLVLHVWAKVEA--DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASE
GLB3_CHITP ----------LSADQISTVqASFDKVKG------DPVGILYAVFKADPSIMAKFTQFAG-KDLESIKGTA
GLB5_PETMA PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSA
LGB2_LUPLU --------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-FLK-GTSEVPQNNP
GLB1_GLYDI ---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-FSG----AS---DP

Helix      EEEEEEEEEEEEEEEEEEE            FFFFFFFFFFFF  FFGGGGGGGGGGGGGGGGGGG
HBA_HUMAN  QVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL--RVDPVNFKLLSHCLLVTLAAHLPAE
HBB_HUMAN  KVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKL--HVDPENFRLLGNVLVCVLAHHFGKE
MYG_PHYCA  DLKKHGVTVLTÄLGAILKK----K-GHHEAELKPLAQSHATKH--KIPIKYLEFISEAIIHVLHSRHPGD
GLB3_CHITP PFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRG---VTHDQLNNFRAGFVSYMKAHT--D
GLB5_PETMA DVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSF--QVDPQYFKVLAAVIADTVAAG----
LGB2_LUPLU ELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG---VADAHFPVVKEAILKTIKEVVGAK
GLB1_GLYDI GVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGNKRIKAQYFEPLGASLLSAMEHRIGGK

Helix       HHHHHHHHHHHHHHHHHHHHHHHHHH
HBA_HUMAN  FTPAVHASLDKFLASVSTVLTSKYR------
HBB_HUMAN  FTPPVQAAYQKVVAGVANALAHKYH------
MYG_PHYCA  FGADAQGAMNKALELFRKDIAAKYKELGYQG
GLB3_CHITP FA-GAEAAWGATLDTFFGMIFSKM-------
GLB5_PETMA -----DAGFEKLMSMICILLRSAY-------
LGB2_LUPLU WSEELNSAWTIAYDELAIVIKKEMNDAA---
GLB1_GLYDI MNAAAKDAWAAAYADISGALISGLQS-----
```

Fig. 2.8. Multiple alignment of the amino acid sequence of seven goblins of different organisms after [129] with the respective identifiers used in the SWISS-PROT database [9]. The line designated by `Helix` defines the mapping onto the three-dimensional structure of the proteins. Deletions of amino acids at certain positions are designated by `-`.

function of genes and proteins can be derived, which can then be verified experimentally in a considerably more goal directed manner.

Such efforts are embedded in the endeavor of biologist and bioinformatics researchers to be able to explain the function of biological organisms. Especially an exact understanding of the human metabolism is of fundamental interest to the pharmaceutical industry. Substances constructed on the genetic level and especially adapted to a certain individual — such is the hope of the researchers — might make a substantially improved treatment of diseases possible, without at the same time causing the often dramatic side effects of classical medicines. Therefore, the sequencing of more and more genetic material and its detailed analysis with respect to structure and cell biological function is especially pushed by pharmaceutical companies.

2.4 Outlook

Markov models represent a formalism, which has received substantial attention in the field of pattern recognition and beyond due the the success of the technique in the area of automatic speech recognition. Therefore, it would be a pointless endeavor trying to list all problems that were ever tackled by applying these methods. Therefore, we will focus on the most important topics besides automatic speech recognition, character and handwriting recognition, and analysis of biological sequences, for which Markov models were used to a larger extent.

Similarly to the processing of speech signals, which merely represent a sequence of measurements of the sound pressure level, hidden Markov models can be applied for the analysis of other series of measurements, as they are, e.g., obtained in material testing (cf. e.g. [206, 230]).

Comparable to online handwriting recognition is the automatic recognition of human gestures (cf. e.g. [23, 58, 162, 219, 247]). However, the trajectories of the hands and arms of a person and if necessary also the respective hand postures need to be extracted from the respective image sequences with costly image processing methods before the statistical analysis of the motion sequences. In this respect these methods are comparable in their structure to a video-based online handwriting recognition system [72, 243] developed on the basis of a method for tracking pen movements during writing in image sequences [161].

As a generalization of gesture recognition the recognition of human actions or behavior can be regarded. Thus in, e.g., [253] motion sequences of tennis players and in [25] of people walking are analyzed. As human gait is quite characteristic for individuals, can easily be observed from a distance, and is hard to conceal, hidden Markov models are increasingly applied to gait recognition in the context of surveillance applications (cf. e.g. [113, 145, 222]). In [93] human motion patterns learned are used for mimicing them by a robot and thus having the demonstrated action carried out by the machine. Extremely special human actions are changes in facial expressions, as they are, e.g., analyzed in [92, 143]

In all these methods first linear sequences of feature vectors are created from the input image sequences. The Markov model-based techniques then start from the linearized feature representations. However, in the literature also methods were proposed, which extend the formalism of hidden Markov models such, that a modeling of two- or even three-dimensional input data is directly possible (cf. e.g. [59, 110, 141, 142, 203]).

In practically all approaches mentioned so far hidden Markov models are used in isolation and not in combination with Markov chain models. This is mainly due to the fact, that for applications, as, e.g., gesture or action recognition only a rather small inventory of segmentation units is used. Therefore, probabilistic restrictions on the respective symbol sequences are not of immediate importance.

On the purely symbolic level Markov chain models, in contrast, are applied for describing state sequences without being complemented by a hidden Markov model. An important application area is the field of information retrieval, where statistical models of texts are described by Markov chain models (cf. e.g. [191]). As a compact representation of documents in principle corresponds to a compression of their content, the same principles also form the foundation of different methods for text compression (cf. [14]). Markov chain models are also applied in slightly modified form as so-called Markov decision processes for, e.g., the solution of planning tasks (cf. e.g. [240]).

Part I

Theory

Introductory Remarks

In the following chapters 3 to 6 of the first part of this book the theoretical foundations of Markov models will be presented. At the beginning a short overview is given over the most important concepts of probability theory and mathematical statistics, which are necessary for the understanding of the following explanations. Afterwards, methods for the description of data distributions in high-dimensional vector spaces will be presented within the topical framework of vector quantization. Virtually as statistical extension of classical vector quantization techniques also methods for the estimation of mixture models on the basis of normal densities are covered. In chapter 5 follows the formal presentation of hidden Markov models and the algorithms necessary for their application. There only typical members of this modeling technique will be dealt with. The most important variants of hidden Markov models will, however, be presented shortly in section 5.8, where the interested reader will also find further bibliographical references. The topic of chapter 6 are Markov chain models, which are referred to as n-gram models in their specialized realization for pattern recognition tasks. In contrast to hidden Markov models the theoretical background is less extensive here. Therefore, the emphasis of the presentation lies on algorithmic solutions, which allow a reliable estimation of the model parameters required. We will concentrate on methods, which from today's point of view can be regarded as standard techniques, even provided that especially in this field a multitude of widely differing methods were proposed. In the same way as in the presentation of hidden Markov models important alternative modeling techniques will only be described briefly at the end of the chapter.

Though a thorough understanding of the application of Markov models for pattern recognition tasks is only possible when simultaneously considering aspects relevant in practice, in the framework of the following formal treatment we will try to present the theoretical core of these methods without disturbing cross-references in a linear, consecutive way. The subsequent second part of this book is then devoted intensively to problems that result from the practical application of hidden Markov and n-gram models.

3

Foundations of Mathematical Statistics

Many events that can be observed in natural processes do not occur according to some well defined law but rather exhibit a random behavior. Therefore, for a single event no prediction about its outcome is possible. However, certain regularities can be derived even for random processes, if their behavior is observed in frequent repetitions and one abstracts from considering events in isolation. Probability theory offers the necessary mathematical models for the treatment of regularities underlying such random processes. Mathematical statistics additionally considers the problem, how the parameters of probabilistic models can be derived from observations.

In the following some important fundamental concepts of probability theory and mathematical statistics shall be introduced that are relevant for the further consideration of Markov models. The goal of this presentation is to illustrate the relevant terms. For a mathematical exact formal treatment and derivation of these concepts the interested reader is referred to the extensive volume of specialized literature.

3.1 Random Experiment, Event, and Probability

Carrying out a procedure which can be repeated arbitrarily often and will produce a random result from a well-defined set of possible outcomes is formally described as a random *experiment*[1]. This can for example be the rolling of a die or the observation of the development of the stock marked. The result of every single experiment is in all those cases within the bounds of the given possibilities uncertain — the faces one to six when rolling dice or the rising and falling of the stock marked indices. Regularities can only be identified when abstracting from isolated events and considering long-term observations only.

A random *event*[2] A corresponds to a single result or a set of potential results of a random experiment, e.g. the showing of the face six, any even-numbered face, or the

[1] If the topic of mathematical statistics is clear from the context the additional word "random" is usually omitted.

[2] As with experiments also events are not labeled "random" if their nature is clear from the context.

rising of the stock market. Such an event A occurs, if for a particular trial the result of the random experiment lies in the set A. The complete set of all possible results of a random experiment is the so-called sample space Ω. Events that correspond to exactly one result of the random experiment are called elementary events. The usual set operations conjunction, disjunction, and complement (with respect to the sample space Ω) can be applied to events.

The *relative frequency* $f(A)$ of an event A that occurred m times during an N-fold repetition of a random experiment is obtained as the quotient of its absolute frequency (or count) $c(A) = m$ and the total number of trials N:

$$f(A) = \frac{c(A)}{N} = \frac{m}{N}$$

A pragmatic derivation of the notion of probability is directly based on the relative frequency of an event. If this concept is generalized in the way, that one abstracts from the actual duration of the observation, the underlying regularity for the occurrence of the event A is defined as its *probability* $P(A)$. It is immediately evident that with an increasing number of observations the variation in the relative frequency $f(A)$ will constantly decrease and approach a uniform, constant value — the probability $P(A)$. This relationship is supported by the so-called "law of large numbers"[3].

If there is already the information available before the observation of A, that event B occurred, one obtains the *conditional probability* $P(A|B)$ of an event A. This probability for the occurrence of A under the condition of B having occurred before can be derived from the probability of A and B occurring jointly and the unconditional probability of B as follows[4]:

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

As $P(A|B)$ is determined *after* the observation of the event B this quantity is also referred to as the *posterior probability* of A. In contrast, the unconditional probability of an event, which is valid *before* the incorporation of additional restrictions, is called *prior probability*.

If the observation of B provides no information about the occurrence of A, the events A and B are said to be *statistically independent*. The conditional probability is, consequently, equal to the unconditional one and the following equality holds:

$$P(A|B) = P(A) \text{ and } P(B) = P(B|A) \quad \text{if } A, B \text{ are statistically independent}$$

The joint probability of statistically independent events can simply be determined by the product of the individual probabilities:

[3] The "law of large numbers", which was stated by Bernoulli, says: The probability, that the relative frequency of an event A will deviate more than an arbitrary but fixed threshold ϵ from its probability $P(A)$, will become arbitrarily small, if the number of observations becomes arbitrarily large, i.e. approaches infinity.

[4] Instead of the set notation for the joint probability $P(A \cap B)$ in the literature frequently the simplified notation $P(A, B)$ is used for the union or conjunctive combination of events.

$$P(A,B) = P(A)\,P(B) \quad \text{if } A, B \text{ are statistically independent}$$

An important relationship for computations involving conditional probabilities is stated by *Bayes' rule*:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3.1}$$

It allows to compute the posterior probability $P(B|A)$ of event B from the conditional probability $P(A|B)$ by taking into account model knowledge about the events A and B in the form of the associated prior probabilities. The conditional dependence in the probability expressions is thus virtually reversed.

When assuming that the sample space Ω is completely partitioned into pairwise disjoint events $B_1, B_2, \ldots, B_n$, then Bayes' rule in its general from is given by:

$$P(B_j|A) = \frac{P(A|B_j)P(B_j)}{\sum\limits_{i=1}^{n} P(A, B_i)} = \frac{P(A|B_j)P(B_j)}{\sum\limits_{i=1}^{n} P(A|B_i)P(B_i)} \tag{3.2}$$

3.2 Random Variables and Probability Distributions

A simplification of the mathematical treatment of random events can be achieved by mapping them appropriately onto the set of real numbers $\mathbb{R}$. Random experiments are then represented by so-called *random variables* that randomly take on certain values from $\mathbb{R}$. A random variable X, which takes on a finite or countably infinite number of values $x_1, x_2, \ldots, x_N$, is called *discrete*. In contrast, so-called *continuous* random variables can take on arbitrary values $x \in \mathbb{R}$.

Random variables are characterized by means of their *distribution function*[5]. The distribution function $F_X(x)$ of a random variable X specifies for every possible value of x, how large the probability is, that values taken on by X are less than or equal to x:

$$F_X(x) = P(X \le x)$$

In the discrete case $F_X(x)$ can easily be computed from the individual probabilities of the elementary events A_i, which are associated with the values x_i of the random variable. The distribution function is obtained by summing over all p_i corresponding to values less than or equal to x:

$$F_X(x) = \sum_{i:x_i \le x} P(A_i) = \sum_{i:x_i \le x} p_i$$

In the continuous case the sum in the above equation is turned into an integral and the integrated continuous function $p_X(x)$ looses its interpretation as a probability of elementary events.

[5] The distribution function is frequently also referred to as *cumulative distribution function*.

$$F_X(x) = \int_{-\infty}^{x} p_X(t)\,dt$$

The quantity $p_X(x)$ is called the *probability density* function or simply the *density* of the random variable X. In the same way as probabilities density values are always non-negative. However, they may also take on values larger than 1. Only the total area or probability mass under the density function must be equal to 1. The probability of some event is obtained as the integral of the density function over an appropriate interval of real numbers. The probability that an arbitrary continuous random variable takes on a certain value is, therefore, always equal to zero.

The concept of random variables can be generalized to vector-valued quantities. In order to simplify the treatment of the subject we will limit the following presentation to the most important continuous case. The relationships to the discrete versions are obtained in complete analogy to one-dimensional or scalar random variables.

A vector-valued n-dimensional random variable $\boldsymbol{X}$ formally constitutes a *random vector*[6], which is composed of n individual random variables X_i. The distribution function of the random vector $\boldsymbol{X}$, which is given by

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = P(X_1 \leq x_1, X_2 \leq x_2, \ldots X_n \leq x_n)$$

is obtained on the basis of a *joint* probability density function $p_{\boldsymbol{X}}(\boldsymbol{x})$ of the individual random variables X_i:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \cdots \int_{-\infty}^{x_n} p_{\boldsymbol{X}}(t_1, t_2, \ldots t_n) dt_1, dt_2, \ldots dt_n$$

The probability density function p_{X_i} of an individual random variable can be computed by integration over all remaining vector components as the *marginal distribution* of the joint density function $p_{\boldsymbol{X}}$ according to:

$$p_{X_i}(x) = \int_{(x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n) \in \mathbb{R}^{n-1}} p_{\boldsymbol{X}}(x_1, \ldots x_n) dx_1, \ldots dx_{i-1}, dx_{i+1}, \ldots dx_n$$

If the individual random variables X_i are statistically independent, the joint density function of the random vector can be obtained simply as the product of all component densities.

$$p_{\boldsymbol{X}}(\boldsymbol{x}) = p_{\boldsymbol{X}}(x_1, x_2, \ldots x_n) = \prod_{i=1}^{n} p_{X_i}(x_i)$$

In any case, however, the joint density $p_{\boldsymbol{X}}$ can in generalization of the notion of conditional probabilities be represented as the product of conditional density functions. In order to illustrate this procedure we will consider here the simple two-dimensional

[6] Random vectors are frequently also referred to as *multivariate random variables* as opposed to the scalar or *uni-variate* case.

case only. For the joint density of a random vector then the following relationships hold:

$$\begin{aligned} p_{\boldsymbol{X}}(\boldsymbol{x}) &= p_{\boldsymbol{X}}(x_1, x_2) \\ &= p_{X_1}(x_1)\, p_{X_2|X_1}(x_2|x_1) \\ &= p_{X_2}(x_2)\, p_{X_1|X_2}(x_1|x_2) \end{aligned}$$

In the general n-dimensional case, however, there exist $n!$ different possibilities for splitting up $p_{\boldsymbol{X}}$ into one-dimensional conditional density functions.

3.3 Parameters of Probability Distributions

For the coarse characterization of probability distributions or the associated random variables, respectively, the parameters expected value and variance are used, which are derived from the actual probability density function. The *expected value* $\mathcal{E}\{X\}$ of a random variable or the corresponding probability distribution denotes the value of X that is taken on in the statistical average. If the distribution is completely known, the expected value can be computed as the first moment of the probability density function:

$$\mathcal{E}\{X\} = \int\limits_{-\infty}^{\infty} x p_X(x)\, dx$$

For discrete distributions one obtains the following relationship:

$$\mathcal{E}\{X\} = \sum_{i=1}^{\infty} x_i p_i$$

When the expected value of a distribution is known, the *variance* characterizes the expected scatter of values taken on by X around $\mathcal{E}\{X\}$. It can be computed as the expected value of the squared difference between values of the random variable and its expected value, which corresponds to the second central moment of the distribution function:

$$\mathrm{Var}\{X\} = \mathcal{E}\{(X - \mathcal{E}\{X\})^2\} = \int\limits_{-\infty}^{\infty} (x - \mathcal{E}\{X\})^2 p_X(x)\, dx$$

In the discrete case one obtains the following rule in analogy to the calculation of the expected value:

$$\mathrm{Var}\{X\} = \sum_{i=1}^{\infty} (x_i - \mathcal{E}\{X\})^2 p_i$$

By means of simple algebraic transformations it can be shown that the variance of a distribution can be alternatively determined using the following relationship, which offers advantages especially in practical applications:

$$\mathrm{Var}\{X\} = \mathcal{E}\{X^2\} - (\mathcal{E}\{X\})^2 \tag{3.3}$$

For random vectors, which in the following — for simplicity — we will be treating as vector-valued random variables, moments of the distribution can be defined in the same way as for scalar quantities. The expected value of a random vector is obtained as direct generalization of the one-dimensional case from the following relationship:

$$\mathcal{E}\{\boldsymbol{X}\} = \int\limits_{\boldsymbol{x}\in\mathbb{R}^n} \boldsymbol{x}\, p_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x}$$

The variance of a multivariate distribution is described by the so-called *covariance matrix*, which is defined as follows:

$$\begin{aligned}\mathrm{Var}\{\boldsymbol{X}\} &= \mathcal{E}\{(\boldsymbol{X} - \mathcal{E}\{\boldsymbol{X}\})\,(\boldsymbol{X} - \mathcal{E}\{\boldsymbol{X}\})^T\} \\ &= \int\limits_{\boldsymbol{x}\in\mathbb{R}^n} (\boldsymbol{x} - \mathcal{E}\{\boldsymbol{X}\})\,(\boldsymbol{x} - \mathcal{E}\{\boldsymbol{X}\})^T p_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x}\end{aligned}$$

Instead of the squared difference between a scalar quantity and the expected value, here the outer product of the respective vector difference is calculated. If the components X_i of a random vector $\boldsymbol{X}$ are statistically independent, the main diagonal elements of the covariance matrix $\mathrm{Var}\{\boldsymbol{X}\}$ correspond directly to the variances $\mathrm{Var}\{X_i\}$ of the individual distributions and the remaining matrix elements vanish.

Similar to the case of uni-variate distributions the rule for calculating the covariance matrix can be rewritten as follows:

$$\mathrm{Var}\{\boldsymbol{X}\} = \mathcal{E}\{\boldsymbol{X}\,\boldsymbol{X}^T\} - \mathcal{E}\{\boldsymbol{X}\}\,\mathcal{E}\{\boldsymbol{X}\}^T \tag{3.4}$$

3.4 Normal Distributions and Mixture Models

In the discrete case the distribution of a uni- or multivariate random variable can — in principle — be defined by storing the probabilities $P(\boldsymbol{x})$ for every outcome $\boldsymbol{x}$ of the random variable in a table. For continuous distributions, however, such an approach is not feasible, as the associated densities can be very general continuous functions. Consequently, in this case it is only possible to work with parametrically defined models, which also for discrete probability distributions yield considerably more compact representations.

The most important parametric distribution in the context of Markov models — and maybe even beyond — is the *normal distribution*[7], which is defined for contin-

[7] Frequently, the normal distribution is also referred to as the *Gaussian distribution* after the German mathematician Carl Friedrich Gauss (1777 – 1855) who worked extensively on the subject. The invention of the normal distribution, however, is attributed to the French mathematician Abraham de Moivre (1667 – 1754).

uous random variables only. For a uni-variate random variable satisfying a normal distribution one obtains the following probability density function[8]:

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\dfrac{(x-\mu)^2}{2\sigma^2}} \tag{3.5}$$

The two parameters μ and σ^2 of the probability density function correspond directly to the expected value and the variance of the normal distribution. For n-dimensional random vectors one obtains the multivariate Gaussian probability density function by using a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{C}$ according to:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{C}) = \frac{1}{\sqrt{|2\pi\boldsymbol{C}|}}\, e^{-\dfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{3.6}$$

In this formula $\boldsymbol{C}^{-1}$ denotes the inverse and $|2\pi\boldsymbol{C}|$ the determinant of the covariance matrix scaled by a factor of 2π.

Many natural processes can be described by means of the normal distribution, as one can frequently assume, that after a sufficiently long time of observation they will satisfy a Gaussian distribution. However, the normal distribution has only a single mode — i.e. a global maximum — at the position of its mean and decays exponentially from there on. Therefore, it is said to be a *uni-modal* probability density function. When aiming at the parametric representation of distributions with multiple local maxima — or modes — this simple model is no longer sufficient.

It can be shown, however, that even general continuous probability density functions $p(\boldsymbol{x})$ can be approximated with arbitrary precision by a linear combination of an infinite number of normal distributions [252]:

$$p(\boldsymbol{x}) \hat{=} \sum_{i=1}^{\infty} c_i\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{C}_i) \qquad \text{with} \qquad \sum_{i=1}^{\infty} c_i = 1$$

Generally, however, with such *mixture density* models only a finite sum of K component densities is considered:

$$p(\boldsymbol{x}) \approx p(\boldsymbol{x}|\theta) = \sum_{i=1}^{K} c_i\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{C}_i) \tag{3.7}$$

The parameters of this model — namely the mixture weights c_i as well as the mean vectors $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{C}_i$ of the individual normal densities — are collected within a parameter set θ. Here, the fact that such a simplified model is easily manageable in practice by far outweighs the inevitable loss in precision caused by the coarser density representation.

[8] Prior to the introduction of the Euro as the standard currency in many European countries, this formula together with a graph showing the typical bell-shape of the probability density function was depicted on the 10-Deutschmark bank note.

3.5 Stochastic Processes and Markov Chains

Though the results, that are generated by random variables over a longer period of time vary randomly, the properties of the process generating these results — the probability distribution associated with the random variable — remain unchanged. In order to be able to treat a variation of these characteristics mathematically and to describe the behavior of statistical procedures with time-varying properties, stochastic processes are used.

A *stochastic process* is defined as a sequence of random variables $S_1, S_2, \ldots$, which take on values s_t from a discrete or continuous domain according to individual probability distributions. Depending on the type of domain one distinguishes between either continuous or discrete stochastic processes. As for the topic of this book only discrete stochastic processes are relevant, all further treatment will be limited to this simpler case. Additionally, one can also interpret the discrete values generated by such processes as discrete states being taken on. Therefore, in the context of discrete stochastic processes one frequently talks about states and state sequences generated by the processes.

The distribution function corresponding to the random variable S_t at time t can in general be dependent on the actual time t itself and the values $s_1, \ldots s_{t-1}, s_{t+1}, \ldots$, which were taken on by the remaining random variables. This very powerful general concept of a stochastic process is, however, usually restricted in a number of aspects.

A stochastic process is said to be *stationary*, if the absolute time t does not make any difference for the behavior, i.e. if the probability distribution is the same for all random variables S_t. The process is further said to be *causal*, if the distribution of the random variable S_t is only dependent on past states $s_1, s_2, \ldots s_{t-1}$. The probability distribution for a discrete, stationary, and causal stochastic process can, therefore, be written as follows:

$$P(S_t = s_t | S_1 = s_1, S_2 = s_2, \ldots S_{t-1} = s_{t-1})$$

If the correspondence between actual values s_t and the associated random variables S_t is uniquely defined by the context this property can be specified in simplified form as:

$$P(s_t | s_1, s_2, \ldots s_{t-1})$$

The causality of the process constitutes a substantial restriction, which, however, for processes evolving over time is immediately obvious. Still with time t advancing, and, therefore, increasing length of the state sequence an arbitrarily long set of dependencies for the probability distribution can be generated.

The so-called *Markov property* represents the additional restriction that the dependence of the properties of the process is limited to a *finite* history — in the case of a *simple* stochastic process even to the immediate predecessor state only. The probability distribution of a discrete, simple, causal, and simple stochastic process, which is also called a *Markov chain* of first order, can be written as:

$$P(s_t | s_1, s_2, \ldots s_{t-1}) = P(s_t | s_{t-1})$$

If the total set of events — i.e. the state space considered — is finite, the necessary conditional probabilities can be compactly combined into a matrix of *state transition probabilities*, which completely describes the property of the process:

$$\boldsymbol{A} = [a_{ij}] = [P(S_t = j | S_{t-1} = i)]$$

Higher-order Markov chains — i.e. with longer time dependencies of the distributions — offer no principal advantages over first-order models. The longer influence of the context can always be coded into a single state by an appropriate extension of the state space. However, such a structural reorganization is not always desirable or possible. Therefore, in the statistical modeling of symbol sequences also Markov chains of higher order play an important role, as we will be seeing in chapter 6.

3.6 Principles of Parameter Estimation

In order to be able to use a statistical model for the description of certain natural processes, the free parameters need to be determined in an appropriate manner. One important prerequisite for this are expectations set up by experts, which essentially determine the type of model to be used. The second important foundation are concrete observations of the process to be described. These can either be real measurements or quantities derived from them.

By taking the selected model type as a basis, estimates of the model parameters can then be computed on this set of sample data. The actual characteristics of those parameter estimates are, however, also dependent on the kind of optimization criterion that is applied by the selected estimation procedure.

3.6.1 Maximum Likelihood Estimation

The most widely used method for estimating parameters of statistical models is constituted by the so-called *maximum likelihood* method (ML). On the basis of a given set of samples $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T\}$ and depending on the actual type of the model sought, estimates $\hat{\boldsymbol{\theta}}$ of the model parameters are determined, so that the probability — or in the continuous case the probability density, respectively — for the observation of the data will be maximized.

In order to simplify the considerations in the following we will assume that the sample set corresponds to a sequence of values generated by random variables $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots \boldsymbol{X}_T$ satisfying an identical probability distribution, the parameters of which are unknown and to be estimated. The individual random variables $\boldsymbol{X}_i$ are furthermore assumed to be statistically independent. Therefore, the joint probability of the data can be obtained as the product of all contributions from the identically parametrized individual distributions:

$$p(\omega|\boldsymbol{\theta}) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{\theta})$$

When maximizing this quantity by a variation of the parameter $\boldsymbol{\theta}$ not the values of the random variables but the model parameters themselves represent the variables. This is made explicit by introducing the so-called *likelihood function*:

$$L(\boldsymbol{\theta}|\omega) = L(\boldsymbol{\theta}|\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_T) = p(\omega|\boldsymbol{\theta})$$

The goal of the maximum likelihood method is now to maximize the value of the likelihood function for a given type of stochastic model and a given sample set ω depending on the parameters $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, L(\boldsymbol{\theta}|\omega)$$

In order to solve this problem the standard approach for finding extreme values of analytical functions can be applied. Here the derivative of $L(\boldsymbol{\theta}|\omega)$ with respect to the model parameters $\boldsymbol{\theta}$ is computed and equated to zero. In most practical cases one obtains exactly one extremum, which then corresponds to the ML estimate for the model parameters sought. In general, however, the result is not uniquely defined.

Often a mathematically simpler treatment of this extreme value problem can be arrived at by not considering the likelihood function itself but is logarithm instead:

$$\hat{\boldsymbol{\theta}}_{\text{ML}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \ln L(\boldsymbol{\theta}|\omega)$$

By applying a monotonic function the extremum remains unchanged. But especially for statistical models on the basis of exponential probability densities as, e.g., normal distributions considerable simplifications are obtained.

In order to illustrate the principle of ML estimation we want to consider a simple example. On a sample set $\omega = \{x_1, x_2, \dots x_T\}$ the parameters of a one-dimensional normal distribution shall be estimated. The logarithmic likelihood function is then given by:

$$\begin{aligned}\ln L(\boldsymbol{\theta}|\omega) = \ln \prod_{t=1}^{T} \mathcal{N}(x_t|\mu, \sigma^2) &= \sum_{t=1}^{T} \ln \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\dfrac{(x_t-\mu)^2}{2\sigma^2}} \\ &= -\frac{T}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{T}(x_t-\mu)^2\end{aligned}$$

As derivatives of this function with respect to the two variables μ and σ^2 one obtains:

$$\begin{aligned}\frac{\partial}{\partial\mu}\ln L(\boldsymbol{\theta}|\omega) &= \frac{1}{\sigma^2}\sum_{t=1}^{T}(x_t-\mu) \\ \frac{\partial}{\partial\sigma^2}\ln L(\boldsymbol{\theta}|\omega) &= -\frac{T}{2\sigma^2} + \sum_{t=1}^{T}\frac{(x_t-\mu)^2}{2\sigma^4}\end{aligned}$$

By equating those formulas to zero one obtains the following values as estimates $\hat{\mu}$ for the mean and $\hat{\sigma}^2$ for the variance of the normal distribution sought as delivered by the ML method:

$$\hat{\mu}_{\text{ML}} = \frac{1}{T}\sum_{t=1}^{T} x_t$$

$$\hat{\sigma}^2_{\text{ML}} = \frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{\mu}_{\text{ML}})^2 = \frac{1}{T}\sum_{t=1}^{T} x_t^2 - \hat{\mu}^2_{\text{ML}}$$

Those two quantities are also referred to as the *sample mean* or the *sample covariance matrix* of a probability distribution defined by a set of samples only.

In the multivariate case one obtains the following relationships for calculating estimates of the mean vector $\hat{\boldsymbol{\mu}}$ and the covariance matrix $\hat{\boldsymbol{C}}$ of an n-dimensional Gaussian density:

$$\hat{\boldsymbol{\mu}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t \tag{3.8}$$

$$\hat{\boldsymbol{C}} = \frac{1}{T}\sum_{t=1}^{T}(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}})^T = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{x}_t\boldsymbol{x}_t^T - \hat{\boldsymbol{\mu}}\hat{\boldsymbol{\mu}}^T \tag{3.9}$$

If it can be safely assumed that the real distribution of the data corresponds to the pre-specified model type and the available sample set is large enough, the ML estimation has several advantageous properties. The parameter estimates $\hat{\boldsymbol{\theta}}_{\text{ML}}$ calculated for the unknown distribution parameters converge to the real parameters $\boldsymbol{\theta}^*$, if the size of the sample set approaches infinity:

$$\lim_{T\to\infty} \hat{\boldsymbol{\theta}}_{\text{ML}} = \boldsymbol{\theta}^*$$

Additionally, the ML estimates exhibit the lowest variance of all possible estimates, so that no other method can deliver an estimate that lies closer to the real parameters.

3.6.2 Maximum a posteriori Estimation

In contrast to maximum likelihood estimation the so-called *maximum a posteriori estimation* (MAP) uses the posterior probability of the model parameters as optimization criterion for a pre-specified type of model and a given sample set.

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\omega)$$

When rewriting this formula using Bayes' rule one obtains the following relationship:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{\theta}|\omega) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \frac{p(\omega|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\omega)}$$

The density function $p(\omega)$ of the sample data itself is independent of the model parameters $\boldsymbol{\theta}$ sought and can, therefore, be ignored in the maximization process. It is, however, required to specify expectations about the actual characteristics of the model parameters in the form of a suitable prior distribution $p(\boldsymbol{\theta})$.

It is for this reason that the MAP estimation offers advantages over the ML method if only a rather limited set of sample data is available. The ML estimation is then still the optimal method for deriving parameter estimates from the data alone. Due to the limited size of the sample set, however, the results obtained will in general be rather unreliable, as no prior expectations are taken into account. When applying the MAP principle those expectations are considered by making the prior distribution $p(\boldsymbol{\theta})$ of the parameters part of the optimization criterion. Putting it in simple terms, with this procedure one realizes an optimal combination of two principle strategies for parameter estimation. Model parameters, for which only few training samples are available, are mainly determined according to the expectations. Those parameters, however, which can be estimated on large amounts of data, are mainly derived from the sample data while largely ignoring the expectations.

3.7 Bibliographical Remarks

On the topic of mathematical and computational statistics a huge number of monographs exist, many of which are of introductory nature or even explicitly intended as textbooks (cf. e.g. [36, 116, 183, 193, 223]).

Among the many monographs the book by Larsen & Marx [134] stands out because of its clear presentation and the many illustrative examples. Good introductions to the fundamental concepts of statistics especially relevant for pattern recognition tasks, furthermore, can be found in the classic books by Fukunaga [80], and Duda, Hart, & Stork [50] as well as in [97, Chap. 3].

4 Vector Quantization

When processing signal data with a digital computer one is always faced with the problem that this data on the one hand needs to be stored as compactly as possible and on the other hand must be represented with sufficient accuracy. As digital representations are necessarily always also finite, it is, therefore, the goal of a so-called *vector quantizer* to map vectors from the input data space onto a finite set of typical reproduction vectors. Through this mapping ideally no information should be lost which is relevant for the further processing of the data. Therefore, one tries to reduce the effort for storage and transmission of vector-valued data by eliminating redundant information contained therein.

Such a coding is for example applied to speech signals before transmitting them over low-bandwidth digital telephone networks. With image data one tries for example to store these as compactly as possible in the memory of a digital camera or to transmit them as feature-length movies to private homes.

Though the meaning of the data is principally not of interest during coding, the most suitable representations are achieved, if "similar" vectors are grouped together when encoding the data and "dissimilar" ones are represented separately. From this point of view the goal of coding corresponds to the one of so-called *cluster analysis*, which aims at finding areas of high density of an unknown data distribution and representing those adequately. The approximate parametric representation of general probability distributions is usually performed by means of mixture densities on the basis of normal distributions (cf. section 3.4 p. 38). In this approach the individual Gaussian densities model the different regions of high density in the data considered — the so-called *clusters*. The corresponding mean vectors can then be viewed as the reproduction vectors of a statistical quantization process.

In the following sections we will first formally define the concept of a vector quantizer and derive conditions for its optimality. Subsequently, the most important algorithms for building vector quantizers will be presented. Finally, in section 4.4 the unsupervised estimation of mixture densities will be treated as a generalization of the vector quantization problem.

4.1 Definition

A vector quantizer — or for short quantizer — Q is defined as a mapping of a k-dimensional vector space $\mathbb{R}^k$ onto a finite subset $Y \subset \mathbb{R}^k$:

$$Q : \mathbb{R}^k \mapsto Y$$

The set $Y = \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots \boldsymbol{y}_N$ of reproduction or prototype vectors $\boldsymbol{y}_i$ is also referred to as the *codebook*. The size N of the codebook is the essential parameter for characterizing a set of vector quantizers[1].

With every vector quantizer Q of size N there always is associated a partition of the considered vector space $\mathbb{R}^k$ into regions or cells $R_1, R_2, \ldots R_N$. In the cell R_i lie all those vectors $\boldsymbol{x} \in \mathbb{R}^k$, which were mapped to the prototype or code word $\boldsymbol{y}_i$ by the quantizer. One obtains the respective cell as the inverse image of the prototype vector under the mapping Q of the vector quantizer:

$$R_i = Q^{-1}(\boldsymbol{y}_i) = \{\boldsymbol{x} \in \mathbb{R}^k | Q(\boldsymbol{x}) = \boldsymbol{y}_i\}$$

As a quantizer Q maps every vector $\boldsymbol{x}$ from the input space to exactly one prototype $\boldsymbol{y}_i$, it defines implicitly a complete and disjoint partition of $\mathbb{R}^k$ into cells R_i, i.e.:

$$\bigcup_{i=1}^{N} R_i = \mathbb{R}^k \qquad \text{and} \qquad R_i \cap R_j = \emptyset \quad \forall i, j \text{ with } i \neq j$$

The behavior of a quantizer Q is then uniquely defined by the specification of the codebook Y used and the associated partition $\{R_i\}$ of the vector space considered.

In practice a vector quantizer can be described as the composition of a coder C and a decoder D. When introducing an index set $I = \{1, 2, \ldots N\}$ one obtains:

$$C : \mathbb{R}^k \mapsto I \qquad \text{and} \qquad D : I \mapsto Y$$

The quantization rule Q is thus obtained as a concatenation of the coding and the subsequent decoding step:

$$Q = D \circ C$$

This representation is especially useful if the transmission of compactly coded data is considered. Then it is in principle sufficient to use a matched pair of coder and decoder and to transmit only the code word indices of the quantized data over the channel[2]. As it needs to be ensured that the decoder uses the correct codebook for the reconstruction of the data from the quantization indices, in practical applications

[1] A vector quantizer with a codebook containing N prototypes is sometimes also referred to as an *N-level quantizer*. This designation should, however, not be confused with the notion of a multi-stage vector quantizer, which for increased efficiency generates the quantization result in multiple subsequent processing steps.

[2] Vector quantizers that use a variable data rate also transmit the quantized indices themselves in compressed form, e.g., by applying a Huffman coding to them (cf. e.g. [85, Chap. 17, pp. 631–633]).

the codebook itself also needs to be communicated in the transmission of the data. Therefore, the size of the codebook has to be taken into account when considering the necessary capacity of the transmission channel and the total compression ratio to be achieved.

A substantial "drawback" of the quantization process is that a vector $\boldsymbol{x} \in \mathbb{R}^k$ will be mapped onto a prototype vector $\boldsymbol{y}$, which will in general be different from the source vector. Therefore, from every single quantization operation results an individual *quantization error* $\epsilon(\boldsymbol{x}|Q)$ depending on the quantization rule Q used. This error can formally be described by using a suitable distance measure $d(\cdot, \cdot)$ for vector-valued data[3]:

$$\epsilon(\boldsymbol{x}|Q) = d(\boldsymbol{x}, Q(\boldsymbol{x}))$$

General statements about the reproduction quality of a certain quantizer Q can, however, only be derived from a global consideration of the quantization error. Therefore, the overall quantization error, which is to be expected in the statistical average, is computed:

$$\bar{\epsilon}(Q) = \mathcal{E}\{\epsilon(X|Q)\} = \mathcal{E}\{d(X, Q(X))\} = \int\limits_{\mathbb{R}^k} d(\boldsymbol{x}, Q(\boldsymbol{x}))\, p(\boldsymbol{x})\, d\boldsymbol{x} \tag{4.1}$$

Here it is assumed that the statistical properties of the vectors $\boldsymbol{x}$ considered can be described by a random variable X, the results of which satisfy a probability density function $p(\boldsymbol{x})$.

4.2 Optimality

Though the size of the codebook constitutes the essential configuration parameter of vector quantizers in practice, conditions for the existence of optimal quantization rules are always considered for a fixed number N of prototype vectors. Then the achieved reproduction quality is the only factor determining the quality of the quantizer. Therefore, the general goal of vector quantizer design is to create that quantizer that with a given fixed codebook size achieves the minimal average quantization error for a certain distribution of input data.

Since a quantizer — as mentioned above — is defined by specifying the codebook *and* the associated partition, starting from both components criteria can be formulated, how the respective other element has to be chosen in an optimal way. A closed form description of optimality for both codebook and partition is, however, not possible.

Nearest-Neighbor Condition

The so-called *nearest-neighbor condition* describes the optimal selection of the partition $\{R_i\}$ for a given codebook Y. Here every single cell needs to be determined in

[3] When using the Euclidean distance here instead of a general distance measure one obtains the quantization error $\epsilon(\boldsymbol{x}|Q) = ||\boldsymbol{x} - Q(\boldsymbol{x})||$.

such a way that it contains all those vectors $\boldsymbol{x} \in \mathbb{R}^k$, which have minimal distance from the associated prototype vector $\boldsymbol{y}_i$:

$$R_i \subseteq \{\boldsymbol{x} | d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_j) \, \forall j \neq i\} \tag{4.2}$$

This means that the corresponding quantizer Q maps a vector $\boldsymbol{x}$ onto its nearest neighbor in the codebook:

$$Q(\boldsymbol{x}) = \boldsymbol{y}_i \quad \text{if} \quad d(\boldsymbol{x}, \boldsymbol{y}_i) \leq d(\boldsymbol{x}, \boldsymbol{y}_j) \, \forall j \neq i \tag{4.3}$$

For the case that a vector $\boldsymbol{x}$ has equal distance from two (or more) codebook vectors it can be mapped arbitrarily to any one of the candidate cells. The resulting quantization error

$$d(\boldsymbol{x}, Q(\boldsymbol{x})) = \min_{\boldsymbol{y} \in Y} d(\boldsymbol{x}, \boldsymbol{y}) \tag{4.4}$$

is not affected by the actual choice[4].

It can easily be shown that the quantization of vectors by means of the nearest-neighbor rule (4.3) minimizes the average expected quantization error for a given codebook Y. Therefore, a lower bound for the term used to compute $\bar{\epsilon}(Q)$ known from equation (4.1) is derived as follows[5]:

$$\bar{\epsilon}(Q) = \int\limits_{\mathbb{R}^k} d(\boldsymbol{x}, Q(\boldsymbol{x})) \, p(\boldsymbol{x}) \, d\boldsymbol{x} \geq \int\limits_{\mathbb{R}^k} \{\min_{\boldsymbol{y} \in Y} d(\boldsymbol{x}, \boldsymbol{y})\} \, p(\boldsymbol{x}) \, d\boldsymbol{x}$$

The comparison of this result with equation (4.4) shows that by means of the nearest-neighbor condition exactly this lower bound of the average quantization error will be reached. Therefore, the partition defined on the basis of equation (4.2) is optimal for the given codebook.

Centroid Condition

The optimal choice of a codebook Y for a given partition $\{R_i\}$ is defined by the so-called *centroid condition*. For a cell R_i of the partition the optimal reproduction vector is given by that prototype vector $\boldsymbol{y}_i$ that corresponds to the *centroid* of the cell:

$$\boldsymbol{y}_i = \mathrm{cent}(R_i)$$

The centroid of a cell R here is defined as that vector $\boldsymbol{y}^* \in R$ from that all other vectors $\boldsymbol{x} \in R$ in the statistical average have minimal distance, i.e.:

[4] For such cases it is suggested in [85, p. 350] to perform the mapping onto the codebook vector with smallest index i.

[5] Deriving the lower bound of the average quantization error in this way is possible because both factors $d(\cdot, \cdot)$ and $p(\boldsymbol{x})$ in the integral take on non-negative values. In order to minimize the integral as a whole it is sufficient to chose $d(\cdot, \cdot)$ to be locally minimal for every vector $\boldsymbol{x}$ of the input space.

$$\boldsymbol{y}^* = \text{cent}(R) \quad \text{if} \quad \mathcal{E}\{d(X, \boldsymbol{y}^*)|X \in R\} \leq \mathcal{E}\{d(X, \boldsymbol{y})|X \in R\} \, \forall \boldsymbol{y} \in R$$

The random variable X again serves to characterize the distribution of the data vectors $\boldsymbol{x}$ in the input space.

If the centroid of R is uniquely defined[6], it can be specified as follows:

$$\text{cent}(R) = \underset{\boldsymbol{y} \in R}{\text{argmin}} \, \mathcal{E}\{d(X, \boldsymbol{y})|X \in R\} \tag{4.5}$$

For the quite commonly used elliptic symmetric distance measures of the form $(\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{y})$, which contain the Euclidean distance $||\boldsymbol{x}-\boldsymbol{y}||$ as a special case if no scaling of the vector space with an inverse scatter matrix $\boldsymbol{C}^{-1}$ is performed, the centroid of a cell is identical to the conditional expected value of the data vectors limited to the region considered:

$$\text{cent}(R) = \mathcal{E}\{X|X \in R\} = \int_R \boldsymbol{x} \, p(\boldsymbol{x}|\boldsymbol{x} \in R) d\boldsymbol{x}$$

As all vectors in the cell R have minimal average distance to the centroid, the use of the centroid as prototype vector also minimizes the average quantization error for the cell in question. If all codebook vectors are determined in this way, the quantization by means of this codebook minimizes in the statistical average the error caused by the quantization process for the given partition. The codebook is, therefore, chosen optimally for the partition at hand.

This can easily be shown by calculating the average quantization error, which is achieved by applying the centroid condition:

$$\bar{\epsilon}(Q) = \sum_{i=1}^{N} \int_{R_i} d(\boldsymbol{x}, \boldsymbol{y_i}) p(\boldsymbol{x}) d\boldsymbol{x} = \sum_{i=1}^{N} P(X \in R_i) \int_{R_i} d(\boldsymbol{x}, \boldsymbol{y_i}) p(\boldsymbol{x}|\boldsymbol{x} \in R_i) d\boldsymbol{x}$$

The total error is obtained by integration over all components resulting from the respective cells and summation over all cells in the partition. This formula can be rewritten by introducing the prior probabilities $P(X \in R_i)$ of the cells and the conditional probability densities $p(\boldsymbol{x}|\boldsymbol{x} \in R_i)$ of vectors limited to a specific cell.

As all cells are disjoint and the sum runs over non-negative terms only, all N partial error terms can be minimized independently:

$$\int_{R_i} d(\boldsymbol{x}, \boldsymbol{y_i}) p(\boldsymbol{x}|\boldsymbol{x} \in R_i) d\boldsymbol{x} = \mathcal{E}\{d(X, \boldsymbol{y_i})|X \in R\} \longrightarrow \min!$$

As the comparison with the definition of the centroid in equation (4.5) shows, this minimization can exactly be achieved by choosing the centroid of every cell R_i as

[6] Depending on the distribution of the vectors within the cell considered, it may happen that the centroid in certain cases is not uniquely defined. The minimum of the mean distance is then achieved in the same way for different prototype vectors. As with the choice of the nearest neighbor the centroid can then be selected arbitrarily among the candidates without affecting optimality.

its associated prototype vector $\boldsymbol{y}_i$, because exactly this choice minimizes the average distance to the other vectors in the cell.

Consequently, for an optimal quantizer codebook and partition are immediately dependent on each other. For the parametrization of the procedure, therefore, the specification of the codebook is sufficient. The partition of the input space results implicitly from the optimal choice of the quantization operation by means of the nearest-neighbor rule. In practice it is, therefore, quite common to conceptually identify the codebook with the associated vector quantizer.

4.3 Algorithms for Vector Quantizer Design

Though in the previous sections conditions were specified for the optimality of a partition for a given codebook or of a codebook for a pre-defined partition, respectively, no analytical closed form solution exists for deriving an optimal vector quantizer of given size for some distribution of data vectors. Nevertheless, iterative methods can be defined that starting from an initial quantizer optimize it step by step and thus try to determine an approximately optimal vector quantizer. All those methods, however, can not guarantee to actually find the optimal solution of the problem. They are, therefore, inherently sub-optimal.

Another problem in the design of vector quantizers lies in the fact, that the quality of the quantizer depends on the probability density $p(\boldsymbol{x})$ of the data vectors considered. In practice this distribution is, however, usually not known and can, therefore, not be described exactly in parametric form. Yet one always assumes that a suitable sample set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T\}$ of example vectors is available. Based on this data the parameters of the actual distribution function can be approximated. In the following treatment of the algorithms for vector quantizer design this sample set will always be considered instead of the total input space $\mathbb{R}^k$.

Lloyd's Algorithm

The idea behind the method known as *Lloyd's algorithm* lies in the exploitation of the dual view on vector quantizers. By means of the procedures defined in the previous section alternatingly the partition and the codebook, respectively, are determined in an optimal way. By applying one iteration of this method an optimal quantizer will in general not be created. It can, however, be shown that an iterative application of the method results in a sequence of vector quantizers, which achieve ever decreasing average quantization errors.

The algorithm, which is summarized in figure 4.1, generates for a given sample set a vector quantizer of size N, i.e. a codebook containing N prototype vectors. At the beginning of the procedure a suitable initial codebook Y^0 is chosen, which, however, needs to be done heuristically. Though the choice of this starting point obviously influences the optimization, Useful results can be achieved in practice by randomly selecting N vectors from the sample set. This disadvantage of Lloyd's

```
Given a sample set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T\}$ of example vectors, the desired code-
book size $N$, and a lower bound $\Delta\epsilon_{\min}$ for the relative improvement of the quan-
tization error.
1. Initialization
   choose a suitable initial codebook $Y^0$ of size $N$
   (e.g. by randomly selecting $N$ vectors $\boldsymbol{y}_i^0$ from $\omega$)
   initialize iteration count $m \leftarrow 0$
2. Optimization of the Partition
   for the current codebook $Y^m$ determine the optimal partition by classifying
   all vectors $\boldsymbol{x}_t$ with $t = 1 \ldots T$ into cells
   $R_i^m = \{\boldsymbol{x} | \boldsymbol{y}_i^m = \underset{\boldsymbol{y} \in Y^m}{\operatorname{argmin}}\, d(\boldsymbol{x}, \boldsymbol{y})\}$
   also determine the average quantization error
   $\bar{\epsilon}(Y^m) = \frac{1}{T} \sum_{t=1}^{T} \min_{\boldsymbol{y} \in Y^m} d(\boldsymbol{x}_t, \boldsymbol{y})$
3. Codebook Update
   for all cells $R_i^m$ with $i = 1 \ldots N$ calculate new reproduction vectors
   $\boldsymbol{y}_i^{m+1} = \operatorname{cent}(R_i^m)$
   these constitute the new codebook $Y^{m+1} = \{\boldsymbol{y}_i^{m+1} | 1 \leq i \leq N\}$
4. Termination
   calculate the relative decrease of the quantization error with respect to the last
   iteration
   $\Delta\epsilon_m = \frac{\bar{\epsilon}(Y^{m-1}) - \bar{\epsilon}(Y^m)}{\bar{\epsilon}(Y^m)}$
   if the relative decrease was large enough, i.e. $\Delta\epsilon_m > \Delta\epsilon_{\min}$
           set $m \leftarrow m + 1$ and continue with step 2
   otherwise Stop!
```

Fig. 4.1. Lloyd's Algorithm for the design of vector quantizers

algorithm is largely avoided by the algorithm that will be presented in the following section.

In the next processing step the optimal partition of the sample set is computed for the current codebook Y^m. The mapping of every element $\boldsymbol{x}_t \in \omega$ onto a cell of the partition corresponds to its classification into that class R_i^m, for which the data element considered has minimal distance from the associated prototype vector $\boldsymbol{y}_i^m$. As the termination of the procedure is decided based on the quantization error achieved, this quantity is also computed during the classification step. Starting from the newly computed partition subsequently the updating of the codebook can be performed. The new codebook simply consists of all the centroids of the optimally chosen cells. The algorithm terminates, if no sufficient relative improvement of the quantization error is achieved any more by the optimization. Here the lower bound $\Delta\epsilon_{\min}$ is a parameter of the method, which needs to be specified by the user[7].

[7] Own experiments have shown that a relative improvement of the quantization error of less than a thousandth does not result in an improvement of the codebook which is relevant for practical applications.

```
Given a sample set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_T\}$ of example vectors, the desired code-
book size $N$, and a lower bound $\Delta\epsilon_{\min}$ for the relative improvement of the quan-
tization error.
  1. Initialization
     choose a suitable initial codebook $Y^0$ of size $N^0$
     (e.g. trivially as $Y^0 = \{\text{cent}(\omega)\}$ with $N^0 = 1$)
     initialize iteration count $m \leftarrow 0$
  2. Splitting
     generate from the current codebook $Y^m$ a new codebook
     with $N^{m+1} = 2\,N^m$ reproduction vectors
     $Y^{m+1} = \{\boldsymbol{y}_1 + \boldsymbol{\epsilon}, \boldsymbol{y}_1 - \boldsymbol{\epsilon}, \boldsymbol{y}_2 + \boldsymbol{\epsilon}, \boldsymbol{y}_2 - \boldsymbol{\epsilon}, \dots \boldsymbol{y}_{N^m} + \boldsymbol{\epsilon}, \boldsymbol{y}_{N^m} - \boldsymbol{\epsilon}\}$
     using a suitable small distortion vector $\boldsymbol{\epsilon}$
  3. Optimization
     optimize the newly created codebook $Y^{m+1}$ by applying Lloyd's algorithm
  4. Termination
     if the desired number of codebook vectors is not yet reached
               set $m \leftarrow m + 1$ and continue with step 2
     otherwise Stop!
```

Fig. 4.2. LBG Algorithm for the design of vector quantizers

The by far major part of the effort caused by this method results from the classification of all vectors in the sample set during the optimization of the partition in step 2. The effort necessary for the calculation of the updated codebook, however, is negligible[8]. It, therefore, makes sense to carry out this step in any case, even though the average quantization error computed corresponds to the previous codebook.

LBG Algorithm

The most problematic aspect of Lloyd's algorithm is its initialization. It is, therefore, a quite obvious goal to modify this part of the algorithm such that the results can no longer be adversely affected by a bad choice of the initial codebook. The *LBG algorithm*, which was named after the initials of its inventors Linde, Buzo & Gray [144], therefore, does not construct the desired vector quantizer of size N directly, but generates a sequence of quantizers with increasing number of codebook vectors.

The algorithm, which is summarized in figure 4.2, starts with the choice of an initial codebook. As at this time the full set of reproduction vectors does not need to be available, simple and robust possibilities exist for this choice to be made. An optimal choice is possible when considering the use of a trivial quantizer which possesses a single reproduction vector only. As then the whole sample set corresponds

[8] When using the Euclidean distance only the sample mean of the cells needs to be determined. One obtains the centroid, therefore, according to $\text{cent}(R) = \frac{1}{|R|}\sum_{\boldsymbol{x}\in R} \boldsymbol{x}$. The summation over the vectors contained in the respective cell can easily be performed incrementally during the classification, so that finally only the normalization on the total number of vectors is necessary.

to the single cell of the associated partition, this trivial codebook consists of exactly the centroid of the sample set[9].

In order to reach the desired codebook size in the course of the procedure the algorithm applies a method for splitting up existing codebook vectors into two respective new prototypes. Therefore, a suitable distortion vector $\boldsymbol{\epsilon}$ of small Euclidean norm will be added to or subtracted from all former prototype vectors resulting in two new codebook vectors $\boldsymbol{y}_i + \boldsymbol{\epsilon}$ and $\boldsymbol{y}_i - \boldsymbol{\epsilon}$ each that replace the original prototype. In total this operation yields a codebook that contains twice as many reproduction vectors as the original one[10].

Of course it can not be expected that the codebook generated in this way is optimal with respect to the quantization error achieved. Therefore, in the third step of the procedure the current codebook is optimized by applying Lloyd's algorithm (steps 2 to 4) known from the previous section. If the desired codebook size is not yet reached, the splitting of codebook vectors is again applied to the optimized quantizer.

Compared to the method after Lloyd the LBG algorithm offers the important advantage that the initialization process is clearly defined. Thus it is avoided that a random but unfortunate choice of the initial codebook causes the iterative optimization of the quantizer to reach an unsatisfactory local optimum only. Just the distortion vector necessary for splitting codebook vectors needs to be specified in a suitable way. The method also reduces the computational costs of the necessary classification as increasingly large codebooks are created step by step. As long as only rough estimates of the parameters of the vector quantizer are available in the beginning of the optimization, the method works on very small codebooks. These can be decoded much more efficiently than the final complete quantizer. During the continuous growth of the codebook refined and more elaborate models are used only when a better approximation of the desired quantization rule is already available.

k-Means Algorithm

Both Lloyd's algorithm and the LBG algorithm are well suited for the design of vector quantizers. However, both require the repeated expensive classification of all data vectors in the sample set during the necessary optimization steps. In contrast, with the so-called *k-means algorithm* there exists a method that is able to generate an approximately optimal codebook with only a single pass through the data to be processed. The procedure that was developed by MacQueen [149] actually results from the idea to use not a single sample mean but multiple means for the characterization of some distribution of data vectors. Yet these mean vectors correspond directly to the reproduction vectors of a vector quantizer of size k.

[9] In order to speed up the procedure in the starting phase one can also choose a small codebook containing $N^0 \ll N$ randomly selected vectors without adversely affecting the final result of the algorithm.

[10] It is also possible to split up only so many prototypes that the desired codebook size is reached, or only those ones generating an especially high local quantization error. In both cases it needs to be assured, however, that the method does not create too small cells.

```
Given a sample set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T\}$ of example vectors and the desired
codebook size $N$.
1. Initialization
   choose the first $N$ vectors of the sample set as initial codebook $Y^0$
   $Y^0 = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N\}$
   initialize iteration count $m \leftarrow 0$
2. Iteration
   for all vectors not yet processed $\boldsymbol{x}_t, N < t \leq T$
   a) Classification
      for $\boldsymbol{x}_t$ determine the optimal reproduction vector $\boldsymbol{y}_i^m$ in the current
      codebook $Y^m$
      $\boldsymbol{y}_i^m = \operatorname*{argmin}_{\boldsymbol{y} \in Y^m} d(\boldsymbol{x}_t, \boldsymbol{y})$
   b) Partition Update
      determine the new partition by updating the cell of the codebook vector
      selected
      $R_j^{m+1} = \begin{cases} R_j^m \cup \{\boldsymbol{x}_t\} & \text{if } j = i \\ R_j^m & \text{otherwise} \end{cases}$
   c) Codebook Update
      determine a new codebook by updating the prototype of the cell modi-
      fied in the previous step
      $\boldsymbol{y}_j^{m+1} = \begin{cases} \operatorname{cent}(R_j^{m+1}) & \text{if } j = i \\ \boldsymbol{y}_j^m & \text{otherwise} \end{cases}$
```

Fig. 4.3. k-means algorithm for the design of vector quantizers

Figure 4.3 shows the associated algorithm. In order to be consistent in notation with the other methods presented, we will denote the size of the codebook by N and not by k as in the original work.

As it is the goal of the method to process every data vector $\boldsymbol{x}_t$ only once, one obtains a surprisingly simple initialization rule. The initial codebook just consists of the first N vectors of the sample set. The following processing steps remind us of the two optimization phases of Lloyd's algorithm. However, in the k-means method they are applied to every vector independently. Thus a vector $\boldsymbol{x}_t$ is first mapped onto the optimal reproduction vector $\boldsymbol{y}_i^m$ in the current codebook Y^m according to the nearest-neighbor rule. Immediately afterwards the parameters of the vector quantizer are updated. Therein it is assumed that by mapping the vector $\boldsymbol{x}_t$ onto R_i^{m+1} only this very cell is changed. Therefore, it is sufficient to compute a new reproduction vector $\boldsymbol{y}_i^{m+1}$ for the modified cell. After a single pass through the sample set the newly created codebook is available.

It might be suspected that such a codebook estimation procedure does not lead to satisfactory results. In practice, however, the method is surprisingly powerful and largely more efficient than Lloyd's algorithm or the LBG method. When quality is concerned the vector quantizers generated by the k-means algorithm are generally not inferior to those created by the other methods. If the sample set consists of a sequence of vectors generated randomly and independently of each other, it can be shown that the method converges asymptotically against the optimal solution, provided that the size of the sample set approaches infinity [149]. In practice this means that the algorithm works especially well for large sample sets, which exhibit no or only very limited correlation between subsequent vectors.

Though the k-means algorithm is fundamentally different from the method attributed to Lloyd, the latter is frequently incorrectly referred to as the k-means algorithm[11]. When efficiency is concerned the method largely outperforms all iterative optimization methods as Lloyd's algorithm or the LBG algorithm with its linear complexity in the size of the sample set. Still the quality of the resulting vector quantizers is comparable.

4.4 Estimation of Mixture Density Models

The result of a vector quantization method describes a given data distribution only by means of the N reproduction vectors within the codebook created. A substantially more precise representation is obtained when using a mixture density model, as then also the characteristics of the local scatter of the data can be described. Because of the rather simple mathematical treatment usually normal distributions are used as components of such a model (cf. also section 3.4).

The simplest though also qualitatively worst method for the estimation of such a mixture model for a given sample set directly builds upon the results of a vector quantization process. After the codebook generation is finished just the parameters of a normal distribution are estimated for all cells R_i of the final partition. The necessary mean vector $\boldsymbol{\mu}_i$ is identical to the centroid $\boldsymbol{y}_i$, which is already known. Merely the sample covariance matrix needs to be calculated for the respective cell (cf. equation (3.9) page 43):

$$\boldsymbol{C}_i = \sum_{\boldsymbol{x} \in R_i} (\boldsymbol{x} - \boldsymbol{\mu}_i)(\boldsymbol{x} - \boldsymbol{\mu}_i)^T$$

Considerably better results are achieved, if the distortion of the vector space caused by the covariance matrix is already considered during the quantization process. The so-called *Mahalanobis distance* represents the associated extension of the Euclidean distance measure.

$$d_{\text{Mahalanobis}}(\boldsymbol{x}, \boldsymbol{\mu}) = (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{C}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})$$

As the comparison with equation (3.6) on page 39 shows, this formula is almost identical with the exponential term of a normal density function. It is, therefore, only a small step ahead to consider such a distribution directly in the quantization process.

As, however, a density function in contrast to a distance represents a measure of membership, the rule for mapping vectors to codebook entries needs to be modified. A vector $\boldsymbol{x}_t$ is now mapped to that cell R_i, for which the corresponding normal distribution $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{C}_i)$ yields the maximal density value:

[11] See also the bibliographical remarks in section 4.5 on page 59

$$R_i = \{\boldsymbol{x} | i = \underset{j}{\mathrm{argmax}} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j, \boldsymbol{C}_j)\}$$

This modification of the quantization rule makes it possible that mixture density models can be created with traditional methods for vector quantizer design. However, the algorithms still assume that the average quantization error should be minimized, which is not an appropriate optimization criterion for a probability distribution.

This can be made clear when considering the fact that for the calculation of the parameters of the normal distributions only vectors from a limited region — the respective cell — are used. The density function itself is, however, defined for the complete vector space and for all possible vectors yields non-negative density valued, which may become arbitrarily small though.

EM Algorithm

For the correct estimation of Gaussian mixture models the so-called *EM algorithm* [42] is used. It defines a very general method for the iterative optimization of statistical models with hidden states or variables. As the general treatment of this method is rather cumbersome and sometimes not quite intuitive, we will focus here on the concrete variant of the algorithm, which is used for estimating mixture models on the basis of normal distributions. A model with N component densities is defined as follows (cf. equation (3.7) page 39):

$$p(\boldsymbol{x}|\theta) = \sum_{i=1}^{N} c_i \, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{C}_i)$$

Rather than representing a clearly bounded region or cell within the vector space the normal distributions statistically represent the distribution of similar vectors or patterns. The set of parameters consisting of the prior probabilities c_i of the individual pattern classes and the associated density parameters $\boldsymbol{\mu}_i$ and $\boldsymbol{C}_i$ will be taken together and denoted by θ.

Principally the EM algorithm always has the goal of maximizing the probability of the observed data $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T\}$ depending on the respective model. The fact that the dependent variable in this quantity is the set of parameters θ and not the sample set is reflected by the transition to the likelihood function:

$$L'(\theta|\omega) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T|\theta)$$

As the further application of a monotonic function does not alter the result of the maximization and as the treatment of normal distributions is considerably simplified by taking the logarithm, usually the logarithm of $L'(\theta|\omega)$ is used, which for simplicity we will be referring to as the likelihood function in the following:

$$L(\theta|\omega) = \ln L'(\theta|\omega) = \ln p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T|\theta) = \sum_{\boldsymbol{x} \in \omega} \ln p(\boldsymbol{x}|\theta)$$

```
Given a sample set $\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots \boldsymbol{x}_T\}$ of example vectors, the desired code-
book size $N$, and a lower bound $\Delta L_{\min}$ for the relative improvement of the likeli-
hood.
1. Initialization
   choose initial parameters $\theta^0 = (c_i^0, \boldsymbol{\mu}_i^0, \boldsymbol{C}_i^0)$ of the mixture model
   initialize iteration count $m \leftarrow 0$
2. Estimation
   for every vector $\boldsymbol{x} \in \omega$ calculate estimates of the posterior probabilities of the
   pattern classes using the current models $\theta^m$
   $P(\omega_i|\boldsymbol{x}, \theta^m) = \frac{c_i^m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^m, \boldsymbol{C}_i^m)}{\sum_j c_j^m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j^m, \boldsymbol{C}_j^m)}$
   calculate the likelihood of the data for the current model $\theta^m$
   $L(\theta^m|\omega) = \ln p(\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T|\theta^m) = \sum_{\boldsymbol{x}\in\omega} \ln \sum_j c_j^m \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_j^m, \boldsymbol{C}_j^m)$
3. Maximization
   calculate updated parameters $\theta^{m+1} = (c_i^{m+1}, \boldsymbol{\mu}_i^{m+1}, \boldsymbol{C}_i^{m+1})$
   $c_i^{m+1} = \frac{\sum_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x}, \theta^m)}{|\omega|}$
   $\boldsymbol{\mu}_i^{m+1} = \frac{\sum_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x}, \theta^m)\, \boldsymbol{x}}{\sum_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x}, \theta^m)}$
   $\boldsymbol{C}_i^{m+1} = \frac{\sum_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x}, \theta^m)\, \boldsymbol{x}\boldsymbol{x}^T}{\sum_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x}, \theta^m)} - \boldsymbol{\mu}_i^{m+1}(\boldsymbol{\mu}_i^{m+1})^T$
4. Termination
   calculate the relative change in the likelihood with respect to the previous
   iteration
   $\Delta L_m = \frac{L(\theta^m|\omega) - L(\theta^{m-1}|\omega)}{L(\theta^m|\omega)}$
   if the relative improvement was large enough, i.e. $\Delta L_m > \Delta L_{\min}$
           let $m \leftarrow m + 1$ and continue with step 2
   otherwise Stop!
```

Fig. 4.4. EM algorithm for the estimation of mixture models.

The name of the EM algorithm is derived from its two main processing steps. First, the values of the non-observable probabilistic quantities are estimated depending on the current model and the given data. This step is called expectation or simple E-step. Afterwards, new model parameters are calculated, which locally optimize the objective function, i.e. the likelihood of the model. This second phase is referred to as the maximization or M-step.

In order to be able to apply the EM algorithm for the estimation of mixture models a sample set of example data and the desired number of component distributions must be given. The hidden variables here consist of the unknown correspondences between data vectors and normal distributions or pattern classes, respectively.

For the initialization of the procedure, which is summarized in figure 4.4, first some suitable initial parameters of the mixture model need to be determined. As the complexity of the model is significantly higher compared to a vector quantizer and, therefore, the method is extremely sensitive to the choice of the initial model, a random initialization is no longer possible. However, a suitable initial model can easily be determined with one of the two simpler methods described earlier. Besides the initial parameters of the component densities $\boldsymbol{\mu}_i^0$ and $\boldsymbol{C}_i^0$ only the mixture weights $c_i^0 = \frac{|R_i|}{\sum_j |R_j|}$ need to be defined, which correspond to estimates of the prior probabilities $P(\omega_i)$ of the pattern classes ω_i.

In the E-step probabilities for the mapping of every vector onto one of the codebook classes ω_i are derived depending on the current model θ^m. These posterior probabilities can be calculated for arbitrary vectors $\boldsymbol{x} \in \omega$ in the following way:

$$\begin{aligned} P(\omega_i|\boldsymbol{x}, \theta^m) &= \frac{P(\omega_i|\theta^m)\, p(\boldsymbol{x}|\omega_i, \theta^m)}{p(\boldsymbol{x}|\theta^m)} = \frac{P(\omega_i|\theta^m)\, p(\boldsymbol{x}|\omega_i, \theta^m)}{\sum_j P(\omega_j|\theta^m)\, p(\boldsymbol{x}|\omega_j, \theta^m)} \\ &= \frac{c_i^m\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^m, \boldsymbol{C}_i^m)}{\sum_j c_j^m\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_i^m, \boldsymbol{C}_i^m)} \end{aligned}$$

Here first Bayes' rule is applied to rewrite the expression. As the overall density function $p(\boldsymbol{x}|\theta^m)$ is exactly described by the current model it can be substituted by the mixture density itself in the second rewriting step.

Furthermore it makes sense to compute the likelihood of the current model as a sum over the logarithmic density values of the mixture during the estimation of the posterior probabilities.

Starting from these probabilities for the mapping between data vectors and pattern classes, in the next step, the M-step, new parameters of the mixture density are derived such, that the likelihood of the current model is maximized given the sample data. One obtains the following relationships for computing estimates of the prior probabilities c_i, the mean vectors $\boldsymbol{\mu}_i$, and the associated covariance matrices $\boldsymbol{C}_i$:

$$c_i = \frac{\sum\limits_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x})}{|\omega|}$$

$$\boldsymbol{\mu}_i = \frac{\sum\limits_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x})\boldsymbol{x}}{\sum\limits_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x})}$$

$$\boldsymbol{C}_i = \frac{\sum\limits_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x})(\boldsymbol{x}-\boldsymbol{\mu}_i)(\boldsymbol{x}-\boldsymbol{\mu}_i)^T}{\sum\limits_{\boldsymbol{x}\in\omega} P(\omega_i|\boldsymbol{x})}$$

In the same way as in the design of vector quantizers it is also necessary to decide in the EM algorithm about the termination of the procedure in a suitable way. In analogy to the termination criterion used for Lloyd's algorithm this decision can be based on the relative improvement of the likelihood $L(\theta|\omega)$, which is obtained for the current model parameters given the sample set.

In contrast to classical vector quantization techniques the mapping of data vectors to pattern classes — i.e. the individual component densities — is not performed via the binary decision of a classifier but probabilistically instead. The estimation of mixture densities with this method is, therefore, sometimes also referred to as "soft vector quantization".

4.5 Bibliographical Remarks

An in-depth treatment of the field of vector quantization can be found in the fundamental monograph by Gersho & Gray [85], which the presentation of the topic in this chapter was partially based on [85, Chap. 10, pp. 309–340 and Chap. 11, pp. 345–400]. A good summary of the most important aspects can also be found in [97, Chap. 4.4, pp. 163–175].

The *k-means algorithm* for the design of vector quantizers was developed by MacQueen [149]. After the initials of its inventors Linde, Buzo & Gray the *LBG algorithm* was named [144]. Descriptions of this method can also be found in [85, pp. 361–362], [81, pp. 395–396] or [97, pp. 169–170]. In contrast the origin of *Lloyd's algorithm* is not completely clear. Later descriptions of the method can be found, e.g., in [144] or [85, pp. 362–370]. In the literature the method is frequently incorrectly referred to as k-means algorithm (cf. e.g. [196, p. 125], [105, p. 11], [81, pp. 394–395], and [97, pp. 166–169]) though it is fundamentally different from this method (see also section 4.3 page 50). The *EM algorithm* for the parameter estimation of statistical models with hidden variables goes back to work of Dempster, Laird & Rubin [42]. Summarizing descriptions of this very general method and proofs of its convergence can, e.g., be found in [95, pp. 29–35] and [97, pp. 170–175].

5 Hidden Markov Models

In the field of pattern recognition signals are frequently thought of as the product of sources that act statistically. The goal of the analysis of such signals, therefore, is to model the statistical properties of the assumed signal sources as exactly as possible. As a basis of the model building merely the observed example data and assumptions about limitations in the model's degrees of freedom are available. However, the model to be determined should not only replicate the generation of certain data as exactly as possible but also deliver useful information for segmenting the signals in some meaningful units.

Hidden Markov models are able to treat *both* above-mentioned modeling aspects. In a two-stage stochastic process information about the segmentation can be derived from the internal states of the models, whereas the generation of the signal data itself takes place in the second stage.

The great popularity of this modeling technique results from its successful application and consequent further development in the field of automatic speech recognition. In this area of research hidden Markov models have effectively superseded all competing approaches and constitute the dominating processing paradigm. Their ability to describe processes or signals evolving over time especially well has in particular caused that the also very successful technique of artificial neural networks is only in very few cases used for speech recognition and comparable segmentation problems. However, there exist a number of hybrid systems consisting of a combination of hidden Markov models and artificial neuronal networks that try to make use of the advantages of both modeling techniques (see section 5.8.2).

5.1 Definition

Hidden Markov models (HMMs) describe a two-stage stochastic process. The first stage consists of a discrete stochastic process, which is stationary, causal, and simple. The state space considered is finite. Thus the process probabilistically describes the state transitions within a discrete, finite state space. It can be visualized as a finite state automaton with edges between any pair of states, which are labeled with transition probabilities. The behavior of the process at a given time t is only dependent on the immediate predecessor state and can be characterized as follows:

$$P(S_t|S_1, S_2, \dots S_{t-1}) = P(S_t|S_{t-1})$$

In the second stage then for every point in time t additionally an output or emission O_t is generated. The associated probability distribution is only dependent on the current state S_t and not on any previous states or emissions[1].

$$P(O_t|O_1 \dots O_{t-1}, S_1 \dots S_t) = P(O_t|S_t)$$

This sequence of emissions is the only thing that can be observed of the behavior of the model. In contrast, the state sequence taken on during the generation of the data can not be observed. It is so to speak "hidden" from which the term *hidden* Markov models is derived. From the outside view on the model — i.e. the observation of its behavior — also the more common reference to the sequence $O_1, O_2 \dots O_T$ of emissions as *observation sequence* is motivated. Individual elements of this sequence we will be calling *observations* in the following.

In the pattern recognition literature the behavior of HMMs is always considered for a finite time interval of length T. For the initialization of the model at the beginning of that period additional start probabilities are used that describe the probability distribution of the states at time $t = 1$. An equivalent termination criterion is usually missing, however. The actions of the model are, therefore, terminated as soon as an arbitrary state at time T is reached. Neither a statistically nor a declarative criterion is used to specifically mark end states[2].

A hidden Markov model of first order, which is usually denoted as λ, is, therefore, completely described by

- a finite set of states $\{s|1 \leq s \leq N\}$ that in the literature are usually only referred to by their indices,
- a matrix $\boldsymbol{A}$ of state transition probabilities

$$\boldsymbol{A} = \{a_{ij}|a_{ij} = P(S_t = j|S_{t-1} = i)\}$$

- a vector $\boldsymbol{\pi}$ of start probabilities

$$\boldsymbol{\pi} = \{\pi_i|\pi_i = P(S_1 = i)\}$$

[1] In the tradition of research at IBM HMMs are described in a slightly different way. There emissions occur during state transitions, i.e. at the edges of the model (cf. e.g. [105, p. 17]). With respect to its expressive power this formulation is, however, completely equivalent to the one which is used in this book and also throughout the majority of the literature, as is also shown in [105, p. 35–36].

[2] In [95, p. 150] analogously to the start probabilities also termination probabilities are defined. The HMM architectures used for analyzing biological sequences usually contain special non-emitting start and end states (see also section 8.4 page 133).

- and state specific probability distributions

  $$\{b_j(o_k) \,|\, b_j(o_k) = P(O_t = o_k|S_t = j)\} \quad \text{or} \quad \{b_j(\boldsymbol{x}) \,|\, b_j(\boldsymbol{x}) = p(\boldsymbol{x}|S_t = j)\}$$

  for the output of the model

However, the output distributions need to be distinguished depending on the type of emissions the model is generating. In the simplest case the outputs are generated from a discrete inventory $\{o_1, o_2, \ldots o_M\}$ and, therefore, have a symbolic nature. The quantities $b_j(o_k)$ then represent discrete probability distributions which can be grouped together in a matrix of output probabilities:

$$\boldsymbol{B} = \{b_{jk}|b_{jk} = P(O_t = o_k|S_t = j)\}$$

For this choice of output modeling one obtains so-called *discrete HMMs*.

If the observations are vector valued quantities $\boldsymbol{x} \in \mathbb{R}^n$ instead, the output distributions are described on the basis of continuous probability density functions:

$$b_j(\boldsymbol{x}) = p(\boldsymbol{x}|S_t = j)$$

Current applications of HMMs for problems of signal analysis exclusively use these so-called *continuous HMMs*, even though the necessity to model continuous distributions significantly increases the complexity of the formalism

5.2 Modeling Emissions

The description of the model's emissions by discrete probability distributions is normally used in the literature for the introduction of HMMs. This variant is also applied to the analysis of biological sequences, as there one can work with discrete symbolic inventories of either the four bases used to build DNA strands or the 20 possible amino acids (cf. e.g. [127]). For applications in the field of signal processing, however, discrete models are hardly considered any more, as they require the use of a vector quantizer, which converts continuous feature representations of speech signals or handwritten script into discrete observation sequences prior to the analysis.

Avoiding this quantization step or including it into the model building process, respectively, considerably increases the expressive power of the HMM formalism. In order to achieve this it is, however, necessary to represent continuous distributions over $\mathbb{R}^n$ in a suitable way. Here a direct representation by the empirically given distribution as in the discrete case is not possible. Parametric representations are, however, only known for a small number of families of distributions, as e.g. the normal or Gaussian distribution. For the desired application area such "simple" models can not be used, as — without exception — they represent uni-modal distributions. With only a single region of high density in the area of the expected value only extremely "well-behaved" data can be described.

In order to be able to nevertheless deal with arbitrary continuous distributions with multiple modes or regions of high density in general, approximative techniques are applied. The most well known and most widely used method consists in the use of *mixture densities* on the basis of Gaussian densities. It can be shown that every general continuous probability distribution $p(\boldsymbol{x})$ can be approximated to arbitrary precision with a linear combination of, in general, infinitely many component normal distributions [252]:

$$p(\boldsymbol{x}) \quad \hat{=} \quad \sum_{k=1}^{\infty} c_k \, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k) \quad \approx \quad \sum_{k=1}^{M} c_k \, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k)$$

If a finite number M of mixtures is used for the approximation, an error results, which, however, can be kept small by using a suitable number of component densities. As a normalization constraint the mixture weights, which may vary between zero and one, need to sum up to unity:

$$\sum_k c_k \; = \; 1 \qquad \text{with} \qquad 0 \leq c_k \leq 1 \; \forall k$$

The general form of continuous HMMs, therefore, uses one mixture density per state j for the description of the output probability density function:

$$b_j(\boldsymbol{x}) \quad = \quad \sum_{k=1}^{M_j} c_{jk} \, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{jk}, \boldsymbol{C}_{jk}) \quad = \quad \sum_{k=1}^{M_j} c_{jk} \, g_{jk}(\boldsymbol{x}) \qquad (5.1)$$

The number M_j of mixture components used may vary from state to state. Each of the normal distributions, which will be denoted by $g_{jk}(\boldsymbol{x})$ in the following, furthermore possesses an individual set of parameters consisting of a mean vector $\boldsymbol{\mu}_{jk}$ and a covariance matrix $\boldsymbol{C}_{jk}$.

A continuous HMM can thus also be considered as a discrete model with a state specific "soft" quantization stage. The discrete output distributions can be found in the mixture weights, and the component densities used define the quantization rule. In contrast to a discrete HMM here no "hard" decisions are made, but the density values of all normal distributions used are incorporated into the calculation.

As for continuous HMMs the number of parameters is drastically increased with respect to the discrete case, a number of techniques was developed to reduce the number of parameters by jointly using parts of the model. Such methods are usually referred to as the *tying* of parameters. They will be treated in greater detail in section 9.2 from page 152 onward.

The most well-known of these approaches are the so-called *semi-continuous HMMs* developed by Huang and colleagues, which are frequently referred to as *tied-mixture models* [94, 95]. In such models only a single set of component densities is used to construct all state-specific output probability densities:

$$b_j(\boldsymbol{x}) \quad = \quad \sum_{k=1}^{M} c_{jk} \, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{C}_k) \quad = \quad \sum_{k=1}^{M} c_{jk} \, g_k(\boldsymbol{x}) \qquad (5.2)$$

This global set of mixture components $g_k(\boldsymbol{x})$ is frequently also referred to as the codebook, as in this variant of the model the relationship to a discrete model with initial quantization stage becomes the most obvious. The definition of the output probability densities is completely analogous to the general continuous case. However, there is no longer a state dependency of the parameters of the underlying normal distributions. Additionally, in semi-continuous HMMs every mixture density consists of the same number M of baseline distributions.

5.3 Use Cases

The application of HMMs for pattern recognition tasks is always backed by the fundamental assumption that the patterns considered are — at least in principle — outputs of a comparable stochastic model and that the properties of this model can be described reasonably well by HMMs.

A relevant question for a given model is, therefore, how well this model describes some pattern — i.e. a certain observation sequence. For this purpose the production probability $P(\boldsymbol{O}|\lambda)$ of the data for a known model — or a reasonable approximation of it — needs to be computed. On the one hand this quantity indicates, how well some model λ that was built for describing certain example data $\boldsymbol{O} = O_1, O_2, \ldots O_T$ is capable of representing the statistical properties of this data. On the other hand it can be used as the basis for a classification decision.

If two or more HMMs λ_i are available, which resemble the statistical properties of patterns from different pattern classes Ω_i, some given sequence of test data $\boldsymbol{O}$ can be classified into that class Ω_j, for which the posterior probability $P(\lambda_j|\boldsymbol{O})$ becomes maximum:

$$P(\lambda_j|\boldsymbol{O}) = \max_i \frac{P(\boldsymbol{O}|\lambda_i)P(\lambda_i)}{P(\boldsymbol{O})} \tag{5.3}$$

When evaluating this expression, the probability $P(\boldsymbol{O})$ of the data itself represents a quantity irrelevant for the classification — or the maximization of $P(\lambda_i|\boldsymbol{O})$ — because it is independent of λ_i and, therefore, constant. Thus for determining the optimal class it is sufficient to consider the denominator of equation (5.3) only:

$$\lambda_j = \operatorname*{argmax}_{\lambda_i} P(\lambda_i|\boldsymbol{O}) = \operatorname*{argmax}_{\lambda_i} \frac{P(\boldsymbol{O}|\lambda_i)P(\lambda_i)}{P(\boldsymbol{O})} = \operatorname*{argmax}_{\lambda_i} P(\boldsymbol{O}|\lambda_i)P(\lambda_i)$$

However, for this procedure in addition the prior probabilities $P(\lambda_i)$ of the individual models need to be specified. For the sake of simplicity these are, therefore, frequently neglected, so that the classification decision is solely based on the production probability $P(\boldsymbol{O}|\lambda_i)$.

Such an approach can be applied to the classification of isolated sample patterns that do not need to be further segmented into smaller units. The models for describing the different pattern classes are defined completely independently of each other. In this way, e.g., isolated word recognition tasks can be solved or local similarities of biological sequences can be analyzed (cf. [52, pp. 87–91]).

The mere evaluation of the production probability makes, however, only classification decisions on completely segmented data possible. Therefore, in the context of HMM technology a different model of usage is of greater importance, which allows for the integrated segmentation and classification of sequences of sample data. Therein one assumes that the possibility exists to associate individual meaningful units of a larger observation sequence with partial structures within a larger model. Usually this is achieved by a constructive approach. First, small models for certain elementary segmental units are created, as e.g. for speech sounds, handwritten characters, or short amino-acid sequences. These can then be combined to form larger compound models taking into account certain restrictions about the sequencing possibilities. In automatic speech recognition thus from single speech sounds models for words and finally whole spoken utterances are constructed. In the field of bioinformatics models for families of proteins can be built in a similar way.

For such a complex HMM the production probability $P(\boldsymbol{O}|\lambda)$ yields hardly any relevant information about the data considered. Instead the internal processes involved in the generation of the observation sequence $\boldsymbol{O} = O_1, O_2, \ldots O_T$ using the model λ need to be uncovered, i.e. the associated state sequence $\boldsymbol{s} = s_1, s_2, \ldots s_T$. On this basis inferences about the partial models involved and also a segmentation of the observation sequence into these units can be achieved. In a stochastic formalism inferences can, however, always be drawn in a probabilistic way only. Therefore one determines that state sequence $\boldsymbol{s}^*$ that maximizes the production probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ of the data for a given model λ. This procedure is usually referred to as the *decoding* of the model, as the generation of observations by the model can be viewed as the coding of the internal state sequence into observable entities.

Finally, the important question needs to be clarified, how a model can be built at all, that sufficiently enough resembles the statistical properties of certain data, in order to be used for purposes of classification or recognition. Though it can be evaluated by considering the production probability, how well a certain model describes given data, no algorithm is known, which could generate the optimal model for a certain task on the basis of example data. It is merely possible to improve an existing model λ such that the optimized model $\hat{\lambda}$ better resembles — in the statistical sense — the example data used. As with most iterative optimization methods also here the choice of a suitable initial model is of essential importance.

The use cases outlined above are frequently described as three fundamental problems for HMMs in the literature[3]. The so-called *evaluation problem* is concerned with the question, how for a given model the probability can be determined to generate a given observation sequence with this very model. The drawing of inferences about the internal processes within HMMs is the aim of the so-called *decoding problem*. The challenging search for the model which will be optimal to represent the properties of certain example data is finally called the *training problem*.

[3] The idea of characterizing the possible use cases of HMMs in the form of three fundamental problems is said to be going back to Jack Ferguson, Institute for Defense Analyses, Alexandria, VA, USA, as Rabiner states without further reference in [195].

When following classical presentations of HMMs, which were influenced by [195] in the majority of cases, for each of these problem there exists an algorithm for its solution. In this view it is, however, totally ignored that in many cases alternative methods exist and that some algorithms can also be used for the solution of different tasks. Therefore, in the following we will break up the quite restrictive structuring of HMM theory into pairs of problems and associated algorithms. Instead, the presentation will be oriented at the respective use case, i.e. we will present methods for evaluation and decoding of HMMs as well as for the estimation of the models' parameters.

5.4 Notation

In the literature a rather homogeneous notation can be found in formal descriptions of HMMs. Starting and transition probabilities, the model itself, as well as many of the derived quantities, which will be presented in the following sections, are denoted with unique symbols. This "standardization" can probably be explained to the major part by the influence of the classical article by Rabiner [195].

The consistence in notation stops, however, as soon as the transition from discrete to continuous models is performed. Due to the very nature of the model emissions now can no longer be described with discrete probability distributions, but need to be represented by continuous density functions. As a further consequence many derived probability quantities are transformed into densities.

In order to achieve an HMM notation with a maximum of uniformity we will keep the "discrete" view on the models in the following, as long as a differentiation or specialized treatment of continuous models is not necessary. This means that we will always be talking about probabilities even if those would need to be eventually densities in the continuous case. The emissions of the models will in these cases also be uniformly denoted as a sequence $\boldsymbol{O}$ of elements $O_1, O_2, \ldots O_T$.

The discrimination between discrete and continuous models will exclusively be made on the basis of the values that can be taken on by these random variables O_t. In the discrete case we will be denoting the observation symbols as o_k, in the continuous case, however, as vectors $\boldsymbol{x} \in \mathbb{R}^N$. In analogy to the notation of probabilities $P(O_t = o_k)$ we will use the notation $p(O_t = \boldsymbol{x})$ or $p(O_t = \boldsymbol{x}_t)$, respectively, for densities. The latter can, by leaving away the explicit specification of the random variable O_t, be transformed into the notation $p(\boldsymbol{x}_t)$ for continuous model outputs of an HMM that is commonly used in the literature.

These conventions will, however, not lead to an absolute consistency in the notation, but as Duda & Hart [49, p. 173] state quite pointedly[4] ...

> *... we shall recall Emerson's remark that a foolish consistency is the hobgoblin of little minds and proceed.*

[4] Original quote from Ralph Waldo Emerson (American Philosopher, 1803 – 1882) from the essay *"Self Reliance"* (1841).

5.5 Evaluation

The most widely used measure for assessing the quality, with which an HMM describes the statistical properties of certain data, is the so-called production probability. It gives the probability that the observation sequence considered was in some arbitrary way — i.e. along an arbitrary state sequence — generated by a certain model. A similar but slightly modified evaluation criterion is obtained, if only that state sequence is considered, which for the given data yields the maximal generation probability.

5.5.1 The Production Probability

In order to calculate the *production probability* $P(\boldsymbol{O}|\lambda)$ of a certain observation sequence $\boldsymbol{O}$ for a given model λ we will first consider an intuitively simple but quite inefficient method.

As the output of observation elements O_t can only result from the states of the model, an observation sequence $O_1, O_2, \ldots O_T$ needs to be generated along a corresponding state sequence $\boldsymbol{s} = s_1, s_2, \ldots s_T$ of the same length. The probability for this happening is obtained as the product of the output probabilities along the state sequence, as the model is given and also the concrete state sequence can be assumed to be fixed:

$$P(\boldsymbol{O}|\boldsymbol{s}, \lambda) = \prod_{t=1}^{T} b_{s_t}(O_t) \tag{5.4}$$

The probability that a given model λ runs through an arbitrary state sequence is in turn simply obtained as the product of the respective state transition probabilities. At the beginning of the sequence in principle the start probability π_{s_1} needs to be used. The notation can, however, be greatly simplified by the definition of $a_{0i} := \pi_i$ and $s_0 := 0$:

$$P(\boldsymbol{s}|\lambda) = \pi_{s_1} \prod_{t=2}^{T} a_{s_{t-1},s_t} = \prod_{t=1}^{T} a_{s_{t-1},s_t} \tag{5.5}$$

By taking together equations (5.4) and (5.5) it can immediately be determined, with which probability a given model λ generates a certain observation sequence $\boldsymbol{O}$ in a specific way — i.e. along a certain state sequence $\boldsymbol{s}$.

$$P(\boldsymbol{O}, \boldsymbol{s}|\lambda) = P(\boldsymbol{O}|\boldsymbol{s}, \lambda) P(\boldsymbol{s}|\lambda) = \prod_{t=1}^{T} a_{s_{t-1},s_t} b_{s_t}(O_t) \tag{5.6}$$

By the interleaving of the calculation procedures an expression results that immediately reflects the internal processes of the model: In turn state transitions are carried out according to a_{s_{t-1},s_t} and state-specific outputs are generated according to $b_{s_t}(O_t)$.

Due to their statistical nature HMMs are principally capable of generating a desired observation sequence $\boldsymbol{O}$ along every arbitrarily chosen sequence of states $\boldsymbol{s}$ of

```
Let $\alpha_t(i) = P(O_1, O_2, \ldots O_t, s_t = i|\lambda)$
  1. Initialization
     $\alpha_1(i) := \pi_i b_i(O_1)$
  2. Recursion
     for all times $t$, $t = 1 \ldots T-1$:
     $\alpha_{t+1}(j) := \sum_i \{\alpha_t(i)\, a_{ij}\}\; b_j(O_{t+1})$
  3. Termination
     $P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$
```

Fig. 5.1. Forward algorithm for the calculation of the production probability $P(\boldsymbol{O}|\lambda)$ of an observation sequence $\boldsymbol{O}$ for a given model λ.

the same length, though eventually with arbitrarily low probability. Therefore, for the calculation of the production probability $P(\boldsymbol{O}|\lambda)$ of a certain observation sequence $\boldsymbol{O}$ all state sequences of length T through the model need to be taken into account as possible "causes". The overall probability is then obtained as the sum over all probabilities for *jointly* generating the observation sequence $\boldsymbol{O}$ and a specific state sequence $\boldsymbol{s}$:

$$P(\boldsymbol{O}|\lambda) = \sum_{\boldsymbol{s}} P(\boldsymbol{O}, \boldsymbol{s}|\lambda) = \sum_{\boldsymbol{s}} P(\boldsymbol{O}|\boldsymbol{s}, \lambda) P(\boldsymbol{s}|\lambda)$$

This "brute force" method thus consists in the enumeration of all possible state sequences of length T through the model, the calculation of the production probability $P(\boldsymbol{O}|\boldsymbol{s}, \lambda)$ along every specific sequence, and the summation of the results obtained. The computational effort for this conceptionally simple method is, however, with $O(TN^T)$ exponential in the length of the observation sequence. Therefore, the method is out of question for practical applications.

Forward Algorithm

The widespread use of HMM technology can essentially be attributed to the fact that for this formalism efficient algorithms are known for the solution of the central problems. All these methods use the Markov property valid for all HMMs, this is their limited "memory" that only allows for the storing of *a single* internal state. If a model λ has taken on a certain state j at time t it is totally irrelevant for the future behavior of the model, on which path this state was arrived at and which outputs were generated in this course. At the respective next point in time $t+1$ it is, therefore, sufficient to consider all possible states at time t only — this is the N states of the model. Thus one can *always* carry out the elementary calculations for HMMs strictly along the time axis in parallel for all model states.

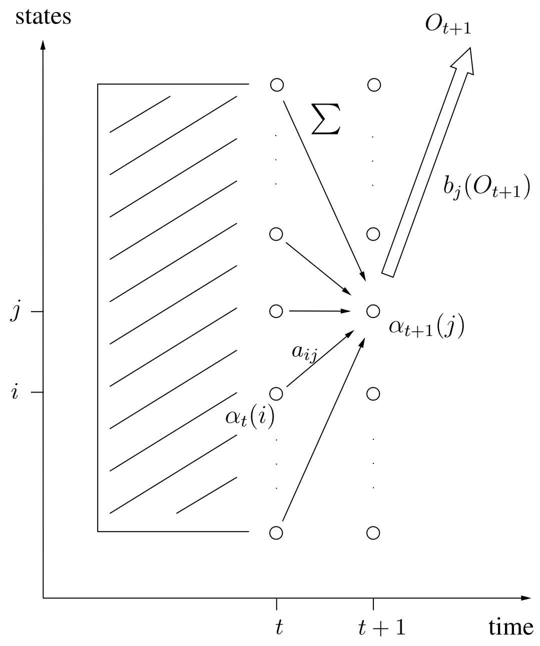


Fig. 5.2. Calculation scheme for determining the forward variables $\alpha_t(i)$ by means of the forward algorithm.

For the calculation of the production probability $P(\boldsymbol{O}|\lambda)$ this principle leads to the so-called *forward algorithm*[5], the actions of which are summarized in figure 5.1. One defines as the so-called *forward variables* $\alpha_t(i)$ the probability that for a given model λ the first part of the observation sequence up to O_t is generated and at time t the state i is reached.

$$\alpha_t(i) = P(O_1, O_2, \ldots O_t, s_t = i|\lambda) \tag{5.7}$$

Based on this quantity a recursive procedure for the calculation of the production probability for the entire observation sequence can be formulated. As already outlined above, the method works in parallel for all model states in a strictly time-synchronous manner. The resulting calculation scheme is graphically visualized in figure 5.2.

For the initialization of the computations or the anchoring of the recursion, respectively, the forward probabilities $\alpha_1(i)$ at the beginning of the time axis, i.e. at time $t = 1$ need to be determined. The probability for generating the first observation element O_1 at the initial point in time and for reaching state i, is obtained as the

[5] As the name indicates already there exists a matching counterpart of the forward algorithm, which is referred to as the backward algorithm. Taken together they constitute the so-called forward-backward algorithm, which will be presented in its entirety during the presentation of the HMM parameter training in section 5.7

product of the start probability π_i for state i and the output probability $b_i(O_1)$ of the observation element in this state:

$$\alpha_1(i) = \pi_i \, b_i(O_1)$$

For the further course of the calculations one can now assume that according to the induction principle the calculation of the $\alpha_t(i)$ is possible for all past times t. It is, therefore, sufficient to specify a rule, how the forward variables $\alpha_{t+1}(j)$ can be computed from these quantities at the next point in time.

In order to generate the prefix of the observation sequence $O_1, O_2, \ldots, O_t, O_{t+1}$, which is extended by the observation element O_{t+1}, and to reach state j, all possibilities need to be considered for generating $O_1, O_2, \ldots, O_t$ — which is equivalent to $\alpha_t(i)$ — and then proceeding by one further step in time. The forward variable $\alpha_{t+1}(j)$ is, therefore, obtained as the sum of the $\alpha_t(i)$ over all possible predecessor states i and inclusion of the respective transition probabilities a_{ij} to the current state. Additionally, the observation element O_{t+1} needs to be generated according to $b_j(O_{t+1})$:

$$\alpha_{t+1}(j) := \left\{ \sum_i \alpha_t(i) \, a_{ij} \right\} b_j(O_{t+1}) \tag{5.8}$$

At the end of the calculations at time T one obtains N different probabilities $\alpha_T(i)$ for generating the observation sequence $\boldsymbol{O}$ in an arbitrary way and finally reaching state i. The overall production probability $P(\boldsymbol{O}|\lambda)$ is obtained by summation over these partial probabilities.

$$P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

5.5.2 The “Optimal” Production Probability

For the mathematical exact determination of the total production probability is was necessary to include all possible paths through the respective model into the computations. For the evaluation of the modeling quality of an HMM this summarizing consideration is, however, not necessarily the only procedure making sense. Therein the possibility is lost to judge the specialization of partial models within the total model. However, when considering only the respective optimal possibility to generate a certain observation sequence with a given model, a model, which is satisfactory on the average, can be discriminated from another one, which works especially well in certain cases.

When disregarding efficiency considerations the optimal production probability $P^*(\boldsymbol{O}|\lambda)$ — i.e. the optimal probability $P(\boldsymbol{O}, s^*|\lambda)$ for generating the observation sequence along a specific path — can be determined by maximization over all individual production probabilities given by equation 5.6:

$$P^*(\boldsymbol{O}|\lambda) = P(\boldsymbol{O}, s^*|\lambda) = \max_s P(\boldsymbol{O}, s|\lambda)$$

```
Let $\delta_t(i) = \max_{s_1, s_2, \ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_1, s_2, \ldots s_{t-1}, s_t = i|\lambda)$
  1. Initialization
     $\delta_1(i) = \pi_i b_i(O_1)$
  2. Recursion
     for all times $t$, $t = 1 \ldots T-1$:
     $\delta_{t+1}(j) = \max_i \{\delta_t(i) a_{ij}\} b_j(O_{t+1})$
  3. Termination
     $P^*(\boldsymbol{O}|\lambda) = P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda) = \max_i \delta_T(i)$
```

Fig. 5.3. Algorithm for the calculation of the maximal production probability $P^*(\boldsymbol{O}|\lambda)$ of an observation sequence $\boldsymbol{O}$ along the respective optimal path for a given model λ.

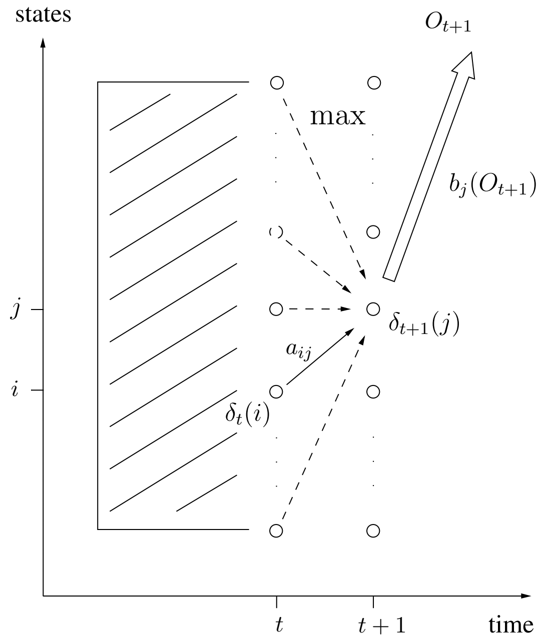


Fig. 5.4. Calculation scheme for the computation of the partial path probabilities $\delta_t(i)$

A much more efficient method for the calculation of this quantity is obtained as a slight variation of the forward algorithm by again applying the considerations about the finite memory of HMMs, which were already pointed out there. The resulting method, for which the computation procedure is summarized in figure 5.3, is part of the *Viterbi algorithm*, which, however, aims at determining the optimal path itself. Therefore, only part of the complete procedure will be described here. The overall algorithm will be presented when treating the model decoding in the following section 5.6.

As an auxiliary quantity for this procedure one defines the maximal probability $\delta_t(i)$ to produce the initial segment of the observation sequence up to time t along an arbitrary path ending in state i:

$$\delta_t(i) = \max_{s_1,s_2,\ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_1, s_2, \ldots s_{t-1}, s_t = i|\lambda) \tag{5.9}$$

The recursive scheme for the calculation of these partial path probabilities $\delta_t(i)$ then works completely analogously to the computation of the forward probabilities $\alpha_t(i)$ in the forward algorithm. The resulting procedure is graphically visualized in figure 5.4. At time $t = 1$ one obtains the initial $\delta_1(i)$ trivially as the product of the start probability π_i of the respective state and the output probability $b_i(O_1)$ of the first element of the observation sequence. An optimization over variable parts of the state sequence is not necessary yet.

$$\delta_1(i) = \pi_i \, b_i(O_1)$$

As the probabilities of the respective optimal partial paths $\delta_t(i)$ can be assumed to be known for all past times according to the induction principle, merely the optimal probabilities $\delta_{t+1}(i)$ of the paths extended by one element need to be computed. Therefore, all possible predecessors i of the current state j are considered. The maximal probability is determined for continuing the paths ending in one of the predecessor states up to the current state according to the transition probability a_{ij}. Finally, the current observation element O_{t+1} needs to be generated according to $b_j(O_{t+1})$.

$$\delta_{t+1}(j) = \max_i \{\delta_t(i) a_{ij}\} \, b_j(O_{t+1})$$

The optimal production probability $P^*(\boldsymbol{O}|\lambda)$ is obtained after finishing the calculations as the maximum over all optimal possibilities $\delta_T(i)$ to reach a certain state i at the final time step T, as with HMMs usually all possible states can be considered as the final states of a valid path[6].

$$P^*(\boldsymbol{O}|\lambda) = P(\boldsymbol{O}, s^*|\lambda) = \max_i \delta_T(i)$$

Thus one obtains analogously to the forward algorithm a calculation scheme for determining the quality of a model λ depending on example data $\boldsymbol{O}$. In contrast to the production probability therein only the optimal possibility for generating the observation sequence is taken into account. The optimal state sequence $\boldsymbol{s}^*$ can, however, not be determined based on the computation steps alone, which were presented so far.

Especially in the field of speech recognition the probability $P^*(\boldsymbol{O}|\lambda)$ of the optimal path is frequently used instead of the total production probability $P(\boldsymbol{O}|\lambda)$. This approach can be justified by the observation that the probability for generating

[6] If specially marked end states should be used (cf. e.g. [52, p. 51]), the maximization needs to be restricted to the appropriate set. Alternatively also additional termination probabilities can be considered when computing the final path probabilities (cf. e.g. [95, p. 150]).

the data $\boldsymbol{O}$ along the optimal path $\boldsymbol{s}^*$ will dominate the summation over all possible paths, which is necessary for the calculation of $P(\boldsymbol{O}|\lambda)$, and, therefore, will be highly correlated with that quantity [157]. Additionally, the calculation of the partial path probabilities $\delta_t(i)$ in practice is considerably more efficient compared to the forward probabilities. As will be explained in more detail in chapter 7, when carrying out computations for HMMs the probabilities are usually transformed into the logarithmic domain. The multiplications necessary for determining the $\delta_t(i)$ are then transformed into summations. The maximizations are not affected by such a monotonic mapping. The summation necessary for the computation of the forward probabilities, however, can only be realized with considerable computational effort in the logarithmic domain.

5.6 Decoding

If the individual states of a larger model can be associated with certain meaningful segments of the modeled data, the consideration of a global quality measure, as e.g. the production probability, is no longer sufficient for the analysis. It is rather necessary to uncover the internal processes during the generation of the observation sequence. As this is in principle, however, possible along every arbitrary state sequence, such an inference is only possible in the probabilistic sense. Therefore, one determines that state sequence $\boldsymbol{s}^*$ that for a given model produces the observations with maximal posterior probability.

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}} P(\boldsymbol{s}|\boldsymbol{O}, \lambda) \tag{5.10}$$

The posterior probability of a certain state sequence $\boldsymbol{s}$ on the right hand side of equation 5.10 can be rewritten as follows by applying Bayes' rule:

$$P(\boldsymbol{s}|\boldsymbol{O}, \lambda) = \frac{P(\boldsymbol{O}, \boldsymbol{s}|\lambda)}{P(\boldsymbol{O}|\lambda)} \tag{5.11}$$

For a maximization of this quantity the contribution of the production probability $P(\boldsymbol{O}|\lambda)$ is, however, irrelevant, as it is constant for a fixed model and a given observation sequence. By exploiting this fact equation (5.10) can be simplified and the optimal state sequence can be determined as follows:

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}} P(\boldsymbol{s}|\boldsymbol{O}, \lambda) = \underset{\boldsymbol{s}}{\operatorname{argmax}} P(\boldsymbol{O}, \boldsymbol{s}|\lambda)$$

This optimal path $\boldsymbol{s}^*$ corresponds exactly to that state sequence, for which the maximal probability $P^*(\boldsymbol{O}|\lambda)$ of generating the observation sequence is obtained. Thus for determining $\boldsymbol{s}^*$ one could in principle use the "brute force" approach for computing the optimal production probability, which was outlined in the previous section 5.5.2. After enumerating all possible state sequences of length T and calculating the respective individual production probabilities using equation (5.6) one selects the state sequence $\boldsymbol{s}^*$ with maximal probability. As this method exhibits an exponential

```
Let $\delta_t(i) = \max_{s_1,s_2,\ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_1, s_2, \ldots s_{t-1}, s_t = i|\lambda)$
  1. **Initialization**
     $\delta_1(i) := \pi_i b_i(O_1)$                                    $\psi_1(i) := 0$
  2. **Recursion**
     for all times $t$, $t = 1 \ldots T-1$:
     $\delta_{t+1}(j) := \max_i \{\delta_t(i) a_{ij}\} \, b_j(O_{t+1})$        $\psi_{t+1}(j) := \operatorname{argmax}_i \{\delta_t(i) a_{ij}\}$
  3. **Termination**
     $P^*(\boldsymbol{O}|\lambda) = P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda) = \max_i \delta_T(i)$
     $s_T^* := \operatorname{argmax}_j \delta_T(j)$
  4. **Back-Tracking of the Optimal Path**
     for all times $t$, $t = T-1 \ldots 1$:
     $s_t^* = \psi_{t+1}(s_{t+1}^*)$
```

Fig. 5.5. Viterbi algorithm for computing the optimal state sequence $\boldsymbol{s}^*$, which maximizes the production probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ of an observation sequence $\boldsymbol{O}$ for a given model λ.

complexity $O(TN^T)$ in the length of the state sequence, however, it is not suitable for practical applications. Rather the method for efficiently calculating $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$, which was described in section 5.5.2, is extended such that not only the optimal score but also the associated state sequence $\boldsymbol{s}^*$ can be determined.

Viterbi Algorithm

The efficient computation of the optimal state sequence $\boldsymbol{s}^*$ by means of the so-called *Viterbi algorithm* uses a recursive procedure, which is quite similar to the forward algorithm and which also exploits the Markov property. As already explained in section 5.5.2, one defines probabilities $\delta_t(i)$ for partially optimal paths, which generate the initial segment of the observation sequence up to O_t with maximal probability and end in state i.

$$\delta_t(i) = \max_{s_1,s_2,\ldots s_{t-1}} P(O_1, O_2, \ldots O_t, s_1, s_2, \ldots s_{t-1}, s_t = i|\lambda)$$

The computation scheme for the $\delta_t(i)$ largely corresponds to the one for determining the forward variables $\alpha_t(i)$. The only difference is that instead of a summation over the probabilities associated with the predecessor states a maximization is performed. The resulting algorithm for the computation of the optimal path is presented in figure 5.5.

In a similar way as in dynamic time warping, which by a comparable scheme computes an optimal path for mapping two patterns onto each other, (cf. e.g. [95, pp. 71–78]), every decision in the computation of the $\delta_t(i)$ is only locally optimal. The globally optimal probability of the optimal state sequence $\boldsymbol{s}^*$ is only known after the evaluation of the $\delta_T(i)$, i.e. after the observation sequence has been considered in its entire length. That state, that maximizes $\delta_T(i)$ denotes the end of the optimal state sequence. The remaining members of the sequence now need to be determined based on the decisions made during the computations of the $\delta_t(i)$. Therefore, one defines so-called "backward pointers" $\psi_t(j)$ along the partial paths, which for every corresponding $\delta_t(j)$ store the associated optimal predecessor state.

$$\psi_t(j) = \underset{i}{\operatorname{argmax}}\, \delta_{t-1}(i) a_{ij}$$

Starting from

$$s_T^* = \underset{i}{\operatorname{argmax}}\, \delta_T(i),$$

the optimal path can then be reconstructed recursively according to

$$s_t^* = \psi_{t+1}(s_{t+1}^*)$$

in reversed chronological order[7].

For practical applications this reversing of the direction of the order of calculations means a significant limitation. The optimal state sequence can, therefore, only be determined *after* the end of the observation sequence is reached, i.e. only after all the data to be analyzed is known in its entirety. Especially for interactive systems it is, however, frequently desirable to be able to generate feedback to the user in the form of partial results — i.e. after a prefix of the optimal state sequence has been decided for — already while the calculations are still ongoing. But such *incremental* decoding methods can in general find sub-optimal solutions only (see also page 213).

5.7 Parameter Estimation

For all application areas of HMMs ideally models should be used, that resemble the statistical properties of certain data as closely as possible. Unfortunately, as already pointed out in section 5.3, no method is known that for a given sample set is able to create a model, which is optimal in some respect. If, however, by applying know-how of experts one succeeds in defining a suitable model structure — i.e. the number of states and the type of output distributions — and useful initial estimates of the model's free parameters, an iterative optimization of the model with respect to the data considered can be performed. This step-by-step improvement of the model parameters up to some quality which is sufficient for the respective task is usually called *training* of the model.

5.7.1 Foundations

The methods presented in the following mainly differ with respect to the measure used for judging the modeling quality of a certain model. In the so-called Baum-Welch algorithm this is achieved by using the production probability $P(\boldsymbol{O}|\lambda)$. In

[7] The decision for the optimal predecessor state is in general not unique. Multiple sequences $\boldsymbol{s}_k^*$ with identical scores maximizing equation (5.6) might exist. In practical applications, therefore, a rule for breaking up such ambiguities is necessary, which might e.g. be a preference for states with lower indices.

Viterbi training only the probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ of the respective optimal state sequence is considered instead.

All these methods are based on the principle that the parameters of a given model λ are subject to a growth transformation, i.e. the model parameters are modified such that the score $P(\ldots|\hat{\lambda})$ of the changed model $\hat{\lambda}$ exceeds the one achieved for the original model. It is, however, possible that a certain model represents a fixed point of the respective optimization method, so that an improvement of the quality measure is no longer possible and the respective score remains unchanged[8]. In general the parameter estimation methods, therefore, guarantee a monotonic increase of the modeling quality only:

$$P(\ldots|\hat{\lambda}) \geq P(\ldots|\lambda)$$

Considered intuitively the parameter estimation methods mentioned are based on the idea to “observe” the actions of the model during the generation of an observation sequence. The original state transition and output probabilities are then simply replaced by the relative frequencies of the respective events. But due to the statistical nature of HMMs the inference of the internal processes is only possible in the probabilistic sense — as already pointed out elsewhere. It is, however, possible to determine the *expected* number of relevant events — i.e. of state transitions and emissions — depending on the given model and the observation sequence considered. When for the sake of simplicity only discrete models are considered[9] and no separate estimation formulas for the start probabilities are given[10], the updated model parameters can in principle be determined as follows:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from } i \text{ to } j}{\text{expected number of transitions out of state } i}$$

$$\hat{b}_i(o_k) = \frac{\text{expected number of outputs of } o_k \text{ in state } i}{\text{total number of outputs in state } i}$$

In order to be able to infer the expected state transitions or outputs of the model, respectively, it is necessary to determine the probability that at a given time t the model was in a certain state i. This probability we want to be referring to as the *state probability* in the following.

For its computation two fundamentally different possibilities exist. If the probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ to create the observation sequence along the optimal path only is considered, it can be verified directly, whether or not a certain state i was present at time t. The state probability $P^*(S_t = i|\boldsymbol{O}, \lambda)$, therefore, takes on the values zero and one only. It can be described by a characteristic function that operates on the optimal state sequence $\boldsymbol{s}^*$:

[8] In practice this means that the chosen quality measure does not change any more within the scope of the available computational accuracy.

[9] Of course the optimization of general continuous mixture densities is possible with all training methods. However, it always represents the most challenging part of the procedure.

[10] The updated start probabilities $\hat{\pi}_i$ can be considered as a special case of the transition probabilities $\hat{a}_{ij}$ and can, therefore, be computed analogously.

$$P^*(S_t = i|\boldsymbol{O}, \lambda) = \chi_t(i) = \begin{cases} 1 & \text{if} \quad s_t^* = i \text{ and } \boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}} P(\boldsymbol{s}, \boldsymbol{O}|\lambda) \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

However, if the quality measure is the production probability $P(\boldsymbol{O}|\lambda)$, for which all paths through the model are considered, the computation of the posterior probability of a certain state at a given time becomes somewhat more demanding.

Forward-Backward Algorithm

In order to compute the posterior probability $P(S_t = i|\boldsymbol{O}, \lambda)$ of a state i at time t for a given observation sequence $\boldsymbol{O}$ and a known model λ of course a "brute force" approach could be used. However, with the forward variables $\alpha_t(i)$ we already got to know quantities in section 5.5, which to a limited extent make predictions about the presence of a state at a given time. Merely the probability is missing for completing the rest of the partial observation sequence from the current state onward.

This quantity is referred to as *backward variable*. It represents the probability for generating the partial observation sequence $O_{t+1}, O_{t+2}, \ldots O_T$ from time $t+1$ onward starting at state j and given the model λ.

$$\beta_t(j) = P(O_{t+1}, O_{t+2}, \ldots O_T | s_t = j, \lambda) \tag{5.13}$$

It can be efficiently computed with the counterpart of the forward algorithm — the so-called *backward algorithm* — and represents the companion piece to the forward variable $\alpha_t(i)$. Both algorithms taken together are frequently considered as a coherent unit in the literature and are referred to as the *forward-backward algorithm*.

In order to compute the desired state probability $P(S_t = i|\boldsymbol{O}, \lambda)$ on the basis of the forward and backward variables let us first rewrite this expression using Bayes' rule as follows:

$$P(S_t = i|\boldsymbol{O}, \lambda) = \frac{P(S_t = i, \boldsymbol{O}|\lambda)}{P(\boldsymbol{O}|\lambda)} \tag{5.14}$$

The production probability $P(\boldsymbol{O}|\lambda)$ can be computed by means of the forward algorithm. The numerator of the right hand side of equation (5.14) directly corresponds to the product of the respective forward and backward variables (cf. equations (5.7) and (5.13)):

$$\begin{aligned} P(S_t = i, \boldsymbol{O}|\lambda) &= P(O_1, O_2, \ldots O_t, S_t = i|\lambda) P(O_{t+1}, O_{t+2}, \ldots O_T | S_t = i, \lambda) = \\ &= \alpha_t(i)\beta_t(i) \end{aligned}$$

The posterior probability of the state i at time t, which is usually denoted as $\gamma_t(i)$, can, therefore, be calculated as follows:

$$\gamma_t(i) = P(S_t = i|\boldsymbol{O}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{O}|\lambda)} \tag{5.15}$$

The evaluation of the backward variable $\beta_t(i)$ — as already indicated by its name — almost corresponds to a mirrored version of the forward algorithm. The anchoring

```
Let
$\alpha_t(i) = P(O_1, O_2, \ldots O_t, s_t = i|\lambda)$    $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots O_T|s_t = i, \lambda)$
1. Initialization
   $\alpha_1(i) := \pi_i b_i(O_1)$                              $\beta_T(i) := 1$
2. Recursion
   for all times $t$, $t = 1 \ldots T-1$:                  respectively $t = T-1 \ldots 1$:
   $\alpha_{t+1}(j) := \sum_i \{\alpha_t(i) a_{ij}\} b_j(O_{t+1})$    $\beta_t(i) := \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$
3. Termination
   $P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$         $P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i)$
```

Fig. 5.6. Forward-backward algorithm for the joint computation of the forward and backward variables, respectively.

of the inductive computation procedure takes place at time t, which is the end of the time interval defined by the observation sequence. Trivially, the probability for not generating any further observations from time T onward is equal to one.

$$\beta_T(i) := 1$$

If according to the induction principle the $\beta_{t+1}(j)$ are assumed to be known for all future times $t+1$, a recursive computation rule for the backward variables $\beta_t(i)$ can be derived by considering all possible continuations of a state sequence starting from the current state i.

$$\beta_t(i) := \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{5.16}$$

At the termination of the procedure one obtains — similarly to the forward algorithm — the production probability $P(\boldsymbol{O}|\lambda)$ by summation over all backward variables $\beta_1(i)$ at time $t = 1$ taking into account the start probabilities and the generation of the first observation element O_1 in the respective state[11]:

$$P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i)$$

In figure 5.6 both algorithms are presented together in order to illustrate the extraordinary symmetry of the computational procedures. The computation scheme resulting for the backward variables is graphically visualized in figure 5.7.

5.7.2 Training Methods

The probabilistic possibility for defining the state probability via $\gamma_t(i)$ or its deterministic variant $\chi_t(i)$, respectively, form the basis for the training methods treated

[11] The fact that the production probability $P(\boldsymbol{O}|\lambda)$ may both be computed via the forward and the backward algorithm can in practice be exploited to check the computations for consistency and accuracy.

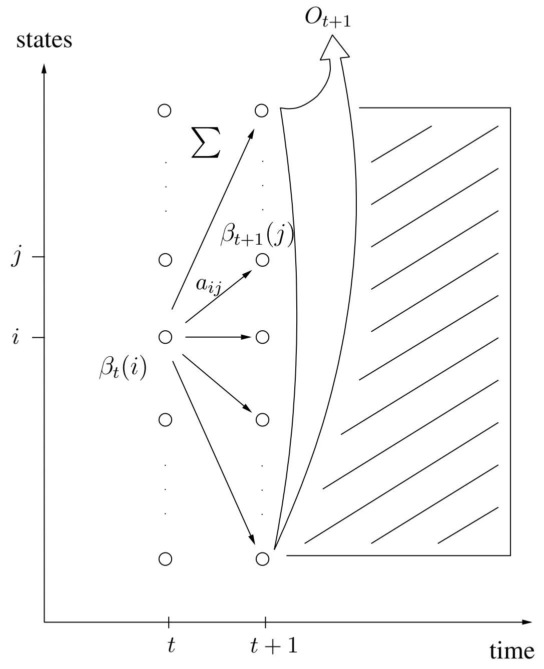


Fig. 5.7. Computation scheme for determining the backward variables $\beta_t(i)$ via the backward algorithm

in the following. By means of the chronological mapping between model states and elements of the observation sequence not only state transition probabilities can be estimated but also the parameters of the state specific output distributions. In discrete models thus also simple estimation formulas can be derived. The increased complexity when using mixture densities for describing the output distributions is due to the fact, that even when considered in isolation no closed form solutions exist for estimating the parameters of such models. In fact *iterative* optimization strategies have to be applied, which, however can be combined with the training methods applied for HMMs as a whole.

Baum-Welch Algorithm

The most widely used method for the optimization of HMMs is given by the so-called *Baum-Welch algorithm*. As the optimization criterion it uses the total production probability $P(\boldsymbol{O}|\lambda)$. Thus the algorithm improves a given model λ depending on certain example data $\boldsymbol{O}$ in such a way that the optimized model generates the training set with equal or greater probability:

$$P(\boldsymbol{O}|\hat{\lambda}) \geq P(\boldsymbol{O}|\lambda)$$

The equality of those both expressions is valid only if a local maximum with respect to the optimization in the space of all possible models was already reached with the parameters of the original model.

In this method all model parameters are replaced by their conditional expected values with respect to the given original model λ and the training data $\boldsymbol{O}$. The Baum-Welch algorithm, therefore, represents a variant of the EM algorithm, which in general optimizes parameters of multi-stage stochastic processes with hidden variables following the maximum likelihood criterion ([42], see also section 4.4 page 55).

The foundations of the algorithm are represented by some quantities, which based on the forward and backward variables allow in the statistical sense inferences about the internal processes of the model λ when generating certain given data $\boldsymbol{O}$. Besides the posterior probability $P(S_t = i|\boldsymbol{O}, \lambda)$ for the occurrence of a state i at time t, which is called $\gamma_t(i)$, posterior probabilities for state transitions and — for continuous models on the basis of mixture densities — also probabilities for the selection of individual mixture components M_t at a given time are required.

The posterior probability $P(S_t = i, S_{t+1} = j|\boldsymbol{O}, \lambda)$ of a transition from state i to state j at time t, which is usually denoted as $\gamma_t(i, j)$ in the literature[12], can in the style of equation (5.15) be calculated as follows:

$$\begin{aligned}\gamma_t(i,j) &= P(S_t = i, S_{t+1} = j|\boldsymbol{O}, \lambda)\\ &= \frac{P(S_t = i, S_{t+1} = j, \boldsymbol{O}|\lambda)}{P(\boldsymbol{O}|\lambda)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(\boldsymbol{O}|\lambda)}\end{aligned}$$

The numerator of the final expression for $\gamma_t(i, j)$ represents the probability for generating the observation sequence with the restriction that a transition from i to state j occurs at time t. The merging of computation paths within the model is graphically visualized in figure 5.8.

In the literature the state probability is frequently defined on the basis of the posterior probabilities for state transitions. The probability $\gamma_t(i)$ of some state i occurring at all at time t can be obtained as the marginal distribution of $\gamma_t(i, j)$ by summation over all possible state successors.

$$\gamma_t(i) = P(S_t = i|\boldsymbol{O}, \lambda) = \sum_{j=1}^{N} P(S_t = i, S_{t+1} = j|\boldsymbol{O}, \lambda) = \sum_{j=1}^{N} \gamma_t(i,j)$$

However, this relationship can only be exploited for times $t < T$, as $\gamma_t(i, j)$ is only defined for those points in time. In practice $\gamma_t(i)$ will, therefore, be computed directly via equation (5.15). Otherwise a suitable extension of the definition of $\gamma_t(i, j)$ will be required.

On the basis of the γ_t all updated parameters $\hat{\lambda}$ for HMMs with discrete output distributions can be computed. The number of state transitions from i to j that can be expected in the statistical average is obtained as the sum of the individual transition

[12] Due to the close connection in their meaning and in order not to unnecessarily make the notation any more complex, the state probability is denoted with a single argument as $\gamma_t(i)$ and the state transition probability with two arguments as $\gamma_t(i, j)$.

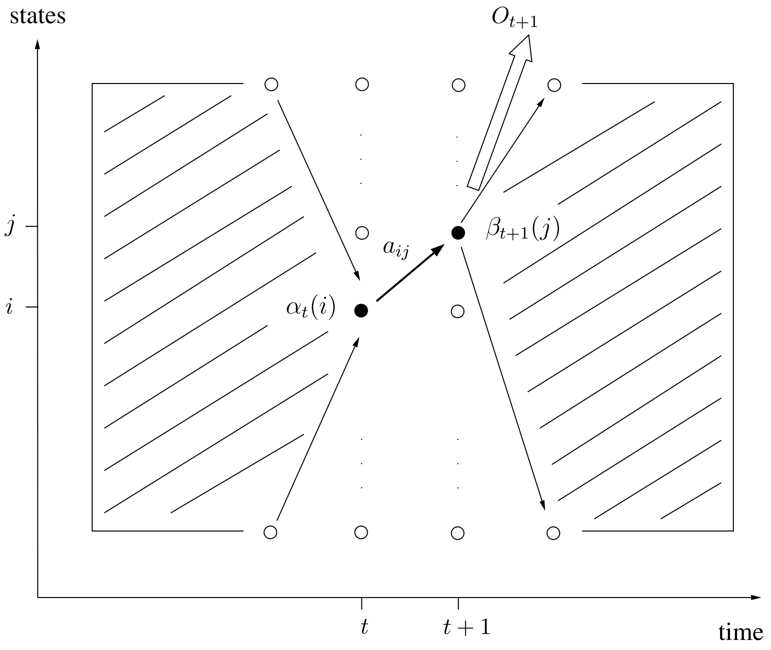


Fig. 5.8. Computation scheme for determining the $\gamma_t(i,j)$

probabilities $\gamma_t(i,j)$ over all points in time $t = 1, 2, \ldots T-1$ to be considered[13]. When normalizing this quantity on the expected total number of transitions out of state i one obtains the improved estimates $\hat{a}_{ij}$ for the transition probabilities of the model:

$$\hat{a}_{ij} = \frac{\sum\limits_{t=1}^{T-1} P(S_t = i, S_{t+1} = j|\boldsymbol{O}, \lambda)}{\sum\limits_{t=1}^{T-1} P(S_t = i|\boldsymbol{O}, \lambda)} = \frac{\sum\limits_{t=1}^{T-1} \gamma_t(i,j)}{\sum\limits_{t=1}^{T-1} \gamma_t(i)} \tag{5.17}$$

As a special case of the transition probabilities one obtains the following simple equation for determining improved start probabilities:

$$\hat{\pi}_i = P(S_1 = i|\boldsymbol{O}, \lambda) = \gamma_1(i) \tag{5.18}$$

The improved discrete output probabilities can in general be obtained in an analogous way. First, the expected number of outputs of a specific symbol o_k in state j is calculated by checking for a match between o_k and the corresponding element of the observation sequence in addition to the occurrence of the respective state. When

[13] As HMMs according to common opinion do not perform a state transition into a specially marked end state when reaching the end of the observation sequence at time T, here the restriction to all prior points in time is necessary.

normalizing this quantity to the expected total number of emissions, which are generated by state j, one obtains estimates $\hat{b}_j(o_k)$ of the discrete output probabilities according to:

$$\hat{b}_j(o_k) = \frac{\sum\limits_{t=1}^{T} P(S_t = j, O_t = o_k | \boldsymbol{O}, \lambda)}{\sum\limits_{t=1}^{T} P(S_t = j | \boldsymbol{O}, \lambda)} = \frac{\sum\limits_{t\,:\,O_t = o_k} P(S_t = j | \boldsymbol{O}, \lambda)}{\sum\limits_{t=1}^{T} P(S_t = j | \boldsymbol{O}, \lambda)} = \frac{\sum\limits_{t\,:\,O_t = o_k} \gamma_t(j)}{\sum\limits_{t=1}^{T} \gamma_t(j)} \tag{5.19}$$

As it can be unambiguously determined whether a certain observation symbol o_k was present at a given time t or not, the probabilities $P(S_t = j, O_t = o_k | \boldsymbol{O}, \lambda)$ contribute positive portions to the sum in the numerator of equation (5.19) only for those points in time. All other terms vanish and the summation can be restricted accordingly and thus be simplified[14].

When modeling output distributions by mixture densities it is necessary, in contrast, to optimize the parameters of the mixtures themselves — i.e. of the individual normal distributions — and the associated mixture weights. If the evaluation of the mixture densities is viewed as some sort of quantization, which realizes a probabilistic mapping of the observations onto abstract but not yet meaningful symbolic units, it becomes immediately evident that the computation of updated mixture weights can be performed in analogy to the procedure for discrete output probabilities.

Therefore one defines, similarly to the state probability, an auxiliary quantity $\xi_t(j, k)$ which represents the probability of selecting in state j the k-th mixture component at time[15] t for generating the continuous observation O_t:

$$\xi_t(j, k) = P(S_t = j, M_t = k | \boldsymbol{O}, \lambda) = \frac{\sum\limits_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}\, c_{jk}\, g_{jk}(O_t)\, \beta_t(j)}{P(\boldsymbol{O}|\lambda)} \tag{5.20}$$

Based on equation (5.20) a formula for the estimation of the mixture weights can be derived as follows:

$$\hat{c}_{jk} = \frac{\sum\limits_{t=1}^{T} P(S_t = j, M_t = k | \boldsymbol{O}, \lambda)}{\sum\limits_{t=1}^{T} P(S_t = j | \boldsymbol{O}, \lambda)} = \frac{\sum\limits_{t=1}^{T} \xi_t(j, k)}{\sum\limits_{t=1}^{T} \gamma_t(j)} \tag{5.21}$$

The updating of the parameters of the k-th mixture component $g_{jk}(\boldsymbol{x})$ of state j — i.e. of the mean vector $\boldsymbol{\mu}_{jk}$ and the covariance matrix $\boldsymbol{C}_{jk}$ — is performed in analogy to the estimation formulas for mixture density models presented in section 4.4

[14] As for a given time t the symbol o_k was either present in the observation sequence or not, the probability $P(S_t = j, O_t = o_k | \boldsymbol{O}, \lambda)$ either takes on the value zero or is equal to $P(S_t = j | \boldsymbol{O}, \lambda)$ which is simply $\gamma_t(j)$.

[15] For time $t = 1$ in equation (5.20) the term $\sum\limits_{i=1}^{N} \alpha_{t-1}(i)\, a_{ij}$ needs to be replaced by the respective start probability π_j of the associated state.

page 55. The posterior probability $P(\omega_i|\boldsymbol{x}, \theta)$ of the individual pattern class merely needs to be replaced by the term $\xi_t(j,k)$. In contrast to the estimation equations (3.8) and (3.9) for individual normal distributions presented in section 3.6 page 41 observations are not incorporated deterministically but probabilistically into the computation process. The probability that a certain observation vector $\boldsymbol{x}_t$ is used for the estimation of the parameters of $g_{jk}(\boldsymbol{x})$ exactly corresponds to the probability $P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)$ of selecting the respective mixture component for its generation at the time in question.

Therefore, one obtains the following formulas for the computation of updated mean vectors $\hat{\boldsymbol{\mu}}_{jk}$ and covariance matrices $\hat{\boldsymbol{C}}_{jk}$ of the individual component densities:

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)\, \boldsymbol{x}_t}{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)} = \frac{\sum_{t=1}^{T} \xi_t(j,k)\, \boldsymbol{x}_t}{\sum_{t=1}^{T} \xi_t(j,k)} \tag{5.22}$$

$$\begin{aligned}\hat{\boldsymbol{C}}_{jk} &= \frac{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)\, (\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jk})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jk})^T}{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)} \\ &= \frac{\sum_{t=1}^{T} \xi_t(j,k)\, (\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jk})(\boldsymbol{x}_t - \hat{\boldsymbol{\mu}}_{jk})^T}{\sum_{t=1}^{T} \xi_t(j,k)}\end{aligned} \tag{5.23}$$

The estimation formula for the covariance matrix $\boldsymbol{C}_{jk}$ of the mixture component can be rewritten by exploiting the relationship given by equation (3.4) on page 38 in order to obtain the following computation scheme, which is simpler in practice, because it only requires a single pass through the data:

$$\begin{aligned}\hat{\boldsymbol{C}}_{jk} &= \frac{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)\, \boldsymbol{x}_t\boldsymbol{x}_t^T}{\sum_{t=1}^{T} P(S_t = j, M_t = k|\boldsymbol{O}, \lambda)} - \hat{\boldsymbol{\mu}}_{jk}\hat{\boldsymbol{\mu}}_{jk}^T \\ &= \frac{\sum_{t=1}^{T} \xi_t(j,k)\, \boldsymbol{x}_t\boldsymbol{x}_t^T}{\sum_{t=1}^{T} \xi_t(j,k)} - \hat{\boldsymbol{\mu}}_{jk}\hat{\boldsymbol{\mu}}_{jk}^T\end{aligned} \tag{5.24}$$

In semi-continuous models all states share a set of component densities $g_k(\boldsymbol{x})$ for building the mixture models, as introduced in section 5.2. As the mixture weights are still state specific, the respective estimation formula remains unchanged. In order to estimate the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{C}_k$ used jointly via the shared component densities it

Let

$$\gamma_t(i) = P(S_t = i|\boldsymbol{O}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{O}|\lambda)}$$

$$\gamma_t(i,j) = P(S_t = i, S_{t+1} = j|\boldsymbol{O}, \lambda) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{P(\boldsymbol{O}|\lambda)}$$

$$\xi_t(j,k) = P(S_t = j, M_t = k|\boldsymbol{O}, \lambda) = \frac{\sum\limits_{i=1}^{N} \alpha_t(i)\, a_{ij}\, c_{jk}\, g_{jk}(O_t)\, \beta_t(j)}{P(\boldsymbol{O}|\lambda)}$$

1. **Initialization**
   Choose a suitable initial model $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ with initial estimates
   - π_i for starting and
   - a_{ij} for transition probabilities as well as for
   - mixture weights c_{jk} and
   - component densities $g_{jk}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{jk}, \boldsymbol{C}_{jk})$ for the definition of output probability density functions $b_{jk}(\boldsymbol{x}) = \sum\limits_k c_{jk}\, g_{jk}(\boldsymbol{x})$.
2. **Optimization**
   Compute updated estimates $\hat{\lambda} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{B}})$ for all model parameters:

$$\hat{a}_{ij} = \frac{\sum\limits_{t=1}^{T-1} \gamma_t(i,j)}{\sum\limits_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{\pi}_i = \gamma_1(i)$$

$$\hat{c}_{jk} = \frac{\sum\limits_{t=1}^{T} \xi_t(j,k)}{\sum\limits_{t=1}^{T} \gamma_t(j)}$$

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum\limits_{t=1}^{T} \xi_t(j,k)\, \boldsymbol{x}_t}{\sum\limits_{t=1}^{T} \xi_t(j,k)} \qquad \hat{\boldsymbol{C}}_{jk} = \frac{\sum\limits_{t=1}^{T} \xi_t(j,k)\, \boldsymbol{x}_t \boldsymbol{x}_t^T}{\sum\limits_{t=1}^{T} \xi_t(j,k)} - \hat{\boldsymbol{\mu}}_{jk}\hat{\boldsymbol{\mu}}_{jk}^T$$

3. **Termination**
   if the quality measure $P(\boldsymbol{O}|\hat{\lambda})$ was considerably improved by the updated model $\hat{\lambda}$ with respect to λ
      let $\lambda \leftarrow \hat{\lambda}$ and continue with step 2
   otherwise Stop!

Fig. 5.9. Baum-Welch algorithm for estimating the parameters of general continuous HMMs. For modifications necessary for discrete or semi-continuous models see the text.

has to be taken into account that the probability for the selection of a certain density at a given time is independent of some concrete state in semi-continuous models. It is obtained from equation (5.20) as the marginal distribution of $\xi_t(j,k)$ by summation over all possible states:

$$P(M_t = k|\boldsymbol{O},\lambda) \quad = \quad \xi_t(k) \quad = \quad \sum_j \xi_t(j,k)$$

If now the probability of the observation vectors is replaced by $\xi_t(k)$ in equations (5.22) and (5.24) one obtains the following estimation formulas for $\boldsymbol{\mu}_k$ and $\boldsymbol{C}_k$ in semi-continuous models:

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum\limits_{t=1}^{T} P(M_t = k|\boldsymbol{O},\lambda)\, \boldsymbol{x}_t}{\sum\limits_{t=1}^{T} P(M_t = k|\boldsymbol{O},\lambda)} = \frac{\sum\limits_{t=1}^{T} \xi_t(k)\, \boldsymbol{x}_t}{\sum\limits_{t=1}^{T} \xi_t(k)} \tag{5.25}$$

$$\hat{\boldsymbol{C}}_k = \frac{\sum\limits_{t=1}^{T} P(M_t = k|\boldsymbol{O},\lambda)\, \boldsymbol{x}_t\boldsymbol{x}_t^T}{\sum\limits_{t=1}^{T} P(M_t = k|\boldsymbol{O},\lambda)} - \hat{\boldsymbol{\mu}}_k\hat{\boldsymbol{\mu}}_k^T = \frac{\sum\limits_{t=1}^{T} \xi_t(k)\, \boldsymbol{x}_t\boldsymbol{x}_t^T}{\sum\limits_{t=1}^{T} \xi_t(k)} - \hat{\boldsymbol{\mu}}_k\hat{\boldsymbol{\mu}}_k^T \tag{5.26}$$

The updating of the model parameters according to the estimation equations (5.17), (5.18), and (5.19) for discrete models, by applying equations (5.17), (5.18), (5.21), (5.22), and (5.24) for models with continuous output distributions, and by means of equations (5.17), (5.18), (5.21), (5.25), and (5.26) for semi-continuous models corresponds to one step in an iteratively optimizing training process. Starting from an initially given model λ^0 the updating of the parameters needs to be repeated until the resulting model reaches a sufficient descriptive quality or no further improvements are to be expected any more. For "fully" continuous models the complete algorithm is put together in figure 5.9.

Viterbi Training

In contrast to the Baum-Welch algorithm in the so-called *Viterbi training* only the probability $P^*(\boldsymbol{O}|\lambda) = P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ that the observation sequence is generated along the best-scoring path $\boldsymbol{s}^*$ is optimized during the training process. It can be shown that the method realizes a growth transformation starting from an existing model λ, so that the modified model $\hat{\lambda}$ achieves a higher or at least equal probability for the optimal path:

$$P^*(\boldsymbol{O}|\hat{\lambda}) \geq P^*(\boldsymbol{O}|\lambda)$$

The method follows the intuitive understanding of the principle of HMM training proceeding in two working phases outlined at the beginning of the chapter. First, the optimal state sequence $\boldsymbol{s}^*$ for the training data is computed depending on the given model parameters by means of the Viterbi algorithm. In the second step the estimates

Let

$$\chi_t(i) = \begin{cases} 1 & \text{if} \quad s_t^* = i \text{ and } \boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}}\, P(\boldsymbol{s}, \boldsymbol{O}|\lambda) \\ 0 & \text{otherwise} \end{cases}$$

1. **Initialization**
Choose a suitable initial model $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ with initial estimates of π_i for starting and a_{ij} for transition probabilities as well as discrete output probabilities $b_j(o_k)$.
2. **Segmentation**
Calculate by means of the Viterbi algorithm (see figure 5.5 page 75) the optimal state sequence $\boldsymbol{s}^*$ for generating the data $\boldsymbol{O}$ given the model λ.
3. **Optimization**
Compute updated estimates $\hat{\lambda} = (\hat{\boldsymbol{\pi}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{B}})$ for all model parameters (except $\boldsymbol{\pi}$, explanations see text):

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \chi_t(i)\, \chi_{t+1}(j)}{\sum_{t=1}^{T-1} \chi_t(i)} \qquad \hat{b}_j(o_k) = \frac{\sum_{t\,:\,O_t=o_k} \chi_t(j)}{\sum_{t=1}^{T} \chi_t(j)}$$

4. **Termination**
if the quality measure $P^*(\boldsymbol{O}|\hat{\lambda})$ was considerably improved by the updated model $\hat{\lambda}$ with respect to λ
let $\lambda \leftarrow \hat{\lambda}$ and continue with step 2
otherwise Stop!

Fig. 5.10. Viterbi training for parameter estimation of discrete HMMs. Extensions for continuous models see text.

for updating the model parameters are determined on the basis of the empirical distributions, which result from the explicit mapping of observations to individual model states along the optimal state sequence.

Formally this mapping can be described via the state probability $\chi_t(i)$ known from equation (5.12), which merely corresponds to the evaluation of a characteristic function on the optimal state sequence $\boldsymbol{s}^*$. When replacing the probabilistic version $\gamma_t(i)$ by the deterministic mapping using $\chi_t(i)$ in the Baum-Welch algorithm one obtains the formulas necessary for computing updated parameters of discrete models by means of the Viterbi training. Estimates for the state transition probabilities, which can principally be obtained by counting state pairs in the optimal state sequences, are obtained as:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} P(S_t = i, S_{t+1} = j|\boldsymbol{s}^*, \boldsymbol{O}, \lambda)}{\sum_{t=1}^{T-1} P(S_t = i|\boldsymbol{s}^*, \boldsymbol{O}, \lambda)} = \frac{\sum_{t=1}^{T-1} \chi_t(i)\chi_{t+1}(j)}{\sum_{t=1}^{T-1} \chi_t(i)} \tag{5.27}$$

Useful estimates for start probabilities are, however, not obtained by means of the Viterbi training. As a special case of equation (5.27) one would obtain a value of one for the start probability $\pi_{s_1^*}$ of the first state s_1^* of the optimal state sequence and zero for all other states. As the start probabilities are hardly of any importance in practical applications, this means no significant limitation of the method.

The discrete output probabilities are directly determined by the empirical distributions. The latter are obtained by simply counting all observation symbols o_k, which are mapped to a certain model state via the optimal state sequence:

$$\hat{b}_j(o_k) = \frac{\sum\limits_{t\,:\,O_t=o_k} P(S_t = j|\boldsymbol{s}^*, \boldsymbol{O}, \lambda)}{\sum\limits_{t=1}^{T} P(S_t = j|\boldsymbol{s}^*, \boldsymbol{O}, \lambda)} = \frac{\sum\limits_{t\,:\,O_t=o_k} \chi_t(j)}{\sum\limits_{t=1}^{T} \chi_t(j)} \tag{5.28}$$

The overall algorithm for the Viterbi training of discrete HMMs is summarized in figure 5.10.

The estimation of continuous mixture models by means of the Viterbi training is, however, considerably more demanding, as no analytical methods are known for deriving optimal parameters of such models from training examples. It is usually assumed, however, that the number M_j of the baseline distributions used remains constant. Nevertheless an estimation of the mixture weights c_{jk} as well as of the parameters of the component densities $g_{jk}(\boldsymbol{x})$ can not be performed directly.

In [231] for this purpose the maximum likelihood criterion for the improvement of the optimal production probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ depending on the model parameters λ is applied. This optimization can be carried out separately for the transition probabilities a_{ij}, as in equation (5.27), and for the output probability densities $b_j(\boldsymbol{x})$. Though one obtains a system of constraint equations, no explicit relationship for computing updated model parameters can be derived. This must rather be achieved in a complex embedded optimization process.

Segmental k-Means

With respect to its theoretical derivation the so-called *segmental k-means algorithm* is identical with the Viterbi training presented in the preceding section [111]. As optimization criterion also the production probability $P(\boldsymbol{O}, \boldsymbol{s}^*|\lambda)$ of the training data along the optimal path through the model is used. In the practical application, however, the procedure offers a solution for the problem of estimating parameters of mixture models on example data only with the embedding of a method for vector quantization — namely the k-means algorithm. Though this practical aspect is mentioned in [111] it is not elaborated as e.g. in [135].

Similar to Viterbi training the method proceeds in two phases. In a first step a segmentation of the training data is generated with the existing model. Subsequently new output probability density functions can be estimated from the resulting mappings between feature vectors and model states without any further reference to the original parameters. The resulting algorithm is summarized in figure 5.11.

As in all training procedures for HMMs suitable initial model parameters need to be chosen before the optimization begins. However, the method can be extended such

Given the number M_j of mixture components to be estimated per model state (frequently $M_j = M$ is chosen identically for all states j)

1. **Initialization**
   Choose a suitable initial model $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$
2. **Segmentation**
   By means of the Viterbi algorithm (see figure 5.5 page 75) compute the optimal state sequence $\boldsymbol{s}^*$ for the generation of the data $\boldsymbol{O}$ given the model λ.
   Compute updated transition probabilities $\hat{a}_{ij}$:
   $$\hat{a}_{ij} = \frac{\sum\limits_{t=1}^{T-1} \chi_t(i)\, \chi_{t+1}(j)}{\sum\limits_{t=1}^{T-1} \chi_t(i)}$$
3. **Estimation**
   For all states j, $0 \leq j \leq N$:
   a) **Cluster Analysis**
      For the partial sample set $\boldsymbol{X}(j)$ compute a vector quantization codebook $Y = \{\boldsymbol{y}_1, \ldots \boldsymbol{y}_{M_j}\}$ and the associated partition $\{R_1, \ldots R_{M_j}\}$, e.g. using the k-means algorithm (see figure 4.3 page 54)
   b) **Updating the Model**
      Compute updated parameters of the output distributions:
      $$\hat{c}_{jk} = \frac{|R_k|}{|\boldsymbol{X}(j)|}$$
      $$\hat{\boldsymbol{\mu}}_{jk} = \boldsymbol{y}_k$$
      $$\hat{\boldsymbol{C}}_{jk} = \frac{1}{|R_k|} \sum_{\boldsymbol{x} \in R_k} \boldsymbol{x}\,\boldsymbol{x}^T - \hat{\boldsymbol{\mu}}_{jk}\, \hat{\boldsymbol{\mu}}_{jk}^T$$
4. **Termination**
   if the quality measure $P^*(\boldsymbol{O}|\hat{\lambda})$ was considerably improved by the updated model $\hat{\lambda}$ with respect to λ
   let $\lambda \leftarrow \hat{\lambda}$ and continue with step 2
   otherwise Stop!

Fig. 5.11. Segmental k-means algorithm for estimating parameters of continuous HMMs.

that also an initialization is possible for HMMs with certain limited model structure. This aspect of the segmental k-means algorithm will be presented in greater detail in section 9.3 page 163.

In the next step the optimal state sequence of the observations considered is computed by means of the Viterbi algorithm. As in the Viterbi training on the basis of this already estimates for the transition probabilities of the updated model can be computed (see equation (5.27)). Furthermore one obtains a mapping between feature vectors $\boldsymbol{x}_t$ and corresponding model states, which can formally be derived from the discrete state probability $\chi_t(i)$ (see equation 5.12 page 78). We want to take together all vectors, which are mapped to a particular state i, to form a partial set $\boldsymbol{X}(i)$ of the considered sample data:

$$\boldsymbol{X}(i) = \{\boldsymbol{x}_t | \chi_t(i) = 1\} = \{\boldsymbol{x}_t | s_t^* = i\}$$

These form the basis for the estimation of new output probability density functions. First a cluster analysis of the respective partial set $\boldsymbol{X}(i)$ is performed using a principally arbitrary method for vector quantization[16]. The parameters of the output distribution of the corresponding model state then result in the same way as with the simple estimation of mixture models (see section 4.4 page 55). As in vector quantization it is, however, necessary to specify the number of desired codebook classes or mixture components, respectively.

When assuming that per model state j exactly one mixture density with M_j individual component densities is used for modeling the output distribution, the mean vectors $\boldsymbol{\mu}_{jk}$ correspond directly to the centroids of the M_j codebook classes, which were determined by the vector quantization procedure for the partial sample set $\boldsymbol{X}(j)$. The covariance matrices are computed as the sample covariances of the feature vectors that were mapped to a specific reproduction vector. The necessary mixture weights correspond to the prior probabilities of the individual codebook classes.

Since in the segmental k-means algorithm the parameters of an HMM including the complex mixture density models are completely newly computed in every step of the optimization on the basis of the example vectors only, the method converges significantly faster than a comparable iterative estimation using the Baum-Welch algorithm.

5.7.3 Multiple Observation Sequences

In general, sample sets that are used for parameter training are subdivided into individual segments — in automatic speech recognition in utterances or turns and in the analysis of biological sequences into protein domains or motifs, respectively. From the view of the HMM formalism these are considered individual observation sequences. In order to be able to estimate model parameters also on such a set of isolated sequences the training procedures do not need to be modified fundamentally. Only the statistics gathered for the updating of the parameters need to be accumulated across all the observation sequences considered. One then obtains modified estimation formulas with an additional outer summation over the observation sequences, which were, however, omitted in the presentation of the individual methods in the preceding sections for the sake of clarity (cf. e.g. [95, pp. 157–158]).

The fundamental principle will now be explained for the estimation of the mean vectors of continuous output probability densities. Therein we assume that for the

[16] For reasons of efficiency it is rather obvious to use the k-means algorithm as vector quantization method, as it achieves good results already within a single pass through the data. However, principally any algorithms for vector quantizer design or the unsupervised estimation of mixture densities could be applied.

training of the models a sample set $\omega = \{\boldsymbol{O}^1, \boldsymbol{O}^2, \ldots \boldsymbol{O}^L\}$ of L individual observation sequences $\boldsymbol{O}^l$ is available. In this case one obtains the updated mean vectors according to:

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum\limits_{l=1}^{L} \sum\limits_{t=1}^{T} \xi_t^l(j,k)\, \boldsymbol{x}_t}{\sum\limits_{l=1}^{L} \sum\limits_{t=1}^{T} \xi_t^l(j,k)}$$

The inner sums in the numerator and denominator, respectively, of this expression correspond to the original estimation formula (5.22) page 84 that considered a single observation sequence only. The mapping probability $\xi_t^l(j,k)$ between feature vectors and mixture components, however, needs to be calculated depending on the l-th observation sequence $\boldsymbol{O}^l$. The accumulated statistics are finally summed over all partial sequences of the sample set.

5.8 Model Variants

Due to their widespread use and their long-lasting development history for hidden Markov models a number of variants in the algorithmic treatment or in the models themselves were proposed. The most important of those aspects shall be outlined shortly in the following. For a detailed treatment of the respective methods the interested reader is referred to the cited specialized literature.

5.8.1 Alternative Algorithms

All recognition systems with implicit segmentation that are based on HMMs in principle use the Viterbi algorithm for decoding. Differences merely result from necessary improvements in efficiency (see section 10.2 page 167) or the incorporation of additional modeling parts as, e.g., statistical language models (cf. chapter 12 page 189).

In contrast for the estimation of model parameters besides the established methods, which were presented in the preceding section, a number of alternative approaches exist. The most well known group of methods applies techniques for discriminative training. Similarly to Viterbi training the goal is therein to improve the probability of the optimal path through the model for given data. However, this is not achieved in isolation. Rather it is attempted to reduce the probability of all competing paths at the same time. In this way a higher discriminative power of the models shall be achieved, which, however, results in a substantially higher computational effort. Mathematically a maximization of the mutual information is performed. Therefore, these methods are frequently found under the topic of maximum mutual information (MMI) in the literature (cf. e.g. [35], [95, pp. 213–214], [97, pp. 150–152]). Closely related to these methods is a technique known as corrective training [8].

5.8.2 Alternative Model Architectures

Besides the "classical" HMM architecture there exists a multitude of variants that by using special modifications attempt to either avoid or at least compensate for the inherent limitations of the models. Especially the improvement of the modeling of duration by HMMs, which is possible rather inadequately with simple transition probabilities only, is the aim of several approaches (cf. e.g. [31, 140], [97, pp. 406–408]). In the main application area of HMMs, however, transition probabilities constitute a rather unimportant modeling aspect compared to output probability densities. For that reason and because of the increased computational effort of alternative duration modeling approaches none of the proposed techniques could establish itself as a standard method yet.

The most well known modification of the classical HMM architecture undoubtedly constitute hybrid systems with a combination of HMMs and artificial neuronal networks (NN) (cf. e.g. [159], [97, p. 458-459]). The neuronal networks, which are incorporated into the model, are used either as a vector quantizer in combination with discrete HMMs [197], as a replacement of the modeling of output distributions on the basis of mixture densities [201], or directly for the estimation of posterior probabilities of individual model states [159]. The combination of the scores obtained such with probabilities derived from the HMM formalism is, however, quite problematic and requires special mapping rules.

Due to the increase in the models' degrees of freedom, which results from the omission of the restrictions to mixture density models, an improved potential can be expected from such hybrid systems. This is also frequently demonstrated exemplarily in the literature. However, for this advantage the poor convergence properties of the training of neuronal networks have to be accepted. Also the optimization of those parameters is usually performed separately from the actual HMM and not in an integrated training process, as with classical model architectures based on mixture densities. As, furthermore, no standardized design strategies exist for hybrid HMM/NN-systems those methods could not establish themselves to date as real or even better alternatives alongside the standard architectures.

5.9 Bibliographical Remarks

Hidden-Markov-Models were named after the Russian mathematician Andrej Andrejewitsch Markov (1856 – 1922) [151]. Books and articles from the mathematical literature mainly treat the theoretical aspects of these models. The primary application area of HMMs, in which they were successfully applied and consequently further developed, is the field of automatic speech recognition. Therefore, the treatment of HMMs within monographs related to the field of pattern recognition is almost always coupled with that complex of topics.

Especially convincing is the treatment of the topic in the monograph by Huang, Acero, & Hon [97]. The core of the model building process is also treated in the partially outdated book by Rabiner & Juang [196] as well as in the monograph by

Jelinek [105]. A nice introduction to the use of HMMs in the domain of bioinformatics is given by Durbin *et al.* [52].

In all the abovementioned works descriptions of the algorithms relevant for HMMs can be found. A summary of those is also offered by the classical article by Rabiner [195], which had a significant influence on the HMM literature and the notation used therein.

The *Viterbi algorithm* was named after its inventor and goes back to his works in the field of coding theory [226]. An in-depth early presentation and analysis of the method is given by [76]. Later descriptions can be, e.g., found in [95, pp. 151–152] or [97, pp. 387–389]. The most widely used method for estimating parameters of HMM is the *Baum-Welch algorithm*, which was developed by Baum and colleagues [12]. Though the algorithm surely was named after its inventors there exists no accessible publication with Welch as co-author. The method represents a variant of the *EM algorithm* [42]. The relationship between the two algorithms is, e.g., presented in [97, pp. 389–393]. A proof of the convergence of the Baum-Welch algorithm can besides [12] also be found in, e.g. [95, pp. 158–164]. The principle method is furthermore described in [195, pp. 342–348], [95, pp. 152–158], and [105, pp. 149–161]. The *Viterbi training* as an alternative method for estimating parameters of HMMs is, e.g., described in [135]. Especially for continuous models the method was elaborated in [231]. Principally comparable to this method is the segmental k-means algorithm, which was developed by Juang & Rabiner [111]. A short description of the method can also be found in [196, p. 427]

6

n-Gram Models

A *statistical language model* in its most general form defines a probability distribution over a set of symbol sequences from some finite inventory. Those methods are referred to as "language models" because their origin, their development, and their spreading in use is closely related to the statistical modeling of texts as well as to the restriction of possible sequences of word hypotheses in automatic speech recognition.

An especially simple yet very powerful concept for the formal description of statistical language models is formed by their representation using Markov chains. The most widely used version of these models today is based on this formalism, the so-called *n-gram models*. The description of the statistical properties of symbol sequences by stochastic grammars is also well understood but considerably more complicated, as the parameter training of such models is technically extremely demanding. Additionally, the definition of the grammar rules needs to be done by experts, as no general inference methods are known for learning these automatically from data. Therefore stochastic grammars have been used for pattern analysis tasks to a rather limited extent only, which is why the term language model is mostly used synonymously for n-gram models in the literature. These models shall, therefore, be considered as the only class of statistical language modeling techniques in the following. For different methods the interested reader is referred to the respective specialized literature.

6.1 Definition

A statistical n-gram model corresponds to a Markov chain of order n-1. The probability $P(\boldsymbol{w})$ of a certain symbol sequence $\boldsymbol{w} = w_1, w_2, \ldots, w_T$ of length T is first decomposed according to Bayes' rule into a product of conditional probabilities:

$$P(\boldsymbol{w}) = P(w_1)P(w_2|w_1)\ldots P(w_T|w_1,\ldots,w_{T-1}) = \prod_{t=1}^{T} P(w_t|w_1,\ldots,w_{t-1})$$

During this factorization, however, with increasing length T of the symbol sequence there possibly result conditional probabilities with arbitrary long dependencies — as already pointed out when introducing stochastic processes. Therefore, for practical applications the maximal length of the context is limited to $n-1$ predecessor symbols. Motivated by the chronological order of symbol or word sequences this context is frequently also referred to as the *history*.

$$P(\boldsymbol{w}) \approx \prod_{t=1}^{T} P\underbrace{(w_t \mid w_{t-n+1}, \ldots, w_{t-1})}_{n \text{ symbols}}$$

The respective predicted symbol w_t and the associated history form a tuple of n symbols, which is the reason why the models are referred to as n-gram models. A concrete n-tuple of symbol is called n-gram and is in the field of automatic speech recognition usually referred to as an *event*.

Thus a given n-gram model defines probabilities for the prediction or evaluation of the occurrence of symbols from a finite inventory within a sequence on the basis of a context of $n-1$ known predecessor elements. From these individual contributions the probability of a certain sequence in total can be calculated directly. The entirety of the conditional probability distributions forms the statistical language model, which in contrast to HMMs is never denoted with an explicit mathematical symbol in the literature. Therefore, in the further elaborations we also abstain from an explicit designation, as long as the implicit assignment of probabilities to the respective model is clear.

As with increasing context length besides the principal difficulties to estimate such a model and to apply it in practice also the required amount of memory increases substantially, the most widely used variants of n-gram models are bi- and tri-grams. Already 4-gram models are, in contrast, hardly used any more in statistical recognition systems, whereas in the field of automatic speech recognition bi-gram models complementing the HMM modeling can be regarded as a standard procedure today. In the more recently opened application areas of statistical modeling techniques like the analysis of handwriting, gestures, or biological sequences, statistical language models to date are used only to a very limited extent. This is mainly due to the fact, that in those research areas in contrast to the field of speech recognition the general availability of sufficiently large database is not yet granted.

6.2 Use Cases

In the same way as for HMMs the usage of statistical language models for the description of texts and other symbol sequences is based on the assumption that their underlying generation principle obeys statistical rules, which can be described or at least sufficiently be approximated using Markov chains.

Therefore also for n-gram models it is a relevant question, how well a given model is able to describe available data. For this purpose essentially the probability of this symbol sequence — or some information theoretic measure derived from it — needs to be computed by means of the model used. Then inferences about the quality of the language model can be drawn or different models can be evaluated with respect to their suitability for describing the respective data. On this basis, e.g., paragraphs of text can be assigned a certain category or a specific topic, if for each of those a suitable model was built.

The second relevant question concerns the construction of the n-gram models themselves. In contrast to the necessary iteratively optimizing procedures used for HMMs, for statistical language models methods are conceivable that compute a model, which is in some sense optimal, directly depending on the sample data. The naive, immediately obvious solution of the problem is to count within the available sample set the absolute frequencies $c(w_1, w_2, \ldots, w_n)$ of all symbol tuples and all possible contexts $w_1, \ldots, w_{n-1}$. The required conditional probabilities $P(w_n|w_1, w_2, \ldots, w_{n-1})$ can then be defined via the relative frequencies $f(w_n|w_1, w_2, \ldots, w_{n-1})$.

$$P(w_n|w_1, w_2, \ldots, w_{n-1}) := f(w_n|w_1, w_2, \ldots, w_{n-1}) = \frac{c(w_1, w_2, \ldots, w_n)}{c(w_1, \ldots, w_{n-1})} \tag{6.1}$$

However, even for a moderately sized inventory of symbols and short context length it must be assumed that the majority of the theoretically possible n-grams are not contained in the sample set considered — even if this is rather large. Those events are then said not to have been observed or to be “unseen”. According to equation (6.1) all conditional probabilities related to such tuples will be defined to be zero. When evaluating the resulting model on new data then necessarily a probability of zero results for every sequence considered that contains a single of those unseen events.

Such a behavior of the model is, however, extremely undesirable, as it can be assumed that the estimation of a vanishing probability is extremely unreliable and only due to the limited size or representativity of the sample set considered. Therefore, in n-gram models the probability distributions determined empirically are always subject to a post-processing or smoothing operation, which is aimed at delivering robust estimates especially for very small conditional probabilities. In those methods lies the relevant algorithmic know-how for applying statistical language models to pattern recognition tasks.

6.3 Notation

In the treatment of n-gram models mainly the terms for computing the individual conditional probabilities are considered and not their evaluation on a longer text. Therein the distinction between the predicted word and the current history is essential. In order to be able to also make this more explicit in the formal treatment, in the following a notation will be used that is inspired by the one defined in [62].

An arbitrary isolated n-gram we will be denoting as $\boldsymbol{y}z$, where z stands for the predicted symbol and $\boldsymbol{y} = y_1, y_2, \ldots y_{n-1}$ for its history. In the notationally simple case of tri- or bi-gram models, all symbols of a tri-gram can be denoted individually — if this is required — as xyz and those of a bi-gram as yz. The conditional n-gram probabilities we will in general be denoting as $P(z|\boldsymbol{y})$ and as $P(z|xy)$ for tri- and $P(z|y)$ for bi-gram models. That symbol within the n-gram, which is the last with respect to the chronological order, will always be referred to as z.

Besides the frequency of occurrence $c(\boldsymbol{y}z)$ of an n-gram $\boldsymbol{y}z$ in the training set considered, also several derived quantities are important, which either characterize properties of n-gram contexts or provide meta information about the empirical frequency distribution. The absolute frequency — or count — of all n-grams with history $\boldsymbol{y}$ we will denote as $c(\boldsymbol{y}\,\cdot)$ by using the joker symbol '$\cdot$'. It is in principle equal to the frequency $c(\boldsymbol{y})$ of the context $\boldsymbol{y}$ of an n-gram $\boldsymbol{y}z$. Differences merely result from border phenomena in the sample set considered. If, e.g., the training text or a specially marked section therein end with $\boldsymbol{y}$, then at this position no further event $\boldsymbol{y}z$ exists having this very context and $c(\boldsymbol{y}) > c(\boldsymbol{y}\,\cdot)$ holds. For this reason we will use $c(\boldsymbol{y}\,\cdot)$ throughout the whole presentation, as this quantity always yields the correct normalization in practical applications. By $d_k(\boldsymbol{y}\,\cdot)$ we will be denoting, how many events with context $\boldsymbol{y}$ occur exactly k times in the training data. Especially important is the frequency $d_1(\boldsymbol{y}\,\cdot)$ of so-called *singletons*, i.e. events that occur exactly once in a given context. The number of different events occurring at all is in generalization of this notation referred to as $d_{1+}(\boldsymbol{y}\,\cdot)$. When also replacing the context restrictions in this quantity by the joker symbol, one obtains the total number of all n-grams $c(\cdot\,\cdot)$ or the number of events occurring k times in total $d_k(\cdot\,\cdot)$. In order to simplify the representation, in the two last cases mentioned the arguments $(\cdot\,\cdot)$ can be omitted completely.

6.4 Evaluation

For the evaluation of a statistical language model — as in many fields of statistical pattern analysis — its descriptive power on unknown data is determined, i.e. on test data that was not used for creating the model. The so-called *perplexity* has established itself as the relevant quality measure [108]. For a given test text $\boldsymbol{w} = w_1, w_2, \ldots, w_T$ of length $|\boldsymbol{w}| = T$ or a test symbol sequence, respectively, one obtains the perplexity $\mathcal{P}$ of the language model considered as the reciprocal value of the geometric mean of the individual symbol probabilities[1]:

$$\mathcal{P}(\boldsymbol{w}) = \frac{1}{\sqrt[|\boldsymbol{w}|]{P(\boldsymbol{w})}} = \frac{1}{\sqrt[T]{P(w_1, w_2, \ldots, w_T)}} = P(w_1, w_2, \ldots, w_T)^{-\frac{1}{T}}$$

[1] In the literature the perplexity is usually defined via the entropy of a language or of an information source (cf. e.g. [95, pp. 97–100], [196, pp. 449-450], [97, pp. 560–562]). However, the introduction of a number of fundamental information theoretic concepts for this derivation only, seems too demanding. Therefore, here precedence was given to a simplified presentation.

The wide-spread use of this measure in the literature is to a substantial extent due the possibility of interpreting it in an intuitively illustrative way. For that one assumes that the text considered was created by an information source, which generates with some probability symbols from a finite vocabulary V. For purposes of the analysis it is now desired to be able to predict this process as exactly as possible, i.e. only as few as possible symbols should be considered for continuing a sequence. However, such a deterministic statement is strictly speaking not possible due to the statistical behavior of the source. In principle all symbols might occur at any time, even though with an arbitrarily low probability. In the statistical average, however, a relationship to the deterministic behavior can be established.

In the "worst" case the generation is performed according to a uniform distribution over the lexicon V, which means that every additional symbol w_t will be generated with probability $P(w_t) = \frac{1}{|V|}$ independently of the context. Then a prediction of successor symbols is not possible, as all words of the lexicon can occur equally likely. The probability of a text of length T is obtained as $\{\frac{1}{|V|}\}^T$ and its perplexity $|V|$ is equal to the size of the lexicon used.

If the generation principle relies on any other probability distribution, which generates certain words with higher and others with lower frequency, always a lower perplexity $\rho < |V|$ is obtained compared to the case of the uniform distribution. The precision of the prediction based on this model now can be related to the one of an "uninformed" source, which acts according to a uniform distribution, with equal perplexity. The vocabulary of that source, which is identical with respect to the evaluation criterion, would consist of exactly $|V'| = \rho < |V|$ words — i.e. less than contained in the original lexicon. It can, therefore, be said that the perplexity of a statistical language model specifies, how many words are in the statistical average likely for the continuation of a symbol sequence, even if at every time from the statistical point of view an arbitrary continuation with eventually lower probability is possible.

From this basic observation results the goal of language modeling, which is to keep the perplexity of texts or symbol sequences to be expected as low as possible by means of precise statistical models. Thereby the generalization capability of the model is essential, as a model too specifically attuned to a certain training situation is useless for practical applications[2].

For n-gram models this is always achieved quite well, if sufficiently precise restrictions for possible successor words can be derived based on the considered context of $n - 1$ words. However, it must be possible to estimate the parameters of the conditional probabilities, which are necessary for describing these restrictions, sufficiently robustly on the available training material.

An important task for automatic speech recognition is the recognition of sequences of digits for telephony applications. Due to the almost complete absence

[2] By completely storing the training material of a language model always a perplexity of 1 can be achieved on this data. However, even with standard techniques arbitrarily low perplexities on the training set are possible, which, however, allow now predictions whatsoever about the modeling quality on unknown texts.

of restrictions for such symbol sequences, here the "worst case" perplexity of 10 can hardly be reduced any more by techniques of statistical language modeling. For "normal" texts in natural languages, however, a significant reduction in perplexity below the unrestricted case is always possible. Therefore, also the compression of texts is an application area for statistical language models (cf. e.g. [14]).

n-gram models are applicable especially well for the English language — not only because this language is probably among the most well studied in this respect. The relatively linear sentence structure with reduced degrees of freedom in word order and the almost complete missing of inflected forms offer ideal conditions for statistical language modeling. Inflecting languages with rather free word order as French or German are considered to be much more challenging. In German additionally the extremely productive creation of compound words is added, which either requires the analysis of complex word constructs or a substantial increase of the possible lexicon (cf. e.g. [2, 86, 229, 241]). A similar problem can be found in agglutinative languages as, e.g., Turkish or Finnish, where many syntactic phenomena are described by concatenating morphemes (cf. e.g. [32, 216]).

The basis of language modeling does, however, not necessarily have to be the orthographic word definition in the respective language. In order to simplify the modeling, e.g., a normalization of the lexicon used (cf. e.g. [1]) or a direct modeling of morpheme sequences by the statistical model (cf. e.g. [32, 86, 242]) can be considered.

6.5 Parameter Estimation

As n-gram models in contrast to HMMs usually contain no hidden state variables, their parameters can in principle be computed directly — i.e. without the need of some iterative optimization procedure — based on example data. Provided that one is content with defining the conditional probabilities via the relative frequencies, the model can directly be specified after counting the events observed in the sample set.

However, for a robust application of n-gram models it is essential, that the problem of events not observed in the data is dealt with. Therefore, in all approaches to statistical language modeling known from the literature the empirical distributions are never used directly, but rather a suitable smoothing is applied to them. The most important task of this smoothing is to replace vanishing probabilities of unseen events by plausible and robust estimates. Due to the normalization conditions necessary for probability distributions this can, however, not be performed in isolation, but the estimates of the remaining events have to be included in the process.

The most widely used class of methods for solving this problem proceeds in two steps. First the empirical distributions are modified such that a re-distribution of probability mass from seen to unseen events takes place. As those manipulations usually result in very small changes only, the probabilities of seen events are hardly modified. The relevant effect of that approach is, therefore, the gathering of "probability mass", in order to be able to define new, very small probabilities for n-grams not observed in the sample set.

In a second step then robust estimates are calculated on the basis of the modified empirical distributions by incorporating one or even several more general distributions. For frequently observed events the influence of this modification can be kept small or be omitted totally. For unseen n-grams it is, however, important not to distribute the gathered probability mass uniformly but on the basis of a more general distribution — usually the one of the associated $(n\text{-}1)$-gram model. Otherwise *all* unseen events would be assigned the same probability.

In practice also for $(n\text{-}1)$-grams robust estimates are not necessarily available directly. Therefore, the two steps of the method outlined above are recursively applied to the resulting model hierarchy from the n-gram down to the uni-gram or zero-gram, respectively. The term *uni-gram model* refers to the prior probability of the words and the term *zero-gram model* is used to denote the uniform probability distribution over a lexicon of given size.

6.5.1 Redistribution of Probability Mass

A quite intuitive possibility to get rid of the problem of non-observed events, consists in raising their frequencies of occurrence, that were determined to be zero, to some positive value. In order not to discard the differences between seen and unseen events, this positive constant — usually 1 — is added to all n-gram counts. If one uses the sum over all newly determined counts of individual events $\sum_z c^*(\boldsymbol{y}z)$ instead of the frequency of the respective n-gram context $c(\boldsymbol{y}\cdot)$ for the computation of the relative frequencies, the normalization condition for probability distributions is not violated by this manipulation.

This relatively old method is referred to as *adding one* or *Laplace's rule* in the literature. Though it is maximally simple on the algorithmic side, it unfortunately achieves *substantially* worse results than the newer methods, which will be presented in the following. This is due to the fact that the probability of rare events is systematically over-estimated (cf. [165]).

All other methods for redistributing probability mass are based on the principle to first gather the counts, which are necessary for removing unseen events, at some other position in the original probability distribution. Hereby not only the normalization condition of the relative frequencies remains unaltered, but it is also possible that a different amount of probability mass for unseen events is obtained depending on the properties of the initial distribution considered and the redistribution strategy applied. Thus it can be controlled to some extent, whether the observation of such an event, which is assumed to be extremely rare, is more or less probable in a certain context.

Discounting

In order to be able to gather probability mass for unseen events without a modification of the occurrence frequencies in total, the empirical counts $c(\boldsymbol{y}z)$ of observed

n-grams need to be reduced by a certain amount $\beta(\boldsymbol{y}z)$. Therefore, this class of methods is referred to as so-called *discounting* [117]. The modified relative frequencies $f^*(z|\boldsymbol{y})$ are obtained directly from the modified n-gram counts $c^*(\boldsymbol{y}z)$ according to:

$$f^*(z|\boldsymbol{y}) = \frac{c^*(\boldsymbol{y}z)}{c(\boldsymbol{y}\cdot)} = \frac{c(\boldsymbol{y}z) - \beta(\boldsymbol{y}z)}{c(\boldsymbol{y}\cdot)} \quad \forall \boldsymbol{y}z, c(\boldsymbol{y}z) > \beta(\boldsymbol{y}z)$$

The discounting function $\beta(\boldsymbol{y}z)$ may be chosen such, that for certain events $\boldsymbol{y}z$ it will become equal to their count $c(\boldsymbol{y}z)$. Then the effect can be observed, that these events first contribute all their probability mass to the redistribution process, but themselves belong to the effectively unseen events after the discounting.

The total probability mass one obtains for improving the empirical distribution and especially for removing unseen events is referred to as the so-called *zero probability* $\lambda(\boldsymbol{y})$. It depends on the respective n-gram context and is calculated as a sum over the accumulated probability mass[3]:

$$\lambda(\boldsymbol{y}) = \frac{\sum\limits_{\boldsymbol{y}z : c(\boldsymbol{y}z)>0} \min\{\beta(\boldsymbol{y}z), c(\boldsymbol{y}z)\}}{c(\boldsymbol{y}\cdot)}$$

In case that the modified relative frequency $f^*(\cdot)$ is directly used as an estimate for the conditional probability of seen events, this quantity denotes, with which probability one of the unseen events can be expected to occur *at all* in a given context. As will be elaborated in more detail in the following section 6.5.2 there also exist methods that perform a redistribution of the zero probability to *all* events in the respective context. For both strategies, however, additional knowledge is required, as otherwise the redistribution would only be possible according to a uniform distribution.

For the definition of a special discounting strategy — i.e. the choice of $\beta(\boldsymbol{y}z)$ — the methods known from the literature can be divided into two groups. All methods, which perform the reduction of a certain n-gram count proportional to its amount $c(\boldsymbol{y}z)$, are referred to as *linear discounting*. If, however, a discounting constant β is specified independently from the actual frequency of events, so-called *absolute discounting* is applied.

In the simplest form of linear discounting $\beta(\boldsymbol{y}z)$ is defined by means of a proportional factor α depending on $c(\boldsymbol{y}z)$.

$$\beta(\boldsymbol{y}z) = \alpha\, c(\boldsymbol{y}z)$$

The modified relative frequencies $f^*(\boldsymbol{y}z)$ for all observed events are consequently obtained as:

$$f^*(z|\boldsymbol{y}) = \frac{(1-\alpha)c(\boldsymbol{y}z)}{c(\boldsymbol{y}\cdot)} = (1-\alpha)f(z|\boldsymbol{y}) \quad \forall \boldsymbol{y}z, c(\boldsymbol{y}z) > 0 \quad \text{and} \quad 0 < \alpha < 1$$

[3] In general the discounting function $\beta(\boldsymbol{y}z)$ can take on values larger than the absolute frequencies $c(\boldsymbol{y}z)$, so that during the modification eventually only the minimum of the respective values is gathered as probability mass.

A good choice for α is given by the relative frequency of singletons, i.e. events observed exactly once in the data [166]:

$$\alpha = \frac{d_1(\cdot\,\cdot)}{c(\cdot\,\cdot)} = \frac{d_1}{c}$$

A more general formulation results, if the proportional constant is not defined globally, but individually for every n-gram context. In this case it becomes equivalent to the respective zero probability $\lambda(\boldsymbol{y})$, so that one obtains the following modified distribution of relative frequencies [62, 166]:

$$f^*(z|\boldsymbol{y}) = \frac{(1-\lambda(\boldsymbol{y}))c(\boldsymbol{y}z)}{c(\boldsymbol{y}\,\cdot)} = (1-\lambda(\boldsymbol{y}))f(z|\boldsymbol{y}) \quad \forall \boldsymbol{y}z, c(\boldsymbol{y}z) > 0 \tag{6.2}$$

The most serious drawback of linear discounting lies in the fact that the counts of frequently observed events are modified the most. This does, however, not comply with the basic statistical assumption backed up by the "law of large numbers" that the more robust estimates are obtained, the more example data is available for a certain event. Furthermore, in the general formulation of equation (6.2) the choice or optimization, respectively, of the zero probability $\lambda(\boldsymbol{y})$ is required for all n-gram contexts. Though this can be performed on additional data [166], the effort for computing the n-gram model is in any case significantly increased.

Substantially easier in the application are, in contrast, methods for absolute discounting. Also with respect to their effectiveness these are among the best methods known. Here large frequencies remain almost unchanged, and mainly events observed rarely contribute to the gathering of probability mass. Therefore, these techniques in general tend to assign smaller probabilities to unseen events than methods for linear discounting do.

In absolute discounting every count $c(\boldsymbol{y}z)$ determined empirically is decreased by a constant amount β. On the basis of the counts $c^*(\boldsymbol{y}z)$ modified such one obtains the following new distribution of relative frequencies:

$$f^*(z|\boldsymbol{y}) = \frac{c^*(\boldsymbol{y}z)}{c(\boldsymbol{y}\,\cdot)} = \frac{c(\boldsymbol{y}z)-\beta}{c(\boldsymbol{y}\,\cdot)} = \quad \forall \boldsymbol{y}z, c(\boldsymbol{y}z) > \beta \tag{6.3}$$

The zero probability $\lambda(\boldsymbol{y})$ can in this case simply be given as:

$$\lambda(\boldsymbol{y}) = \frac{\sum\limits_{\boldsymbol{y}z:c(\boldsymbol{y}z)>0} \beta}{c(\boldsymbol{y}\,\cdot)} = \beta\,\frac{d_{1+}(\boldsymbol{y}\,\cdot)}{c(\boldsymbol{y}\,\cdot)}$$

In all variants of absolute discounting the discounting constants β are always chosen such, that they do not lie above the original frequencies. This is especially true for the widely used choice of $\beta \leq 1$, which already achieves very good results in practice.

Under certain marginal conditions the discounting constants can be chosen better — though not necessarily optimal. If for every context length exactly one constant is used, one obtains the following upper bound for its choice according to [165]:

$$\beta \leq \frac{d_1(\cdot\cdot)}{d_1(\cdot\cdot) + 2d_2(\cdot\cdot)} = \frac{d_1}{d_1 + 2d_2} < 1$$

In [34] even three different constants are proposed for events seen once, twice, or more frequently. The improvements achieved with this refinement of the model are in the majority of cases quite marginal, however.

6.5.2 Incorporation of More General Distributions

The gathering of probability mass alone is, however, not sufficient in order to determine suitable estimates for conditional n-gram probabilities. The redistribution of the zero-probability for removing unseen events or also for the support of less robust estimates always requires the incorporation of additional, more reliable knowledge. As this knowledge can not be contributed by experts, but also needs to be determined from the example data, for that purpose only probability distributions can be considered, which are less complex and, therefore, can be estimated more easily than the n-gram model to be improved.

For tasks of language modeling the strategy most frequently applied for choosing such more general distributions is the shortening of the context restriction of the n-gram models by one word. The resulting $(n\text{-}1)$-gram distribution is less specific than the initial model and, therefore, chances are also higher that its parameters can be estimated robustly on the available data.

For the combination of the more general distributions with the smoothed empirical distributions two fundamental classes of methods exist. When applying *interpolation*, a weighted averaging of the two parts of the model is performed. In the case of the so-called *backing off* the more general distribution is only taken into account for redistributing the gathered zero probability in a clever way onto unseen events.

Interpolation

A group of methods widely used in the field of language modeling for improving empirically determined estimates $f(z|\boldsymbol{y})$ of a special distribution $P(z|\boldsymbol{y})$ are *interpolation methods* (cf. [106]). By a linear combination with a suitably chosen more general distribution $q(z|\boldsymbol{y})$ one obtains more robust estimates for the special probability distribution considered:

$$P(z|\boldsymbol{y}) = (1-\alpha)\, f(z|\boldsymbol{y}) + \alpha\, q(z|\boldsymbol{y}) \quad 0 \leq \alpha \leq 1 \tag{6.4}$$

This approach is based on the assumption that principally *all* empirically determined frequencies $f(z|\boldsymbol{y})$ can be estimated better — i.e. more robustly — by also taking into account additional knowledge in the form of the more general distribution $q(z|\boldsymbol{y})$. Thus, vanishing frequencies and those that are based on a sufficient number of samples are treated in the same way.

In this method the interpolation weight α needs to be chosen such, that on the one hand the coarser information of the more general distribution does not dominate the special estimates too much, but on the other hand suitably supports less robust or vanishing relative frequencies. On the basis of additional training material, which was not used for determining the empirical distributions, the interpolation weight α can either be determined experimentally or be optimized in a mathematically exact way[4].

As more general distribution in the context of language modeling almost always the $(n\text{-}1)$-gram distribution $P(z|\hat{\boldsymbol{y}})$ associated with the n-gram model $P(z|\boldsymbol{y})$ considered is chosen. Thus the generalization lies in the shortening of the context restriction $\boldsymbol{y} = y_1, y_2, \ldots y_{n-1}$ by one word to yield $\hat{\boldsymbol{y}} = y_2, \ldots y_{n-1}$, which achieves a lower specificity of the model and at the same time a better trainability of its parameters.

$$q(z|\boldsymbol{y}) = q(z|y_1, y_2, \ldots y_{n-1}) \leftarrow P(z|y_2, \ldots y_{n-1}) = P(z|\hat{\boldsymbol{y}})$$

For tri- and bi-gram models with this strategy $q(\cdot)$ can easily be given as follows:

$$\begin{aligned} q(z|xy) &\leftarrow P(z|y) \\ q(z|y) &\leftarrow P(z) \end{aligned}$$

As now the more general distribution $q(\cdot)$ again consists of an n-gram model, the interpolation principle is in general applied recursively here:

$$\begin{aligned} P(z|\boldsymbol{y}) &= (1-\alpha_n)\, f(z|\boldsymbol{y}) + \alpha_n\, q(z|\boldsymbol{y}) \\ &= (1-\alpha_n)\, f(z|y_1 \ldots y_{n-1}) \\ &\quad + \alpha_n\, [(1-\alpha_{n-1})\, f(z|y_2 \ldots y_{n-1}) \\ &\qquad + \alpha_{n-1}\, q(z|y_2 \ldots y_{n-1})] \\ &= \cdots \qquad\qquad\qquad\qquad \forall i:\ 0 \leq \alpha_i \leq 1 \end{aligned}$$

Thus by repeated shortening of the n-gram context for choosing the respective more general distribution $q(\cdot)$ one obtains a linear combination[5] of all empirically determined relative frequencies as the final model[6].

$$\begin{aligned} P(z|\boldsymbol{y}) = &\quad \lambda_n\ f(z|\boldsymbol{y}) \\ &+ \lambda_{n-1}\ f(z|y_2 \ldots y_{n-1}) \\ &+ \quad \cdots \\ &+ \quad \lambda_1\ f(z) \\ = &\textstyle\sum_{i=1}^{n} \lambda_i\ f(z|y_{n-i+1} \ldots y_{n-1})\ \forall i:\ 0 \leq \lambda_i \leq 1 \quad \text{and} \quad \sum_i \lambda_i = 1 \end{aligned}$$

[4] For the optimal choice of the interpolation weight α the training material of the n-gram model can not be used, as on this data the most special model — i.e. the empirical distribution $f(z|\boldsymbol{y})$ itself — would always be optimal. One, therefore, would obtain $\alpha = 0.0$ [105, p. 63]. Rather, additional material needs to be available that is exclusively used for the optimization of the interpolation weight.

[5] For the sake of clarity here the resulting products of the pairwise interpolation weights α_i were replaced by new constants λ_i.

[6] The incorporation of a zero-gram model usually is not necessary, if one assumes, that every word of the lexicon V considered was observed at least once. However, if this is not the case, as, e.g., the sample set and the lexicon were defined completely independently of each other, the interpolation scheme can be extended by the additional term $\lambda_0 \frac{1}{|V|}$.

From this the interpolation formula for robust tri-gram models developed by Jelinek & Mercer and used in the tradition of research at IBM can be obtained as a special case (cf. e.g. [105, pp. 60–61]):

$$P(z|xy) = \lambda_3 \, f(z|xy) + \lambda_2 \, f(z|y) + \lambda_1 \, f(z)$$

In the formulation so far the interpolation method directly started from the empirically determined relative frequencies $f(\cdot)$ and made no use of the reduced frequency distribution $f^*(\cdot)$ generated in the preparatory step described previously. The principal possibility for its incorporation into the interpolation becomes evident the most easily, if *linear discounting* is used for obtaining $f^*(z|\boldsymbol{y})$:

$$f^*(z|\boldsymbol{y}) = (1 - \alpha) f(z|\boldsymbol{y}) \quad \forall \, \boldsymbol{y}z$$

This expression corresponds directly to the first term in the sum of the classical interpolation equation (6.4). When distinguishing between sufficiently frequently observed events, for which estimates $f^*(z|\boldsymbol{y})$ exist, and effectively unseen n-grams, one obtains a generalized interpolation rule. Therein the reduced frequency distribution $f^*(z|\boldsymbol{y})$ is combined with the general distribution $q(z|\boldsymbol{y})$, with the interpolation weight being equal to the gathered zero-probability $\lambda(\boldsymbol{y})$:

$$P(z|\boldsymbol{y}) = \begin{cases} f^*(z|\boldsymbol{y}) \; + \; \lambda(\boldsymbol{y}) q(z|\boldsymbol{y}) & c^*(\boldsymbol{y}z) > 0 \\ \lambda(\boldsymbol{y}) q(z|\boldsymbol{y}) & c^*(\boldsymbol{y}z) = 0 \end{cases} \tag{6.5}$$

As in this case the interpolation weights are not defined for a distribution in general, but are chosen depending on the context $\boldsymbol{y}$, the method is also called *non-linear interpolation* [165].

The crucial conceptual drawback of interpolation models is directly caused by their very design principle. When assuming, that counts, which were determined on a large basis of sample data, already represent robust estimates, it does not seem to be useful to modify these by an interpolation with a coarser model. In the end this leads to a smoothing of the resulting distributions, which at the same time also causes a partial loss of their specificity.

Backing Off

The principle of *backing off*, which dates back to Katz [117], in contrast to interpolation methods incorporates the more general distribution only then into the computation of the n-gram model, if the reduced frequencies $f^*(\cdot)$ vanish. In all other cases those values are directly adopted as the conditional probabilities of the respective event. The redistribution of the gathered zero probability $\lambda(\boldsymbol{y})$ onto the unseen events is, however, performed proportional to the more general distribution $q(\cdot)$.

$$P(z|\boldsymbol{y}) = \begin{cases} f^*(z|\boldsymbol{y}) & c^*(\boldsymbol{y}z) > 0 \\ \lambda(\boldsymbol{y}) \, K_{\boldsymbol{y}} q(z|\boldsymbol{y}) & c^*(\boldsymbol{y}z) = 0 \end{cases} \tag{6.6}$$

In order to guarantee the normalization constraint for the distribution $P(z|\boldsymbol{y})$ constructed such, an additional scaling factor $K_{\boldsymbol{y}}$ needs to be introduced. It makes sure, that the zero probability is completely — i.e. with weight 1.0 — incorporated into the resulting model according to a probability distribution $K_{\boldsymbol{y}} q(z|\boldsymbol{y})$ defined for unseen events only. The scaling factor can, therefore, be computed as the reciprocal value of the probability sum that is obtained for unseen events according to the more general distribution:

$$K_{\boldsymbol{y}} = \frac{1}{\sum\limits_{\boldsymbol{y}z\,:\,c^*(\boldsymbol{y}z)=0} q(\boldsymbol{y}z)} \tag{6.7}$$

In the same way as with interpolation methods the more general distribution $q(\cdot)$ is generally chosen by simply shortening the n-gram context. Likewise the principle of backing off is recursively applied to the whole hierarchy of n-gram models. For a back-off tri-gram model one thus obtains the following computation scheme:

$$P(z|xy) = \begin{cases} f^*(z|xy) & c^*(xyz) > 0 \\ \lambda(xy)\,K_{xy} \begin{cases} f^*(z|y) & c^*(xyz) = 0 \wedge c^*(yz) > 0 \\ \lambda(y)\,K_y \begin{cases} f^*(z) & c^*(yz) = 0 \wedge \ c^*(z) > 0 \\ \lambda(\cdot)\,K_\cdot \frac{1}{|V|} & c^*(z) = 0 \end{cases} \end{cases} \end{cases}$$

In combination with absolute discounting by backing off one obtains very easily extremely powerful n-gram models for a wide spectrum of applications (cf. [34, 62]). Even with considerably more expensive methods in general only marginally lower perplexities can be achieved.

6.5.3 Optimization of Generalized Distributions

In the previous section it was assumed that the more general distributions necessary for interpolation methods or backing off can be obtained by shortening the context of the respective n-gram model. This heuristic definition is, however, not necessarily the best possible strategy.

This can be made illustratively plausible with the example of a word z, which in fact occurs very frequently in a given sample set, but only in a certain context y. Depending on the text category this could be for example “York” in the context “New” or “machine” after the preceding phrase “support vector”. In all other contexts y', however, z was never observed and both in interpolation as in backing off the more general distribution $q(z|y')$ is decisive for determining the conditional probability $P(z|y')$. If one chooses $q(z|y') \leftarrow P(z)$, the predicted probability $P(z|y')$ is proportional to the frequency of occurrence $c(z)$ of the considered word z. Therefore, one obtains a relatively high probability for its observation in a context y', in which z never occurred in the sample data. A relatively low probability would be more plausible, as in this situation the considered word is seen frequently in exactly one context, but was not observed in any other. A more general distribution, which comes close to that intuitive idea, defines $q(z|y')$ proportional to the number of different contexts, in which a certain word was observed. The more unique the context specificity of an event is, the lower the respective predicted probability would be compared to other events.

In the method proposed by Kneser & Ney for determining the more general distribution, $q(\cdot)$ is not defined heuristically, but derived in an analytical way [121]. Using two different optimization criteria principally comparable distributions $q(\cdot)$ are obtained, which are not based on the absolute frequencies of events, but on the number of contexts, in which a word was observed.

In the first approach proposed it is required, that the conditional $(n\text{-}1)$-gram distribution $P(z|\hat{\boldsymbol{y}})$, which results from shortening the context, can also be obtained as the marginal distribution of the joint distribution $P(\boldsymbol{y}, z|\hat{\boldsymbol{y}})$. The more general distribution $q(z|\boldsymbol{y})$ is then given by:

$$q(z|\boldsymbol{y}) = \frac{d_{1+}(\cdot\hat{\boldsymbol{y}}z)}{d_{1+}(\cdot\hat{\boldsymbol{y}}\cdot)} = \frac{d_{1+}(\cdot\hat{\boldsymbol{y}}z)}{\sum\limits_{z'} d_{1+}(\cdot\hat{\boldsymbol{y}}z')} \tag{6.8}$$

In the second derivation, which applies the principle of *leaving-one-out* — a special technique of parameter estimation by means of cross validation (cf. [166] or [57]), a similar formula for the computation of $q(\cdot)$ is obtained:

$$q(z|\boldsymbol{y}) = \frac{d_{1}(\cdot\hat{\boldsymbol{y}}z)}{d_{1}(\cdot\hat{\boldsymbol{y}}\cdot)} = \frac{d_{1}(\cdot\hat{\boldsymbol{y}}z)}{\sum\limits_{z'} d_{1}(\cdot\hat{\boldsymbol{y}}z')} \tag{6.9}$$

Thus in the first case $q(z|\boldsymbol{y})$ is proportional to the number of different contexts $\hat{\boldsymbol{y}}$, in which the word z to be predicted is occurring. In the expression (6.9) determined by leaving-one-out from these contexts only those are considered, in which z was observed *exactly once*. In general this quantity $d_1(\cdot\hat{\boldsymbol{y}}z)$ also accounts for a significant portion of the total number of different contexts $d_{1+}(\cdot\hat{\boldsymbol{y}}z)$, so that the two possibilities for the choice of $q(\cdot)$ do not differ substantially. In practical applications frequently cases occur, though, in which $d_1(\cdot\hat{\boldsymbol{y}}z)$ vanishes for certain, mostly frequent words, because these were always observed more than once in their respective contexts. Therefore, equation (6.8) is better suited for the robust estimation of optimized more general distributions.

Though the principle of determining more general distributions for n-gram models in an optimal way is immediately convincing, the advantages of the method are rather small in practice.

For the derivation of equations (6.8) and (6.9) the authors in fact rely on backing off. In practice with the proposed smoothing technique significant improvements in the modeling quality are, however, only obtained in combination with non-linear interpolation[7]. As in backing off the more general distribution is used for the evaluation of unseen events only, this behavior is quite plausible. An optimization there

[7] Inconsistently Kneser & Ney themselves use non-linear interpolation in order to demonstrate the potential of their smoothing technique in experimental evaluations [121].

has a conceivable effect only, if a large number of unseen events occur in the test set. This means, however, that the n-gram model used is rather unsuitable for the task, which will rarely happen in practice. In contrast, with non-linear interpolation the more general distribution is *always* incorporated into the probability computations and, therefore, has a significantly greater effect on the modeling quality.

The possible improvements are, however, obtained with an extremely high computational effort, which already for tri-gram models by far dominates the other computations necessary for building the model. Responsible for this is the repeated evaluation of expressions of the form $d_k(\cdot \boldsymbol{y} z)$. These counts can not be determined locally due to the leading joker symbol. Rather, always all contexts $x\boldsymbol{y}$ have to be taken into account, which makes searching a large portion of the stored count data necessary.

6.6 Model Variants

In the literature besides the "classical" definition of n-gram models also several variants of the basic concept were proposed. These in general try to overcome limitations or deficiencies of the standard formalism.

6.6.1 Category-Based Models

The most well known variant of the n-gram technique tries to exploit the fact, that natural languages alongside a syntagmatic structure exhibit a paradigmatic structure, too. This means that in different linguistic contexts not only a certain word, but a whole group of words or phrases can occur likewise. In every case one obtains a syntactically well formed utterance, even if its meaning in general will be altered by such an exchange.

Thus in the sentence *"I moved to Dortmund one year ago"* the city name could be replaced by an arbitrary one as, e.g. *Los Angeles* or *Paris*. Additionally the number could be in principle varied arbitrarily and the respective unit of time might be exchanged. But also in many other positions paradigmatic replacements could be made.

When trying to capture such a situation with an n-gram model, one will notice, that then all possible combinations need to occur at least once in the sample set considered. Still no paradigmatic rules can be represented in an abstract way. In order to make, e.g., the occurrence of city names probable in all relevant contexts, *all* these names would need to be observed in the respective context. Therefore, all words, which should be exchangeable in a certain context, are grouped to form a word class or *category*. Thus *Dortmund*, *Los Angeles*, and *Paris* would form the category of city names in our small example.

The language models extended in this way are referred to as *category-based n-gram models* or simply as *class n-grams*, provided that the thematic context is unique. The categories used should comprise such words that — in the statistical sense — occur in similar contexts or generate comparable restrictions for the joint occurrence of other words or categories. In [29, Sect. 3] Brown and colleagues describe this situation quite illustratively as follows:

"Clearly, some words are similar to other words in their meaning and syntactic function. We would not be surprised to learn that the probability distribution of words in the vicinity of Thursday is very much like that for words in the vicinity of Friday. Of course, they will not be identical: we rarely hear someone say Thank God it's Thursday! or worry about Thursday the 13^{th}*."*

When using a categorial n-gram model for the computation of the probabilities of word sequences the possible sequences of categories, which might correspond to these, must be taken into account. As in general the mapping of words to categories is not unique, in principle any sequence $\boldsymbol{C} = C_1, C_2, \ldots C_T$ of length T can be associated with a word sequence $\boldsymbol{w} = w_1, w_2, \ldots w_T$ with a certain probability $P(w_1, w_2, \ldots w_T | C_1, C_2, \ldots C_T)$. The probability of the word sequence is then obtained as the marginal distribution over all possible category sequences of the same length:

$$\begin{aligned} P(\boldsymbol{w}) &= P(w_1, w_2, \ldots w_T) \\ &= \sum_{C_1, C_2, \ldots C_T} P(w_1, w_2, \ldots w_T, C_1, C_2, \ldots C_T) \\ &= \sum_{C_1, C_2, \ldots C_T} P(w_1, w_2, \ldots w_T | C_1, C_2, \ldots C_T) P(C_1, C_2, \ldots C_T) \end{aligned}$$

As a first simplification one now defines, that the occurrence of a word at a specific position is not dependent on the total sequence of categories, but only on the respective corresponding element. The probability of the word sequence can then be rewritten as:

$$P(w_1, w_2, \ldots w_T | C_1, C_2, \ldots C_T) = \prod_{t=1}^{T} P(w_t | C_t)$$

Furthermore one describes the joint probability of the category sequence by approximating it with an n-gram model:

$$P(C_1, C_2, \ldots C_T) \approx \prod_{t=1}^{T} P(C_t | \, C_{t-n+1}, \ldots, C_{t-1})$$

The evaluation of a word sequence by means of a categorial language model restricted in this respect is then obtained according to:

$$P(\boldsymbol{w}) \approx \sum_{C_1, C_2, \ldots C_T} \prod_{t=1}^{T} P(w_t | C_t) P(C_t | \, C_{t-n+1}, \ldots, C_{t-1}) \tag{6.10}$$

If also the category n-gram model used is restricted to consist of a bi-gram model only, one obtains the following simplified computation rule:

$$P(\boldsymbol{w}) \approx \sum_{C_1, C_2, \ldots C_T} \prod_{t=1}^{T} P(w_t|C_t) P(C_t|C_{t-1}) \tag{6.11}$$

The comparison with equation (5.6) on page 68 immediately shows, that this model is equivalent to a discrete HMM, the internal states of which correspond to the categories and which generates words from a certain lexicon as outputs. Unfortunately, this also means, that the mere evaluation of the model is substantially more expensive than in the case of an "ordinary" n-gram model. For the latter only the multiplication of a sequence of conditional probabilities was necessary. For a categorial bi-gram model, however, the probability of a word sequence can be computed only by means of the forward algorithm. In contrast to the general formulation of a categorial n-gram model in equation (6.10), where in principle category sequences of arbitrary length are allowed as contexts, this nevertheless means a significant improvement in efficiency.

The most dramatic reduction in the evaluation effort of categorial n-gram models is achieved by the use of strictly disjoint categories. Then a unique mapping between the word and the corresponding category sequence is possible. Equation (6.10) can then be further simplified to:

$$P(\boldsymbol{w}) \approx \prod_{t=1}^{T} P(w_t|C_t) P(C_t|\, C_{t-n+1}, \ldots, C_{t-1}) \tag{6.12}$$

The main advantage of categorial models lies in the fact, that they require a significantly smaller number of parameters as opposed to "ordinary" word-based n-gram models. Therefore, they can be estimated robustly also on limited training material and promise improved generalization capabilities on unknown data, provided that the paradigmatic conditions were captured sufficiently well.

A "conventional" n-gram model is, however, always then superior to a categorial model, if large amounts of training material are available. Therefore, category-based n-gram models frequently are only used as a supplement to an existing word-based model. They then serve the purpose to provide robust estimates for the more general distributions required, for which it is the primary concern to define useful probabilities for rare or completely unobserved events (cf. e.g. [171, 204]).

The need to specify a suitable category system, however, makes the application of categorial models substantially more difficult. In some domains, which are also analyzed on a syntactic-semantic level, the linguistic categories lend themselves to be used directly for the purpose of language modeling (cf. e.g. [172]). In general more powerful but also significantly more costly are, however, methods for automatically determining a suitable category system on the basis of a sample set only (cf. e.g. [29, 122, 172]).

6.6.2 Longer Temporal Dependencies

The most important limitation in the modeling capabilities of n-gram language models results from the fact, that in practice only rather short contexts of usually no more than two predecessor words can be taken into account. The training sets usually available are not sufficient to robustly estimate parameters of more complex models. Therefore, in the literature a number of methods was proposed, which allow to capture longer context dependencies while at the same time achieving a reduced model complexity.

In so-called long-distance bi-grams pairs of words are considered in certain predefined distances, in order to create language models, which are from a computational point of view similar to bi-gram models. The total model then results from a linear combination of the component models [96]. A similar principle is also applied by the so-called distance tri-grams where word triplets are generated by skipping one context word at a time [153].

In contrast to such explicit techniques for taking into account longer contextual restrictions are methods, which try to adapt an existing model in a suitable way to a concrete textual or thematic context, respectively. Some methods of this group of techniques will be described under the aspect of model adaptation in section 11.3 page 186.

6.7 Bibliographical Remarks

Markov chain models originated from the work of the Russian mathematician Andrej Andrejewitsch Markov (1856 – 1922), after whom they were also named. He used such a modeling the first time for the statistical analysis of the character sequences in the text of “Eugene Onegin”, a novel in verse by Alexander Sergeyevich Pushkin [151]. In the mathematical technical literature mainly theoretical aspects of Markov chain models are covered. Their practical application occurs primarily in the field of automatic speech recognition or the statistical modeling of texts, respectively, where the models are referred to as *n-gram* or *language models*.

A very thorough treatment of the topic can be found in the monograph by Huang, Acero & Hon [97, pp. 558–590]. The treatment of language models in [105] is, however, strongly limited to the view of the author on the topic. Bell *et al.* describe n-gram models in the context of text compression [14]. A compact introduction into the subject is given in the article by Federico *et al.* [62]. Especially the historical perspective on the development of techniques for language modeling is covered by Rosenfeld [200].

Stochastic grammars, which similarly as n-gram models can be used for the statistical description of sequencing constraints in symbol chains or texts, respectively, are, e.g., described in [97, pp. 554–558]. An overview of language modeling techniques in general can be found in [97, Chap. 11, pp. 545–553] or [181, p. 417].

The probably oldest method for removing vanishing probability estimates of unobserved events is *adding one* (cf. e.g. [62], “Method A” in [249], “Jeffrey’s estimate” in [165]). The method, which is also referred to as *Laplace’s rule*, is to date still applied in the analysis of biological sequences [52, p. 108,115]. Considerably better model parameters are obtained by *interpolation methods*, which date back to works by Jelinek & Mercer ([106], cf. also [104] or [105, S. 62ff]). Ney, Essen &

Kneser proposed the use of *non-linear interpolation* [165]. The principle of *backing off* was developed by Katz [117]. The gathering of the probability mass required is performed by means of *linear discounting* ([117], cf. also [62, 166]) or *absolute discounting* ("Method B" in [249], [165, 166]). A method for the optimization of the more general distributions required for both backing off and interpolation was developed by Kneser & Ney [121]. Comparisons of different smoothing techniques can be found in Federico *et al.* [62] and also in Chen & Goodman [33, 34], where extensive experimental evaluations are reported. *Category-based n-gram models* are described in, e.g., [29] or [97, pp. 565–570].

Part II

Practice

Introductory Remarks

In the chapters 3 to 6 of the preceding part of this book the theoretical foundations of Markov models were introduced. Though they are extremely important for the conceptional understanding of the properties of HMMs and n-gram models, they are not sufficient for realizing working implementations of these techniques and systems that are successful in practical applications. For this purpose additionally know-how related to practice is necessary, the presentation of which, despite its great importance, is usually disregarded in the literature. In his monograph on statistical methods for speech recognition Jelinek proceeds in a especially extreme way with respect to practical aspects, by clarifying right at the beginning in a remark [105, p. 11]:

> *"As presented here, the algorithm contains the idea's essence. In practice, [...], many refinements are necessary that are the results of intensive experimentation. [...] This is the case with the vast majority of algorithms presented in this book: We describe the basic idea that must then be worked out in practice."*

Especially from a pioneer of automatic speech recognition and of statistical methods in the field of pattern analysis one would have expected more than such a disappointing statement.

In contrast to that we want to put an emphasis of this book on the practical applications of Markov model technology. In the following chapters essential aspects of HMMs and n-gram models relevant in practice will be covered, in order to achieve not only a theoretical understanding of the reader, but to also enable him, to realize his own applications of the techniques presented.

At the beginning methods will be presented for the numerically stable handling of probabilities, which are ubiquitous when dealing with Markov models. Chapter 8 is dedicated to the challenging problem of configuring Markov models, i.e. the choice of a suitable model architecture. Different methods, which make a robust estimation of parameters possible even for complex models are presented in chapter 9. In the subsequent chapter 10 efficient algorithms for model utilization are described. Additionally, techniques are presented, which achieve a gain in efficiency in practical applications by a reorganization of the models' representation. The topic of chapter 11 is the adaptation of models to the operating conditions, which will in general be different from the ones of the training phase. Especially for hidden Markov models such methods have gained increased importance lately. In the last chapter of the part on practical aspects of Markov models several methods for integrated search are presented, which make the joint use of HMMs and n-gram models for challenging tasks possible.

In a successful recognition system based on Markov models be it for spoken language or handwritten script in general methods from *all* topics mentioned above are implemented. In contrast, methods for model adaptation, integration of n-gram models, and the robust estimation of continuous distributions hardly play a role in the analysis of biological sequences.

7

Computations with Probabilities

The handling of probability values in real computing systems at first sight seems to be a trivial problem, as the range of values of those quantities is limited to the interval $[0.0 \ldots 1.0]$. Nevertheless problems arise especially in longer computational procedures, as *extremely small* values lying close to zero need to be represented and manipulated.

That such nearly vanishing probability values can appear quickly in computations for Markov models, can be illustrated by the following example. The computation of the probability $P(\boldsymbol{s}|\lambda)$ of a state sequence $\boldsymbol{s} = s_1, s_2, \ldots s_T$ for a given model λ is performed by multiplying all transition probabilities involved (cf. equation (5.5)):

$$P(\boldsymbol{s}|\lambda) = \prod_{t=1}^{T} a_{s_{t-1},s_t}$$

Even if in every state only two successor states with equal probability could occur, i.e. all a_{s_{t-1},s_t} are equal to 0.5, one obtains already from a length of the observation sequence of $T > 100$ onwards numerical values smaller than $5 \cdot 10^{-100}$, which can hardly be represented in today's digital computers[1]. And yet in these considerations strongly simplifying assumptions were made, which will never be met in reality. Additionally, in usual situations the handling of considerably longer observation sequences will be required, as $T = 100$ in automatic speech recognition usually corresponds to utterances of one second in length[2] and in handwriting recognition of only a few consecutive characters. Furthermore, scores for partial paths also include output probabilities or probability densities, which themselves tend to become quite small numerically.

[1] On practically all modern computer architectures floating point numbers are represented in formats, which were standardized in the framework of the ANSI/IEEE standard 854 [100]. With respect to their absolute value then single precision numbers can be represented in the range of approximately $3.4 \cdot 10^{38}$ to $1.4 \cdot 10^{-45}$ and such with double precision in the range of $1.8 \cdot 10^{308}$ to $4.9 \cdot 10^{-324}$.

[2] Feature vectors are extracted from the speech signal with a spacing of 10 ms in practically all current speech recognition systems.

In order to be able to effectively counteract the phenomenon of *de facto* vanishing probabilities, in practice one banks on an improved representation of those quantities and, if necessary, also on their algorithmic limitation to suitable lower bounds.

7.1 Logarithmic Probability Representation

The probably oldest method for improving the dynamic range of probability values in computations consists in scaling these appropriately (cf. [195], [95, pp. 241–242]).

Unfortunately, this is only possible locally as especially in longer computations the occurring quantities tend more and more to extremely small probability values. The necessary scaling factors, therefore, need to be determined for every normalization operation anew, which makes the method extremely costly and error prone.

Therefore, the representation on a negative logarithmic scale has become the method of choice for handling even extremely small probability values (cf. [95, pp. 243–244]). One no longer uses the real probability or density values p in computations but applies the following transformation:

$$\tilde{p} = -\log_b p \tag{7.1}$$

After this transformation the formerly multiplicative probability values can be interpreted as additive constants.

The choice of the base b of the logarithm used has no appreciable influence on the representational possibilities. As the natural logarithm to base e, however, makes an additional simplification of the evaluation of normal densities possible (cf. page 42), it is mainly used for this purpose. Also in practically all standard computing libraries efficient implementations of the natural logarithm and its inverse are available as `log()` and `exp()`, respectively. In the following we will, therefore, use the pair of functions $\ln x = \log_e x$ and e^x for the transformation of probability values to and from the logarithmic domain.

By this transformation the original range of values $[0.0 \ldots 1.0]$ for probabilities is mapped to the entire non-negative floating-point numbers that can be represented. As density values can also become larger than 1.0 — though only in rather rare cases in practice — their logarithmic representation comprises principally even the whole dynamic range of floating-point numbers. The resolution of the negative-logarithmic representation is limited though due to the limited precision of today's floating-point formats. However, the accuracy is sufficient for the practical application of the method.

In order to be able to fully exploit the advantages of the logarithmic representation it is necessary to carry out all computations on probabilities as continuously as possible in this domain. This is achieved easily as long as only multiplications and maximizations are involved in the calculations. Thus, e.g., the formula for computing the partial path probabilities $\delta_{t+1}(i)$ in the Viterbi algorithms (cf. equation (5.9)) or figure 5.5) is transformed from

$$\delta_{t+1}(j) = \max_i \{\delta_t(i) a_{ij}\}\, b_j(O_{t+1})$$

in linear, i.e. “normal” probability representation, into

$$\tilde{\delta}_{t+1}(j) = \min_i \left\{ \tilde{\delta}_t(i) \,+\, \tilde{a}_{ij} \right\} \,+\, \tilde{b}_j(O_{t+1})$$

when using the negative-logarithmic representation.

It is, however, considerably more complex, to carry out computations with logarithmically represented probability quantities, if these require a summation. In the most unfavorable of situations the quantities $\tilde{p}_1$ and $\tilde{p}_2$ involved would need to be first converted to the linear domain, then to be added, and finally to be transformed to the logarithmic domain again:

$$\tilde{p}_1 +_{\log} \tilde{p}_2 \;=\; -\ln\left(e^{-\tilde{p}_1} + e^{-\tilde{p}_2}\right)$$

This extremely costly operation can be simplified considerably by applying the so-called *Kingsbury-Rayner formula* ([118], see also [136, p. 29]) and thus saving one exponentiation[3]:

$$\begin{aligned} \tilde{p}_1 +_{\log} \tilde{p}_2 &= -\ln(p_1 + p_2) = -\ln(p_1(1 + \tfrac{p_2}{p_1})) = \\ &= -\left\{\ln p_1 \,+\, \ln(1 + e^{\ln p_2 - \ln p_1})\right\} \\ &= \tilde{p}_1 \,-\, \ln(1 + e^{-(\tilde{p}_2 - \tilde{p}_1)}) \end{aligned} \tag{7.2}$$

When applying equation (7.2) in fact still an exponentiation and a computation of the logarithms has to be performed[4]. However, sequences of computations involving summations of probability values can then be carried out in the logarithmic domain in an integrated manner.

The forward and backward variables $\alpha_t(i)$ and $\beta_t(j)$ (cf. equations (5.7) and (5.13) and figure 5.6) and also the probability sums for the normalization of n-gram scores (cf. equation (6.7)) can be calculated with sufficient accuracy by applying the method of Kingsbury & Rayner. The observable loss in efficiency in the training phase of a model constitutes a negligible limitation only, compared to the extension of the dynamic range.

However, the evaluation of output probability density functions described by mixture models requires the summation over all component densities even when decoding the model (cf. equation (5.1)):

$$b_j(\boldsymbol{x}) = \sum_{k=1}^{M} c_{jk}\, g_{jk}(\boldsymbol{x})$$

In this case the application of the “logarithmic summation” after equation (7.2) is not advisable, as per density calculation M exponentiations and computations of the

[3] A computation can even be avoided completely, if one or both values $\tilde{p}_i$ already lie below the smallest logarithmic probability value, that can be represented, or if the difference between $\tilde{p}_1$ and $\tilde{p}_2$ is too large and, therefore, the term $e^{-(\tilde{p}_2 - \tilde{p}_1)})$ vanishes.

[4] The precision of the “logarithmic summation” can even further be improved, if one uses, e.g. the function `log1p()` available in the standard C-library for directly computing $\ln(1 + x)$ also for small x.

logarithm are necessary in addition to the computational effort in the linear domain. If one is not able to work with "ordinary" probabilities in such cases (cf. section 7.3), the output probability density $b_j(\boldsymbol{x})$ can be approximated by the respective maximum of the component densities with slightly reduced accuracy:

$$b_j(\boldsymbol{x}) \approx \max_k \{c_{jk}\, g_{jk}(\boldsymbol{x})\}$$

As for this approximative calculation no summations are necessary any more, the computation of $\tilde{b}_j(\boldsymbol{x})$ can now completely be performed in logarithmic representation:

$$\tilde{b}_j(\boldsymbol{x}) \approx \min_k \{\tilde{c}_{jk} + \tilde{g}_{jk}(\boldsymbol{x})\}$$

7.2 Lower Bounds for Probabilities

The logarithmic representation of probability values presented in the previous section can, however, only transfer numerically vanishing quantities into a range of values that can be manipulated by digital computers. If "'real" zeros appear, e.g. probabilities of unobserved events, these would need to be mapped to the largest representable positive floating-point number[5]. In the same way as the equivalent quantities in linear representation, such values dominate all calculation processes, in which they are used. Effectively this leads to situations where states or state sequences, in the evaluation of which a single vanishing probability occurs, can no longer be part of any solution.

Similarly to the creation of statistical language models (cf. section 6.5), however, such a behavior is in general not desired in statistical models, as it can never be safely assumed, that a probability computed as zero and the associated final rejection of certain solutions is reliable enough. Rather, one always assures that probability values as, e.g., transition or output probabilities, which were either defined individually or computed, may never fall below a certain minimal value $p_{\min}$. In negative-logarithmic representation this means to use a respective maximal value $\tilde{p}_{\max} = -\ln p_{\min}$. Such a lower bound is also referred to as the *floor* and the associated procedure as *flooring*.

As along paths through HMMs or n-gram models the accumulated scores grow depending on the length of the path, no absolute limitation is introduced here. Rather, one relies on the fact, that the use of suitably limited individual values in combination with the increased dynamic range of the logarithmic representation always ensure the computability and comparability of the results. In practice this assumption is *de facto* justified as concrete observation or word sequences are always of finite length.

The limitation of $b_j(\boldsymbol{x}) > b_{\min}$ already proposed in [195] causes two effects in the practical application. During the training phase of the model it is avoided that certain states are not considered for parameter estimation, the current parameters of

[5] When applying equation (7.1) in a naive way, even an error in the floating-point computation would result.

which are still totally unsuitable for the generation of certain observations. In decoding the limitation avoids that paths through states with vanishing output probabilities are immediately discarded as possible solutions. A principally similar effect can be achieved by limiting the mixture weights according to $c_{jk} > c_{\min}$.

Though this so-called flooring is a mandatory measure for the robust application of Markov models, it has the crucial disadvantage, that the necessary bounds need to be determined heuristically and, therefore, in general also need to be optimized in the context of the application. Furthermore, always interactions with methods for the limitation of the search space arise, as those try to eliminate less promising solutions from the search space early.

7.3 Codebook Evaluation for Semi-Continuous HMMs

When using semi-continuous HMMs a method for the evaluation of the densities within the commonly shared codebook has proven useful in practice, which can essentially be viewed as a technique for reducing the dynamic range of all quantities involved.

The output probability densities $b_j(\boldsymbol{x})$ of semi-continuous HMMs are defined as mixture densities over a common inventory of component densities (cf. equation (5.2)):

$$b_j(\boldsymbol{x}) \quad = \quad \sum_{k=1}^{M} c_{jk}\, g_k(\boldsymbol{x}) \quad = \quad \sum_{k=1}^{M} c_{jk}\, p(\boldsymbol{x}|\omega_k)$$

The individual mixtures $g_k(\boldsymbol{x})$ therein correspond to the class-conditional densities $p(\boldsymbol{x}|\omega_k)$ of the features depending on the codebook classes ω_k. Using Bayes' rule this quantity can be related to the posterior probability $P(\omega_k|\boldsymbol{x})$ of the respective class:

$$p(\boldsymbol{x}|\omega_k)\, P(\omega_k) \;=\; p(\boldsymbol{x}, \omega_k) \;=\; P(\omega_k|\boldsymbol{x})\, p(\boldsymbol{x})$$

In this expression still neither the prior probabilities $P(\omega_k)$ of the codebook classes nor the density $p(\boldsymbol{x})$ of the data itself is known. However, the fact can be exploited that in the semi-continuous modeling a global codebook is used. Ideally this should approximate reasonably well the distribution of all feature vectors — namely $p(\boldsymbol{x})$ — with the baseline distributions contained therein. Therefore, $p(\boldsymbol{x})$ can be approximated by a mixture density as follows:

$$p(\boldsymbol{x}) \quad \approx \quad \sum_{m=1}^{M} P(\omega_m)\, p(\boldsymbol{x}|\omega_m)$$

When further assuming that the partitioning of the feature space into classes ω_k generates regions of high density of approximately equal size, the prior probabilities $P(\omega_k)$ can be approximated by a uniform distribution. The conditional probability $P(\omega_k|\boldsymbol{x})$ of a class ω_k can then be computed depending on the feature vectors $\boldsymbol{x}$ as follows:

$$P(\omega_k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\omega_k)\,P(\omega_k)}{p(\boldsymbol{x})} \approx \frac{p(\boldsymbol{x}|\omega_k)\,P(\omega_k)}{\sum_{m=1}^{M} P(\omega_m)\,p(\boldsymbol{x}|\omega_m)} \approx \frac{p(\boldsymbol{x}|\omega_k)}{\sum_{m=1}^{M} p(\boldsymbol{x}|\omega_m)}$$

In practice this corresponds to a scaling of the density values $p(\boldsymbol{x}|\omega_k)$ such that they sum up to unity, a normalization operation which is widely used in the application of semi-continuous HMMs [95, p. 200]. The definition of the output probabilities is then achieved on the basis of the posterior probabilities of the codebook classes.

$$b'_j(\boldsymbol{x}) = \sum_{k=1}^{M} c_{jk}\,P(\omega_k|\boldsymbol{x}) = \sum_{k=1}^{M} c_{jk}\,\frac{p(\boldsymbol{x}|\omega_k)}{\sum_{m=1}^{M} p(\boldsymbol{x}|\omega_m)}$$

The semi-continuous model can, therefore, be considered directly as the combination of a "soft" vector quantization stage, which computes $P(\omega_k|\boldsymbol{x})$, and a discrete HMM with output probabilities $b_j(\omega_k) = c_{jk}$.

7.4 Probability Ratios

When decoding HMMs for biological sequences often a modification of the computation of output probabilities is performed. The output probability $b_j(o_k)$ for the symbol o_k in state j given by the model is normalized onto a suitable background distribution, which is identical for all states:

$$b'_j(o_k) = \frac{b_j(o_k)}{P(o_k)}$$

In the simplest case $P(o_k)$ is assumed as a random model, i.e. as a uniform distribution over all K possible symbols of the output alphabet:

$$b'_j(o_k) \approx \frac{b_j(o_k)}{\frac{1}{K}}$$

One obtains a modified probability $P'(\boldsymbol{O}, \boldsymbol{s}|\lambda)$ for the generation of the sequence $O_1, O_2, \ldots O_T$ considered and jointly running through a certain state sequence $s_1, s_2, \ldots s_t$ for a given model λ (see also equation (5.6), page 68):

$$P'(\boldsymbol{O}, \boldsymbol{s}|\lambda) = \prod_{t=1}^{T} a_{s_{t-1},s_t} \frac{b_{s_t}(O_t)}{P(O_t)} = \prod_{t=1}^{T} \frac{1}{P(O_t)} \prod_{t=1}^{T} a_{s_{t-1},s_t} b_{s_t}(O_t) = \frac{P(\boldsymbol{O}, \boldsymbol{s}|\lambda)}{\prod_{t=1}^{T} P(O_t)}$$

The normalization of the original output probabilities $b_j(o_k)$ onto a background distribution causes a non-linear length normalization of the path score $P'(\boldsymbol{O}, \boldsymbol{s}|\lambda)$. If only a random model is used, one obtains a linear length normalization by the factor of $\left(\frac{1}{K}\right)^T$.

As a logarithmic probability representation is also used in the analysis of biological sequences, the concrete score for the output of a symbol o_k in state j is computed as follows:

$$\tilde{b}'_j(o_k) = \ln \frac{b_j(o_k)}{P(o_k)} = \ln b_j(o_k) - \ln P(o_k) \tag{7.3}$$

The procedure is referred to as *log-odds scoring* in the literature (cf. [128], [52, pp. 108–110]). By this normalization step it is assured that the scores of the optimal state sequence determined for different biological sequences can be compared directly. Otherwise these would vary largely depending on the length of the observation sequence considered and, consequently, could not constitute a useful basis for a classification decision.

When for the purpose of simplification considering the decision over a single sequence only, a rejection criterion can easily be defined by using log-odds scoring. This approach in principle corresponds to the use of a very general separate HMM — a so-called *garbage model* — for modeling in total the observation sequences to be expected. If this model achieves a higher total probability than the model associated with a special class of patterns, a rejection is performed.

In the field of automatic speech recognition the use of garbage models can be found for the detection of unknown words. There one creates a general HMM, e.g. for sequences of speech sounds of the language considered, in addition to the models for words from the recognition lexicon. If the garbage model achieves the best score for a certain acoustic event, it can be assumed, that this part-of-speech was not sufficiently well described by the models of the “known” words and, therefore, is not part of the recognition lexicon — in other words it is “unknown” (cf. [4, 91, 112, 256]).

8 Configuration of Hidden Markov Models

For n-gram language models the vital configuration parameters are given by the size of the lexicon used and the length of the context to be considered. When creating HMMs for a certain application, in contrast, it is not immediately clear, what model size should be chosen, which type of emission modeling should be used, and whether the number of possible paths through the model could eventually be restricted in a suitable way.

8.1 Model Topologies

In the main application areas of HMM-based modeling — automatic speech and handwriting recognition as well as the analysis of biological sequences — the input data to be processed exhibits a chronological or linear structure. Therefore, it does not make sense for such applications to allow arbitrary state transitions within an HMM, as it is the case for so-called *ergodic* models (see figure 8.1(d)).

Rather, one assumes, that the models are run through in causal chronological sequence and, therefore, the model states can be sorted linearly. Transition probabilities to states, that describe data segments lying backwards in time, are constantly set to zero. In graphical representations of HMMs such edges, which are excluded from possible state sequences, are omitted for the purpose of simplification.

The most simple model topology that can be derived from this assumption is found in the so-called *linear* HMMs. As shown schematically in figure 8.1(a), in these models only transitions to the respective next state and to the current state itself are possible with some positive probability. By the self-transitions or *loops* one achieves that the model is able to capture variations in the temporal extension of the patterns described.

A larger flexibility in the modeling of duration is achieved, if also the skipping of individual states within a sequence is possible. One then obtains so-called *Bakis* models. This topology, which is widely used in the field of automatic speech and handwriting recognition, is exemplarily depicted in figure 8.1(b). For the modeling of acoustic events Bakis models offer the possibility to describe to some limited extent

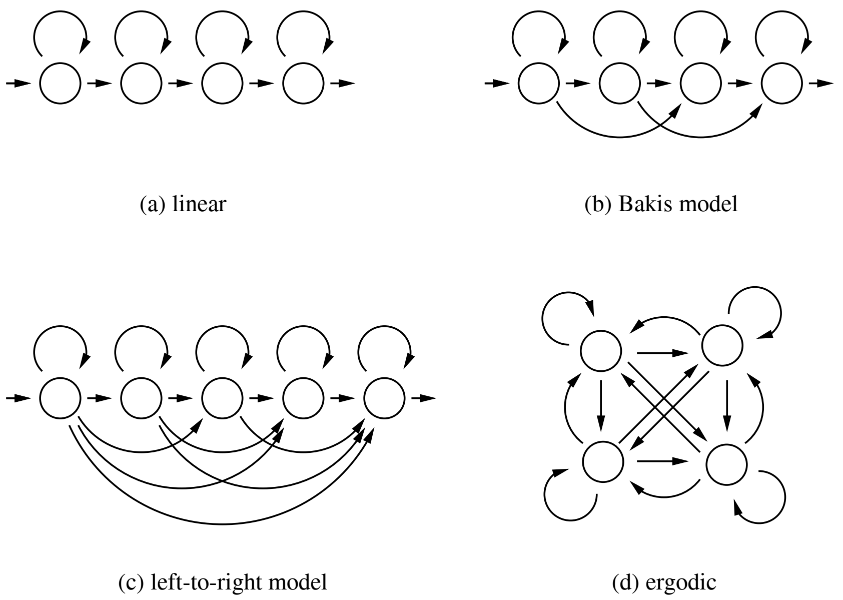


Fig. 8.1. Schematic representation of different HMM topologies: (a) linear model, (b) Bakis model, (c) left-to-right model, and (d) completely connected structure of an ergodic model.

articulatory reductions caused by mutual influences between neighboring segments, which lead to the omission of short portions of the signal.

Larger variations in the temporal structure of the data can be described by so-called *left-to-right* models. In this model topology, which is shown in figure 8.1(c), longer parts of the data to be processed may be missing as an arbitrary number of states may be skipped in forward direction within a sequence. Only jumping back to "past" states within a model is not allowed.

The more freely the model topology is chosen the more parameters are to be trained and the more variable are also the possible paths through the model. The choice of a certain baseline topology, therefore, always represents a compromise between flexibility and tractability. As not only the parameter training becomes more difficult for larger numbers of successors per model state, but also the effort in decoding is increased, linear models undoubtedly represent the most efficient model topology. In contrast, the best compromise between number of parameters, decoding cost, and flexibility is offered by Bakis models, which are used in many of today's systems (see also part III).

8.2 Modularization

In the theoretical view on HMMs always exactly one model λ with N states exists, which is considered as a unit. In practice it is, however, hardly possible to specify overall models directly without any suitable measures of modularization. The most obvious possibility for structuring larger HMMs is based on the segmentation units, which are considered in the respective application domain. In the field of speech or handwriting recognition these usually are spoken or written words from a certain language[1]. As will be shown in the next section, on this basis complex models for spoken utterances or written texts can be constructed.

The individual word model can in principle directly be created with one of the topologies presented in the previous section. However, therein the problem arises, that for every model as a whole sufficiently many training samples need to be available. For a system intended for the recognition of isolated digits or only a few command words a suitable sample set could still be created. However, for recognition systems with large vocabularies such a whole-word modeling is totally unfeasible.

It is, therefore, necessary to consider a further modularization of the segmentation units considered, in order to assure the trainability of the parameters of the overall model by a suitable reuse of partial models. Simultaneously one obtains an HMM, which is considerably more compact with respect to the number of independent parameters used (see also section 9.2). As this technique was developed and brought to perfection in the field of speech recognition, the resulting elementary models, which describe certain segments of words, are also referred to as *sub-word units*.

In recent years different methods for the definition of sub-word units were developed. The solutions found are frequently based on principles quite specific to the application. Therefore, we do not attempt to give a complete overview of the proposed approaches here. Rather, in the following sections methods shall be described, which are applicable in a relatively general way. For the more detailed treatment of the topic the interested reader is referred to the associated specialized literature and the references given therein (cf. e.g. [136, Chap. 6, pp. 91–114], [105, Chap. 3, pp. 39–56], [97, pp. 428–439]).

8.2.1 Context-Independent Sub-Word Units

The most simple methods for the definition of sub-word units are based on the segmentation of words into a sequence of short segments. It is most natural to use the orthographic representation for this purpose. One then obtains a segmentation into a sequence of characters, which is used in many systems for automatic handwriting recognition. In the field of speech recognition, however, it makes sense to start from a phonetic transcription of the words considered. The symbolically represented

[1] When analyzing biological sequences a somewhat different philosophy is followed in structuring HMMs. The most important of those techniques are described in section 8.4.

sequence of speech sounds associated with a word gives a relatively good indication of its actual realization as an acoustic event[2]. In both cases one obtains a quite compact inventory of approximately 50 to 100 units[3], which is ideally suited for the construction of arbitrary written or spoken words, and, therefore, can be used universally[4]. However, the accuracy of such a modeling is rather low in comparison, as the realizations of the individual units are heavily influenced by their context and they, therefore, may vary largely in their concrete realizations.

The correct number of states for context independent phone or character units results from the length of the associated signal segments to be expected. In linear topology the number of model states, furthermore, corresponds to the minimal length of possible events, as all states have to be passed through exactly once. Thus for speech recognition applications one obtains three to six states and up to approximately 15 for character models in handwriting recognition.

8.2.2 Context-Dependent Sub-Word Units

The main drawback of simple phone or character models lies in the fact, that they can not capture sufficiently the variability of the units described, which results from their embedding in different contexts. In the early days of automatic speech recognition one tried to tackle this problem among others by using longer sub-word units or by defining segment boundaries between partial models in regions of the signal, which were assumed to be approximately stationary (cf. e.g. [136, pp. 93–94]). An elegant method, which became a standard procedure meanwhile, consists in discriminating elementary units simply depending on their respective contexts. However, the context itself is not included in the acoustic event modeled. Thus one obtains so-called *context-dependent sub-word units*.

The most prominent representatives of this kind of models are the so-called *triphones*, which date back to Schwartz and his colleagues at BBN ([212, 213], cf. also [136, pp. 95–96]). They correspond to phone units in the context of the left and right immediately neighboring speech sound, so that three phone symbols are necessary to uniquely define a triphone. Thus a triphone in the same way as a *monophone* — i.e. a context-independent phone model — describes the realization of only a *single* speech sound, however, of a very special one.

[2] In case that no phonetic transcription is available or its generation causes too much effort, even in the field of automatic speech recognition satisfactory results can be achieved based on an orthographic modeling of words [205, 209].

[3] For English one uses — depending on the basis of definition — approximately 45 different phonetic units, for German the size of the inventory ranges from 40 to 50. In character or handwriting recognition for most languages models for all characters of the alphabet — in upper and lower case, if necessary — and for digits have to be created. For English one, therefore, obtains 62 elementary models and 69 for German including the umlauts.

[4] For character or handwriting recognition the inventory of character and digit models needs to be complemented by HMMs for punctuation symbols and white space. In speech recognition systems it is likewise indispensable to define a model for speech pauses. Furthermore, it may make sense to use additional models for human and non-human noises, hesitations, and other spontaneous speech effects in challenging applications (cf. e.g. [210]).

We want to illustrate the resulting triphone modeling with the example of the word `speech`. In the phonetic transcriptions this word corresponds to the symbol sequence `/spitS/`, when using the SAMPA alphabet (cf. [233]). Without the consideration of the context the necessary model for the speech sound `/i/` would not be distinguished from occurrences in, e.g., `achieve` (`/@tSiv/`), `cheese` (`/tSiz/`), or `reality` (`/riEl@ti/`). The corresponding triphone `p/i/t`, however, restricts the use of this model quite exactly to the respective phonetic context.

The basic idea of the triphone modeling can be generalized in different ways. In the case that one wants to describe contexts in even more detail, this can be achieved by specifying a larger number of context phones. Principally arbitrarily long contexts are allowed in the so-called *polyphones* [208]. Two left and right context phones each are considered by so-called *quinphones*, as they are, e.g., used in the speech recognition system BYBLOS (cf. [18], see also section 13.2).

The main advantage of context-dependent models, namely their high degree of specialization achieved, also represents their main disadvantage. When starting from an inventory of approximately 50 phonetic units one obtains a set of 125 000 potential triphones. Even if not all of these phone combinations can actually occur in the data, it is nevertheless hardly possible in practice to supply a sufficient amount of training material for each of these highly specialized models. Therefore, comparable models or similar model parameters need to be grouped together, in order to ensure the trainability. Different methods for achieving this will be presented in section 9.2. By the techniques applied in principle always the context restrictions are relaxed, so that modeling units result that can be used more generally. In conjunction with triphone models one then arrives at *generalized triphones* ([136, pp. 103–106], [137], cf. also [97, pp. 432–436]).

Though context dependency is clearly an issue in any larger HMM architecture built on the basis of elementary units, it has to date only been applied extensively in the field of automatic speech recognition and to some extent also for the online recognition of handwriting (cf. [125, 224]). The gains achieved for offline recognition of handwriting are, however, quite small, as the nature of the contextual influence can not be as clearly described yet for this task (cf. [69]).

8.3 Compound Models

Already in the subdivision of models for spoken or written words we have implicitly assumed, that more complex models can be created from existing partial HMMs by concatenation. Such construction principles are either explicitly or implicitly applied in order to define compound models for different recognition tasks. This principle is also the basis of Profile-HMMs, which will be the topic of the following section 8.4. However, due to their rather special nature these models will be described separately.

When by concatenation of the respective sub-word units HMMs for all words of the recognition lexicon were created, a total model for the recognition of isolated spoken or written words can be defined by a parallel connection by all individual word models. Figure 8.2(a) schematically shows the resulting model structure. All

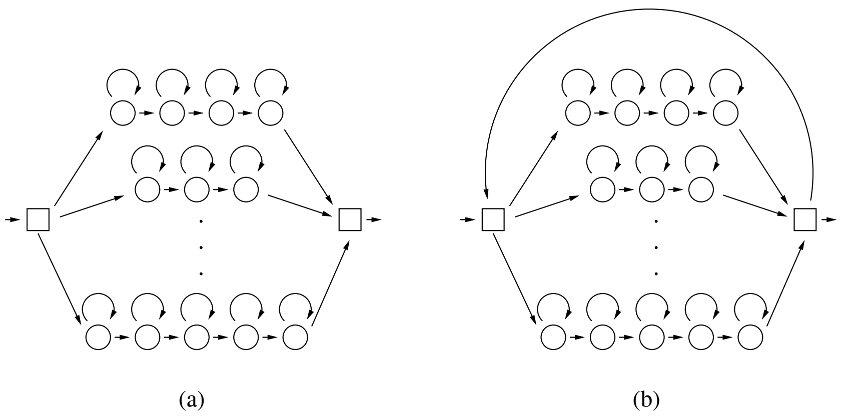


Fig. 8.2. Schematic representation of HMM structures for (a) isolated and (b) connected word recognition. Model states are represented by circles and non-emitting states by squares.

start and end states, respectively, of the individual word models are also start and end states of the total model. The representation of such complex HMMs can be considerably simplified by the introduction of so-called *non-emitting states*, which do not generate any outputs and have uniformly distributed transition probabilities to their respective successor states. They merely serve for grouping together of edges, so that the simplified total HMM has only one start and end state, respectively.

In automatic speech recognition the importance of systems for isolated word recognition is constantly decreasing. For the automatic processing of forms, in contrast, one can frequently assume that the writing contained in a certain field should correspond to exactly one word or phrase. Also in the recognition of machine-printed or handwritten texts in many cases the text lines are first segmented into a sequence of words. Subsequently an HMM for the recognition of words typed or written in isolation can be applied.

However, for the recognition of continuously spoken speech using HMMs always arbitrary sequences of words have to be processed by the statistical model. When adding a looping edge to the isolated word recognition model, which connects the end state of the model to the start state, a model structure results that is represented in figure 8.2(b) . In order to restrict the search through such a model performed during decoding to plausible word sequences, usually in addition to the HMM an n-gram language model is used (see also chapter 12).

In case that the word or segment sequences to be expected in a certain domain are heavily restricted or formalized, these can also be directly represented as a graph of partial HMMs. Such a model architecture was first used successfully in the speech recognition system HARPY [148]. Due to the inherent limitations of HMMs in this way only regular languages can be captured, as the resulting total model also is just another HMM. Figure 8.3 shows the example of a simple grammar, which is coded

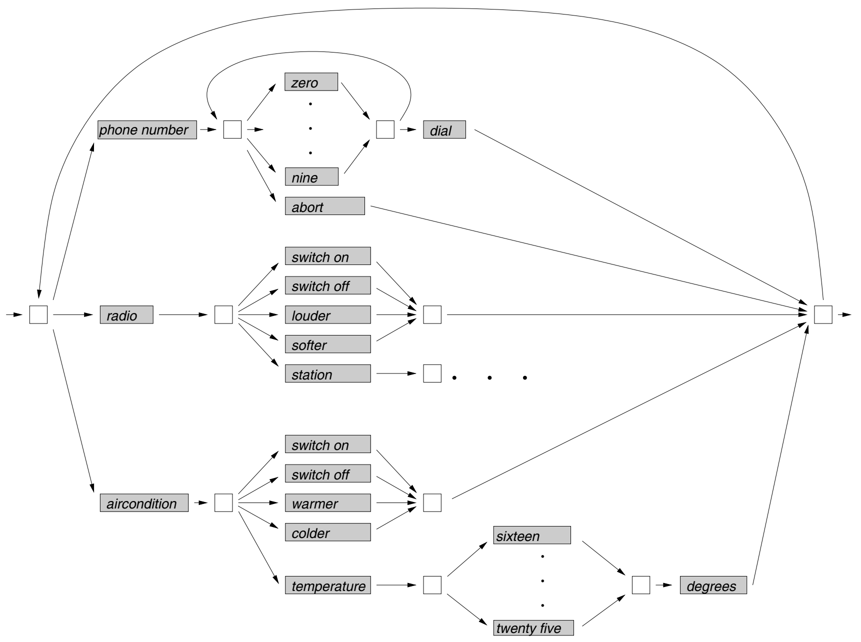


Fig. 8.3. Example of a grammar coded into the structure of an HMM that could be used by a speech recognition system for controlling non-safety-relevant functions in a car.

within the HMM structure. A similar model could be used in the near future in luxury cars for the control of so-called non-safety-relevant vehicle functions — e.g. mobile phone, radio, or air condition — by spoken language.

As during the decoding of grammar HMMs only word sequences can be found which are valid in the respective domain, such models have difficulties in rejecting invalid input. The probabilistic restriction of potential word sequences by a language model is, in contrast, less vulnerable in this respect.

8.4 Profile HMMs

The most widely used HMM structure for applications in the domain of bioinformatics consists in the so-called *profile HMMs* proposed by Krogh and colleagues ([129], cf. also [52, 55]). The structuring of the models is immediately derived from the data, which form the basis of the parameter training. Similarities between proteins are analyzed or represented, respectively, by a position-wise alignment of sequences of amino acids. One obtains so-called *multiple alignments* (see also section 2.3 page 22). Within these the respective positions can be identified, where different sequences exhibit large similarities and thus form a so-called consensus for all proteins considered. Figure 8.4 shows an example of a multiple alignment with

```
HBA_HUMAN     ...VGA--HAGEY...
HBB_HUMAN     ...V----NVDEV...
MYG_PHYCA     ...VEA--DVAGH...
GLB3_CHITP    ...VKG------D...
GLB5_PETMA    ...VYS--TYETS...
LGB2_LUPLU    ...FNA--NIPKH...
GLB1_GLYDI    ...IAGADNGAGV...
                 ***  *****
```

Fig. 8.4. Section from the multiple alignment of seven globins shown in figure 2.8 page 25. The columns marked with * are treated as matches in the associated profile HMM (after [52, p. 106]).

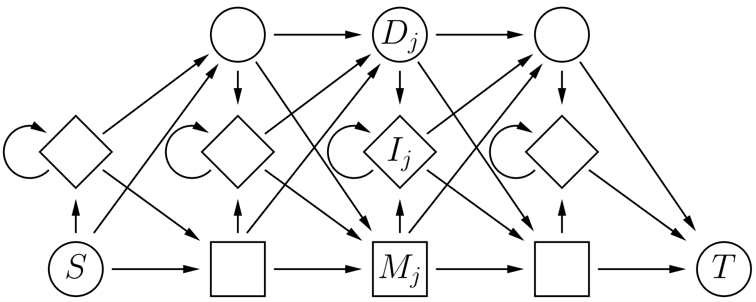


Fig. 8.5. Schematic representation of the structure of a profile HMM as it is obtained on the basis of a multiple alignment. Match states are represented by squares, insert states by rhombuses, and delete states by circles. Additionally, for the grouping together of edges one non-emitting start and end state exist.

the respective consensus columns marked. Every row corresponds to the amino acid sequence of a protein with similar biological function within a cell. As in concrete sequences parts may be missing and others may appear in addition to the consensus, also insertions and deletions must be allowed when building multiple alignments.

In a profile HMM now exactly three types of states exist, which directly correspond to these possible distinctions. The so-called match states describe a position within an amino acid sequence, which belongs to the consensus with respect to the family of sequences considered, i.e. appears in very similar form in the majority of training samples. In order to insert or delete symbols in or from a sequence, so-called insert and delete states are used, respectively. Figure 8.5 shows a profile HMM as it is obtained for the multiple alignment from figure 8.4. Additionally, the model contains specially marked start and end states, which in the same way as delete states do not generate emissions[5]. The possible state transitions within the model allow for passing through from left to right. By means of the delete states it is possible to

[5] In principle these are ordinary non-emitting states. However, in this case transition probabilities are used, which are specially adapted to the model.

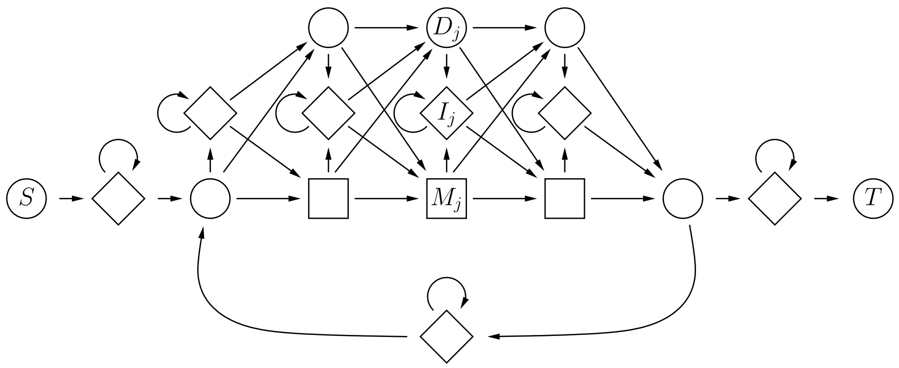


Fig. 8.6. Schematic representation of the extended structure of a profile HMM for the detection of eventually multiple partial sequences embedded into a longer sequence.

skip one or more match states. Additionally, between match states as well as at the beginning and end of the model arbitrarily many amino acids can be inserted into a sequence via the insert states. The required discrete output probability distributions of the match states are estimated on the consensus positions of the multiple alignments, which defines the protein family to be modeled. In analogy the distributions for modeling symbol insertions are obtained.

In a similar way as a model for the recognition of isolated words a profile HMM can be used for determining the degree of similarity between an unknown sequence and the protein family, which is described by the model[6]. Furthermore, an existing multiple alignment can automatically be extended with alignments of new sequences by using a profile HMM trained on it. Therefore merely the optimal state sequence through the model needs to be determined, which then yields the association of the individual sequence elements with the positions in the alignment.

The insert states at the beginning and the end of the model can describe a prefix or suffix of a sequence to be analyzed. The corresponding output distributions are, however, specialized on the actual initial and final sequences occurring in the respective multiple alignment. Therefore the finding of eventually more partial sequences, that match the structure of the protein family, within a longer sequence of amino acids is only possible with an extension of the model. This was proposed by Eddy and colleagues and is part of the analysis tool HMMER ([55], see also section 15.1 page 222). The principle structure of such a model is shown in figure 8.6. It uses additional insert states that serve to bridge partial sequences, which have nothing in common with the amino acid sequences described by the core model. A looping edge allows — in a similar way as in connected word recognition — to describe a sequence of the data modeled. Thus it is possible to detect matches with members of

[6] With PFAM there exists a library of ready-made profile HMMs which is available on the Internet [10].

the protein family considered a several positions within a longer sequence of amino acids.

8.5 Modeling Emissions

After the structure of the HMMs to be used is specified for a certain application, it remains to be decided, how the associated modeling of the model's emissions is to be achieved.

In the field of bioinformatics this decision is rather easy, as there the data to be processed usually consist of discrete symbols of DNA base pairs or amino acids. Unless additional properties of the data are considered (cf. e.g. [189]), discrete output distributions using four or 20 different symbols, respectively, are sufficient.

When describing continuous feature vectors in the fields of speech and handwriting recognition, however, mixture models need to be used (see also section 5.2). Unfortunately no generally applicable method exists, though, for specifying the possible degrees of freedom of the models to be used in a suitable way. Rather experience in dealing with the respective data and frequently also the experimental evaluation of different parametrizations are required.

In particular it is necessary to determine the degree of specialization of the output probability densities, i.e. to specify, how many model states define mixtures based on the same inventory of component densities. Additionally, the sizes of the respective codebooks must be determined. These decisions in general represent a compromise between the precision of the model, its generalization capabilities, and the effort needed for decoding, i.e. the computation time.

In semi-continuous models usually some hundred up to a few thousand densities are used for the common codebook shared by all states. The more individually mixture densities are associated with model states, the less training samples are available and the less component densities can be estimated. As a single normal density per model state is, however, hardly appropriate, in continuous HMMs that do not use mixture tying generally 8 to 64 component densities are used.

In the same way as in the use of context-dependent modeling units the danger is also for HMMs on the basis of mixture models, that one tries to estimate too many independent density models on limited data. However, also to output probability densities comparable methods can be applied as they are used for the optimization of HMM state spaces (see also section 9.2). How the modeling of emissions is concretely done for typical recognition systems in different application scenarios will also be covered in the course of the system descriptions in part III.

9 Robust Parameter Estimation

When applying HMMs in practice one — as always in the field of statistical pattern recognition — is faced with the problem to robustly estimate the parameters on the available training samples. Yet the situation is not as severe as in the case of n-gram models, for which no useful models could be created without suitable measures (cf. section 6.5). However, also with HMMs one is confronted with the so-called *sparse data problem* when working with more complex model architectures. Then chances are high, that the model parameters can either no longer be computed due to numerical reasons, or that an overfitting of the model to the sample data considered occurs. This can in extreme cases lead to the situation that the models estimated have learned the sample set "by heart", i.e. they describe nothing but known samples. Such a behavior can, however, be diagnosed by an accompanying evaluation on an independent cross-validation set and the training procedure can then be stopped at a suitable position.

Yet such a decision via "emergency break" is not helpful in order to modify the configuration of the model purposefully for a simplified parameter estimation. Therefore, it would be extremely useful in practice, to have a well-founded basis for deciding, how much training material is necessary for dimensioning a given model. Unfortunately, this question can be answered from the mathematical side only by the law of large numbers, which was — from the view of statistical pattern recognition — rephrased pointedly by Robert L. Mercer (IBM) [156]:

"There is no data like more data!"

A very rough estimate of the amount of data necessary can be derived from the assumption, that there exists a linear relationship between the number of model parameters P and the minimal size T of the sample set, i.e. that P is directly proportional to T:

$$T \propto P \tag{9.1}$$

Even though such a trivial model is hardly justified from a mathematical point-of-view, it can nevertheless be used quite well in practice for approximately defining lower bounds on the number of training samples that need to be available per model parameter[1]. Furthermore, the relationship can also be exploited in reverse for approximately defining, how many model parameters can be trained from a sample set of given size. As the model quality in general increases with an increased number of parameters, one is always interested in the most special model, which can still be estimated robustly.

As soon as it is clear that a certain parameter set can hardly be trained in a satisfactory way with the available sample data, the factors need to be considered that cause such a situation of data sparseness. Primarily the following three reasons or a combination of them have to be taken into consideration:

Model Complexity

It has been tried to estimate too many independent models or modeling parts on the basis of the available data. In classification problems this number usually is bounded from below by the number of pattern classes that need to be distinguished. Frequently, however, many more considerably more specialized models are used for representing variants of the patterns considered more precisely and thus improving the model quality as a whole.

Dimensionality

The number of features used is too large. Especially if features are determined heuristically, a large number of characteristic quantities can be computed, that – considered in isolation – make a certain contribution for the solution of the problem. The adding of new components to feature vectors, however, also leads to a considerably increase in the demand for sample data. If this can not be satisfied, the modeling quality decreases despite the putatively improved description of the data, as the parameters can no longer be estimated robustly — a situation which is often referred to as the so-called *curse of dimensionality* (cf. e.g. [49, p. 95]).

Correlation

Mutual dependencies exist within the data to be modeled which need to be described by additional parameters. Here the following considerations will concentrate on the correlation between components of the feature vectors, as they are frequently found in heuristically generated feature sets. Correlations over the temporal sequence of the features, however, are not captured parametrically in HMMs, but approximately via the model structure.

The effective size of the parameter set required consequently is defined by the specificity of the model, the dimension of the feature space, and the need for describing correlations in the data. An improvement in the trainability can in general be achieved by reducing the complexity with respect to one or more of these modeling aspects.

[1] A proportionality factor of, e.g., 5 achieves good results in the estimation of mixture models. For establishing individual HMM states usually 50 to 100 samples are sufficient.

In the following section we will at first consider analytical methods, which allow to optimize a given feature representation such, that the model built on top of it requires less parameters without a change in its descriptive quality. For this purpose either internal correlations are approximately eliminated or the dimension of the feature vectors is reduced in total. Then methods for the reduction of model complexity are presented in section 9.2. They are all based on the principle of identifying "similar" parameters within the model, by applying expert knowledge or by means of automatic procedures, and then merging these parameters during the training process. Thus the number of parameters required is reduced, so that the trainability and eventually also the quality of the model as a whole can be improved.

Additionally, for all iteratively optimizing training procedures, as they are used for HMMs, it is of fundamental importance to start the optimization process with a suitable initial estimate. Therefore, methods for determining initial model parameters for HMMs will be presented at the end of the chapter.

9.1 Feature Optimization

The computation of features, which are most suitable for certain signal analysis tasks, is the topic of countless publications in the field of pattern recognition. Here we want to deal with this problem not in such fundamental form, however, but only consider methods, which allow to optimize the properties of an existing feature representation with respect to the model used.

Therefore, we will assume, that a feature extraction rule exists, which is principally suited for the task at hand. This procedure, which will mostly consist of a combination of heuristic methods, yields a baseline representation for the signals to be analyzed in the form of n-dimensional feature vectors $\boldsymbol{x} \in \mathrm{I\!R}^n$. The probability distribution of this data is in general unknown. However, we also assume that the required distribution parameters can sufficiently accurately be determined on the basis of a representative set of sample vectors:

$$\omega = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N\}$$

In general the statistical properties of these baseline features are not optimally matched with the subsequent modeling. Therefore, in the following we will consider a class of methods, which by a suitable transformation carry out an optimization of the data representation with respect to the properties and capabilities of the subsequent modeling.

Therein, we will limit ourselves to *linear* transformations, which map every feature vector $\boldsymbol{x}$ from the baseline representation onto a corresponding feature vector $\boldsymbol{y}$ with equal or smaller dimension. Such transformations are completely defined by a transformation matrix $\boldsymbol{T}$:

$$\boldsymbol{y} = \boldsymbol{T}\boldsymbol{x} \qquad \text{with} \quad \boldsymbol{x} \in \mathrm{I\!R}^n, \boldsymbol{y} \in \mathrm{I\!R}^m, \boldsymbol{T} \in \mathrm{I\!R}^m \times \mathrm{I\!R}^n \quad \text{where} \quad m \leq n$$

The limitation to linear transforms is made for reasons of simplicity and, because the properties of such operations are especially well understood mathematically.

Furthermore, we always want to apply the feature transformation in a *centered* way, i.e. after a compensation of the sample mean $\bar{\boldsymbol{x}}$:

$$\boldsymbol{y} = \boldsymbol{T}(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

From this modification only translation of the transformed feature vectors by the vector $-\boldsymbol{T}\bar{\boldsymbol{x}}$ results, which can easily be corrected again, if necessary. The fundamental statistical properties of the data, however, are not affected by this operation, and a centered feature transform is better suited as a starting point for the further considerations.

After applying a certain transformation to the original data ω one obtains a transformed sample set

$$\tilde{\omega} = \{\boldsymbol{y}_k \mid \boldsymbol{y}_k = \boldsymbol{T}(\boldsymbol{x}_k - \bar{\boldsymbol{x}}), 1 \leq k \leq N\}$$

which replaces the original one in the further modeling process. The goal of all methods presented is to determine the transformation $\boldsymbol{T}$ such that “noise”, which is irrelevant for the modeling, will be eliminated from the data to the greatest possible extent. Simultaneously, however, relevant differences shall be preserved and thus the modeling process shall be simplified in the end.

9.1.1 Decorrelation

As the baseline representation of the features is in general the result of a heuristic procedure, it has to be assumed, that statistical dependencies exist between the individual components x_i of the feature vectors $\boldsymbol{x}$. Figure 9.1(a) illustrates this situation with a simple two-dimensional example. In the distribution shown large values of the first feature x_1 mostly go along with the same of the second feature x_2. The data shown can be described by a two-dimensional normal distribution with mean $\boldsymbol{\mu} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and covariance matrix $\boldsymbol{K} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$. When training the respective model thus six parameters need to be estimated from the example data.

In the distribution shown in figure 9.1(b) the components of the feature vectors are, in contrast, statistically independent and therefore in particular uncorrelated. This data, therefore, can be described by a simplified model, which only uses a diagonal covariance matrix. Thus two model parameters can be saved and a robust estimation of the remaining parameters — according to the rule-of-thumb put up at the beginning in equation (9.1) — can be achieved on approximately 30% less samples.

Of course such a reduction has a more notable effect in real systems, e.g. from the domain of automatic speech recognition. With an inventory of $M = 10\,000$ normal distributions and a feature dimension of $n = 39$ one obtains $10\,000 \cdot \frac{3n+n^2}{2} = 8.19 \cdot 10^6$ independent parameters for a system with full covariance matrices, but only $10\,000 \cdot 2n = 7.8 \cdot 10^5$ degrees of freedom when using diagonal covariance matrices. Thus when using “diagonal” models either a tenth of the training material is sufficient or the ten-fold number of densities can be estimated.

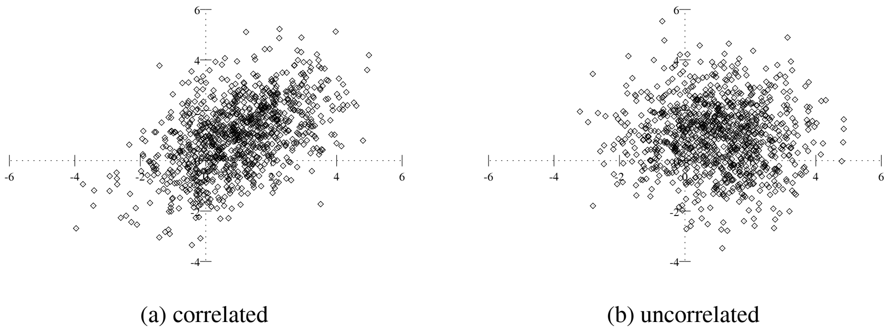


Fig. 9.1. Examples of a simple two-dimensional distribution with (a) correlated and (b) uncorrelated feature vector components.

However, statistical dependencies between components of a feature vector cannot principally be excluded in practice. Though the possibility exists to at least achieve a decorrelation of the features by a transformation of the feature space. In the case of normally distributed data this is equivalent to the creation of statistically independent feature vectors.

Principal Component Analysis I

The so-called *principal component analysis*[2] (PCA) computes for a distribution of data vectors given either parametrically or empirically a new coordinate system, which is oriented such, that the correlation between the vector components vanishes. Furthermore, the new coordinates are chosen such, that the largest variance of the data in the transformed feature space occurs along the first coordinate and constantly decreases for higher vector components (cf. e.g. [45, pp. 302ff], [50, pp. 115–117]). Those vectors defining the coordinates of the transformed vector space are referred to as principal components of the data analyzed.

The computation of principal components consists in an analysis of the scatter characteristics of the data considered. These are characterized by the so-called *scatter matrix*. In case that only a single distribution is considered this is equivalent to its covariance matrix[3]. The total scatter matrix $\boldsymbol{S}_T$ of a sample set $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N$ of data vectors is defined as (cf. also equation (3.9)):

[2] The method is also known as *Karhunen-Loève transform* in the literature. However, there the transformation is not applied in a centered way, so that the Karhunen-Loève transform is only completely equivalent to the principal component analysis for zero-mean data.

[3] In presentations that assume zero-mean data or that compensate the sample mean during the transformation, the so-called *correlation matrix* is used, which is defined as follows: $\boldsymbol{C} := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{x}_i^T$

Orthonormal Transformations

A special class of linear transformations is formed by the *orthonormal transforms*. They are equivalent to the expansion of the data $\boldsymbol{x} \in \mathbb{R}^n$ to be transformed according to a system of orthonormal base vectors. Such a transform, therefore, only causes a rotation of the original data onto a new system of orthogonal coordinates.

The rows of the transformation matrix $\boldsymbol{T}$ correspond exactly to the vectors $\boldsymbol{t}_i$, which form the new orthonormal basis of the $\mathbb{R}^n$:

$$\boldsymbol{T} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots \boldsymbol{t}_n]^T \qquad \text{with} \quad \boldsymbol{t}_i^T \boldsymbol{t}_j = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad ||\boldsymbol{t}_i|| = 1 \quad \forall i, j$$

Due to the pairwise orthogonality of the row-vectors of $\boldsymbol{T}$ one obtains

$$\boldsymbol{T}^T \boldsymbol{T} = \boldsymbol{I}$$

from which it follows directly, that for orthonormal transforms the inverse $\boldsymbol{T}^{-1}$ of the transformation matrix is identical with its transpose:

$$\boldsymbol{T}^{-1} = \boldsymbol{T}^T$$

By exploiting this property it can easily be verified, that the norm of the vectors is preserved by an orthonormal transformation:

$$||\boldsymbol{y}|| = \sqrt{\boldsymbol{y}^T \boldsymbol{y}} = \sqrt{[\boldsymbol{T}\boldsymbol{x}]^T \boldsymbol{T}\boldsymbol{x}} = \sqrt{\boldsymbol{x}^T \boldsymbol{T}^T \boldsymbol{T} \boldsymbol{x}} = \sqrt{\boldsymbol{x}^T \boldsymbol{x}} = ||\boldsymbol{x}||$$

This is equivalent to the fact, that Euclidean distances between data vectors are not changed by the transformation.

$$\boldsymbol{S}_T := \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \tag{9.2}$$

Here $\bar{\boldsymbol{x}}$ denotes the sample mean (cf. also equation (3.8)):

$$\bar{\boldsymbol{x}} := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}_i$$

After applying a transformation $\boldsymbol{T}$ one obtains for the transformed sample set $\tilde{\omega}$ the total scatter matrix:

$$\tilde{\boldsymbol{S}}_T = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{T}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})[\boldsymbol{T}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})]^T = \boldsymbol{T} \boldsymbol{S}_T \boldsymbol{T}^T$$

In order to decorrelate the data, therefore, a transformation $\boldsymbol{T}$ is sought, which "diagonalizes" $\boldsymbol{S}_T$, i.e. causes $\tilde{\boldsymbol{S}}_T$ to become a diagonal matrix. However, the relative position of the data vectors with respect to each other should remain unchanged during this operation. Therefore, the transformation in question needs to be orthonormal, as by this class of transformations Euclidean distances in the vector spaces involved are not affected (cf. sidebar *Orthonormal Transformations* on page 142).

Unfortunately, the property of the total scatter $\tilde{\boldsymbol{S}}_T$ being a diagonal matrix cannot be derived by formulating a criterion and analytically computing a suitable transformation. Rather, one needs to resort to the fact known from linear algebra, that every non-singular matrix can be brought to diagonal form by a suitably chosen orthonormal transform (cf. sidebar *Diagonalization of Symmetric Matrices* on page 145).

Thus the diagonalization of the total scatter matrix $\boldsymbol{S}_T$ is achieved my means of the transpose $\boldsymbol{\Phi}^T$ of its eigenvector matrix $\boldsymbol{\Phi}$, the columns of which consist of the n normalized eigenvectors $\boldsymbol{\phi}_i$ of $\boldsymbol{S}_T$. For the mere diagonalization of $\tilde{\boldsymbol{S}}_T$ the arrangement of these vectors to a matrix is in fact irrelevant. In the fact of the following considerations, however, which also include a reduction of the dimensionality, $\boldsymbol{\Phi}$ is constructed such, that the eigenvectors are arranged according to the size of the associated eigenvalue. The first column of $\boldsymbol{\Phi}$ thus corresponds to the eigenvector with the largest associated eigenvalue and the last column to the one with the smallest.

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots \boldsymbol{\phi}_n] \quad \text{with} \quad \boldsymbol{S}_T\boldsymbol{\phi}_i = \boldsymbol{\phi}_i\lambda_i \quad \text{and} \quad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \tag{9.3}$$

When applying this transformation to zero-mean data vectors

$$\boldsymbol{y} = \boldsymbol{\Phi}^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

one obtains the following transform of the scatter matrix:

$$\tilde{\boldsymbol{S}}_T = \boldsymbol{\Phi}^T\boldsymbol{S}_T\boldsymbol{\Phi} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & \mathbf{0} \\ & \lambda_2 & & \\ & & \ddots & \\ \mathbf{0} & & & \lambda_n \end{bmatrix}$$

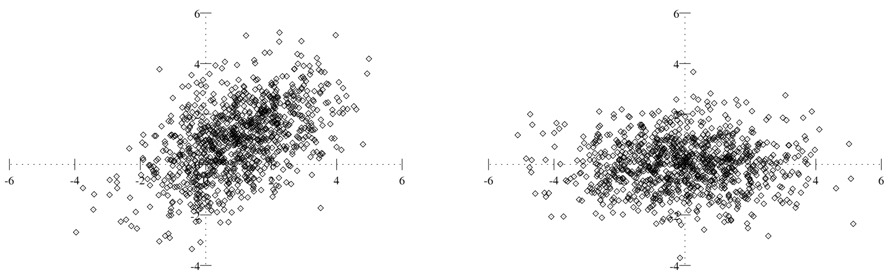
The total scatter matrix $\tilde{\boldsymbol{S}}_T$ of the transformed sample set thus corresponds to the eigenvalue matrix $\boldsymbol{\Lambda}$ of $\boldsymbol{S}_T$, as $\boldsymbol{\Phi}^T$ simultaneously is the inverse of $\boldsymbol{\Phi}$. In the transformed space the variance of the data along the coordinates is thus given by the eigenvalues λ_i. All correlations between vector components vanish, however. Therefore, the goal of bringing the scatter matrix of the optimized feature representation to diagonal form is achieved.

For the example used in the introduction one obtains

$$\boldsymbol{\Phi} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\Lambda} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

as eigenvector and eigenvalue matrix. The transformation of the sample data according to

$$\boldsymbol{y}_i = \boldsymbol{\Phi}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$$



(a) correlated (b) decorrelated

Fig. 9.2. Example for the decorrelation of a simple two-dimensional distribution: (a) original data (see also figure 9.1(a)) and (b) after the decorrelation by means of PCA.

yields the distribution shown in figure 9.2(b). The diagonalized total scatter matrix $\tilde{\boldsymbol{S}}_T$ of the transformed data is given by the eigenvalue matrix $\boldsymbol{\Lambda}$.

The decorrelation of a given feature set always causes the total scatter matrix to become a diagonal matrix and thus, considered globally, the correlations between the components of the transformed feature vectors to vanish. When assuming that the data are approximately normally distributed, one even obtains features, which are approximately statistically independent.

If such a data set is described by mixture models it is, therefore, justified — within certain limits — to use only diagonal covariance matrices and thus save a considerable number of model parameters. As, however, only *global* correlations are eliminated by principal component analysis, nevertheless within individual regions of high density within the data local statistical dependencies can occur, as shown exemplarily in figure 9.3. By the simplified modeling these are captured only inadequately or not at all. Usually, however, one gives precedence to the considerable reduction of the number of parameters over the possibility to model such class regions more exactly. In the example shown simply more baseline densities need to be estimated for the description of an "elongated" region in feature space when using diagonal models only.

Whitening

In practical applications it is in general undesirable that the dynamic range numerically varies largely between individual components of the feature vector. Therefore, it may make sense to also normalize the variance parts after the decorrelation of the features by means of the principal component analysis in order to simplify the further model building.

Diagonalization of Symmetric Matrices

Every non-singular symmetric matrix $\boldsymbol{Q}$ can be brought to diagonal form by means of a suitable orthonormal transformation $\boldsymbol{T} = \boldsymbol{\Phi}^T$ (cf. e.g. [80, pp. 27–28]). Here $\boldsymbol{\Phi}$ corresponds to the so-called *eigenvector matrix* of $\boldsymbol{Q}$:

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \dots \boldsymbol{\phi}_n] \quad \text{with } \boldsymbol{\phi}_i^T \boldsymbol{\phi}_j = \begin{cases} 1 & i = j \\ 0 & \text{otherwise} \end{cases} \quad \text{and } ||\boldsymbol{\phi}_i|| = 1 \, \forall i, j$$

The column vectors $\boldsymbol{\phi}_i$ of $\boldsymbol{\Phi}$ are formed by the normalized eigenvectors[4] of $\boldsymbol{Q}$ with the associated eigenvalue λ_i and thus satisfy the eigenvalue equation:

$$\boldsymbol{Q}\boldsymbol{\phi}_i = \boldsymbol{\phi}_i \lambda_i \qquad \forall i$$

On the basis of the eigenvalues λ_i the so-called *eigenvalue matrix* is defined in correspondence to the eigenvector matrix as follows:

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}$$

Thus as a generalization of the eigenvalue equation

$$\boldsymbol{Q}\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{\Lambda}$$

holds, and one obtains the following decomposition into eigenvector and eigenvalue matrix for $\boldsymbol{Q}$:

$$\boldsymbol{Q} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$$

When applying the transformation $\boldsymbol{\Phi}^T$ to $\boldsymbol{Q}$

$$\boldsymbol{\Phi}^T\boldsymbol{Q}\boldsymbol{\Phi} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T\boldsymbol{\Phi} = \boldsymbol{\Lambda}$$

one obtains as a result the diagonal matrix of the eigenvalues of $\boldsymbol{Q}$.

[4] The computation of eigenvalues and eigenvectors of matrices is a challenging numerical problem. Because of its relatively general applicability the most widely used method for solving this problem is the so-called *power method* (cf. e.g. [109, p. 233]). For symmetric matrices — as scatter matrices always are — in practice good results can also be achieved by the so-called *Jacobi method* (cf. e.g. [109, p. 196], [194, p. 360]).

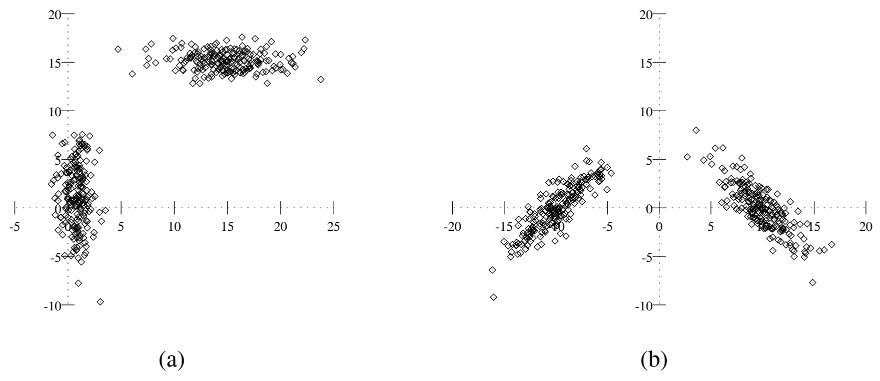


Fig. 9.3. Example of a data distribution with multiple regions of high density and local correlations (a) before and (b) after applying principal component analysis.

When additionally applying the transformation

$$\boldsymbol{\Lambda}^{-\frac{1}{2}} = \begin{bmatrix} \lambda_1^{-\frac{1}{2}} & & & 0 \\ & \lambda_2^{-\frac{1}{2}} & & \\ & & \ddots & \\ 0 & & & \lambda_n^{-\frac{1}{2}} \end{bmatrix}$$

to the data decorrelated by PCA, this effects that the final total scatter matrix $\hat{\boldsymbol{S}}_T$ will become the identity matrix $\boldsymbol{I}$.

$$\hat{\boldsymbol{S}}_T = \boldsymbol{\Lambda}^{-\frac{1}{2}} \tilde{\boldsymbol{S}}_T \boldsymbol{\Lambda}^{-\frac{1}{2}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\Phi}^T \boldsymbol{S}_T \boldsymbol{\Phi} \boldsymbol{\Lambda}^{-\frac{1}{2}} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\Lambda} \boldsymbol{\Lambda}^{-\frac{1}{2}} = \boldsymbol{I}$$

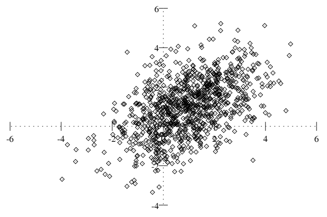
The total transformation of the data, which is necessary for achieving this, i.e. the combination of decorrelation and subsequent variance normalization is referred to as *whitening*:

$$\boldsymbol{z} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{y} = \boldsymbol{\Lambda}^{-\frac{1}{2}} \boldsymbol{\Phi}^T (\boldsymbol{x} - \bar{\boldsymbol{x}})$$

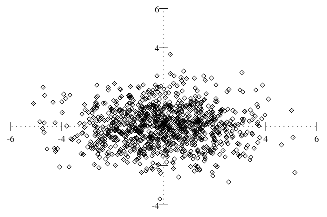
The variances in all coordinates are normalized to 1 by this transformation. In particular this means that after a whitening the global scatter characteristics of the data are invariant to additional orthonormal transformations, as the total scatter was made to become an identity matrix.

In contrast to principal component analysis this transformation is not orthonormal, so that Euclidean distances between the transformed vectors are *not* preserved. Intuitively this is also immediately clear, as the feature space is scaled along the coordinates by a factor of $\frac{1}{\sqrt{\lambda_i}}$, respectively.

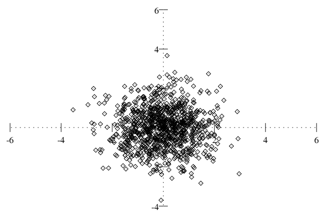
A whitening of the example data from figure 9.4(a) results in the distribution shown in figure 9.4(c). One obtains a radially symmetric point cloud, as the variance of the distribution is equal to 1 in both coordinates.



(a) original



(b) decorrelated



(c) variance normalized

Fig. 9.4. Example for the combined decorrelation and variance normalization by whitening: (a) original data, (b) after decorrelation, and (c) after variance normalization.

9.1.2 Dimensionality Reduction

As a second important method for optimizing feature representations in addition to a decorrelation of the data frequently a reduction of the dimensionality is performed. The methods applied build on top of each other, so that the decorrelation step is either implicitly or explicitly performed in the preparation phase of the actual dimensionality reduction.

Principal Component Analysis II

The simplest method for reducing the dimension of the features consists in not using all n eigenvectors of the scatter matrix $\boldsymbol{S}_T$ for the construction of the transformation matrix $\boldsymbol{\Phi}^T$ of the principal component analysis. Rather only the ones corresponding to the $m < n$ largest eigenvalues of $\boldsymbol{S}_T$ are selected (see equation (9.3)). Thus one obtains a mapping of the feature vectors $\boldsymbol{x} \in \mathbb{R}^n$ onto a lower dimensional space, in which the largest variance components of the original data are preserved.

Of course a certain error arises from this reduction of the vector dimension. This can be described quantitatively, if for every original vector $\boldsymbol{x}$ its reconstruction

$$\boldsymbol{x}' = \sum_{i=1}^{m} y_i \boldsymbol{\phi}_i$$

on the basis of the lower dimensional representation $\boldsymbol{y}$ is generated and the resulting average reconstruction error ϵ is calculated. It can be shown (cf. [45, p. 304-305]) that the expected error is given by the sum of the eigenvalues of those eigenvectors of $\boldsymbol{S}_T$ that were not considered for the construction of the transformation matrix $\boldsymbol{\Phi}^T$:

$$\epsilon = \mathcal{E}\{||\boldsymbol{x} - \boldsymbol{x}'||^2\} = \mathcal{E}\{||\sum_{i=m+1}^{n} y_i \boldsymbol{\phi}_i||^2\} = \sum_{i=m+1}^{n} \lambda_i$$

Thus selecting the eigenvectors associated with the largest eigenvalues for the transformation not only maximizes the variance preserved in the vector components but also minimizes the reconstruction error resulting from the dimensionality reduction. By considering the proportions of the eigenvalues λ_i it can be estimated, on which dimension $m < n$ the data available can be reduced with only negligible losses in the accuracy of the representation.

A reduction of dimensionality by PCA is, however, mainly used when analytically generating features directly from signal data and hardly for the optimization of existing feature representations with respect to mixture models or HMMs[5]. In fact, the global maximization of the preserved variance does not take into account the separability of the pattern classes considered.

Linear Discriminant Analysis

In contrast to a simple dimension reduction on the basis of global variance criteria the so-called *linear discriminant analysis* (LDA) tries to determine a feature space transform that improves the separability of the pattern classes considered while simultaneously reducing the dimensionality. (cf. e.g. [80, pp. 441–459], [50, pp. 117–124]). For the computation of the transformation matrix, therefore, criteria are applied that characterize the distribution of class regions and their separability on the basis of the features available. Consequently, for applying LDA a *labeled sample set* is mandatory. Intuitively a transformation is sought that generates class regions, which are as compact as possible, without at the same time modifying the total variance of the data.

For the further considerations we will, therefore, assume, that within the available sample set ω subsets ω_κ are defined, which contain features of patterns originating from a class Ω_κ only:

$$\omega = \bigcup_\kappa \omega_\kappa$$

The compactness of these classes can be described by the average variance of the features *within* the respective regions. From this one obtains the so-called within class scatter matrix $\boldsymbol{S}_W$:

[5] In the field of automatic speech recognition a further important reason for this is, that the *cepstral coefficients* used as features in the majority of systems can already be assumed to be approximately decorrelated [158].

$$\boldsymbol{S}_W := \sum_\kappa p_\kappa \sum_{\boldsymbol{x} \in \omega_\kappa} (\boldsymbol{x} - \bar{\boldsymbol{x}}_\kappa)(\boldsymbol{x} - \bar{\boldsymbol{x}}_\kappa)^T \tag{9.4}$$

Here $\bar{\boldsymbol{x}}_\kappa$ denotes the mean of the respective class region, i.e. the conditional sample mean of the feature vectors from class Ω_κ. The class-conditional scatter matrices are weighted according to the prior probabilities p_κ of the respective classes, which can be estimated from the proportion of feature vectors from Ω_κ contained in the sample set:

$$p_\kappa = \frac{|\omega_\kappa|}{|\omega|}$$

By means of the so-called between class scatter matrix $\boldsymbol{S}_B$ the relative position of the individual class regions with respect to each other can be described. It is, however, not completely clear, how $\boldsymbol{S}_B$ should be defined. Therefore, different approaches can be found in the literature (cf. e.g. [79, p. 260]). The simplest method is to compute a scatter matrix on the basis of the individual class centers $\bar{\boldsymbol{x}}_\kappa$:

$$\boldsymbol{S}_B := \sum_\kappa p_\kappa (\bar{\boldsymbol{x}}_\kappa - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_\kappa - \bar{\boldsymbol{x}})^T \tag{9.5}$$

This definition also allows to derive an analytical relationship between the tree scatter matrices introduced so far. The total scatter matrix is then obtained as the sum of the between and the within class scatter matrix:

$$\boldsymbol{S}_T \;=\; \boldsymbol{S}_B \;+\; \boldsymbol{S}_W$$

As optimization criteria for the LDA different measures may be considered, all of which basically try to capture the relation between within and between class scatter in the sample set (cf. e.g. [80, pp. 446–447]). On the basis of such a compactness criterion then a between class scatter as small as possible with a distance of the pattern classes among each other being as large as possible at the same time can be defined as the optimization task.

In the literature for this the following criterion is used most frequently[6]:

$$\operatorname{tr}\{\boldsymbol{S}_W^{-1}\boldsymbol{S}_B\} \rightarrow \max!$$

After a quite lengthy mathematical derivation, which the interested reader can find in the respective specialized literature (cf. e.g. [80, pp. 448–459]), it can be shown, that this maximization problem — similar as in the case of the principal component analysis — leads to the solution of an eigenvalue problem. The transformation matrix $\boldsymbol{\Phi}^T$ of the LDA sought, is created from the $m < n$ eigenvectors ϕ_i of the matrix $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ corresponding to the largest eigenvalues:

$$\boldsymbol{\Phi} = [\phi_1, \phi_2, \ldots \phi_m] \qquad \text{with} \quad \boldsymbol{S}_W^{-1}\boldsymbol{S}_B\phi_i = \phi_i\lambda_i \quad \text{and} \quad \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \tag{9.6}$$

[6] As shown in [87], this criterion can also be applied for the evaluation of different feature sets with respect to their suitability for a certain modeling task.

In order to avoid problems in the eigenvalue computation for $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$, which may result from the necessary inversion of $\boldsymbol{S}_W$ or from the fact, that $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ not necessarily is a symmetric matrix, the solution can also be reduced to the simultaneous diagonalization of the matrices $\boldsymbol{S}_W$ and $\boldsymbol{S}_B$ (cf. e.g. [80, pp. 31–34]).

In this method the computation of the transformation matrix for the LDA is performed in two steps. First the between scatter matrix $\boldsymbol{S}_W$ is transformed into an identity matrix by applying a whitening. Therefore one calculates the eigenvalue and eigenvector matrix of $\boldsymbol{S}_W$:

$$\boldsymbol{S}_W\boldsymbol{\Phi} = \boldsymbol{\Phi}\boldsymbol{\Lambda}$$

Then the original data is transformed according to

$$\boldsymbol{y} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

and as the new between class scatter matrix one obtains the identity matrix:

$$\tilde{\boldsymbol{S}}_W = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}^T\boldsymbol{S}_W\boldsymbol{\Phi}\boldsymbol{\Lambda}^{-\frac{1}{2}} = \boldsymbol{I}$$

Of course the between class scatter matrix $\boldsymbol{S}_B$ is also modified by this process, but in general it will not be brought to diagonal form:

$$\tilde{\boldsymbol{S}}_B = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}^T\boldsymbol{S}_B\boldsymbol{\Phi}\boldsymbol{\Lambda}^{-\frac{1}{2}}$$

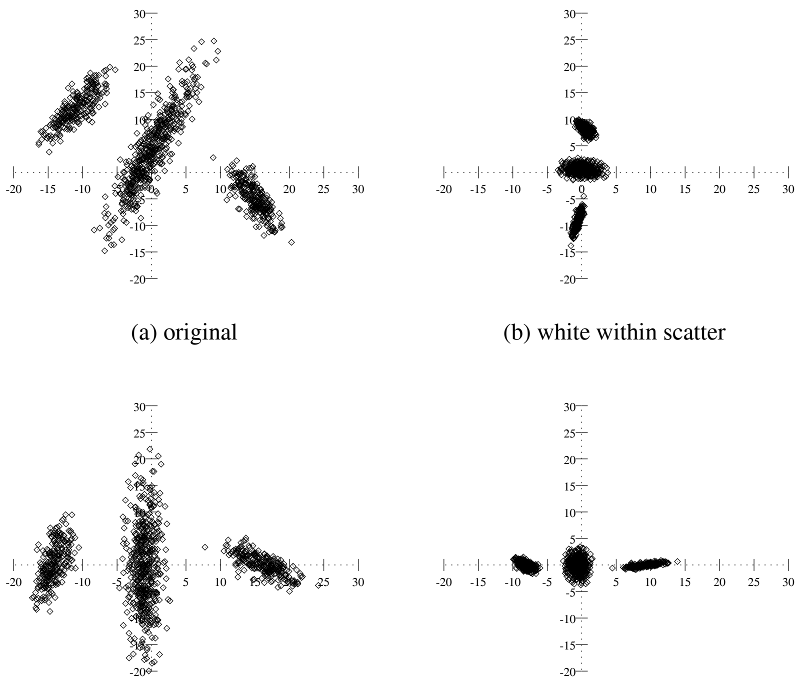
In the second step the fact is exploited, that the within class scatter, which has become an identity matrix, is now invariant to further orthonormal transforms. Therefore, one carries out a principal component analysis of the class centers and thus diagonalizes $\tilde{\boldsymbol{S}}_B$. For this one computes the eigenvector matrix $\boldsymbol{\Psi}$ of $\tilde{\boldsymbol{S}}_B$ and carries out a second transformation step:

$$\boldsymbol{z} = \boldsymbol{\Psi}^T\boldsymbol{y} = \boldsymbol{\Psi}^T\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}^T(\boldsymbol{x} - \bar{\boldsymbol{x}})$$

It can be shown, that the resulting total transformation matrix $\boldsymbol{\Psi}^T\boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{\Phi}^T$ except for the normalization of the eigenvectors is identical with the one that is obtained from the direct solution of the eigenvalue problem for $\boldsymbol{S}_W^{-1}\boldsymbol{S}_B$ (cf. e.g. [80, pp. 31–33], [45, p. 330]). This means, that both transformations map the feature space to the same new coordinate system with the difference, that in the two-stage method additionally a scaling by $\boldsymbol{\Lambda}^{-\frac{1}{2}}$ is performed.

The advantage of the second method — besides its greater clarity — lies in the fact, that it exhibits a better numerical stability. Neither a matrix inversion nor the calculation of eigenvalues for a not necessarily symmetric matrix need to be performed.

The effects of the LDA transformation on a sample distribution with three pattern classes is illustrated in figure 9.5. In the upper left the original distribution is shown as well as its LDA transform after the application of the one-step method. For comparison the figure shows in the upper right the intermediate result, which is generated by the whitening of the within class scatter matrix. The final result of the two-stage



(a) original (b) white within scatter

(c) direct LDA (d) two-stage LDA

Fig. 9.5. Example of linear discriminant analysis: (a) original data, (b) after whitening the within class scatter matrix, (c) result of the direct solution and (d) of the two-stage method.

LDA transformation, which is shown in the lower right, then results from this by the application of a PCA to the between class scatter matrix. Except for a scaling and a mirroring of the data it is identical with the result of the one-stage method.

A serious problem in the application of linear discriminant analysis, that should not be overlooked when deriving mathematical solutions, lies in the fact that for LDA a labeling of the sample set is mandatory. Therefore, the question arises, how the underlying pattern classes should be chosen in a suitable way. In fact this choice directly influences the between and within class scatter matrices and, in the end, the result of the whole transformation.

From a more detailed analysis of the significance of the between class scatter matrix at least a lower bound for the number of pattern classes can be derived. Because of its very definition (cf. equation (9.5)) $\boldsymbol{S}_B$ has at most rank $K - 1$, where K is the number of pattern classes used. As only the non-zero of the K class centers $\boldsymbol{x}_\kappa$ is taken into account in the construction of $\boldsymbol{S}_B$, at most $K - 1$ linearly independent vector components can exist. The between class scatter matrix, therefore, spans a $K-1$-dimensional sub-space and consequently only possesses $K-1$ positive eigenvalues. In the two-stage computation of the LDA a diagonalization of $\boldsymbol{S}_B$ forms the last step, which then can only map to a sub-space with a dimension m smaller than $K-1$.

LDA was used extensively for purposes of automatic speech recognition in the tradition of the Philips research systems (cf. [5, 88, 221] and [167]). There the underlying pattern classes are practically always defined on the basis of the model states of the context independent sub-word unit HMMs used. Therefore, the number of classes by far exceeds the feature dimension. The use of elementary segmental units as, e.g., speech sounds — i.e. considerably fewer pattern classes — is agreed to be considerably less powerful in comparison.

These results were also confirmed in extensive experiments of our own. They suggest the assumption, that what matters most when choosing the class definition is to ensure a robust estimation of the between class scatter matrix, as it cannot be seriously expected, that a linear transform is actually capable of perfectly separating a few hundred pattern classes. At the same time a more sophisticated class definition also assures, that the within class scatter more precisely approximates the local situations within individual regions of high density. In total it can be concluded, that the number of classes used for an LDA transformation should considerably exceed the dimension of the feature vectors.

9.2 Tying

The methods presented in the previous section implicitly decrease the number of model parameters by reducing the degrees of freedom present in the data — e.g. the dimension of the feature vectors themselves. In contrast, methods that try to group "similar" model parameters together work towards an explicit reduction of the number of parameters. As a consequence their estimation on the available sample data is improved and finally the robustness of the model as a whole is increased. These methods, which can be applied to many different parts of the modeling, usually are referred to as *tying*. Three principal methods can be distinguished in such a procedure for the merging of model parameters.

In *constructive* tying the merging of model parameters implicitly results from the construction of more complex models from building blocks, which are given by certain elementary models. All copies of those subunits reused then naturally reference the same set of parameters.

In contrast to this the following two approaches take an already existing model as a basis, the trainability of which is to be improved by parameter tying.

By *generalization* of modeling parts, which are realized in a very special way and for which, therefore, only few training samples are available, more general units can be derived. The parameters of these can be estimated on a broader basis of data and thus also more robustly. The special models are then no longer used themselves, but are replaced by suitable generalizations. The necessary generalization rules usually are set up by exploiting expert knowledge. The creation of the generalized models themselves can, in contrast, be performed automatically.

A similar result can also be achieved by *agglomerative* tying. On the basis of already existing model parameters similar parameters can be identified by a suitable distance measure. By applying an algorithm for cluster analysis to the parameter space groups of model parameters can be constructed, which serve a similar "purpose" within the overall model and for which a sufficient number of training samples is available. These methods have the big advantage of being able to compute a suitable form of tying for an arbitrary model automatically in a data-driven manner. However, it should not be forgotten that this method in fact proceeds in a quite paradoxical way: The parameter clusters, which are supposed to ensure robust trainability, are determined on the basis of initial parameters, which could just *not* be estimated reliably. In practice one nevertheless achieves good results with data-driven tying for a wide range of applications despite of this Munchhausen-like trick[7].

How these principles of tying are applied to existing or yet to be constructed models is of course left to the developer. Depending on the task at hand different procedures may make sense. Unfortunately, no strategy can be given, which would perform optimally in any case. Therefore, in the following we want to give an overview over the most well known methods as an "aid to decision-making". First we will present such methods that apply tying at the level of partial models. Afterwards, we will consider comparable methods that identify similar groups of model states or merge parameters within the mixture models used.

9.2.1 Model Subunits

The creation of a complex HMM is generally not achieved by a direct training of all parameters. Rather, one constructs larger models on the basis of smaller partial models with well defined descriptive potential. This procedure not only leads to a modularization of complex HMMs, but also allows the especially economic exploitation of the available example data for the parameter training.

This method was developed and brought to perfection in the field of automatic speech recognition. But also in recognition tasks structured similarly, as e.g. the recognition of handwriting, the use of so-called *sub-word units* can be considered state of the art (see section 8.2 page 129). In contrast, for the analysis of biological sequences hardly any measures are taken, in order to build the models created in a modular way on the symbolic level.

As described in section 8.2, HMMs for complex units, as e.g. whole words, spoken utterances, or sections of a text, are created starting from a limited inventory of baseline models. An implicit tying of model parameters results from the fact, that though new model states originate from every replication of an elementary model, these always share the same set of parameters as all other copies of the original model.

[7] In the fantastic stories about the supposed adventures of the German Baron Münchhausen (1720 – 1797) he claims to have pulled himself from a swamp by his own hair.

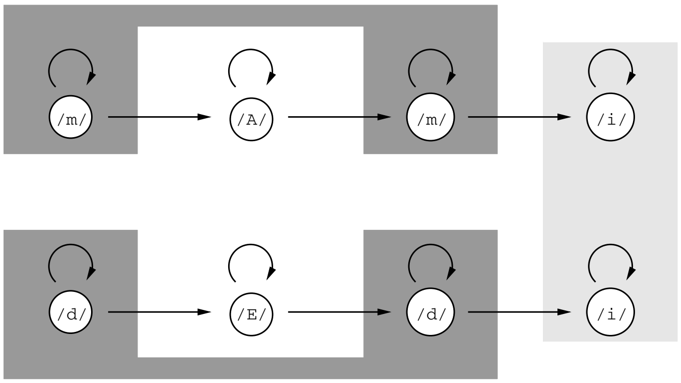


Fig. 9.6. Example of simple parameter tying on the level of elementary models, as it results from the model construction process.

Figure 9.6 shows a simple example for this situation from the domain of automatic speech recognition. For the description of a simple "baby's vocabulary" we want to create models for the words `mommy` and `daddy`. As in the phonetic transcriptions /`mAmi`/ and /`dEdi`/ two pairs of speech sounds appear at different positions, one obtains already a word-internal tying of the respective models (/`m`/ and /`d`/). Here it is assumed, of course, that for such a toy application speech sounds occurring in different contexts can be modeled identically. When concatenating both words to a primitive compound model it furthermore becomes clear, that the parameters of the partial model for the speech sound /`i`/ are identical in the overall model and, therefore, can also be tied.

In simple cases of model-driven tying thus parameters of partial models named identically are shared within a complex overall model. When constructing HMMs through composition of existing models this simply corresponds to a reuse of parameter sets. For the training phase this means, that all example sequences contained in the sample set, that correspond to an arbitrary copy of a model, are taken into account for the estimation of the shared set of parameters.

Model Generalization

Model-driven tying becomes more complex, if the similarity of partial models can not be derived directly from their names, but must be determined by inference processes. Such methods are mainly applied in conjunction with the use of context dependent sub-word units (see also section 8.2.2 page 130). Then the variety of possible models usually is so large, that for the majority of those a direct estimation of robust parameters can not be performed. Therefore, one tries to group similar models together, in order to ensure their trainability.

A rule-based method for the generalization of context dependent sub-word units, which are constructed according to the principle of triphones, was proposed in [208].

$$\begin{tikzcd}
 & & \texttt{t/r/I} \arrow[dl] \arrow[dr] & & \\
 & \text{consonant }\texttt{/r/I} \arrow[dl] \arrow[dr] & & \texttt{t/r/}\text{vowel} \arrow[dl] \arrow[dr] & \\
\texttt{\_ /r/I} \arrow[dr] & & \text{consonant }\texttt{/r/}\text{vowel} \arrow[dl] \arrow[dr] & & \texttt{/t/r/\_} \arrow[dl] \\
 & \texttt{\_ /r/}\text{vowel} \arrow[dr] & & \text{consonant }\texttt{/r/\_} \arrow[dl] & \\
 & & \texttt{/r/} & &
\end{tikzcd}$$

Fig. 9.7. Example for the generalization of context restrictions of triphones

For such models the generalization of the context restriction on a symbolic level can either be achieved by a shortening of the segmental extension of the context or by an explicit introduction of contextual super-classes.

Figure 9.7 shows, starting from the triphone model `t/r/I`, a possible simple generalization hierarchy. As every generalization of the context restriction applies either to the left or to the right context, for every triphone model always two possible generalizations exist. The generalization of an individual context is performed in two steps in the example. First coarse phonetic classes as vowels, nasals, plosives and the like are used and finally, the context restriction is omitted totally. At the end of the generalization process thus the respective context independent phone model is reached.

If the context restriction applies to more than only one neighboring unit as, e.g., in so-called polyphones, the generalization of the respective models is performed by simple shortening of the context's extension. At the end of the generalization hierarchy again the context independent variant of the modeled unit is reached — for polyphones the respective monophone.

In the resulting hierarchy of generalized models now those need to be selected, which are sufficiently special and at the same time can be trained robustly. These are then used for constructing more complex HMMs instead of the specialized original context dependent units. A good indication of the trainability of HMMs is given by the frequency of occurrence of the respective units in the available sample data. In the generalization hierarchy one then chooses the most special models that occur more frequently than a certain threshold in the training material. In practice good results are achieved with minimal frequencies in the range of 50 to a few hundred occurrences.

Model Clustering

A different view on the generalization relation between HMMs for more or less specialized sub-word units results when methods are used on the basis of *decision trees*, which are also known as *classification and regression trees* [26]. In contrast to the definition of generalization relations between context dependent sub-word units criteria are formulated that combine similar units into groups or clusters (cf. e.g. [131]). Therefore, these techniques constitute methods for cluster analysis, which, however, work on the symbolic representation of the models and within which similarity relations are defined by rules.

We want to explain the principal approach briefly using an example for the grouping of similar triphone models. Here, every node in the decision tree corresponds to a possible group of triphones with identical central phone. A certain model is assigned to such a model group by a sequence of binary decisions starting from the root node of the tree. In every tree node, therefore, a criterion set up by experts is evaluated, which tests for a certain property of the model context[8]. Depending on whether or not the property is true for the triphone considered either the left or the right successor of the current tree node is reached. The individual leaf nodes of the decision tree correspond to one HMM each, which is used for the actual modeling of all triphones that are assigned to this node by the decision rules.

By means of this technique a grouping of context dependent sub-word units is possible, which can be parametrized in a very detailed way. The main disadvantage of the method is, however, that the necessary decision rules need to be set up by experts. Furthermore, the respective rule sets tend to become quite large in practice and are given only exemplarily or as quite limited abridgments in the literature (cf. [131]).

The disadvantage, that for the grouping of models first expert knowledge needs to be formalized suitably, is tried to be avoided by agglomerative techniques that derive similarities between different partial models in a purely data-driven manner. Still a suitable distance measure between the parameters of the HMMs considered needs to be pre-specified for these methods. Then groups of models, which have similar parameters and can thus be represented by a shared HMM, can be computed automatically by applying a method for cluster analysis. As one of the first researchers Kai-Fu Lee used such a purely data-driven method for determining suitable generalized triphones [136, pp. 103–106]. As the model structure of individual HMMs may in general vary and as in the distance calculation at least multiple states of a model have to be taken into account, automatic techniques for tying on the level of partial models did not prevail. However, the application of equivalent methods on the level of model states can be considered state of the art today. Therefore, we want to present the respective techniques within the framework of the automatic generation of state clusters in the following section.

[8] In the literature it is said quite often that nodes “ask questions”, which is, however, a rather inappropriate anthropomorphic description of this automatic procedure.

9.2.2 State Tying

The tying of individual HMM states represents a generalization as opposed to the merging of similar model subunits. There parameter groups may be chosen with substantially increased granularity. At the same time the application of automatic procedures becomes easier, as the structure of the units processed is considerably simpler than the one of partial models.

Simple Duration Modeling

On the state level generally no constructive tying is applied, as in contrast to model subunits no symbolic meaning within more complex models is associated with individual states. Only for the modification of the duration modeling of HMMs rarely constructive state-based tying is applied. An individual model state describes data segments, the duration of which satisfies a geometric distribution. The maximal probability is taken on for a segment length of a single time step independently from the actual values of the transition probabilities. For longer segment durations the occupancy probability decays exponentially (cf. [95, p. 218]). By linear concatenation of multiple identical copies of a respective state this duration behavior can be modified such, that it satisfies a binomial distribution. In linear models thus a minimal segment duration is specified, which corresponds to the number of replicated states. By means of the transition probabilities the state group can be parametrized such, that the maximal probability of occupancy is achieved for an arbitrary longer segment duration.

It can be assumed, that a binomially distributed segment duration is better suited for the modeling than an exponentially decaying distribution. However, such rather moderate changes of the duration properties (cf. also section 5.8.2) hardly have a decisive influence on the modeling quality in practice, as it is always dominated by the substantially more important part of the output distributions.

State Clustering

In contrast, the rule-based or data-driven generation of state clusters has established itself as state of the art. Such techniques are applied in many current systems in order to achieve in a flexible and conceptual simple way an optimal exploitation of the training data and simultaneously a high modeling quality.

Especially for the field of automatic speech recognition a number of rule based methods were proposed. There linguistic-phonetic knowledge about similarities between the states within context dependent phone models is represented in the form of decision trees (cf. [26], see also page 156). These can then be used to generate a generalization hierarchy on possible groups of model states. Such state groups describe phonetic events, which appear in many different contexts as part of a speech sound. Parameter sets are then trained for those clusters, for which sufficiently many samples are available and which allow a modeling as specialized as possible (cf. e.g. [78, 124, 173], [97, pp. 432–436]). The main disadvantage of those methods lies in

```
Given the parameters of an HMM, for which the number of states shall be optimized, a distance measure $d(C_i, C_j)$ for model states or state clusters, respectively, and a suitable termination criterion.
 1. **Initialization**
    Create an individual state cluster for all model states $i$
    $C_i = \{i\} \quad \forall i, 0 \leq i \leq N$
    and combine these to form the initial set of clusters $\mathcal{C}$:
    $\mathcal{C} = \bigcup_{i=1}^{N} \{C_i\}$
 2. **Selection**
    Choose from the current set $\mathcal{C}$ of state clusters that pair $C_j, C_k$ that minimizes the distance measure:
    $(C_j, C_k) = \underset{C_p, C_q \in \mathcal{C}}{\operatorname{argmin}} d(C_p, C_q)$
 3. **Reorganization**
    Construct a new state cluster $C$ by merging $C_j$ and $C_k$:
    $C \leftarrow C_j \cup C_k$
    Remove the original clusters $C_j$ and $C_k$ from $\mathcal{C}$ and insert the newly created cluster $C$ instead:
    $\mathcal{C} \leftarrow \mathcal{C} \setminus \{C_j, C_k\} \cup \{C\}$
 4. **Termination**
    **if** the termination criterion is not yet satisfied for the current set $\mathcal{C}$ of state clusters
              continue with step 2
    **otherwise** Stop!
```

Fig. 9.8. Algorithm for the automatic computation of a set of state clusters for a given HMM.

the fact, that it is necessary to acquire extremely specialized expert knowledge and to represent it suitably for the application within an automatic clustering method.

This problem is avoided, if the state clusters are determined in a purely data driven manner. Therefore first a similarity or distance measure between states needs to be defined, which can be computed only on the basis of the state parameters or the samples associated with the state. Subsequently state groups, for which parameters can be trained robustly, can be identified by a principally arbitrary method for vector quantization or cluster analysis.

Figure 9.8 shows a simple greedy algorithm for the automatic generation of state clusters in HMMs of arbitrary structure. It proceeds similarly to the methods described in [136, p. 104] or [255]. The algorithm assumes that for the parameter sets of individual states or state groups C_j and C_k a suitable distance measure $d(C_j, C_k)$ was specified. Furthermore, a criterion must be given, which defines when the clustering algorithm is to be terminated, as otherwise all states will be merged into a trivial cluster.

In the initialization step first per given model state a simple state cluster is generated, which contains that state only. Taken together these form the initial set of clusters. In every optimization phase now that pair of clusters C_j and C_k is selected that has minimal distance with respect to the chosen distance measure. Those two clusters are subsequently merged. If the termination criterion is not yet satisfied for the current set of clusters, the selection of the nearest neighboring state clusters will be repeated. Otherwise the procedure is finished.

A simple and robust possibility for specifying a termination criterion can — similarly to the tying of partial models — be defined on the basis of the frequency of the clusters generated in the training data. Then it merely needs to be checked, whether all state clusters generated so far cover a sufficient number of feature vectors. If this condition is met for every cluster, one assumes that the modified model can be trained robustly. The minimal number of training samples per state cluster is, however, application dependent and in general needs to be determined empirically. Similarly to the tying of partial models minimal frequencies from 50 up to a few hundred training samples achieve good results in practice, which could be confirmed in numerous experiments of our own.

The most important model parameters of an HMM state consists in the respective output probability density function. Transition probabilities are, in contrast, of inferior importance and are not taken into account when determining state clusters. However, it is quite problematic to define a distance measure for general output distributions described by mixture density models. Therefore, in practice usually suitable simplifications are made.

The model parameters used to determine the state clusters do not need to correspond to those, which are finally used for the modified model. This can be chosen arbitrarily after the optimization of the model structure. Therefore, a simpler modeling can be used as the basis of the cluster analysis applied to the state space. In [255] output distributions with only a single normal density are used to determine the state clusters. As a distance measure between states or state clusters, respectively, then the divergence can be used, which is defined for two multi-variate normal densities as follows (cf. e.g. [80, pp. 458–459]):

$$\begin{aligned} d_{\text{divergence}}(\mathcal{N}_j, \mathcal{N}_k) = & \frac{1}{2}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_k)^T(\boldsymbol{K}_j^{-1} + \boldsymbol{K}_k^{-1})(\boldsymbol{\mu}_j - \boldsymbol{\mu}_k) + \\ & + \frac{1}{2}\,\mathrm{tr}\{\boldsymbol{K}_j^{-1}\boldsymbol{K}_k + \boldsymbol{K}_k^{-1}\boldsymbol{K}_j - 2\boldsymbol{I}\} \end{aligned} \tag{9.7}$$

A considerable simplification of this computation rule results if only diagonal covariance matrices are taken into account, as they are also used in [255]. There the clustering is furthermore limited to model states, which occur in context dependent models having the same base phone. As termination criterion a minimal number of 100 training samples per state group created is required. The HMM structure optimized such then serves as a basis for the creation of a more complex model where the output distributions are described by mixture density models. A comparable approach using a single normal density per state while clustering is described in [167].

A distance measure for mixture models, which can be evaluated easily, was proposed in [51]. As the underlying modeling merely uses one global covariance, the distance of two state clusters can be defined on the basis of the mean vectors of the respective component densities only. The maximal distance between all possible pairs of mean vectors $\boldsymbol{\mu}_l$ and $\boldsymbol{\nu}_m$ is taken to be the cluster distance and can be computed as follows:

$$d(C_j, C_k) = \max_{\boldsymbol{\mu}_l \in C_j, \boldsymbol{\nu}_m \in C_k} ||\boldsymbol{\mu}_l - \boldsymbol{\nu}_m||$$

This distance measure is referred to as furthest neighbor criterion, as only the base densities lying the furthest apart of each other are decisive of the clustering result.

For discrete HMMs and also for mixture models, which are based on shared codebooks as, e.g., semi-continuous HMMs, state specific output distributions can be compared in an especially simple way. As the underlying baseline densities are not modified by the state clustering or are not even existing in the discrete case, the distributions are completely defined by the mixture weights c_{jk} or the discrete output probabilities for individual symbols, respectively.

For such discrete distributions the entropy $H(C_j)$ can be computed, which represents a measure for how special or how general the generation of outputs is performed in the model state or state cluster C_j considered:

$$H(C_j) = -\sum_{k=1}^{K} c_{jk} \ln c_{jk}$$

The entropy of a discrete distribution is non-negative and takes on its maximal value $\ln K$ for a uniform distribution. It becomes zero, if the distribution of model outputs is uniquely defined, i.e. exactly one discrete output probability or mixture weight takes on the value one and all others vanish (cf. e.g. [50, pp. 630–631], [97, pp. 120–122]).

The merging of states and the modeling of the respective outputs by a *single* shared distribution causes an increase of the entropy. This can be used as a distance measure for the generation of state clusters. In order to avoid, that large clusters, which cover a huge number of feature vectors, dominate the grouping process, the increase in entropy is weighted by the number of underlying training samples. For a pair of clusters C_i and C_j, for which the output distributions are given by mixture weights c_{ik} and c_{jk}, the weighted increase in entropy is then defined as follows (cf. [136, p. 105]):

$$d(C_i, C_j) = (N_i + N_j) H(C_i \cup C_j) \; - \; N_i H(C_i) \; - \; N_j H(C_j)$$

Here N_i and N_j denote the number of training samples, which are assigned to the respective cluster. The new mixture weights of a state cluster C_m, which was formed from C_i and C_j, result as the weighted averages of the individual mixture weights according to:

$$c_{mk} = \frac{N_i}{N_i + N_j} c_{ik} + \frac{N_j}{N_i + N_j} c_{jk}$$

They can, therefore, be computed very easily during the clustering process.

9.2.3 Tying in Mixture Models

An even more fine-grained form of tying as compared to state clusters can be achieved by merging similar parameters within the mixture models used to describe the output distributions.

Mixture Tying

The most well known variant of tying on the level of mixture densities is represented by the so-called *semi-continuous* or *tied-mixture HMMs* (see also section 5.2 page 63). There for all mixture densities within an HMM a shared set of baseline densities is used, i.e. the individual components of the mixtures are tied across all distributions. One then obtains the following definition for the output probability densities (cf. equation (5.2)):

$$b_j(\boldsymbol{x}) \quad = \quad \sum_{k=1}^{M} c_{jk}\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{K}_k) \quad = \quad \sum_{k=1}^{M} c_{jk}\, g_k(\boldsymbol{x})$$

Semi-continuous HMMs offer a number of practical advantages as opposed to "fully continuous" models, where the state specific mixtures are described completely independently of each other.

By using a single shared codebook it can be assured in a simple way, that all baseline densities can be reliably estimated on the sample data available. As vector quantization procedures often tend to produce class regions of similar size, when computing an initial codebook for a semi-continuous HMM it only has to be assured, that on average sufficiently many feature vectors per parameter of the baseline distributions are available in the sample set. By a subsequent training of the model the individual densities are modified, though. But only in exceptional cases this causes the parameter estimation to fail due to sparseness of example data. Additionally, in semi-continuous models the dynamic range of the scores can effectively be limited during the computation process by transforming the density values to posterior probabilities (see section 7.3).

The principal idea of semi-continuous HMMs, namely to use shared baseline densities for the modeling of mixture densities, can be generalized to the use of multiple codebooks. Which model states share a common codebook, however, needs to be specified by the developer. It makes sense to merge those states in this respect, from which it can be expected that their output distributions describe similar data.

The most well known example of such a modeling are the so-called *phonetically-tied mixture HMMs*. In this modeling developed for purposes of automatic speech recognition all states of models share common codebooks, which from a phonetic view belong to similar phones or phone classes. When using triphones as sub-word units then, e.g., all models with the same central phone share an inventory of baseline densities.

Clustering of Densities

Besides constructive techniques also automatic methods exist for the tying of baseline densities of mixture models. As, however, for individual component densities in general no symbolic description of the context of occurrence can be specified, no methods on the basis of decision trees exist. In contrast, agglomerative techniques can be applied in complete analogy to the automatic determination of state clusters, as presented in the previous section. One thus obtains an even finer granularity in the merging of similar model parameters. As distance measures between individual densities, e.g., the divergence can be used similarly to the generation of state groups (see equation (9.7)).

In [51] a measure is applied, which can be computed extremely easily. It only evaluates the weighted Euclidean distance of the respective mean vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ of the density groups C_i and C_j:

$$d(C_i, C_j) = \frac{N_i\, N_j}{N_i + N_j}\; ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||$$

Here N_i and N_j denote the number of training samples, which are associated with the respective density. A more expensive method, which also takes into account the covariance matrices of the densities involved, was proposed in [114].

In contrast to the generation of state clusters the merging of parameters on the level of mixture densities in general does not aim at defining more general modeling units, which can be estimated robustly on a given sample set. Rather, all methods serve the primary purpose of speeding up the evaluation of the model by reducing its complexity.

Tying of Covariance Matrices

In mixture density models the mean vector of a baseline density describes the center of a region of high density in the data distribution considered. With respect to the covariance matrices, which define the local distortion of the feature space, the mean vectors, therefore, are of greater importance. Especially when using full covariance matrices, however, these require considerably more training samples for a robust estimation. Consequently, in larger mixture models the covariance matrices are tied frequently, i.e. shared across multiple or all baseline densities.

The most radical approach of this type is applied in automatic speech recognition systems, which originated from the tradition of Philips research (cf. e.g. [167, 221]). In contrast to the usual definition of continuous HMMs there output probability densities are defined on the basis of Laplacian densities with a position vector $\boldsymbol{r}_{jk}$ and a scaling factor $v(l)$ per dimension of the feature space:

$$g_{jk}(\boldsymbol{x}|\boldsymbol{r}_{jk}, \boldsymbol{v}) = \frac{1}{\prod_{l=1}^{n} 2v(l)}\; e^{-\sum\limits_{l=1}^{n} \frac{|x(l) - r_{jk}(l)|}{v(l)}}$$

In essence this corresponds to the use of Gaussian densities with mean vectors $\boldsymbol{r}_{jk}$ and a commonly shared diagonal covariance matrix, the elements of which are given by the scaling factors $v(l)$. The success of such a drastic simplification of the covariance modeling is most likely due to the fact, that in the respective recognition systems always also a linear discriminant analysis is applied. Its first computation step consists in bringing the covariance matrices of the individual pattern classes to the form of identity matrices approximately (see section 9.1.2 page 148). Thus the modeling of the local distortion of the feature space considerably looses importance and can also for mixture models with some 10 000 or several 100 000 densities be successfully described by a single global diagonal covariance matrix.

9.3 Initialization of Parameters

The training methods for HMMs presented in section 5.7 always require the specification of a suitable initial model, the parameters of which are then iteratively improved. How such a starting point of the optimization procedures should be chosen is, however, hardly dealt with in the literature, though this decision fundamentally influences the whole subsequent parameter training. For models with some practical relevance it is by no means sufficient, to choose initial parameters randomly or according to a uniform distribution.

As the only formally defined method the *segmental k-means* algorithm offers the possibility to define initial parameters for an HMM training [135]. This method for parameter estimation alternatingly carries out a segmentation of the training data and a computation of new model parameters on the basis of segmentation *alone* without changing the model structure (see page 88). For the initialization of the method it is, therefore, sufficient to specify a segmentation of the training set and the desired model structure. The latter in general comprises a modularization into model subunits (see section 8.2), the choice of a suitable topology for the baseline models (see section 8.1), and the specification of the number of baseline densities, which shall be used for describing the output distributions.

In the segmental k-means algorithm the initial segmentation of the sample set is derived from a reference annotation of the data. The manual mapping of feature vectors to model states is, however, *de facto* not possible in practice and even a manual segmentation of data into elementary description units as, e.g., phones or characters requires an extraordinary effort, which can only be afforded in the rarest of cases.

Therefore, a radically simplified approach for the initialization of linearly structured models was proposed in [135]. From the reference annotation first by concatenation of the respective HMMs a modeling framework for every section of the training data can be derived. In the field of automatic speech recognition one, e.g., creates an overall HMM for the respective utterance by concatenation of word models according to the orthographic transcription. As one assumes a principally linear passing through the respective model states, these can be assigned linearly to the available feature vectors in a very simple way. Though the initial segmentation created thus will in general be quite inadequate, it is usually sufficient for the definition of a starting point for the subsequent optimization of the model by the training process.

Considerably better results can be achieved, if more accurate segmentation information is taken into account for the initialization of the model parameters. This is usually derived from an already existing recognition system, which is used for segmenting the training data into a sequence of words, phones, or characters[9]. The linear mapping between feature vectors and model states is then merely performed within those rather small segments, so that an acceptable accuracy is achieved.

As the choice of an initial model on the basis of an optimized segmentation in general has a positive effect on the parameter training, even without changes in the model structure, after such a second initialization phase an improvement in the modeling quality will be achieved. However, with a sufficiently precise segmentation it is in particular possible, to initialize the training of a complex model considerably better than on the basis of an uninformed linear mapping. Therefore, large HMM systems are frequently built in a sequence of design phases. One starts with a rather simple model and refines its structure in a number of subsequent optimization steps. The parameters of the next more complex model are then newly initialized with the improved segmentation information derived from the previous model.

[9] In the rare case that for a given sample set a manual segmentation on the level of elementary units is available, this can of course be used alternatively in this phase of the method.

10

Efficient Model Evaluation

With the Viterbi algorithm in section 5.6 a method for decoding HMMs was presented, and also for the evaluation of n-gram models algorithms exist as, e.g., the so-called backing-off described on page 106. However, these methods only represent the basic foundations on which algorithms for the efficient integrated evaluation and decoding of Markov models are developed in practice.

All these methods are based on the idea to save "unnecessary" computations as much as possible. In some cases this can be achieved by a suitable reorganization of the representation of the search space or of the model itself. In the majority of the methods, however, "less promising" solutions are explicitly discarded early from the further search process. Thus the search space is reduced to a tightly focused "promising" part. This procedure is frequently referred to as search space *pruning*. The problem with all those techniques is to find a method for identifying "less promising" solutions. With absolute certainty this is only possible *after* the actual optimal solution is known. Every premature decision holds the risk of possibly deleting the hypothesis actually sought for from the search space. In many cases, however, its exact calculation is out of question anyway in practice because of the tremendous computational costs. Therefore, one usually accepts that manageable efficient methods in general only produce approximate solutions. Such methods are then also called *suboptimal*.

The following sections give an overview over the most important methods for the efficient evaluation of Markov models. At the beginning methods for speeding up the computation of output probability densities on the basis of mixture models are presented. Then the standard method for the efficient application of Viterbi decoding to larger HMMs is described. In section 10.3 methods are explained that apply techniques of search space pruning for the acceleration of the parameter training of HMMs. The chapter concludes with a section on tree-like model structures, which can be used both in HMMs and in n-gram models in order to increase the efficiency when processing these models.

10.1 Efficient Evaluation of Mixture Densities

In typical applications of continuous HMMs, where output distributions are approximated by mixture densities, the evaluation of these models may dominate the total resulting search effort. Especially if a large number of baseline densities is used for representing the emissions as precisely as possible, perhaps more than half of the decoding effort results from the evaluation of probability density functions (cf. [182]). Therefore, in most larger HMM-based systems methods for limiting the effort in the mixture evaluation are applied. Unfortunately however, the actual procedures are not always well documented in the literature.

All these methods are based on the principle to use some method, which can be evaluated extremely fast, in order to identify a subset of baseline densities, which make relevant contributions to the computation of the required output probability densities. For all mixtures in this subset then the density is computed exactly. For all remaining distributions, however, some constant value is used[1] as an approximation of the actual density value, which is expected to be quite small. The overall effort in decoding the mixture density is reduced, as these nearly vanishing densities are "inferred" rather than computed explicitly.

The most well known method for predicting the actual baseline densities to be evaluated is the so-called *Gaussian selection*, which was first proposed in [19] (cf. e.g. [41, 123, 182]). By means of a method for vector quantization first the available Gaussian densities are put together into groups. In order to improve the subsequently performed accelerated density evaluation overlaps between the density groups are explicitly allowed. For every group of baseline densities furthermore the centroid is calculated. During the decoding of the HMM then only those Gaussian densities are computed exactly that lie in the density group, the centroid of which has minimal distance to the current feature vector. As a group of densities in general defines only a short list of those distributions, which seem promising for being evaluated exactly, the technique is also referred to as the method of *Gaussian short-lists* in the literature. Depending on the system configuration an acceleration of the density evaluation by a factor of 3 to 10 can be achieved with this method.

A similar principle is the basis of the method proposed in [207], which uses a sequence of vector quantization processes that are carried out more and more exactly. First a coarse model of the available densities is evaluated, which results from not using all available dimensions of the feature vectors in the representation. For promising density candidates then in one or more subsequent processing steps more exact and thus more costly computations are carried out.

In [77] a method was proposed, which is especially suited for baseline densities with diagonal covariance matrices, as they are frequently used in larger HMM-based systems. The regions, in which one of these densities contributes a relevant amount to a mixture density, are approximated by hyper-cuboids. By representing the feature

[1] The actual choice of this lower bound for vanishing density values is actually a quite critical parameter for the overall performance of the system, which is, e.g., empirically investigated in [123].

space as an n-dimensional tree, then very efficiently those hyper-cuboids can be determined, in which a certain feature vector lies. For the associated densities then the actual values are computed exactly. The resulting error of the approximation defined such can be controlled by a suitable choice of the extension of the hyper-cuboids.

An extensive empirical investigation of different methods for accelerating the evaluation of mixture densities is presented in [180]. As the best performing single technique the method of Gaussian selection described above is identified, i.e. the preselection of certain Gaussian densities by a vector quantization process. As a modification with respect to the original method [19], however, the densities in the three to five density groups lying closest to the current feature vector are evaluated exactly. The authors report a reduction in the effort for evaluating mixture densities by a factor of more than five and a overall reduction in the computational effort to approximately 30 %.

10.2 Beam Search

The Viterbi algorithm for decoding HMMs represents with its linear time complexity already a considerable improvement in efficiency as opposed to the naive computation scheme with exponential costs. However, the method still has quadratic complexity in the number of model states. By choosing a suitable restricted model topology (cf. section 8.1) it can be assured in fact, that per computation of a partial path probability only a few possible predecessor states have to be considered in the normal case. Nevertheless, the Viterbi matrix of the $\delta_t(i)$ grows linearly with the number of model states, such that a complete evaluation in practice is possible for quite limited problems only.

In a speech recognition system with large vocabulary the acoustic-phonetic model usually consists of several 10 000 states, for which at every time partial path scores would need to be computed when applying the Viterbi algorithm directly. All these states, however, encode quite different acoustic events, wherefore one can assume, that the majority of possible state sequences represents spoken utterances, which hardly exhibit any similarities with the signal analyzed and, therefore, actually would not need to be considered at all.

During the decoding process these “less promising” solutions need to be identified as early as possible but also sufficiently reliably. For this an absolute threshold of the path score is certainly out of question, as the values of $\delta_t(i)$ in total vary largely in dependence on the data considered and especially with the length of the observation sequence. A real vanishing of the partial path probabilities is furthermore usually avoided by the methods for flooring — i.e. the limitation of individual contributions to the path score to certain minimal values — presented in section 7.2. Therefore, only the *differences* in the scores can give indications for promising partial results or for such paths, which could be eliminated.

```
1. Initialization
   initialize the set of active states with the non-emitting state 0
   $\mathcal{A}_0 \leftarrow \{0\}$
2. Propagation
   for all times $t$, $t = 1 \ldots T$:
   • initialize the locally optimal path score $\tilde{\delta}_t^* \leftarrow \infty$
   • for all $i \in \mathcal{A}_{t-1}$ and all $j \in \{j | j = \text{succ}(i)\}$
     – compute the local path score from state $i$ to its successor $j$
       $\tilde{\delta}_t(i,j) = \tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{b}_j(O_t)$
     – update the partial path score for state $j$, if necessary
       if $\tilde{\delta}_t(j)$ has not yet been computed or $\tilde{\delta}_t(i,j) < \tilde{\delta}_t(j)$
               $\tilde{\delta}_t(j) \leftarrow \tilde{\delta}_t(i,j)$
               $\psi_t(j) \leftarrow i$
     – update the locally optimal path score
       if $\tilde{\delta}_t(i,j) < \tilde{\delta}_t^*$
               $\tilde{\delta}_t^* \leftarrow \tilde{\delta}_t(i,j)$
   • determine the set $\mathcal{A}_t$ of active states
     – initialize the new set of active states $\mathcal{A}_t \leftarrow \emptyset$
     – add all successors $j$ of active states $i$ which lie within the beam
       for all $i \in \mathcal{A}_{t-1}$ and all $j \in \{j | j = \text{succ}(i)\}$
               if $\tilde{\delta}_t(j) <= \tilde{\delta}_t^* + \tilde{B}$
                       $\mathcal{A}_t \leftarrow \mathcal{A}_t \cup \{j\}$
3. Termination
   determine the optimal end state
   $\hat{s}_T^* := \operatorname{argmin}_{j \in \mathcal{A}_T} \tilde{\delta}_T(j)$
4. Backtracking of the approximately optimal path
   for all times $t$, $t = T - 1 \ldots 1$:
   $\hat{s}_t^* = \psi_{t+1}(\hat{s}_{t+1}^*)$
```

Fig. 10.1. Beam search algorithm for the efficient computation of the approximately optimal state sequence $\hat{s}^*$ in HMMs with large numbers of states. Probability quantities are represented in the negative-logarithmic domain (cf. section 7.1).

In the so-called *beam search* algorithm developed by Lowerre ([148, pp. 25–31], cf. also [147], [97, pp. 606–608])[2] on the basis of the relative differences in the partial path scores an extremely robust dynamic criterion for pruning the search space is used. The search is thus virtually focused within a tight "beam" around the currently best partial solution.

The basic idea of the method is to restrict the evaluation of the partial path scores in the Viterbi matrix to so-called active states, that is to a rather limited set. As active those states are defined, for which the scores $\delta_t(i)$ do not deviate too much from the locally optimal solution $\delta_t^* = \max_j \delta_t(j)$. The maximally admissible score

[2] Despite the enormous practical relevance of the method hardly any descriptions of it can be found in the relevant monographs and the interested reader is referred to the difficult to access original work of Lowerre [148].

difference is specified by a factor B proportional to the currently optimal score δ_t^*. The set of active states $\mathcal{A}_t$ at time t is, therefore, defined as follows:

$$\mathcal{A}_t = \{i|\delta_t(i) \geq B\,\delta_t^*\} \quad \text{with} \quad \delta_t^* = \max_j \delta_t(j) \text{ and } 0 < B \ll 1$$

In the further evaluation of the Viterbi matrix for the respective next time step $t+1$ then only these active states are considered as potential predecessors when computing the local path scores (cf. equation (5.9)). One thus obtains the following modified calculation rule:

$$\delta_{t+1}(j) = \max_{i \in \mathcal{A}_t} \{\delta_t(i) a_{ij}\}\, b_j(O_{t+1}) \tag{10.1}$$

The proportionality factor B is chosen as a very small positive constant and typically lies in the range of 10^{-10} to 10^{-20}. The smaller B is the more states lie with their scores in the interval $[B\,\delta_t^* \ldots \delta_t^*]$ and thus are active, i.e. are considered in the search process.

This concept can intuitively be understood better when using the negative-logarithmic representation of probability quantities (cf. section 7.1). The proportionality factor B then becomes an additive offset $\tilde{B}$, and one obtains the following rule for determining the set of active states:

$$\mathcal{A}_t = \{i|\tilde{\delta}_t(i) \leq \tilde{\delta}_t^* + \tilde{B}\} \quad \text{with} \quad \tilde{\delta}_t^* = \min_j \tilde{\delta}_t(j) \text{ and } \tilde{B} > 0 \tag{10.2}$$

Thus around the respective locally optimal path score $\tilde{\delta}_t^*$ a “corridor” of constant width is created, within which the optimal solution is sought for. The only free parameter $\tilde{B}$ of this search method, therefore, is usually referred to as the *beam width*.

For an implementation of the method it is, however, inconvenient to select from the usually quite large total number of states a small part of active states as potential *predecessors*, as given in equation (10.1). At time $t+1$ nevertheless all possible states j would need to be considered in order to check, whether their predecessors lie in the set of active states $\mathcal{A}_t$. Such a verification, however, is with respect to the effort necessary almost comparable to an actual computation of the respective score.

Therefore, in the beam search one changes the “viewing direction” during the recursive calculation of the partial path probabilities. Instead of considering all potential predecessors the calculations are directly propagated from the respective active states to the potential *successors*. In order to be able to formulate this procedure exactly, we define a successor relation between states of an HMM as follows:

$$j = \text{succ}(i) \quad \Leftrightarrow \quad a_{ij} > 0$$

For “entering” into the model we, furthermore, introduce a non-emitting state 0, which has all possible start states as successors:

$$j = \text{succ}(0) \quad \Leftrightarrow \quad \pi_j > 0$$

The resulting algorithm, which is summarized in figure 10.1, now proceeds as follows: First the set of active states is initialized by the single non-emitting state 0.

Afterwards follows the propagation of the path probabilities, which is carried out for all times $t = 1 \ldots T$.

At the beginning of every propagation step the locally optimal score $\tilde{\delta}_t^*$ is not yet known and is set to a suitable large value. Then for all active states i the respective possible successors j are considered. Given a certain fixed predecessor state i one obtains the score $\tilde{\delta}_t(i, j)$ for a path to state j. The optimal partial path score of state j is now computed anew, if no solution was known so far, or updated, if an improved score results from successor i. Simultaneously the pointers $\psi_t(j)$ necessary for tracking back the optimal path are updated. For the completion of a propagation step then the set $\mathcal{A}_t$ of states is determined that will be further active.

As soon as this computation scheme reaches the end of the observation sequence, i.e. time T, the approximately optimal path $\hat{\boldsymbol{s}}^*$ can be determined starting from its last sequence element $\hat{s}_T^*$ in principally the same way as in the Viterbi algorithm (cf. also figure 5.5). The sub-optimal solution found by beam search will in general be different from the theoretically optimal state sequence $\boldsymbol{s}^*$. However, in practice the exhaustive search of the whole Viterbi matrix necessary for determining $\boldsymbol{s}^*$ exactly is possible in the rarest of cases only due the computational effort required.

The beam search algorithm constitutes the core of every method for decoding HMMs and can in slightly modified form also be applied to the search with n-gram models (cf. chapter 12). In the realization of the algorithm presented here and shown in figure 10.1 of course further improvements in efficiency are possible, that may result from the chosen internal representation of the HMMs. Thus for example the explicit computation of $\mathcal{A}_t$ can be avoided, if the decision about active states is made only at the beginning of the respective next propagation step. Also the evaluation of the output probabilities $b_j(\boldsymbol{O}_t)$ is in fact not required until after the recombination of the potential paths.

10.3 Efficient Parameter Estimation

The computational effort necessary for parameter estimation usually is not of primary concern in efficiency considerations, as the design and parametrization process of HMMs can be performed in advance. However, in all methods applied for this purpose an iterative optimization is required, which causes the running time of an individual optimization step to contribute multiple times to the total effort. Furthermore, frequently different parametrizations or configurations need to be created in order to select the best performing version after completed parameter training. It is, therefore, quite obvious to also try to avoid "unnecessary" computations in training, unless this measures have some negative effects on the quality of the parameter estimation.

10.3.1 Forward-Backward Pruning

In the Baum-Welch algorithm for every segment of the training data the complete matrix of forward and backward probabilities needs to be evaluated. In fact the idea of the beam search can also be applied to this method in order to only carry out the calculation for relevant parts of this search space (cf. [95, p. 244]).

One defines during the computation of the forward and backward variables for every time t active states depending on the optimal value $\alpha_t^* = \max_j \alpha_t(j)$ or $\beta_t^* = \max_j \beta_t(j)$ of the respective probability quantity.

The set of active states, which are considered during the computation process, is then defined as

$$\mathcal{A}_t = \{i | \alpha_t(i) \geq B\,\alpha_t^*\} \quad \text{with} \quad \alpha_t^* = \max_j \alpha_t(j) \text{ and } 0 < B \ll 1$$

for the forward variable and as

$$\mathcal{B}_t = \{i | \beta_t(i) \geq B\,\beta_t^*\} \quad \text{with} \quad \beta_t^* = \max_j \beta_t(j) \text{ and } 0 < B \ll 1$$

for its counterpart in the backward algorithm (cf. also equation (10.2)). As in the *beam search* algorithm the constant B specifies the beam width and thus implicitly the size of the part of the total search space, which is searched for solutions.

Thus one obtains modified recursive computation rules for the forward and backward variables, within which only the states active at the respective previous time step are considered (cf. equations (5.8) and (5.16)):

$$\alpha_{t+1}(j) := \sum_{i \in \mathcal{A}_t} \{\alpha_t(i) a_{ij}\}\, b_j(O_{t+1}) \tag{10.3}$$

$$\beta_t(i) := \sum_{j \in \mathcal{B}_{t+1}} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \tag{10.4}$$

The state probability $\gamma_t(i)$, which is central for the parameter training and which is essentially obtained as a product of the forward and backward probability (see equation (5.15)), vanishes as a consequence of the search space pruning for all those times t and states i, for which no calculation of $\alpha_t(i)$ or $\beta_t(i)$ is performed. Especially at the beginning of the parameter training the proportionality factor B should be chosen very small and the beam width consequently large (cf. page 169), such that only a moderate limitation of the search is achieved and that possible solutions are not excluded due to still unsatisfactory estimates of the model parameters.

A simplification of the method is obtained, if the principle of the beam search algorithm is only applied within a modified forward algorithm. The calculation of the backward variables then can be limited to those states, for which a non-vanishing value of $\alpha_t(i)$ was determined, without the need to compute $\mathcal{B}_t$ explicitly.

10.3.2 Segmental Baum-Welch Algorithm

A further possibility for limiting the search effort during Baum-Welch training results, if the chronological structure of the data is exploited for reorganizing the model training. When assuming that the observation sequences to be processed represent signals, which evolve linearly in time, and using appropriately limited model topologies, "unnecessary" computations can be avoided. From the reference annotation, which defines the sequence of meaningful units for every section $\boldsymbol{O}$ of the sample set — e.g. the orthographic transcription of a spoken utterance —, one again obtains a linearly organized HMM by concatenation of the corresponding partial models for, e.g., words. Here interactions between the partial models can only occur to a limited extent at model boundaries. In contrast, the association of a state of a partial model with a far away segment of the data is extremely unlikely. Therefore, one may assume, that the forward and backward probabilities will take on positive values only "within" the segments, which are defined by the mapping between a partial model and the corresponding data. Consequently, the associated forward and backward matrices will approximately have a block-diagonal structure.

When neglecting the interactions across segment boundaries, the block structure can be fixed *before* the beginning of the training and the evaluation of the probability quantities can be limited accordingly. As the basis for this decision serves the segmentation of the data by means of the current model. As this is not yet optimally adapted to the sample set, one only obtains an approximate solution of the segmentation problem.

Within the segments determined such one can then compute the statistics necessary for the parameter estimation locally by means of the Baum-Welch algorithm. The cumulative statistics for determining updated model parameters result in complete analogy to the method for handling multiple observation sequences, which was described in section 5.7.3 on page 90, as a suitable summation over all individual contributions[3].

The combination of a segmentation of the training data by means of the Viterbi algorithm and a subsequent parameter training only on the individual segments found corresponds to the evaluation of a block-diagonal matrix of forward and backward probabilities. In [43] this method is called segmented training. However a certain loss in the accuracy or flexibility, respectively, of the method may result from the strict specification of the segment boundaries *before* the start of the parameter training. In the so-called semi-relaxed training this disadvantage is avoided by working with overlaps between the respective neighboring blocks or segments [43].

By means of a "pre-segmentation" of the data before the application of the training procedure especially on long observation sequences considerable improvements in efficiency can be achieved. If, for example, an HMM with 100 states shall be trained on a sequence of length $T = 1000$, for forward and backward probabilities 100×1000 matrices, i.e. $2 \cdot 10^5$ values each, need to be computed. In case that the data is partitioned into 10 segments of an average length $T' = 100$, which are associated with partial models of on average 10 states each, in contrast, only matrices of size 10×100 need to be processed ten times and thus only $2 \cdot 10^4$ probabilities must be evaluated. Therefore, already in this simple case one obtains a considerable

[3] In fact already the segmentation of a larger sample set into partial observation sequences corresponds to this procedure.

improvement in the efficiency by a factor of 10, which is the number of segments used.

10.3.3 Training of Model Hierarchies

In HMM-based recognition systems one is principally always interested in the use of models that are as specific as possible. However, for the description of rare events usually also general HMMs are required, which then need to be trained in addition to the special models. In the field of speech recognition it is, for example, common practice to also include general context independent phone models besides the widely used triphone models into an overall recognition system.

In order to be able to correctly estimate the parameters of different modeling components, normally a separate training of the respective models would be required. However, usually a generalization hierarchy exists between the HMMs. Therefore, under certain conditions the parameters of more general models can approximately be derived from the statistics gathered during the estimation of more special HMMs.

Let us assume, that not only a unique relation exists between special and general models, but that also the states of specialized models can be uniquely associated with the ones of the more general HMMs. In continuous HMMs furthermore the output probability density functions of the corresponding model states need to be "compatible", i.e. they must use the same set of baseline densities. This is trivially the case for semi-continuous HMMs, but can also be achieved in all other variants of tied mixture models.

Under these conditions the statistics for the estimation of the parameters of the respective general states are obtained by simple summation over the values, which were computed for the associated special model states. As an example we want to consider a triphone modeling, which uses the same number of states for all baseline models. The respective n-th state of the triphone model $\texttt{x/a/y}$ then corresponds to the n-th state of the monophone model $\texttt{/a/}$. For updating the model parameters of the context independent model all training samples have to be considered, which are assigned to one of the associated special model states. Therefore, one obtains the state probability $\gamma_t(\texttt{/a/}_n)$ for the n-th monophone state approximately as a sum over all the state probabilities of the respective triphone states:

$$\gamma_t(\texttt{/a/}_n) = \sum_{\texttt{x}} \sum_{\texttt{y}} \gamma_t(\texttt{x/a/y}_n)$$

In analogy to this the mapping probability $\xi_t(j, k)$ for the mixture components can be approximated. On the basis of those two quantities then estimates for the model parameters of the general HMM can be computed in the usual way (cf. section 5.7 page 76).

However, the parameter estimates obtained such constitute only approximate solutions. In fact the existing model parameters of the general model would need to be used for correctly determining the state probability and all other statistics. As the general models in practice are only used to complement the modeling with robust baseline models, it is mostly irrelevant, whether their parameter estimates were only determined approximately.

10.4 Tree-like Model Organization

Besides the use of special algorithms also modifications of the model structure can be used, in order to increase the processing efficiency of Markov models. Both for HMMs and for n-gram models tree-like structures offer good possibilities here.

10.4.1 HMM Prefix Trees

In all larger HMM model structures the number of model states by far exceeds the number of different state parameter sets used. Methods for tying of model parameters either group similar parameter sets in a data-driven manner, in order to ensure their robust trainability, Alternatively, they exploit structural properties of the models created, in order to reuse partial models at different positions (cf. section 9.2, page 152). However, tying alone can not reduce the size of the total model, as then in general unwanted paths through the model would become possible.

In HMM systems with large inventories of segmentation units, as they are common in the fields of speech or handwriting recognition, a rather simple possibility exists for deriving a compression of the model structure from the tying of partial models. In such systems the necessary word models of the recognition lexicon are usually built from a sequence of elementary sub-word units — e.g. phone or character models. The total model then principally results from a parallel connection of the respective individual models (cf. section 8.3, page 131). For the correct evaluation of the path score however, it is, irrelevant, whether identical state sequences at the beginning of a word model were passed through separately, or whether these were combined into a single sequence. The structure of the overall model, therefore, can be simplified considerably by merging identical model prefixes of the individual partial models contained within the total recognition lexicon. The resulting tree-like model structure is referred to as *prefix tree* or phonetic prefix tree, if identical sequences of phone models are represented only once (cf. [163, 164, 175]).

In systems for speech or handwriting recognition that use large lexica this technique can be considered to be standard today. Depending on the application a compression of the search space by a factor of 2 to 5 can be achieved. The gain in efficiency is even bigger, as the reduction of the model complexity mostly affects the beginning of word models. In this area the biggest search effort arises, as the scores of the partial paths are quite unsure after only a few feature vectors belonging to a new segmentation unit [164]. However, when using such a “tree lexicon” it is not known before a leaf node is reached, which word from the recognition lexicon was hypothesized. This fact needs to be taken into account appropriately, if an n-gram model shall be integrated into the search procedure (cf. section 12.3, page 194).

Figure 10.2 shows an example for the organization of the recognition lexicon of a hypothetical speech recognition system in the form of a phonetic prefix tree. The

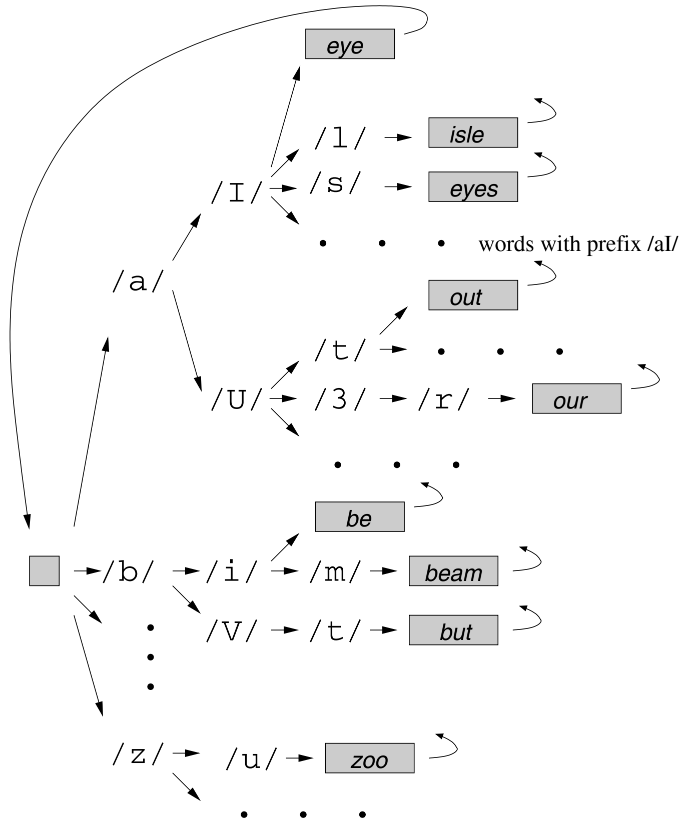


Fig. 10.2. Example of the tree-like organization of the recognition lexicon for a hypothetical speech recognition system.

root node of the tree serves the purpose of merging paths and corresponds to a non-emitting HMM state. From there now no longer all start nodes of all word models are reached as within a linear organization of the lexicon. Rather, the successor nodes of the root of the prefix tree collect the paths trough all word models, which start with the same phone model, respectively. As soon as a phone sequence corresponds to a complete word from the recognition lexicon, the path loops back to the root node. However, such a word model can also be the prefix of additional paths in the lexicon tree.

10.4.2 Tree-like Representation for n-Gram Models

At first sight n-gram models seem to have no appreciable structure at all. The conditional probabilities $P(z|\boldsymbol{y})$ for all combinations of predicted word z and history $\boldsymbol{y}$ could principally be stored in a table. However, such a trivial representation is manageable only for very small lexica and very limited context lengths in practice.

For a tri-gram model with a lexicon of $20\,000$ words already $8 \cdot 10^{12}$ probabilities would need to be stored, which in 4-byte floating-point format would take up approximately 29 terra-bytes of memory. In order to be able to still work with such models, therefore, an efficient representation is mandatory.

The possibilities for structuring the storage of n-gram models immediately result from the scheme used to compute the model parameters. A dramatic reduction in the amount of memory required already results, if only scores of *observed* events are represented explicitly. The conditional probabilities of unseen n-grams can be computed by backing off or interpolation when needed (cf. section 6.5.2, page 104). As the majority of events is not observed in practice, thus a reduction in the storage requirements to manageable sizes is achieved[4]. The necessary computational effort, however, is almost negligible, as only a few multiplications are required.

However, for the computation of n-gram scores of unseen events the zero probability of the potential contexts as well as the parameters of the necessary more general distributions need to be stored. Thus one obtains a hierarchical organization of the model parameters. As the more general distributions are usually also represented by n-gram models, this principle can be applied recursively.

The parameters of such an n-gram model hierarchy could principally be stored in individual tables [237]. However, a much more flexible storage method results from a tree-like representation of the overall model. Individual tree nodes then represent different observed events. In every node the n-gram score and the normalized[5] zero probability is stored, which results for $n + 1$-grams with this history. The successor relation in the tree corresponds to an extension of the event $\boldsymbol{y}$ by a word z to the right. Thus the successors of node $\boldsymbol{y}$ are all $n + 1$-grams $\boldsymbol{y}\cdot$. When only considering, how a certain n-gram is found in that representation, one deals with a *prefix tree*.

Figure 10.3 illustrates this representation of n-gram models with the example of a tri-gram model. In order to be able to specify the score $P(z|\boldsymbol{y})$ of an observed event in a simple way, in the example the principle of backing off is applied. Then the score is directly given by the respective value of the reduced frequency distribution $f^*(z|\boldsymbol{y})$. Furthermore, for reasons of simplification the respective $n - 1$-gram model shortened by one context symbol is used as the more general distribution. The root note $\perp$ of the tree, therefore, represents the zero-gram model, which assigns a probability of $\frac{1}{|V|}$ to every event, where V denotes the lexicon used. As the recursive computation procedure for n-gram scores terminates here, no zero probability is stored in the root node. The successors of the tree root, i.e. the tree nodes on the first level, store the parameters of the uni-gram model and at the same time define the

[4] A further considerable compression of n-gram models can be achieved, if rare events are neglected. For singletons — i.e. n-grams observed only once — this results from the application of absolute discounting with a discounting constant $\beta = 1$. If the modeling quality is not of primary concern but mostly a representation as compact as possible should be generated, additionally also parameters of other rarely observed n-grams can be eliminated from the model.

[5] If the more general distributions are not combined with the special model by interpolation but via backing off, the normalization factor $K_{\boldsymbol{y}}$ must be taken into account (cf. equation (6.7), page 107).

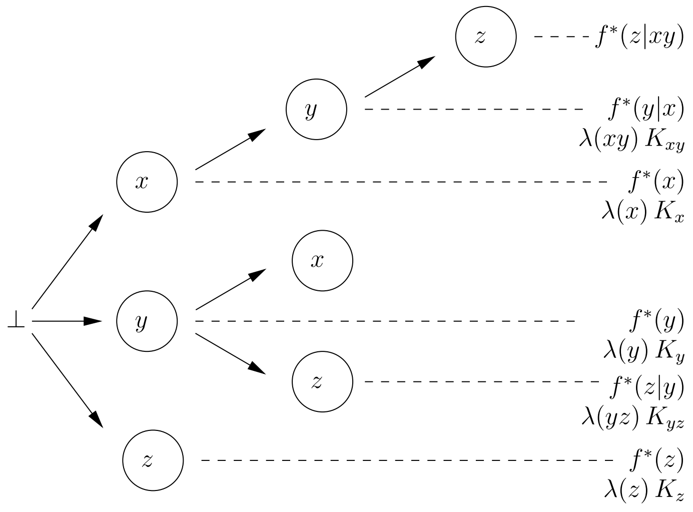


Fig. 10.3. Prefix tree for storing n-gram parameters exemplarily shown for a tri-gram model.

respective histories of the observed bi-grams. Over the tree nodes of the second level, which contain the bi-gram parameters, one finally reaches nodes, which correspond to observed tri-gram events. These will always be leaf nodes. However, depending on the respective sample set also bi-gram or uni-gram nodes can result that have no successors in the tree.

In a tree-like representation of an n-gram model, which is based on the recursive computation scheme for the model parameters, three principally different access types can be distinguished, which are summarized in figure 10.4. In the style of the terminology used in conjunction with cache memories we will speak about an *n-gram hit*, if the score of an observed event can be retrieved by a direct access to the respective tree node. For unseen evens, however, one obtains an *n-gram miss*. This can occur once or multiple times until a corresponding more general event is found. For such an access to succeed, always the complete history of the event considered needs to be represented. If this is not the case, we will call this a *history miss*. In this case the longest suffix of the current history needs to be determined, for which still model parameters exist. Consequently, a shortening of the n-gram context by at least one, but eventually more symbols is performed.

The representation of an n-gram model by means of a prefix tree presented above allows an efficient processing of n-gram hits. However, already for an n-gram miss a second pass through the tree from the root is necessary in order to determine the score of the respective $n-1$-grams. If this does not exist either, the procedure might need to be repeated multiple times. A similar problem arises for history misses. First, an access with the complete history must be tried in order to find out, that this does not exist in the current model. Afterwards the context can be shortened by one symbol

n-gram hit :
The score of an observed event xyz is retrieved, i.e. $c^*(xyz) > 0$.
$$P(z|xy) \leftarrow f^*(z|xy)$$
n-gram miss (simple):
The event xyz was not observed, i.e. $c^*(xyz) = 0$, but there exist other n-grams with history yz, i.e. $c(yz) > 0$.
$$P(z|xy) \not\leftarrow f^*(\cdot|xy) \Rightarrow P(z|xy) \leftarrow \lambda(xy)K_{xy}P(z|y)$$
history miss (simple):
The history xy of an n-gram does not exist in the current model, but only its suffix y.
$$P(z|xy) \not\leftarrow f^*(\cdot|x\cdot) \Rightarrow P(z|xy) \leftarrow P(z|y)$$

Fig. 10.4. Possible access types for n-gram model parameters for the example of a tri-gram model using backing off.

and another access can be tried. This procedure needs to be repeated until an existing suffix of the original history is found.

For both n-gram as well as history misses unnecessary search effort results from the prefix representation, which also increases considerably for n-gram models of higher order. Therefore, it is quite evident to reorganize the tree representation such, that the abovementioned access types can be processed efficiently. As within n-gram contexts not their prefix but the respective last symbols, which lie closest to the predicted word, have the most significance, it is obvious to store the n-gram histories in a suffix tree. As the nodes for predicted words need to be accessible from the respective history nodes, as in the simple prefix tree, we will call this representation scheme of n-gram parameters a combined suffix-prefix tree.

As shown exemplarily for a tri-gram model in figure 10.5, a path from the tree root encodes the history of an n-gram in reverse, i.e. starting from the direct successor of the word to be predicted. By traversing the tree from the root thus automatically the longest existing suffix of a given n-gram history is found. A history miss, therefore, causes no additional search effort at all. The scores of the words z, which were observed in a certain context, are stored compactly as a table per tree node. As at this point the passing through the symbol sequence of an n-gram is quasi reversed again, these scores can not be stored in the tree nodes themselves. These only serve the purpose of representing n-gram contexts in a suffix coding. An n-gram hit can be processed in the combined suffix-prefix representation as efficiently as in the simple prefix tree. As the n-gram contexts become more special when proceeding along a path from the root, one obtains the reversed sequence of the respective more general n-gram histories. In case of an n-gram miss, therefore, the necessary parameters of the more general distributions can be found by simply backtracking the path previously passed through from the root.

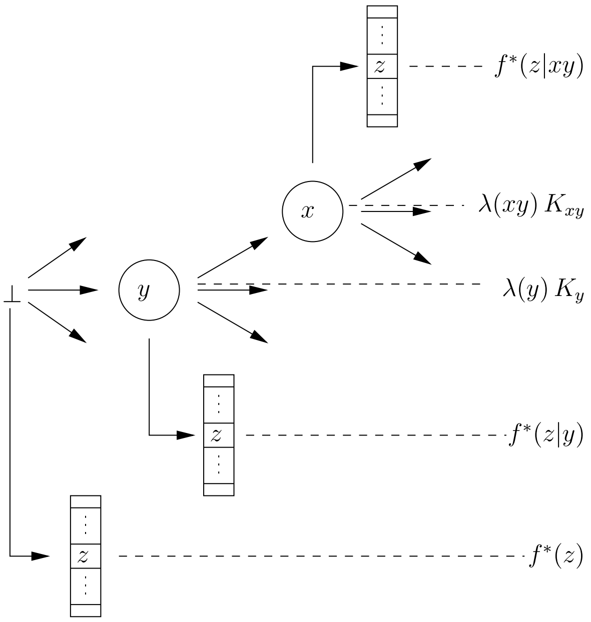


Fig. 10.5. Combined suffix-prefix tree for storing the parameters of an n-gram model for the example of a tri-gram model.

Model Adaptation

Both HMMs and n-gram models are normally created by using some sample set for training and are afterwards applied for the segmentation of *new* data. This is by definition not part of the training samples and can never be in practical applications. The characteristic properties of this test data can thus only to a limited extent be predicted on the basis of the training material. Therefore, in general differences between training and testing material will occur that can not be captured by the statistical models created and in the end adversely affect the quality of the results achieved.

Such changes in the characteristics of the input data can be illustrated with examples from automatic speech recognition. Already rather simple deviations in the conditions of the signal recording often lead to significantly increased error rates. This can be caused by the use of a different microphone or be due to the fact, that the training data was gathered in a quiet office environment, but the recognition system is used out-doors or in a vehicle. Also unexpected interfering noises caused by wind, cars driving by, or conversations of persons that can be heard in the background on a party lead to a reduced system performance. But also changes in the characteristics of the user of the system cause severe problems. Even so-called speaker-independent recognition systems degrade in performance, if they are confronted with an unknown dialect or if children use a speech recognition system that was trained for the speech of adult persons.

All these changes concern the statistical properties of the data to be processed. In contrast, we do not want to consider differences on the semantic-pragmatic level, if, for example, it is tried to order pizza from a flight information system, or a speech recognition system is fed with seismic data.

11.1 Basic Principles

It is the common goal of all methods presented in the following sections, to compensate differences, which concern the statistical properties of the data, between the training and testing conditions of a recognition system based on Markov models. A seemingly obvious solution of the problem might be to train specialized models for every single purpose. However, this approach fails for two important reasons. On the one hand, testing material is by its very nature unknown, as already pointed out above, and only in a laboratory environment conditions can be set up, which ignore this fact. On the other hand for the training of statistical models large amounts of data are required. It should, therefore, not be hoped that for every special purpose a complete representative training set can be collected anew.

To a quite limited extent, however, specialized material will always be available. At the least the actual test data itself can be considered as an example for the current properties of the input data to be expected. Often it is even possible to collect a few up to date samples before the actual application of the recognition system. This so-called adaptation set can then be used for adjusting the given system parameters to the respective task. In standard dictation systems, for example, the user is required to read some given texts as adaptation material, before the system is ready to use.

In contrast to the training of a model, its adaptation consists in the transformation of a given general parametrization by means of an extremely limited adaptation set into a special parameter set, which is better suited for the processing of the data actually observed during the current application of the model. Therefore, by their very nature also adaptation techniques are methods for parameter estimation. They are different from classical estimation methods, because they need to cope with some orders of magnitude fewer training samples.

If adaptation data is available in addition to training and testing material, the adaptation of the models can be performed prior to their actual application. This adaptation in the manner of batch processing is also called *batch adaptation*. In this approach for simplification and for improving the results the correct annotation of the adaptation set can be given and one obtains a *supervised* adaptation method. Often, however, this is not possible due to the effort required. Then the annotation of the adaptation data needs to be determined automatically on the basis of the existing general model. The procedure is then called *unsupervised* adaptation. An unsupervised method also is the only possibility, if the adaptation can not be performed on separate sample data but directly during the evaluation of the models on the test data itself. Such so-called *online* adaptation methods are undoubtedly the most flexible methods for model adaptation, as they do not require any changes in the natural use of the respective system. Furthermore, the parameters can be *continuously* adapted to changes in the characteristics of the data.

11.2 Adaptation of Hidden Markov Models

The adaptation of a general HMM to special changed characteristics of the data to be processed can principally be performed by means of the training methods known from section 5.7, page 76 (cf. e.g. [48]). However, then a quite large adaptation set needs to be available, in order to be able to robustly estimate the specialized HMM parameters. In the case of speaker adaptation, i.e. the adaptation of an existing speech recognition system to a new user, then at least a few hundred utterances of the respective person need to be available as adaptation material.

If only quite few example data is available for the adaptation of a given model, the parameter estimates, which are, e.g., obtained by applying the Baum-Welch algorithm, are extremely unreliable. Therefore, these are not directly used as specialized models, but are interpolated with the parameters of the general model in a suitable way. The interpolation weights are usually dependent on the amount of sample data, which are available for estimating certain model parameters. Thus it can be achieved, that parameters, for which may adaptation examples are available, contribute to the special model almost without modification. However, for rarely observed events the known parameters of the general model prevail.

Formally this corresponds to a parameter estimation technique following the principle of *maximum a posteriori estimation* (MAP, cf. section 3.6.2). In contrast to the frequently applied maximum likelihood estimation (ML, cf. section 3.6.1), which is also the basis of the Baum-Welch algorithm, here the posterior probability of the model parameters θ is maximized for given data ω (cf. e.g. [97, pp. 445–446]):

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(\theta|\omega) = \underset{\theta}{\operatorname{argmax}}\, P(\theta)\, p(\omega|\theta)$$

A model adaptation on the basis of the MAP principle offers the advantage, that the estimates computed converge against the ones of an ML estimation with increasing size of the adaptation set. Thus the MAP estimation becomes equivalent to the result of a standard training procedure if truly large amounts of adaptation material are available.

However, for the application of this principle it is necessary to define estimates for the prior probability $P(\theta)$ of the parameters themselves. Furthermore, a fast specialization on only few example data should not be expected, as then the original model parameters still dominate the MAP estimates. Especially for partial models, states, or probability densities, for which no adaptation samples are available, no improved parameters can be derived by this method[1].

A radically different way for the specialization of models to certain application areas is followed by methods, which were proposed for improving the modeling of variations in speaking rate by automatic speech recognition systems [154, 215, 251]. There different specialized models, e.g. for slow, medium, and fast speech, are created in advance by classifying the available training material accordingly and splitting it up into separate sample sets. The adaptation to the actual task is then performed in a relatively simple way by choosing the model, which matches the data best. In the simplest case the speaking rate of the test data is determined by a classifier, and then the respective model is used for the segmentation [154]. Better results are obtained by a method, which, however, is considerably more demanding. There all available models are used in parallel to generate a segmentation of the test data. Afterwards one decides for the result — and consequently also for the respective model — that achieved the best score [251].

Though in this method an adaptation of the HMMs is performed, it can principally only capture variations, which were anticipated in advance in the design process

[1] This effect can be partially avoided in practice by a suitable tying of parameters (cf. section 9.2, page 152).

and which are covered by sample data from the training set. With this method it is not possible to react upon an unknown variation of the test data.

Maximum-Likelihood Linear-Regression

The most successful method for the fast adaptation of HMM parameters on very limited data is the so-called *maximum likelihood linear regression* (MLLR) [138, 139]. This technique meanwhile belongs to the set of standard methods for applying HMMs to pattern recognition problems (cf. [97, pp. 447ff]).

In contrast to classical training methods, which attempt to compute new estimates for *all* model parameters independently on the sample data, MLLR creates a specialized model by means of a transformation. As opposed to the adapted HMM itself this has only quite few free parameters, which can be estimated robustly also on a small adaptation set. As the transformation modifies all model parameters jointly, also such parameters can be adapted, for which no adaptation samples were observed.

In order to simplify the method, MLLR assumes that the most important parameters of an HMM are the mean vectors of the probability densities used. In contrast to transition probabilities, mixture weights, and also covariance matrices changes in the statistical properties of the data affect these the most. By means of an affine transformation the mean vectors of the general model are adapted to a modified data distribution[2]. Every mean vector $\boldsymbol{\mu}$ is transformed into an adapted mean vector $\boldsymbol{\mu}'$ by the following formula:

$$\boldsymbol{\mu}' = \boldsymbol{A}\,\boldsymbol{\mu} + \boldsymbol{b}$$

Here $\boldsymbol{A}$ denotes the rotational part of the transformation and the vector $\boldsymbol{b}$ the associated translation in the feature space. When forming an extended mean vector $\tilde{\boldsymbol{\mu}} = [1, \boldsymbol{\mu}^T]^T$ and a combined transformation matrix $\boldsymbol{W} = [\boldsymbol{b}, \boldsymbol{A}]$, the formula can be compactly written with a single matrix multiplication only:

$$\tilde{\boldsymbol{\mu}}' = \boldsymbol{W}\,\tilde{\boldsymbol{\mu}}$$

The optimization criterion of the method is the production probability of the data, analogously to the Baum-Welch algorithm. Therefore, the transformation of the model parameters needs to be determined such, that the probability for generating the adaptation data by the specialized model is maximized.

As far as the amount of available sample data permits, also multiple independent transforms on different groups of probability densities can be applied to improve the accuracy of the MLLR method. These groups of densities, for which a separate parameter transformation is estimated, are called *regression classes*[3]. As all mean vectors of densities from the same regression class are adapted in an identical way,

[2] By means of suitable extensions of the MLLR method also the parameters of covariance matrices can be adapted [83]. The rather moderate improvements, however, do usually not justify the considerably increased additional effort in practice.

[3] If all mean vectors are adapted by a single affine transformation only, a single regression class comprising all densities is used.

such distributions should be grouped, which describe similar parts of the model. An obvious choice is, for example, to group all densities of a codebook jointly used by multiple states into a regression class.

For computing estimates of the MLLR transformation of one or more regression classes a mapping between feature vectors $\boldsymbol{x}_t$ from the adaptation set and codebook classes is required. In general this is performed probabilistically in the same way as when estimating the distribution parameters of continuous HMMs by means of the Baum-Welch algorithm (cf. page 80). The quantity $\xi_t(j,k)$ represents the probability for using the k-th mixture component of state j for generating the data at time t. It can be computed on the basis of the forward and backward probabilities (cf. equation 5.20, page 83). Through a formal derivation, which essentially corresponds to the one of the Baum-Welch algorithm, one obtains the following constraint equation for determining the MLLR transformation:

$$\sum_{t=1}^{T} \sum_{g_{jk} \in R} \xi_t(j,k) \boldsymbol{K}_{jk}^{-1} \boldsymbol{x}_t \tilde{\boldsymbol{\mu}}_{jk}^T = \sum_{t=1}^{T} \sum_{g_{jk} \in R} \xi_t(j,k) \boldsymbol{K}_{jk}^{-1} \boldsymbol{W}_{\boldsymbol{R}} \tilde{\boldsymbol{\mu}}_{jk} \tilde{\boldsymbol{\mu}}_{jk}^T \qquad (11.1)$$

Here R represents the respective regression class, which consists of a set of baseline densities g_{jk} of the mixture densities used within the overall model. The associated transformation matrix is denoted by $\boldsymbol{W}_R$. Additionally the feature vectors $\boldsymbol{x}_t$ as well as the inverse covariance matrices $\boldsymbol{K}_{jk}^{-1}$ and the extended mean vectors $\tilde{\boldsymbol{\mu}}_{jk}$ of the densities enter into the calculation.

However, for the constraint equation in the general form given above no closed form solution exists. Frequently the covariance matrices of normal distributions are described by diagonal covariances only in practical applications of HMMs. In this constrained case the transformation matrices $\boldsymbol{W}_R$ can be computed in a row-wise procedure (cf. [139], [97, p. 449]).

A considerable simplification of the MLLR method for practical applications results, if the mapping of codebook classes to adaptation samples is uniquely determined. In the same way as in Viterbi training or the segmental k-means algorithm one used the Viterbi criterion to first establish a unique association between model states and feature vectors. In case of a supervised adaptation for this the correct annotation of the sample data is given, e.g. the correct orthographic transcription of an utterance in the case of automatic speech recognition. In contrast, if the MLLR method is applied in an unsupervised manner, the segmentation can also be generated without the respective restrictions by means of the existing model alone.

On the basis of the optimal state sequence the respective optimal codebook class can then be determined easily as the baseline density $\boldsymbol{W}_R$ of the output distribution of the optimal state s_t, which has maximal posterior probability:

$$m_t = \underset{k}{\operatorname{argmax}}\, c_{s_t k}\, g_{s_t k}(\boldsymbol{x}_t) = \underset{k}{\operatorname{argmax}}\, c_{s_t k}\, \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_{s_t k}, \boldsymbol{K}_{s_t k})$$

Furthermore one assumes, that the covariance matrices of the individual densities can be neglected or can be assumed to be identity matrices, respectively. One then obtains an estimation procedure, which determines the parameter transformation such, that the mean squared error between the feature vectors $\boldsymbol{x}_t$ and the associated mean vectors $\boldsymbol{\mu}_{s_t m_t}$ is minimized ([139, Sect. 4.1: Least Squares Regression], [74]). When using a single regression class only the following simple formula for computing the transformation matrix results:

$$\boldsymbol{W} = \left\{ \sum_{t=1}^{T} \boldsymbol{x}_t \tilde{\boldsymbol{\mu}}_{s_t m_t}^T \right\} \left\{ \sum_{t=1}^{T} \tilde{\boldsymbol{\mu}}_{s_t m_t} \tilde{\boldsymbol{\mu}}_{s_t m_t}^T \right\}^{-1} \tag{11.2}$$

If multiple regression classes are used the computation of the individual transformation matrices follows the same scheme. Then only those adaptation vectors $\boldsymbol{x}_t$ are taken into account, for which the associated codebook class $g_{s_t m_t}$ belongs to the respective regression class.

Doubtlessly the MLLR method offers the fastest possibility for the adaptation of an existing HMM to changed task conditions. When using a single regression class only with a feature dimension of 39, which is typical for speech recognition systems, less than 1600 parameters need to be estimated for the global transformation matrix $\boldsymbol{W}$. In practice this can be achieved in satisfactory quality with less than a minute of speech material.

Of course the quality of the adaptation method can principally be arbitrarily increased by the use of many different regression classes. However, then also the demand for adaptation data increases similarly, and in the end the method does not offer any advantages as opposed to a traditional training procedure. In order to achieve an optimal compromise between accuracy of the adaptation and robustness in the estimation of the transformations' parameters, methods were proposed, which try to optimally define the number of regression classes depending on the size of the adaptation set [22, 139], or which, starting from a single regression class only, increase their number step by step, as soon as sufficient data becomes available [138].

An extension of the MLLR technique, where multiple independent transformations are combined, was proposed by Digalakis and colleagues [21, 22, 46, 47]. For the purpose of speaker adaptation on the training material first adaptation transforms are estimated for typical speaker groups. The number of regression classes used is optimized depending on the available adaptation material. For the adaptation of a general HMM to a specific test speaker then an optimal linear combination is computed from all available transformations. The estimation of the combination weights requires considerably less data than the computation of an MLLR adaptation rule. Therefore, the method is suited very well for the fast adaptation of large models to changed task conditions.

11.3 Adaptation of n-Gram Models

For n-gram models it is even more evident than for HMMs, that the estimation of a completely new specialized model on a limited adaptation set cannot be achieved. In fact even on large training sets it is absolutely necessary to determine useful estimates for probabilities of unseen events by smoothing the empirical distributions (cf. section 6.5, page 100).

11.3.1 Cache Models

So-called *cache n-gram models* ([130], cf. also [38, 103]) are based on the idea, that in texts such words or word combinations re-occur with high probability, that were already used before. As it is practically impossible to update statistics for bi-grams or n-grams of higher order with only a few adaptation samples, for the modeling of this effect only special uni-gram statistics are gathered. In order to be able to better predict the occurrence of a word w_t at the position t in a text, one calculates the frequency $c(w_t|w_{t-T}, \ldots w_{t-1})$ of its occurrence in a certain section of past text material of length T, i.e. in the so-called *cache*. The uni-gram probability based on the cache data is then obtained as follows:

$$P_{\text{cache}}(w_t) = \frac{c(w_t|w_{t-T}, \ldots w_{t-1})}{T}$$

The simple cache uni-gram model estimated such is subsequently interpolated with a standard n-gram model $P(z|\boldsymbol{y})$:

$$P_{\text{cache}}(z|\boldsymbol{y}) = \lambda\, P(z|\boldsymbol{y}) + (1 - \lambda)\, P_{\text{cache}}(z)$$

However, the required interpolation weight λ needs to be determined experimentally or be specified heuristically.

In the work of Kuhn & De Mori [130], who coined the term cache n-gram model, a category-based tri-gram model is used. On the basis of the cache memory merely the probabilities $P(w_i|C_j)$ for mapping between categories and word symbols are adapted, so that a good robustness of the model is achieved.

11.3.2 Dialog-Step Dependent Models

A completely different approach for the adaptation of the language modeling is pursued by the so-called *dialog-step dependent language models* (cf. e.g. [53, 192, 236, 238]). In combination with a system for carrying out a natural language dialog it is possible, to use the prediction of the dialog system for the selection of a language model, which is adapted to the class of utterances to be expected. In fact no real adaptation of the model parameters on the basis of example data is performed, but a selection based on the internal state of the system.

By splitting up the training material into separate dialog-steps, however, the data for the estimation of the individual n-gram models is considerably reduced [192]. Especially for rare types of dialog-steps possibly not enough material might be available for robustly estimating a specialized n-gram model. In such cases either a suitable grouping of the available dialog states needs to be performed [238], or the robustness of the dialog-step dependent models needs to be improved by interpolation with a general language model [236].

11.3.3 Topic-Based Language Models

A similar basic idea as the dialog-step dependent modeling is followed by the so-called *topic-based language models* (cf. e.g. [16, 38, 103]). In this modeling it is assumed, that within large text material certain topics can be identified and that the associated text sections exhibit specific statistical properties. In order to better describe the overall material it is, therefore, quite obvious to use specialized partial models for the individual topics.

As in contrast to dialog-step dependent language models the mapping of a certain text section to a topic and thus to a specialized n-gram model can not be determined uniquely, topic-based n-gram models are mostly used in the form of so-called *topic mixtures*. All partial models $P(z|\boldsymbol{y}, T_i)$ created for a special topic T_i are subsumed into an overall model by a linear combination. The most frequently used is a combination on the level of individual conditional probabilities for predicting the respective following word:

$$P_{\text{topic}}(z|\boldsymbol{y}) = \sum_i \lambda_i \, P(z|\boldsymbol{y}, T_i) \quad \text{with} \quad \sum_i \lambda_i = 1$$

In contrast to this, in [103] the overall probability of a test text $w_1, w_2, \ldots w_T$ is defined as a weighted sum of the probability contributions, which every individual topic-dependent n-gram model produces for the total word sequence. When using bi-gram models the following computation rule results:

$$P_{\text{topic}'}(w_1, w_2, \ldots w_T) = \sum_i \lambda_i \prod_{t=1}^{T} P(w_t|w_{t-1}) \quad \text{with} \quad \sum_i \lambda_i = 1$$

As opposed to the purely local interpolation of the models this definition can exclusively be used for the re-scoring of segmentation hypotheses, which are completely available in a multi-pass recognition system. A time-synchronous combination with the partial path probabilities of an HMM is not possible for such a model.

As for dialog-step dependent n-gram models the individual topic models are trained in advance on text material, which is representative for the respective topic. The adaptation of the overall model to a particular task is then achieved by the adaptation of the interpolation weights λ_i. This can either be performed by an unsupervised estimation procedure, as the EM algorithm [38], or on the basis of methods for topic identification, as they are known from the field of information retrieval [16, 150].

12

Integrated Search Methods

The probably most challenging applications of Markov model technology are recognition tasks with very large inventories of segmentation units. Typical examples are dictation systems with lexica of some 10 000 or 100 000 words or systems for handwriting recognition with almost unlimited vocabulary. The modeling of the segmentation units — i.e. the spoken or written words — by means of HMMs has quasi become standard, as in less complex systems too. However, for such demanding recognition tasks additional restrictions of the possible or plausible sequence of segments are indispensable, in order to keep the search effort manageable. The description of such restrictions by n-gram models offers the decisive advantage over other methods, that two compatible formalisms are used and thus a combined application is possible more easily and with greater success.

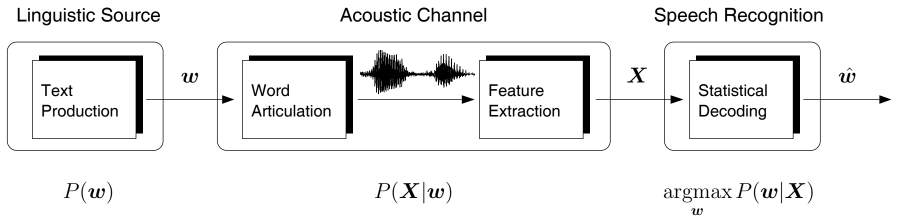


Fig. 12.1. Information-theoretic channel model of speech generation and recognition.

Formally this procedure can be derived from a modeling assumption, which is mainly used in the field of automatic speech recognition and presumably dates back to Jelinek and colleagues [107, 108]. This integrated probabilistic model of signal generation and decoding embeds the segmentation problem into the vocabulary of concepts known from information theory. Figure 12.1 schematically shows the so-called *channel model*, as it was formulated for the problem of speech generation and recognition.

There one assumes that first a hypothetical information source generates a sequence of symbolic units $\boldsymbol{w} = w_1, w_2, \ldots w_N$ with a certain probability $P(\boldsymbol{w})$. Thus, e.g., a human speaker or writer would formulate a word sequence in his brain with the goal to later pronounce it or write it down. The actual realization is then performed in a second step, where the symbolic information is transformed or encoded into a signal representation. The thought of word sequence is thus pronounced as an acoustic signal or realized as writing on a sheet of paper. The signal produced is now transmitted over a potentially noisy channel, recorded by a sensor, and transformed into a parametric representation as a sequence of feature vectors $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T$. For reasons of simplicity the last two steps are subsumed into one for the further considerations. Consequently, a coding of the symbol sequence $\boldsymbol{w}$ into a feature vector sequence $\boldsymbol{X}$ is performed with a certain probability $P(\boldsymbol{X}|\boldsymbol{w})$.

The process of decoding or recognition now tries to reconstruct the original "ideal" word sequence on the basis of these features. Due to the probabilistic nature of the generation, which also captures potential errors of the transmission channel, this is not uniquely possible. Therefore, one decides for the solution $\hat{\boldsymbol{w}}$, which maximizes the posterior probability $P(\boldsymbol{w}|\boldsymbol{X})$ of the word sequence given the observed data $\boldsymbol{X}$. Thus the following decoding rule results:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, P(\boldsymbol{w}|\boldsymbol{X})$$

As parametric models for the direct description of the probability $P(\boldsymbol{w}|\boldsymbol{X})$ for inferring the word sequence given the data are generally not known, the expression is rewritten using Bayes' rule as follows, in order to reduce the problem to probabilistic quantities, which can actually be modeled:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, P(\boldsymbol{w}|\boldsymbol{X}) = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, \frac{P(\boldsymbol{w})\, P(\boldsymbol{X}|\boldsymbol{w})}{P(\boldsymbol{X})}$$

A further simplification results from the fact, that the occurrence probability $P(\boldsymbol{X})$ of the data itself is constant with respect to the maximization and, therefore, can be neglected when determining the optimal word sequence $\hat{\boldsymbol{w}}$:

$$\hat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, P(\boldsymbol{w})\, P(\boldsymbol{X}|\boldsymbol{w}) \tag{12.1}$$

Both probability expressions involved can now be associated with potential modeling parts of a statistical recognition system.

The probability $P(\boldsymbol{w})$ for generating a symbol sequence $\boldsymbol{w}$ can easily be represented by a Markov chain, i.e. by a statistical n-gram model. The quantity $P(\boldsymbol{X}|\boldsymbol{w})$, in contrast, is interpreted as the probability to generate a certain observation sequence $\boldsymbol{X}$ with a given model — i.e. the one corresponding to the word sequence $\boldsymbol{w}$. This part can thus be described by means of a hidden Markov model. Furthermore it can be seen from equation (12.1) that the scores of both parts of the model can principally be combined by simple multiplication.

At this point the considerations are discontinued in most presentations found in the literature. Unfortunately, the integrated use of HMMs and n-gram models is not quite so easily achieved in practice.

On the one hand the dynamic ranges of the two score components usually differ largely. Therefore, without a suitable compensation the HMM score $P(\boldsymbol{X}|\boldsymbol{w})$ practically always dominates the score $P(\boldsymbol{w})$ of the language model[1]. Therefore, the two quantities are combined in a weighted fashion using a constant ρ according to (cf. e.g. [97, p. 610])[2]:

$$P(\boldsymbol{w})^{\rho}\, P(\boldsymbol{X}|\boldsymbol{w}) \tag{12.2}$$

In negative-logarithmic representation (cf. section 7.1) this corresponds to a weighted sum of the scores, where the constant ρ, which is referred to as the *linguistic matching factor*, serves the purpose of adjusting the usually significantly lower costs of the language model to the magnitude of the HMM score. The choice of ρ, unfortunately, is task specific and suitable values need to be determined in experimental evaluations[3].

On the other hand equation (12.1) only specifies, that the total score of a word sequence $\boldsymbol{w}$, which is completely available, can be computed by multiplying the scores of the two model components. It is, however, not evident, how this should be achieved within a search process. But without efficient search methods, as the Viterbi or the beam search algorithm, the decoding of HMMs is not feasible in practice. Therefore, instead of the evaluation of certain final solutions, partial path scores $\delta_t(i)$ need to be combined with parts of the n-gram score during the Viterbi search. The basis for this is the factorization of $P(\boldsymbol{w})$ into individual conditional probabilities $P(z|\boldsymbol{y})$ for the prediction of a word z in the context of $\boldsymbol{y}$ (cf. sections 6.1 and 6.3).

Despite the great importance of respective solutions for more complex applications of Markov models, their presentation is frequently neglected in the literature. For current systems for speech or handwriting recognition it is often difficult to find out from the respective publications, which algorithmic solutions were chosen for the problem of combining HMMs and n-gram models. The primary reason for this is probably, that the descriptions are mostly spread across many quite short conference publications and that the focus of those is more on details of the solution than on

[1] This is mainly due to the fact, that probabilities on largely differing time-scales enter into the calculations for $P(\boldsymbol{w})$ and $P(\boldsymbol{X}|\boldsymbol{w})$, respectively. For determining the HMM score state transition and output probabilities accumulate per frame, i.e. with the "clock pulse" of the signal. The score of the word sequence, in contrast, is generated by multiplying one conditional probability per word and, therefore, comprises one to two orders of magnitude fewer probability components.

[2] Frequently the reason for a weighted combination of n-gram and HMM scores being necessary is said to be, that the models were generated on different data and, therefore, would not be completely compatible. However, this is at most of marginal importance, as, e.g., experiments within the German Verbmobil project showed clearly. For the extensive evaluation 1996 the training data for HMMs and language model were identical. A weighting of the scores was still necessary, though.

[3] For the combination of n-gram models and semi-continuous HMMs in several speech and handwriting recognition systems developed in our labs values of $4 \leq \rho \leq 7$ achieved good results.

integrated approaches. Unfortunately, even in monographs on the topic hardly any satisfactory treatments of the problem can be found[4].

Therefore, we want to present the most important methods for the integration of HMMs and n-gram models in the following. At the beginning the probably oldest and also simplest technique is presented, where on the basis of partial models a model network is created, in which the n-gram scores serve as transition probabilities. In multi-pass search methods long-span n-gram restrictions are applied only in a second search process in order to reduce the total search effort. In combination with efficient representations of the recognition lexicon as a prefix tree it is necessary to create copies of the resulting search space, in order to be able to integrate the n-gram score directly into the search. The actual methods differ depending on whether the replicated search trees are identified by the n-gram history or by the respective starting time in the signal. The chapter concludes with a presentation of a flexible technique for the integrated time-synchronous search in combined HMM/n-gram models, which is also capable of handling long-span context restrictions efficiently.

12.1 HMM Networks

Technically the most simple and probably also the most obvious possibility for combining n-gram models and HMMs offer the so-called *HMM networks*, which for their origin in the field of speech recognition are also referred to as *word networks* (cf. [105, pp. 81–82], [196, pp. 453–454]).

A comparable method was first used in the framework of the HARPY system, yet without an integrated n-gram model [148]. All utterances accepted by the system were encoded as sequences of the respective word HMMs within a model graph.

If an n-gram model is to be included into such a structure, this can principally be achieved at the edges between consecutive words in the form of transition probabilities. When using a bi-gram model a simple “looped” lexicon model is sufficient for this. An example of such an HMM network is shown in figure 12.2. At the edges between the partial models a and b the respective bi-gram score $P(b|a)$ can be combined with the path score of the underlying HMM. As no predecessor words are available at the beginning of the search process, the uni-gram scores of the language model are used as start probabilities of the model.

However, if n-gram restrictions with larger context lengths, as e.g. tri-gram models, are used, this is no longer possible in the framework of this model structure. Due to its inherent limitations an HMM is only capable of considering *exactly one* context during the search in the form of the current predecessor state. In the model structure from figure 12.2 this encodes only the current predecessor word.

In order to be able to represent a longer sequence of context words uniquely, therefore, suitable copies of the partial models need to be created, which are specialized to certain long-span context restrictions. Figure 12.3 schematically shows a

[4] A notable exception is the book by Huang and colleagues, where a good overview over possible techniques is given [97, Chap. 13, pp. 645–662].

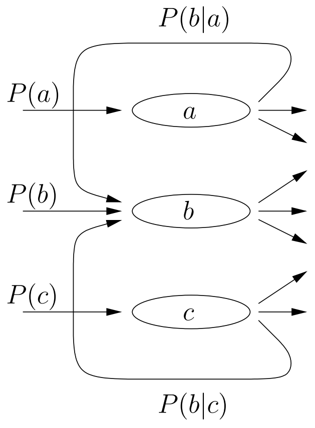


Fig. 12.2. HMM network for a trivial lexicon $\{a, b, c\}$ and the use of a bi-gram language model. For clarity only two loop edges are shown explicitly.

simple example of an HMM network, which allows for the incorporation of a tri-gram model. For every word model a copy exists per potential predecessor word. The search starts — as in the case of the bi-gram model — with the uni-gram score. After passing through a word model then bi-gram scores can be incorporated before the respective context dependent copies of the partial models are reached. These now represent the two-symbol tri-gram context uniquely. Therefore, from here on tri-gram scores can be used as transition probabilities provided that the transition is made to the right context-coding copy of the successor word.

The comparison with the simpler HMM network from figure 12.2, which was sufficient for the integrated decoding of a bi-gram model, shows, that a considerably higher effort is necessary. In general the required number of copies of the word models involved grows exponentially with the length of the n-gram context. Therefore, HMM networks are used almost exclusively in combination with bi-gram models in practice.

12.2 Multi-Pass Search

In order to avoid the exponentially increasing effort required by HMM networks for higher order n-gram models, but nevertheless to exploit their restrictions, methods for *multi-pass search* rely on an application of modeling parts with increasing complexity in multiple subsequent steps. For that procedure one assumes that the additional restrictions of an $n+k$-gram, as opposed to the ones of the n-gram model being also available, mainly lead to a different sorting of the solutions found, but not to completely new ones. If this assumption is fulfilled, the integrated search can use a language model of lower complexity — usually a bi-gram model — for the generation of a set of alternative solutions. These are subsequently re-scored with a

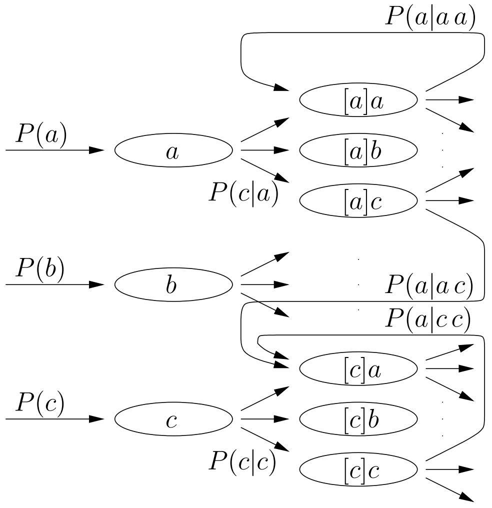


Fig. 12.3. HMM network for a trivial lexicon $\{a, b, c\}$ and a tri-gram language model. Per word model y copies $[x]y$ exist for coding the necessary context restrictions.

more demanding model (cf. e.g. [17, 84, 89]). The computational costs caused by decoding an n-gram model of higher order is thus limited to "relevant" search areas, which were already hypothesized in the first phase of decoding.

The main disadvantage of such a multi-pass strategy is that the data to be segmented needs to be completely available before a substantial part of the computations — i.e. the re-scoring steps — can be carried out. For batch processing applications, as, e.g., the transcription of broadcast news or the search for interesting samples in large genetic databases, this restriction can be neglected. This is different for interactive applications, as, e.g., in multi-modal man-machine communication. With a multi-pass analysis procedure, e.g., in the field of speech recognition demanding computations can not be started before the end of the spoken utterance. A fast reaction on inputs from the user, therefore, is hardly possible with such a method.

12.3 Search Space Copies

As already pointed out in section 10.4.1, a substantial increase in the efficiency of HMM search within large lexica can result from a tree-like representation of the models. However, the considerable compression of the HMM state space achieved in this way also has a serious drawback. When a leaf node of the tree is reached, which encodes the end of a word, the identity of the successor word is not known. Therefore, in an integrated model the n-gram score can not merely be combined with the HMM score in the form of a simple transition probability. Rather, the information about the word context needs to be saved until after the prefix tree is passed through anew and another leaf node is reached.

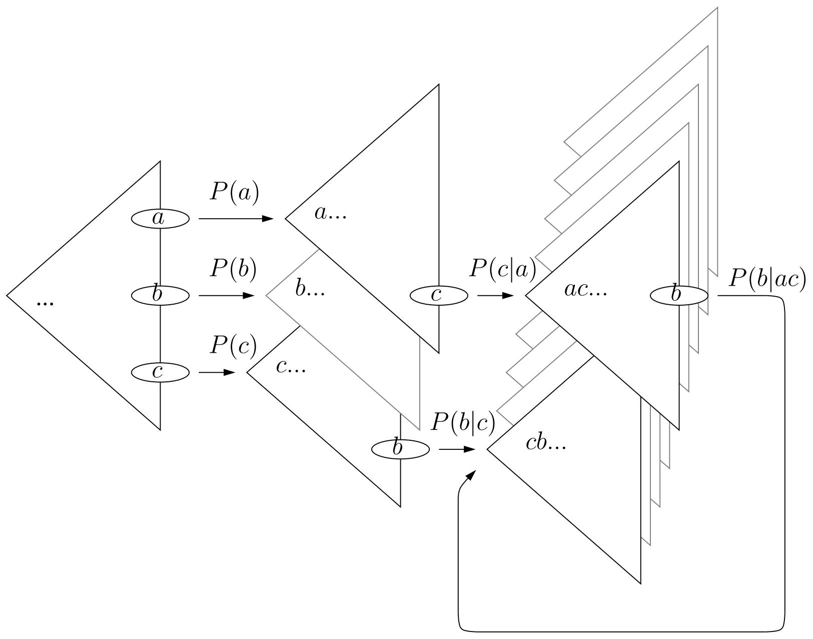


Fig. 12.4. Example of the overall search space when using context-based search tree copies and a tri-gram model with trivial lexicon $L = \{a, b, c\}$.

12.3.1 Context-Based Search Space Copies

In order to achieve the correct combination of an n-gram language model with an HMM organized as a prefix tree, a method was proposed, where dynamic copies of the search space are generated depending on the respective context to be encoded [164]. The algorithm, which originally was developed for bi-gram models only, was later extended to also handle tri-gram language models [178].

Essentially the method corresponds to the use of a compressed network of word nodes. The resulting model graph is dynamically created as required, which, however, is principally also possible for HMM networks. Figure 12.4 exemplarily shows the resulting overall search space when using a tri-gram model with trivial lexicon. The comparison with figure 12.3 makes clear, that the language model score is integrated in a principally similar way as in an HMM network. However, not before the end of the model actually predicted the score can be evaluated. It is thus taken into account in the search process with a delay of one word.

The resulting search procedure can in the same way as the method of HMM networks be realized as a direct extension of the beam search algorithm. In contrast to multi-pass search strategies it thus can be applied in a strictly time-synchronous way.

However, also the use of search space copies becomes extremely demanding for n-gram models of higher order. The exponential growth of the search space can

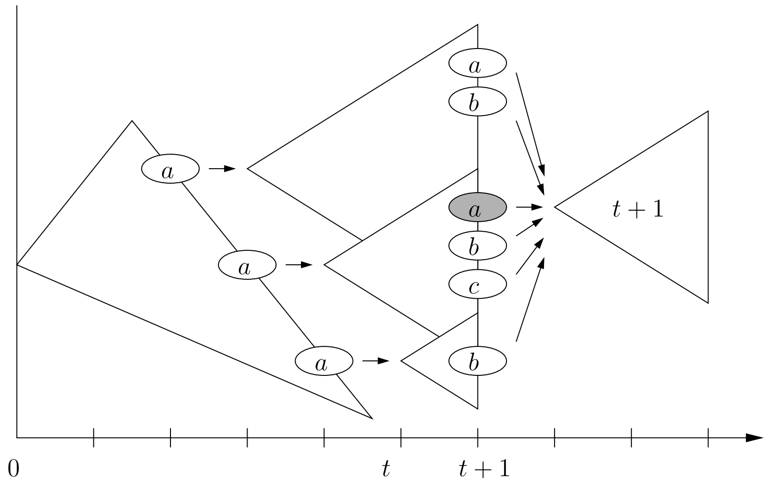


Fig. 12.5. Example of the overall search space when using time-based search tree copies and a bi-gram language model with trivial lexicon $L = \{a, b, c\}$.

not be counteracted arbitrarily even by a tree-like model organization or pruning strategies. This can also be inferred from the fact, that the use of such methods is only documented in the literature up to the complexity of trigram models (cf. e.g. [6, 163, 178]). Still, this method probably is the most efficient technique for the combined decoding of HMMs and bi-gram models.

12.3.2 Time-Based Search Space Copies

A variant of the method presented in the previous section was proposed in [178] and independently of that also in [71]. There the search space copies are not generated depending on the language model context, but on the starting time of the respective hypothesis.

Figure 12.5 exemplarily shows a part of the resulting search space organization. In contrast to the context-based creation of tree copies in this method different search trees are reached from hypotheses of the same word, which end at different times. On the other hand *all* hypotheses with the same end time t are continued in only *a single* search tree with starting time $t + 1$, even though they were possibly created in different search trees. Only the score δ_t^* of the optimal hypothesis ending at time t is taken into account. It serves as the starting score for the newly created search tree copy.

In order to still be able to correctly integrate n-gram models into the search, however, for the evaluation of the language model score in a word end node all potential hypotheses ending in the predecessor tree have to be considered. As for the partial path of the current solution the optimal score δ_t^* was used up to time t, this

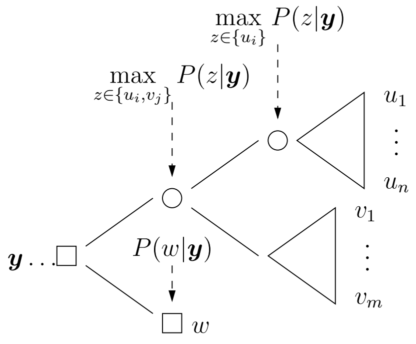

$\max_{z\in\{u_i\}} P(z|\boldsymbol{y})$
$\max_{z\in\{u_i,v_j\}} P(z|\boldsymbol{y})$
u_1
u_n
v_1
v_m
$\boldsymbol{y}\ldots$
$P(w|\boldsymbol{y})$
w

Fig. 12.6. Schematic representation of the computation of the language-model look-ahead in a hypothetical model tree.

score must be replaced by the path score $\delta_t(\ldots z)$ of the respective predecessor node for computing the optimal combined score.

In the form described above bi-gram models can be integrated into the search using this method. Already for tri-gram language models, however, an additional context symbol needs to be taken into account, which makes the formal presentation of the method considerably more complex [176].

12.3.3 Language-Model Look-Ahead

The primary task of a statistical language model within an integrated search process is to provide additional knowledge that allows to restrict the search to relevant areas as tightly as possible. Therefore, it is a disadvantage that n-gram scores are evaluated only with rather large temporal difference at the transitions between partial models or words, respectively.

The goal of the so-called *language-model look-ahead*, therefore, is to be able to incorporate language model restrictions already within a search space organized as a prefix tree [62, 220]. As the identity of the respective successor model is only known when a leaf node is reached, this can be achieved only approximately (cf. also [175, 177, 179]).

For that purpose one determines for every model state in the prefix tree all leaf nodes that can be reached from there. In order to create a hypothesis it is necessary to reach any one of these and to then take into account the respective language model score. The current path score is, therefore, at least augmented by the respective optimum of these n-gram scores. Thus in every model state the n-gram score, which is maximum for all reachable word ends, can be combined with the path score.

Figure 12.6 schematically shows a search tree, which was generated as a copy of an HMM prefix tree within a larger search space. For reasons of simplicity we assume, that the tree copies were created depending on the language model context. In the example considered let this context be given by $\boldsymbol{y}$.

With language-model look-ahead for every model state the optimal achievable n-gram score is determined depending on the current n-gram context. If from a state only a single leaf node w can be reached, this is exactly given by $P(w|\boldsymbol{y})$. However, if multiple paths to a set of leaf nodes $v_1, v_2 \ldots v_n$ exist, the optimal expected language model score is obtained as:

$$\max_{z \in \{v_1, v_2 \ldots v_n\}} P(z|\boldsymbol{y})$$

The preliminary n-gram score defined in this way, therefore, can be combined with the HMM score already *before* a leaf node is reached. Consequently, in combination with pruning techniques a focusing of the search can be achieved, as language model restrictions are taken into account as early as possible. As the set of reachable word end nodes is in general reduced by every state transition, the preliminary language model score computed for the respective predecessor state needs to be replaced by its actual value after each state transition.

12.4 Time-Synchronous Parallel Model Decoding

In all integrated search algorithms on the basis of search tree copies we assumed so far, that the scores contributed by the HMMs and the n-gram model, respectively, were combined according to equation (12.2) and that all decisions of the search method were taken on the basis of this combined score only. However, in practice it is often desirable to be able to parametrize score-based search space restrictions that follow the principle of beam search separately for HMM states and sequences of segment hypotheses, respectively. Furthermore, the use of a combined score in all areas of the search has the consequence, that the order of the n-gram models used directly influences the complexity of the search on the level of HMM states.

Therefore, an integrated search technique was developed in [70], which allows a separation of the state based search process from the level of hypotheses. It originated as a further development of the method presented in section 12.3.2, which works on the basis of time-based search tree copies. The essential difference to the latter method is, that partial path scores within all copies of the HMM search space are scored *without* the influence of the language model. The combined score is only evaluated during the search in the space of segmentation hypotheses. There principally n-gram models of arbitrary order can be used. The complexity of the respective language model does not effect the HMM search, which is realized as beam search extended to the processing of time-based tree copies.

One obtains a time-synchronous search method, which consists of two coupled search processes running in parallel. In the first by means of HMMs segmentation hypotheses are generated, which are passed on to the language model-based search at every time step. In the second search process these are combined into sequences of hypotheses and evaluated by incorporating the n-gram model.

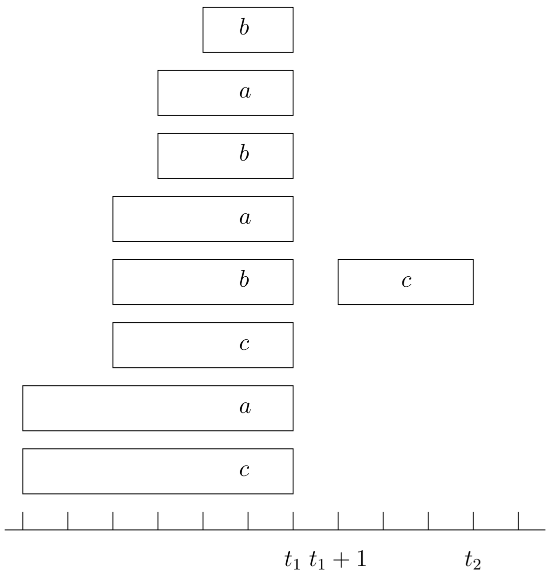


Fig. 12.7. Exemplary representation of hypotheses created by the HMM-based search with a trivial recognition lexicon $L = \{a, b, c\}$ at a given end time t_1. The hypothesis $c(t_1 + 1, t_2)$ also shown represents a solution, which later needs to be integrated into the overall search space.

12.4.1 Generation of Segment Hypotheses

The goal of the hypothesis generation is the creation of a dense lattice of segment hypotheses *without* the influence of the language model context. Starting at times $t_s = 1 \ldots T$ all promising segment hypotheses $\hat{w}_i(t_s, t_e)$ are computed. Here principally from all starting times t_s to be considered a search with all HMMs of the recognition lexicon needs to be carried out. Thus one obtains a copy of the HMM search space for every t_s. The search within these, however, is not performed in isolation but strictly time-synchronously. The results of the hypothesis generation can thus be collected for every end time t_e and be passed on to the integrated language model search, as exemplarily illustrated in figure 12.7.

In order to eliminate less promising search paths the path hypotheses within all search tree copies are subject to a joint beam pruning (see section 10.2, page 167). For increased efficiency the models of the recognition lexicon are organized as a prefix tree (see section 10.4.1, page 174). This means no limitation for the proposed search method, as the language model scores are handled completely separately from the HMM-based search. Therefore, it is absolutely sufficient, that the identities of hypotheses are determined not before a leaf node of the prefix tree is reached. The initial score for starting a new search tree copy at time t_s results as the HMM-based path score of the hypothesis chain that is optimal according to the integrated search and ends at time t_s–1. Thus it is assured, that the scores of all paths followed in

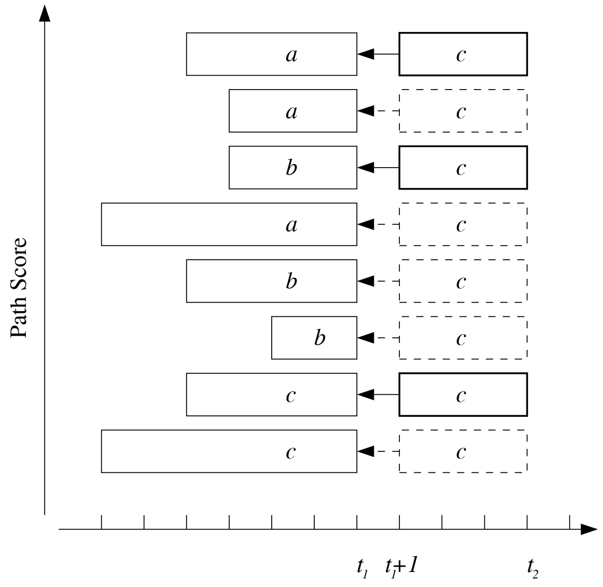


Fig. 12.8. Representation of the configuration known from figure 12.7 after the incorporation of the hypothesis $c(t_1 + 1, t_2)$ into the search space. The resulting search paths with final hypothesis c are ordered from top to bottom according to a fictitious score. With dashed lines those path extensions are drawn, which encode a language model context that is redundant for the future search.

different search tree copies always stay comparable and can be used for the mutual pruning of the search space.

12.4.2 Language Model-Based Search

The basis of the language model search is the generation of a search graph on the level of hypotheses. It is created from the individual hypotheses, which are generated time-synchronously by the HMM-based search. Per hypothesis end time t_e considered, for all $\hat{w}_i(t_s, t_e)$ all possible concatenations with predecessor hypotheses $\hat{w}_j(\ldots, t_s{-}1)$ already found in the search space are created.

The goal of the method is to determine that path within this huge search space, which is optimal with respect to the combined HMM and n-gram score. For this purpose all path hypotheses are evaluated, which were newly created on the basis of $\hat{w}_i(t_s, t_e)$. Here the evaluation of a language model score is possible with arbitrary context length, as the complete context of the preceding hypothesis sequence is known.

In order to effectively avoid an explosion of the search space, a pruning strategy is necessary, which can be applied in a strictly time-synchronous manner. Therefore, all search paths scored by the language model, which were extended up to the current end time t_e by new hypotheses $\hat{w}_i(\ldots, t_e)$ are subject to a beam pruning. Depending on the best path score determined for this set of hypotheses only paths within a constant score window are considered by the future search. All remaining solutions are discarded from the search space.

In order to ensure that all relevant n-gram contexts are also available for further path expansions, it is sufficient to save only those of the "surviving" path hypotheses, which differ in the respective n–1 final hypotheses. Figure 12.8 shows this exemplarily for the configuration of segmentation hypotheses known from figure 12.7.

In the case of a bi-gram model, therefore, per time t_e only a single path with end hypothesis $\hat{w}_i(\ldots, t_e)$ is generated for the i-th entry of the recognition lexicon. For higher order n-grams, however, path hypotheses need to be discriminated according to longer suffixes. In order to be able to manage this efficiently and without the need for costly backtracking of the respective paths, for the n–2 predecessor nodes of a path end a hash key is computed, which together with the identity of the last hypothesis compactly encodes the n-gram history. As this coding is not necessarily unique, not all different n-gram contexts can be discriminated. Informal experiments have shown, however, that this substantial simplification does not lead to measurable losses in quality of the results.

The important advantage of this integrated time-synchronous search method as opposed to comparable approaches lies in the fact, that the scoring of the language model context is *not* transferred into the HMM-based search, as, e.g., in [163]. The complexity of the n-gram models used, therefore, does not put additional burden on the already quite considerable cost of the search within the HMM state space.

The strict separation of HMM-based and language model based search besides a good organizational separability of the internal processes also leads to a better localization of measures for increasing efficiency. Thus the beam width used for the search within the HMM state space, e.g., is completely independent of the relative weighting of the two score components as well as of the method for search space pruning used on the level of hypotheses.

Part III

Systems

Introductory Remarks

In the previous chapters both the theoretical foundations of Markov-model technology were presented and the most important methods were explained, which allow their successful application in practice. However, all these methods have configuration parameters, which can only be optimized for a certain application scenario. Furthermore, not every combination of individual techniques automatically leads to an improved performance of the overall system

Therefore, in the following chapters we want to present successful systems for speech and handwriting recognitions as well as for the analysis of biological sequences. They will serve as examples for demonstrating, which methods from the wealth of available techniques are applied, how they are parametrized in the respective application, and how their combination leads to a powerful overall system. As for a certain application besides Markov model-based techniques also application specific methods are of great importance, these system presentations partially take us beyond the topic of this book. However, for application specific methods we will limit ourselves to the explanation of the principal procedures and refer the interested reader to the respective technical literature. The Markov model techniques used, however, were mostly presented in the scope of the theory and practice parts of this book. Therefore, for those we will always make references to the respective sections, in which they are further explained. Additionally, the configuration parameters used are given and their combination with other methods is explained.

In the literature the efficiency of certain pattern analysis methods is proved by presenting evaluation results. This procedure has an especially long tradition in the field of automatic speech recognition (cf. [82, 254]), where several standardized test scenarios were defined. Nevertheless, also in this area it is extremely difficult to compare systems by the evaluation results documented. Frequently the evaluation configurations considered are slightly different and the results often are hard to interpret for someone less familiar with the respective application. Furthermore, performance measurements once published are quickly outdated. Therefore, in the following we do not want to attempt to compare the systems presented by evaluation results. The interested reader can find those easily in the cited literature.

The following chapters 13 to 15 are devoted to complete systems for one of the three main application areas of Markov models, respectively: Automatic speech recognition, handwriting recognition, and the analysis of biological sequences. As the author's experience in using Markov models is based on extensive research and development of his own, also systems will be presented, which are based on ESMERALDA [63, 67] — an open-source software tool for the development of pattern analysis systems based on Markov models. It allows the flexible configuration of the models and offers concrete implementations for most of the methods relevant in practice that were presented in the second part of this book.

13 Speech Recognition

Though a number of commercial speech recognition systems are available on the market, the problem of automatic speech recognition should by no means be considered to be solved, even if the media and also some researchers sometimes make that impression. As by now a multitude of techniques is required to build a competitive speech recognition system, there are only a few systems left, which can sustain their position at the top of the international research community.

The probably best documented research systems are the ones developed by Hermann Ney and colleagues at the former Philips research lab in Aachen, Germany, and later at the RWTH Aachen University, Aachen, Germany. In the following presentations we want to put the emphasis on the newer works at RWTH Aachen University. However, many aspects of the systems within the research tradition are identical with those developed by Philips. Afterwards we want to present the speech recognizer of BBN, which in contrast to most systems developed by private companies is documented by several scientific publications. The chapter concludes with the description of a speech recognition system of our own developed on the basis of ESMERALDA.

13.1 Recognition System of RWTH Aachen University

The recognition system of RWTH Aachen University, which we will be shortly referring to as the *Aachen recognizer* in the following, is a speech recognition system for large vocabularies, which as one of only a few systems of its kind works time-synchronously. The models and algorithms used are not applied for the analysis of the complete speech signal in multiple successive phases, but are evaluated progressively in only one pass through the data on the utterance to be analyzed. The properties of the system presented in the following are described in [115, 217] and [15].

Feature Extraction

As practically all modern speech recognition systems the Aachen recognizer uses mel-frequency cepstral coefficients (MFCC) as features. During the frequency analysis of short sections of the signal — so-called frames —, which are extracted from the speech signal every 10 ms with a length of 25 ms, the frequency axis is warped approximately logarithmically according to the melody scale following the processing within the human auditory system (cf. [257, p. 112] or [97, p. 34]) Afterwards, by means of a cosine transform of the power spectrum its coarse structure is represented numerically (cf. e.g. [97, pp. 306–315]).

The system uses 15 cepstral coefficients as well as their time derivatives, which are computed by a regression analysis over a few neighboring frames. Additionally, the signal energy and its first and second time derivative are used as features.

In order to optimize the separability of the pattern classes, the initial feature representation is subject to a linear discriminant analysis (see section 9.1.2, page 148). For this purpose three successive feature vectors are stacked to form a 99-dimensional vector and subsequently are mapped onto a 35-dimensional sub-space. The pattern classes used are defined by associating HMM states with feature vectors.

As a further improvement of the feature representation the Aachen recognizer also uses a so-called vocal tract length normalization (VTLN). It consists in a scaling of the frequency axis with the goal of eliminating spectral effects, which are caused by the different lengths of the vocal tract — namely the oral and nasal cavities and the pharynx — of different persons (cf. [232]).

Acoustic Modeling

The modeling of acoustic events in the Aachen recognizer is performed on the basis of continuous HMMs (see section 5.2, page 63). As sub-word units triphones are used. Each of these models is realized as a Bakis model with three states (see section 8.1, page 127). In order to ensure the trainability of the model parameters, by means of decision trees model states are automatically grouped into clusters, for which sufficiently many training samples are available (see section 9.2.2, page 157). Additionally, in all mixture densities used for the modeling of output distributions only a single shared diagonal covariance is estimated for the underlying normal distributions (see page 162).

The recognition lexicon is compactly represented as a phonetic prefix tree, in order to be able to efficiently process also large vocabularies of up to 65 000 words (see section 10.4, page 174). A model for speech pauses as well as different models for human and non-human noises are also part of the lexicon.

Language Modeling

Refined techniques for language modeling and for its integrated use in the search substantially contribute to the success of the Aachen recognizer. In most configurations tri-gram models with so-called non-linear interpolation are used. In the terminology of this book this corresponds to the application of linear discounting for gathering probability mass and the subsequent interpolation with the respective more general distribution (see page 101 as well as section 6.5.2, page 104).

Search

In order to be able to correctly combine an n-gram language model with a tree lexicon, the Aachen recognizer uses an extension of the time-synchronous beam search, where depending on the respective word context copies of the lexicon tree are generated (see section 12.3, page 194). For tri-gram models, therefore, model states need to be distinguished according to two predecessor words. Not before a word end node is reached, the respective n-gram score can be determined and taken into account by the search. For improving this combination of HMMs and n-gram models in the Aachen recognizer additionally a so-called language-model look-ahead is used. With this technique the maximal n-gram scores to be expected are already combined with the path scores within the acoustic model tree, in order to make an early incorporation of the language model restrictions possible and thus achieve an improved efficiency of the search method (see section 12.3.3, page 197).

13.2 BBN Speech Recognizer BYBLOS

In contrast to the Aachen recognizer the BYBLOS system developed by BBN uses a multi-pass search strategy. The modeling parts used are evaluated in subsequent processing steps, which always use more exact and expensive models and thus more and more restrict the potential final solution. The details of the system presented in the following are taken from [18] and [169].

Feature Extraction

The feature extraction of BYBLOS similar to the Aachen recognizer computes 14 mel-cepstral coefficients every 10 ms for speech frames with a length of 25 ms. Together with an energy feature and the first and second time derivatives of these parameters one obtains a 45-dimensional feature vector. The modeling quality is improved by applying a gender-dependent vocal tract length normalization. Furthermore, the cepstral coefficients are normalized to zero-mean[1] and unit variance on a per-utterance basis. The energy feature, in contrast, is normalized to the respective energy maximum.

Acoustic Modeling

BYBLOS uses HMMs of different complexities for the subsequent decoding phases. The basis forms a so-called phonetically-tied mixture HMM, in which all triphone models with the same central phone use a shared codebook of 512 densities (see section 9.2.3, page 161). Furthermore, quinphone models are estimated for phones

[1] This corresponds to a quite simple variant of cepstral mean normalization, as it is also used in the ESMERALDA system. Further explanations can, therefore, be found in section 13.3, page 210.

in the context of *two* left and right neighbors, respectively. In order to ensure the trainability of these extremely specialized models, state clusters are formed automatically. For all states of one of these clusters the mixture densities are then defined on the basis of 80 normal distributions. The tri- and quinphones required are described by Bakis models with five states. BBN achieves a further specialization of the modeling by using gender-dependent HMMs. This means that both for triphones as well as for the considerably more complex quinphone models to complete inventories for male and female speakers are estimated.

Language Modeling

As most speech recognition systems BYBLOS also uses a tri-gram model for the restriction of the search space on the level of words. The robustness of the model is improved by defining a suitable category system, in order to be able to successfully combine models that were estimated for different domains.

Search

In order to make an efficient evaluation of the complex HMMs and n-gram models possible, BYBLOS uses multiple subsequent decoding phases. First, on the basis of the triphone models and by using the so-called fastmatch method [168], which only uses a bi-gram language model and a compressed phonetic prefix tree, an initial restriction of the search space is determined[2]. Subsequently the quinphone models are adapted on the basis of the current solution by applying MLLR (see section 11.2, page 184). By means of the adapted models a refinement of the space of possible solutions is computed. Afterwards another adaptation step is carried out, which also takes into account sub-word units that extend across word boundaries. The acoustic model already adapted twice is now used together with the tri-gram language model for computing a first intermediate result. This then serves as the basis for another complete pass through the decoding phases outlined above.

13.3 ESMERALDA

ESMERALDA[3] is an integrated development environment for pattern recognition systems based on Markov models. It implements[4] most methods for the estimation and application of HMMs and n-gram models presented in this book (cf. [63, 67]). The core of the ESMERALDA architecture, which is shown in figure 13.1, is defined by an incremental recognition system. In different parametrizations and in

[2] The respective efficient decoding method was patented by BBN (cf. [168]).

[3] **E**nvironment for **S**tatistical **M**odel **E**stimation and **R**ecognition on **A**rbitrary **L**inear **D**ata **A**rrays

[4] ESMERALDA is free software and distributed under the terms of the GNU Lesser General Public License (LPGL) [68].

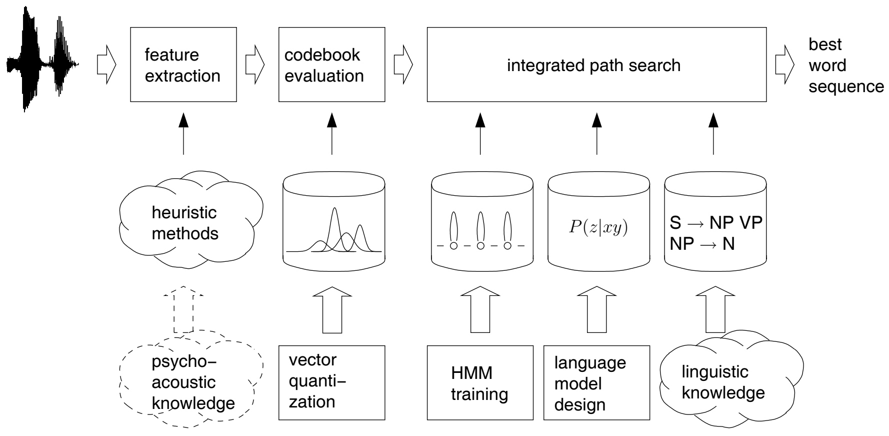


Fig. 13.1. Architecture of the ESMERALDA system: Components of the incremental recognition system are shown (feature extraction, codebook evaluation, path search) as well as the knowledge sources used (HMMs, n-gram model, grammar) and the methods provided for the design of the respective models.

a variety of scenarios it was applied for tasks of automatic speech recognition [11, 24, 64, 70, 71, 119, 120, 132, 185, 205, 218, 227, 228, 234, 235, 250, 251], for the analysis of prosodic structures [27, 28], for music segmentation [190], for automatic handwriting recognition [65, 66, 72, 73, 243, 244, 245, 246] (see also section 14.3), and for the analysis of biological sequences [186, 187, 188, 189] (see also section 15.3).

In the following we want to present the important properties of the speech recognition systems developed on the basis of ESMERALDA. For reasons of simplicity we will always be talking about *the* ESMERALDA recognizer, even though there exist many different system configurations and also other application areas for the techniques used.

The ESMERALDA recognizer similarly to the Aachen recognizer works strictly time-synchronously and also was conceptually influenced by the research at Philips in its development. As important differences to related systems it offers the possibilities to use a purely declarative grammar as a language model and to incrementally generate recognition hypotheses already during the processing of spoken utterances that are not yet completely available. The basic system features put together in the following are documented in [63, 70, 71, 227].

Feature Extraction

The ESMERALDA recognizer, similarly to the systems presented in the previous sections, computes mel-cepstral coefficients every 10 ms on frames with a length of 16 ms. In order to eliminate adverse effects, which are caused by changes in the signal recording conditions, as, e.g., by the use of a different microphone, a dynamic cepstral mean subtraction is applied (cf. e.g. [199, 239] or [97, pp. 522–525]).

For that purpose a moving cepstral average, which is computed on energetic parts of the utterance, is subtracted from the current cepstrum. Thus changed properties of the recording channel can be compensated well. To a quite limited extent also a form of speaker adaptation is achieved. In addition to the cepstrum an energy feature is computed. By estimating a histogram of the signal energies in a certain temporal context it is possible, to map the energy coefficients to a an interval between 0.0 and 1.0 with respect to the current local minimum or maximum, respectively. In the same way as the cepstral mean subtraction this normalization operation can be applied in a strictly causal manner, i.e. without any knowledge about "future" parts of the signal.

The 39-dimensional feature vector is formed from 12 cepstral coefficients, the normalized signal energy, as well as the first and second order time derivatives of these parameters. The smoothed temporal derivatives are computed by means of regression polynomials on a window of five neighboring frames.

Acoustic Modeling

The acoustic modeling in the ESMERALDA recognizer is performed by means of semi-continuous HMMs (see section 5.2, page 63) with codebook sizes of some hundred up to a few thousand densities with diagonal covariance matrices[5].

As sub-word units triphones with linear model topology are used for all speech recognition applications. The number of states of the individual models varies and is determined by the shortest segment length found for a corresponding speech unit in the sample set. In order to ensure the trainability of the triphone models, state clusters are created automatically, which are supported by a certain minimal number of feature vectors in the training set[6]. Only those states are clustered, which occur in triphones for the same baseline phone and at the same model-internal position. The distance between state clusters is determined by the increase in entropy caused by merging the output distributions (see section 9.2.2, page 157).

The recognition lexicon is organized as a phonetic prefix tree. Besides the mandatory silence model it also contains models for human and non-human noises as well as for hesitations.

Statistical and Declarative Language Modeling

The ESMERALDA recognizer uses statistical bi-gram models during the search by default. However, also the time synchronous application of higher-order n-gram models is possible. As the most important reduction in perplexity for word-based

[5] ESMERALDA also supports the use of full covariance matrices. However, the large amount of training samples required to estimate such models and the higher costs in decoding cause the simpler diagonal models to become the more favorable alternative. The reduced capabilities for describing correlations between feature vector components are usually outweighed by the additional precision in modeling achieved when using larger numbers of densities.

[6] Good results are achieved for a threshold of 75 samples. More compact and eventually also more robust models are obtained by requiring a few hundred feature vectors to support a state cluster.

language models usually is achieved when moving from bi- to tri-gram models, the dramatically increased computational cost for even more complex models can hardly be justified in practice[7]. Even though ESMERALDA provides all techniques for the robust estimation of word-based n-gram models presented in chapter 6, in practice absolute discounting in combination with backing off smoothing is used (see page 101 and 106).

As an important extension compared to other recognition systems the ESMERALDA recognizer offers the possibility to use a declarative grammar in much the same way as a statistical language model to guide the recognition process. In this way expert knowledge about utterances from a certain domain can be exploited without the need to gather large amounts of text material for the training of an n-gram model. An LR(1) parser (cf. e.g. [3, pp. 215–247]) derives pseudo-scores for word sequences from the applications of rules defined by the grammar. Thus the restrictions of the grammar are not enforced, so that also ungrammatical utterances can be recognized and the use of a grammar is also possible for domains, where no full coverage can be achieved. The application of a grammar can always be combined with a statistical language model, which is available additionally.

In integrated systems for automatic speech *understanding* it is further possible, to output the grammatical structures determined during the recognition process directly as structured recognition hypotheses. Thus the interface between speech recognition and understanding is raised from the level of simple word hypotheses to the one of domain-specific constituents (cf. [24, 184]).

Incremental Search

The ESMERALDA recognizer uses the strictly time-synchronous method for searching combined HMM and n-gram models that was described in detail in section 12.4, page 198. There search space copies of the acoustic model tree are created depending on the respective starting time. The search on word hypothesis level is strictly separated from that, which principally makes it possible to take into account arbitrarily long n-gram contexts.

In order to be able to generate results in interactive applications already while the current utterance is still being processed, the ESMERALDA recognizer uses a method for the incremental generation of recognition hypotheses. It assumes, that hypotheses for a certain part of the signal become more and more stable, the farther the time-synchronous decoding has proceeded beyond this point. Therefore, every 50 ms the currently optimal solution is tracked back a certain time interval into "history" and the prefix of the hypothesis sequence lying *before* that is output as a partial recognition hypothesis. As the actual optimal solution can not be known before the end of the utterance is reached, such a method necessarily will produce sub-optimal results. With a time delay of two seconds the incremental generation achieves results,

[7] In recognition experiments without the use of a lexicon only on the basis of phone sequences, however, substantial improvements in the accuracy of the results can be achieved by using higher-order n-gram models [70].

which are almost identical with those obtained by a completely optimizing method. With 750 ms delay one obtains an acceptable reaction time to spoken language input and only a rather low increase in the error rate by approximately 5% relative.

14 Character and Handwriting Recognition

In contrast to the field of automatic speech recognition, where Markov model-based methods currently represent the state-of-the-art, HMMs and n-gram models are still a rather new approach for the recognition of machine-printed or handwritten texts. This might be due to the fact, that data obtained from writing or print can in general be segmented into manageable segments, as, e.g., words or characters, much more easily than spoken language. Therefore, especially in the field of OCR, but also for the processing of forms or for the reading of address fields, a number of well established methods exists, that rely on the classic distinction between segmentation and classification. Segmentation-free methods on the basis of Markov models, however, are mainly applied by researchers, which previously gathered experiences with this technology in the field of automatic speech recognition.

In the following we will first present the OCR system of BBN, which explicitly uses the BYBLOS system originally developed for the purpose of automatic speech recognition. It shows the principal approach for off-line processing, even though the recognition of machine-printed characters is obviously easier than the recognition of handwriting. Afterwards, we will present the online handwriting recognition system developed by Rigoll and colleagues at the University of Duisburg, Germany, which applies a number of well-know Markov model-based techniques. The chapter concludes with a presentation of our own system for offline handwriting recognition, which just like the speech recognition systems presented in section 13.3 was developed using the ESMERALDA tool-kit.

14.1 OCR System by BBN

The OCR system developed at BBN is a Markov model-based recognition system for machine-printed text, which is capable of handling very large vocabularies and can also work with “unlimited” vocabulary. As already mentioned above it is based on the speech recognition system BYBLOS, which was presented in section 13.2, page 209. In order to be able to apply this system to OCR problems, some modifications are required, which are documented in [13] and which will be presented briefly in the following.

Feature Extraction

The basis of offline character recognition is the optical capturing of a complete document page with a resolution of 300 to 600 dpi[1]. In order to convert such a document image into a chronologically organized signal, first the individual text lines are segmented. After a compensation of the rotation of the page image it is ensured, that lines are oriented horizontally. They then can be identified by simply searching the minima in a horizontal projection histogram of the gray level image. Then every text line is subdivided into a sequence of small vertical stripes, which overlap each other by two thirds. The width of the stripes is $1/15$ of the line height, which was normalized in order to compensate for different font sizes.

For feature extraction the individual text stripes are again subdivided into 20 overlapping cells, which correspond to small rectangular image regions. On the basis of these cells now feature vectors are generated. Per cell first the average intensity and its horizontal and vertical derivative are computed. In a quadratic window of four cells furthermore the local derivative and the local correlation of the Gray values is determined. For every text line one thus obtains a sequence of 80-dimensional feature vectors, which are computed per stripe in the direction of the text.

This can be viewed as sliding a small analysis window, which has the width of one text stripe, along the text line image — a method, which is referred to as the *sliding window approach*. The technique was first applied to the problem of offline character recognition by researchers at BBN [214]. It is fundamental for transforming text image data into a chronological sequence of feature vectors, which can later be fed into an HMM-based analysis system. Today the sliding window approach can be found in the majority of HMM-based offline recognizers.

Modeling Machine Print

The statistical modeling of machine-printed text is performed on the basis of context independent character models. Together with the models for punctuation symbols and white space, BBN uses 90 elementary models for English texts and 89 for Arabic. For each of these 14 states are defined, which are connected according to the Bakis topology. Therefore, both a linear passing through the state sequence and a skipping of individual states is possible (see section 8.1, page 127).

The output probability densities of the particular model states are defined on the basis of a shared set of component densities similar to semi-continuous HMMs (see section 5.2, page 63). However, this is performed separately for subsets of 10 features from the total 80-dimensional feature vector. All partial output probability densities use an inventory of 64 Gaussians. The total density for a feature vector is obtained by multiplying all eight partial density values.

[1] abbreviation for *dots per inch*

The training of the models is achieved by means of the Baum-Welch algorithm (see page 80), where in the same way as in automatic speech recognition only the orthographic transcription of every text line — i.e. the actual character sequence — is given. At the beginning of the training all HMM parameters are initialized uniformly. Which initial parameters are used for the eight mixture codebooks, unfortunately, is not documented in the literature. However, it may be assumed, that by means of a method for vector quantizer design (see section 4.3, page 50) initial codebooks are generated in an unsupervised manner.

Language Modeling and Search

Just in the same way as in automatic speech recognition for OCR a certain lexicon can be given. The restriction of potential word sequences can then be achieved in the usual way by a word-based n-gram model. BBN achieved with this method and a lexicon of 30 000 words an error rate on the level of characters of less then 1% for English texts and of a little more than 2% for Arabic documents.

However, if no constraints are to be imposed on the expected words, the recognition needs to be performed on the basis of characters alone. Then it is said that an "unlimited" lexicon is used. By applying an n-gram model on the level of characters the missing restrictions of a fixed recognition lexicon can be compensated to some extent. The error rates that can be achieved, however, increase by a factor of 2 to 3 if merely a character tri-gram is used.

The combination of HMMs for individual characters and an n-gram model for the restriction of the search space is achieved by a multi-pass search strategy in the BYBLOS system (see page 210).

14.2 Online Handwriting Recognition System of the University of Duisburg

The online handwriting recognition system developed at the University of Duisburg, Germany, in the research group of Gerhard Rigoll[2] is a writer *de*pendent system for the recognition of isolated words. The writing style, however, is not restricted. As one of the very few systems in the field of handwriting or optical character recognition it uses context dependent sub-word units, in order to achieve a modeling, which is as precise as possible even for large vocabularies. The system's details put together in the following are taken from [126, 198] and [124].

Feature Extraction

In the Duisburg system the pen trajectories are captured by using a graphics tablet by WACOM. Such devices typically provide measurements with a sampling rate of

[2] Meanwhile Gerhard Rigoll heads the Institute for Human-Machine Communication at the Technical University of Munich, Germany.

approximately 200 Hz. Besides the pen position, which is determined with a precision of 2540 lpi[3] and for an elevation of the pen of up to 5 mm over the tablet, the data also comprises the pen pressure in 256 discretized levels. In order to compensate for variations in writing speed, which are highly person specific, the raw data is first re-sampled. In that process the new samples are usually placed equidistantly along the pen trajectory. However, in the Duisburg system depending on the local trajectory parameters the distance between the samples is optimized such, that even for fast changes in the writing direction a sufficiently accurate resolution is ensured.

On the preprocessed trajectory data then four types of features are calculated: The orientation α of the vector connecting two successive pen positions represented as $\sin\alpha$ and $\cos\alpha$, the difference of successive orientations $\sin\Delta\alpha$ and $\cos\Delta\alpha$, the pen pressure, and a local representation of the region surrounding the current pen position in the form of a gray level image. For this so-called *bitmap* feature first the pen trajectory is locally represented as a 30×30 binary image and then sub-sampled in a raster of 3×3 pixels. The nine gray values defined such are then used as additional features.

Modeling Handwriting

In the same way as in the system by BBN context independent models of characters form the basis of the statistical modeling of handwriting in the Duisburg system. For the German language 80 elementary models are used, which are defined as linear HMMs with 12 states for characters and 4 for punctuation symbols.

In order to improve the precision of the representation, additionally so-called trigraph models are used, which are the character-based equivalent to triphones, i.e. character models in the context of the respective left and right neighboring symbols. As with a potential inventory of 80^3 trigraphs these models can not be trained robustly, on the one hand only those trigraphs are represented that sufficiently often occur in the training data. On the other hand robust generalizations of parameters for unseen states are generated by automatically computing state clusters by means of decision trees (see section 9.2.2, page 157).

For the modeling of emissions discrete and continuous HMMs as well as a hybrid approach, which incorporates neuronal networks, are investigated. The best results for a recognition lexicon of 200 000 words are achieved by the hybrid system, which combines discrete HMMs with a vector quantization stage on the basis of neuronal networks. There the parameters of both parts of the model can be optimized jointly by a method for discriminative training.

Search

As the Duisburg online recognition system is not used for the processing of word sequences, no n-gram language model is applied in the decoding stage. In order

[3] abbreviation for *lines per inch*

to make the search in the extremely large recognition lexica of up to 200 000 entries efficient, the necessary sub-word units are represented as a prefix tree (see section 10.4.1, page 174).

14.3 ESMERALDA Offline Recognition System

In contrast to most approaches for offline handwriting recognition, which consider isolated phrases, e.g., in postal addresses, or only isolated words written in certain fields of a form, it is the goal of the system described in the following, to writer independently recognize complete handwritten texts. The principal approach is comparable with the OCR system by BBN. A detailed description of the methods used can be found in [246].

Preprocessing

As opposed to machine-printed texts a suitable preprocessing of optically captured handwriting data is of fundamental importance. Similar to the system by BBN, after a position normalization of the document page first a segmentation of the individual text lines is performed by evaluating the horizontal projection histogram. However, in handwritten texts the baseline of the writing in general is not strictly horizontal — which is referred to as the skew of the line or the so-called baseline drift — and individual characters or words are usually not written completely upright but with some varying inclination with respect to the vertical — the so-called slant. Therefore, it is tried to compensate these variabilities before the actual feature extraction by normalization operations. After a global correction of the line orientation the skew is corrected locally together with the slant of the writing, so that also variations within a line can be captured approximately. As the ESMERALDA offline recognizer is primarily intended for the recognition of texts in video data, then a local binarization of the text line image is performed. Thus it is ensured, that intensity variations of both writing and background do not adversely affect the subsequent feature extraction process[4]. As a final preprocessing step, the text line image is normalized in size. For this purpose first local extrema of the contour of the writing are determined. Then the line image is re-scaled such, that the average distance between these matches a predefined constant.

Feature Extraction

Just in the same way as in the OCR system by BBN the ESMERALDA offline recognizer uses the sliding window technique to convert text line images into sequences of feature vectors. Pre-segmented and normalized text line images are subdivided into

[4] Informal experiments showed, that the purely gray level based features of the BBN system could not be applied successfully to the processing of handwriting data. For the purpose of OCR, however, those features immediately achieved a convincing system performance.

small stripes or analysis windows, which are four pixels wide and overlap each other by half. For each of these windows nine geometrical features are computed from the associated stripe of the binarized text image. The first group of features describes the coarse shape of the writing within the local analysis window. The average distance of the lower baseline to both the upper and the lower contour of the writing is computed, and the distance of the center of gravity of the text pixels to the baseline. These features are then normalized by the core size, i.e. the distance between upper and lower baseline, in order to increase the robustness against variations in the size of the writing. Furthermore, three local directional features are calculated describing the orientation of the lower and upper contour as well as the gradient of the mean of the column-wise pixel distributions. Finally, the average number of black-to-white transitions per column, the average number of text pixels per column, and the average number of text pixels between upper and lower contour are calculated.

In order to be able to consider a wider temporal context to some extent in the feature representation , the 9-dimensional baseline feature set is complemented by a discrete approximation of the temporal derivatives of the features, which is computed over a context of five analysis windows by linear regression.

Handwriting Model

The statistical modeling of handwriting is performed on the basis of semi-continuous HMMs (see section 5.2, page 63) with a shared codebook of approximately $2,000$ Gaussians with diagonal covariances. A total of 75 context independent HMMs are created for modeling 52 letters, ten digits, twelve punctuation symbols, and white space. The number of model states is automatically determined depending on the length of the respective unit in the training material. All these models use the Bakis topology in order to be able to capture a wider variability in the length of the character patterns described. (see section 8.1, page 127).

Language Modeling and Search

In order to make handwriting recognition with "unlimited" lexicon possible only on the basis of character models, sequencing restrictions between HMMs for individual symbols are represented by means of a character n-gram models of increasing complexity[5]. For all models estimated the raw n-gram probabilities were smoothed by applying absolute discounting and backing off (see pages 101 and 106). The integrated decoding of HMMs for describing the handwriting and n-gram language models defining the sequencing restrictions is achieved in the ESMERALDA offline recognizer by applying the time-synchronous search method described in detail in section 12.4 on page 198.

[5] In [246] lexicon free experiments are reported for bi-gram up to 5-gram models.

15

Analysis of Biological Sequences

When considering applications of HMMs in the field of bioinformatics one gets the impression, that there people strive for influencing as many details of the models as possible by expert knowledge, i.e. specifying them manually in a heuristic way. Despite the almost finished sequencing of the human genome this might currently be the only promising method for a discipline, which still suffers chronically from the lack of data.

Here parallels can be drawn to the times when statistical speech recognition was in its infancy. Then also substantial influence was exerted on the model structure. With the availability of ever larger sample sets, that were annotated or could be annotated semi-automatically, such techniques, however, were applied to a ever lesser extent and replaced by methods, which allowed to estimate the necessary configurations from example data automatically. Usually also the former expensive model structures, which needed to be optimized manually, gave way to drastically simplified structures with considerably increased degrees of freedom.

Furthermore, in the field of biological sequence analysis there still seems to be a need for justifying the application of statistical methods on the basis of Markov models. In several publications on the subject, as, e.g., in [55], sections can be found, in which it is explicitly pointed out, that HMMs principally achieve the same results as older methods for score-based sequence comparison. Usually only the underlying mathematical theory is given as their main advantage and less frequently the automatic trainability of the model parameters required.

In the following we will first briefly present the two most important software tools, which were developed for the analysis of biological sequences by means of hidden Markov models. In these systems Markov chain models play just as little a role as the detection of genes within a complete genome. In the final section of the chapter we will present a system for the classification of proteins, which was developed on the basis of ESMERALDA and which follows a radically different approach in many respects.

15.1 HMMER

The *HMMER* system (pronounced *hammer*) was developed by Eddy and colleagues, first at Washington University, Saint Luis, and meanwhile at the Janelia Farm Research Campus of the Howard Hughes Medical Institute, Chevy Chase, [56]. The underlying techniques are described in detail in the monograph [52].

Model Structure

As model structure HMMER exclusively uses profile HMMs (see section 8.4, page 133). Already in version 2.2 (current version is 2.3.2) their structure was extended such, that besides the detection of similar sequences also the search for similar partial sequences was possible, which are embedded in longer sequences of amino acids. The model structure denoted as "Plan 7" results from the one shown in figure 8.6 on page 135 by reducing the potential state transitions within a group of match, insert, and delete states from 9 to 7[1]. Then no transitions from delete to insert states and vice versa are possible.

Parameter Estimation

The estimation of model parameters in HMMER is exclusively performed on the basis of already available multiple alignments that are created by experts. As the mapping between the columns of the alignment and model states is fixed and is not modified, the empirical distributions of the amino acids can directly be given. In order to improve the robustness of the estimates, prior probabilities of the parameter values are incorporated following the principle of MAP estimation (see section 3.6.2, page 43). For the necessary densities mixtures of Dirichlet distributions are used as statistical models (cf. [52, pp. 116–117]). However, an iterative optimization of the model created such by means of the Baum-Welch algorithm or a similar method is not performed.

When searching for sequences the decision about the similarity to the sequence family considered is made on the basis of the ratio between the score of the model and an appropriately chosen background model (see section 7.4, page 124). Therefore, HMMER offers the possibility to automatically adjust the parameters of the background distribution and the necessary thresholds in a calibration step.

Interoperability

An essential aspect of HMMER is the interoperability with other systems. Ready-made profile HMMs, which are available in a certain data format, as, e.g., the one

[1] The name "Plan 7" for the structure improved with respect to the older "Plan 9" model architecture is an allusion to the title of a science fiction movie — sometimes referred to as the worst movie ever made. In the documentation of HMMER frequently similar humorous cross references can be found.

defined in the PFAM database [10], can be used directly for sequence analysis by the software tools. Furthermore, for arbitrary models HMMER is capable of applying the search algorithms provided to different sequence databases available on the Internet, as e.g. the protein database SWISS-PROT [9].

15.2 SAM

The *Sequence Alignment and Modeling System* (SAM) resulted from works of Hughey and colleagues at the University of California, Santa Cruz, CA [98]. From the principal algorithmic capabilities it is very similar to HMMER. As model structure profile HMMs[2] are used though without the modifications of the "Plan 7" architecture. The parameter estimation is performed on pre-existing multiple alignments by default. In order to enable a purely score-based decision about the similarity of sequences, also a calibration of models is provided. In the same way as HMMER, SAM offers the possibility to import ready-made models from external data formats and to directly apply models for the search on sequence databases.

The important difference between HMMER and SAM lies in the fact, that the latter system offers different iteratively optimizing training methods for profile HMMs. By default the Baum-Welch algorithm is used (see page 80), which in the SAM documentation, however, is only referred to as EM algorithm. In order to modify the convergence properties and to avoid finding local maxima only during the optimization procedure, the parameter estimation can be overlaid by a noise process following the principle of simulated annealing. There the parameter estimates are randomly modified according to a statistical model, while the degree of the modification decreases with increasing number of training iterations. For a substantially accelerated but also qualitatively less satisfactory parameter training SAM offers the application of the Viterbi training (see page 86). Furthermore, in SAM there exist various heuristic methods for optimizing the structure of the model to be estimated during the training process depending on the current model parameters. However, all these methods are highly specific to the problem of modeling protein sequences by profile HMMs.

In contrast to HMMER, SAM furthermore offers the possibility to generate multiple alignments from initially un-aligned sequences in a purely data-driven manner. For HMMER multiple alignments need to be pre-specified and can only be augmented by additional members of the respective sequence family, which were found during the similarity search in certain databases.

Despite these distinguishing features one may say, that from the purely discrete view on sequence data both systems implement all relevant aspects of research on the topic of profile HMMs. In contrast to the fields of speech and handwriting recognition both HMMER as well as SAM are available on the Internet as software tools for several system architectures and with detailed documentation.

[2] The profile HMM structure with match, insert, and delete states is incorrectly referred to as a "linear HMM" in the SAM documentation.

15.3 ESMERALDA

Though it might seem obvious that sequence data, which can be described by a series of symbols taken from a small finite alphabet, is modeled using discrete HMMs, alternative data representations can lead to considerably improved modeling capabilities. The ESMERALDA-based system for modeling families of remotely homologous proteins, which was developed by Thomas Plötz and the author at Bielefeld University, Germany, follows such a radically different approach. As the first system of its kind it used a biologically motivated *continuous* multi-channel representation of protein sequence data in conjunction with semi-continuous HMMs for describing families of remotely homologous sequences [186, 187, 188, 189].

Feature Extraction

The function of the residues within a protein sequence — i.e. the individual amino acids — can be abstractly characterized on a symbolic level, but it is actually realized by their bio-chemical properties. Therefore, the symbolic sequence representation of proteins is first mapped onto a suitable continuous representation of relevant bio-chemical properties of amino acids. As it is not clear, which single property might be the most relevant, multiple properties complementing each other — here 35 — are considered, which are defined by so-called amino acid indices. Therefore, protein sequences are now represented as multi-channel real-valued signals, where each channel encodes one bio-chemical property of amino acids.

This signal representation is then analyzed locally by sliding along a small analysis window with a length of 16 residues and an overlap of 15. Thus for every position in the original symbolic sequence representation a local analysis is performed. For this purpose first a channel-wise discrete wavelet transform is applied using two-stage multi-scale analysis. Secondly, after a first reduction of the dimension per channel to 11 the collection of coefficients obtained such is subject to a principal component analysis obtaining a 99-dimensional feature vector (cf. [186, 188]).

Statistical Models of Proteins

The statistical models of proteins are based on semi-continuous HMMs (see section 5.2, page 63). When structured equivalently to profile HMMs the difference to the classical modeling of proteins is completely concentrated in the modified data representation. However, as continuous models based on mixture densities in conjunction with a feature representation encoding local context information are much more powerful than discrete models, the tight structural restrictions of profile HMMs, which encode the prior expert knowledge, no longer are of fundamental importance. Rather, the actual structures can be inferred from training data within the framework of a greatly simplified model topology. Therefore, so-called bounded left-right models were introduced [189]. These are principally equivalent to ordinary left-to-right models with the notable distinction, that the distance for skipping states forward in the model is limited to a maximum number of states. As arbitrarily long forward skips are hardly necessary to represent sequential data, this measure ensures, that the number of transition probabilities reduced thus can actually be trained on the sample data available.

As the problem of data sparseness is especially extreme in the field of bioinformatics, it is of fundamental importance to define robust procedures for parameter training. Usually this is achieved by incorporating certain prior knowledge into the estimation process or by heuristic methods for model regularization, i.e. suitable modifications of the parameter estimates. In the ESMERALDA-based protein modeling framework a training procedure was defined, which combines several estimation steps producing more and more specialized models. As semi-continuous HMMs are used for the initial estimation of the shared set of codebook densities annotated data is not required. Therefore, this can be achieved on large databases of general protein sequences. The remaining model parameters, namely transition probabilities and mixture density weights, can then be estimated on family-specific data. As this data usually is too limited to achieve a robust re-estimation of the codebook densities, too, these are merely adapted to the special data by means of maximum likelihood linear regression (see section 11.2, page 184).

References

[1] Adda G, Adda-Decker M, Gauvain JL, Lamel L (1997) Text normalization and speech recognition in French. In: Proc. European Conf. on Speech Communication and Technology, vol 5, pp 2711–2714

[2] Adda-Decker M (2001) Towards multilingual interoperability in automatic speech recognition. Speech Communication 35(1–2):5–20

[3] Aho AV, Sethi R, Ullman JD (1986) Compilers: Principles, Techniques, and Tools. Addison-Wesley, Reading, Massachusetts

[4] Asadi A, Schwartz R, Makhoul J (1990) Automatic detection of new words in a large vocabulary continuous speech recognition system. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Albuquerque, pp 125–128

[5] Aubert X, Haeb-Umbach R, Ney H (1993) Continuous mixture densities and linear discriminant analysis for improved context-dependent acoustic models. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Minneapolis, vol II, pp 648–651

[6] Aubert X, Dugast C, Ney H, Steinbiss V (1994) Large vocabulary continuous speech recognition of Wall Street Journal data. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Adelaide, vol II, pp 129–132

[7] Avendaño C, Deng L, Hermansky H, Gold B (2004) The analysis and representation of speech. In: Greenberg S, Ainsworth WA, Popper AN, Fay RR (eds) BLA, Springer Handbook of Auditory Research, Springer, Berlin Heidelberg, pp 63–100

[8] Bahl LR, Brown PF, de Souza PV, Mercer RL (1993) Estimating Hidden Markov Model parameters so as to maximize speech recognition accuracy. IEEE Trans on Speech and Audio Processing 1(1):77–83

[9] Bairoch A, Apweiler R (2000) The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. Nucleic Acids Research 28(1):45–48

[10] Bateman A, Birney E, Durbin R, Eddy SR, Howe KL, Sonnhammer ELL (2000) The Pfam protein families database. Nucleic Acids Research 28(1):263–266

[11] Bauckhage C, Fink GA, Fritsch J, Kummert F, Lömker F, Sagerer G, Wachsmuth S (2001) An integrated system for cooperative man-machine in-

teraction. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation, Banff, Canada, pp 328–333

[12] Baum L, Petrie T, Soules G, Weiss N (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. Ann Math Statist 41:164–171

[13] Bazzi I, Schwartz R, Makhoul J (1999) An omnifont open-vocabulary OCR system for English and Arabic. IEEE Trans on Pattern Analysis and Machine Intelligence 21(6):495–504

[14] Bell TC, Cleary JG, Witten IHW (1990) Text Compression. Prentice Hall, Englewood Cliffs, NJ

[15] Beyerlein P, Aubert XL, Haeb-Umbach R, Harris M, Klakow D, Wendemuth A, Molau S, Pitz M, Sixtus A (1999) The Philips/RWTH system for transcription of broadcast news. In: Proc. European Conf. on Speech Communication and Technology, Budapest, vol 2, pp 647–650

[16] Bigi B, De Mori R, El-Béze M, Spriet T (1997) Combined models for topic spotting and topic-dependent language modeling. In: Furui S, Huang BH, Chu W (eds) Proc. Workshop on Automatic Speech Recognition and Understanding, IEEE Signal Processing Society, pp 535–542

[17] Billa J, Ma K, McDonough JW, Zavaliagkos G, Miller DR, Ross KN, El-Jaroudi A (1997) Multilingual speech recognition: The 1996 Byblos Callhome system. In: Proc. European Conf. on Speech Communication and Technology, Rhodes, Greece, vol 1, pp 363–366

[18] Billa J, Colhurst T, El-Jaroudi A, Iyer R, Ma K, Matsoukas S, Quillen C, Richardson F, Siu M, Zvaliagkos G, Gish H (1999) Recent experiments in large vocabulary conversational speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[19] Bocchieri E (1993) Vector quantization for efficient computation of continuous density likelihoods. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Minneapolis, vol 2, pp 692–695

[20] Bogert BP, Healy MJR, Tukey JW (1963) The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In: Rosenblatt M (ed) Proceedings of the Symposium on Time Series Analysis (1962, Providence, Rhode Island), Wiley, New York, pp 209–243

[21] Boulis C, Digalakis VV (2000) Fast speaker adaptation of large vocabulary continuous density HMM speech recognizer using a basis transform approach. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul, pp 3630–3633

[22] Boulis C, Diakoloukas VD, Digalakis VV (2001) Maximum likelihood stochastic transformation adaptation for medium and small data sets. Computer Speech & Language 15:257–285

[23] Brand M, Oliver N, Pentland A (1997) Coupled Hidden Markov Models for complex action recognition. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, pp 994–999

[24] Brandt-Pook H, Fink GA, Wachsmuth S, Sagerer G (1999) Integrated recognition and interpretation of speech for a construction task domain. In: Bullinger

HJ, Ziegler J (eds) Proceedings 8th International Conference on Human-Computer Interaction, München, vol 1, pp 550–554

[25] Bregler C (1997) Learning and recognizing human dynamics in video sequences. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition, San Juan, Puerto Rico, pp 568–57

[26] Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and Regression Trees. The Wadsworth Statistics/Probability Series, Wadsworth Publishing Company, Belmont, California

[27] Brindöpke C, Fink GA, Kummert F, Sagerer G (1998) An HMM-based recognition system for perceptive relevant pitch movements of spontaneous German speech. In: International Conference on Spoken Language Processing, Sydney, vol 7, pp 2895–2898

[28] Brindöpke C, Fink GA, Kummert F (1999) A comparative study of HMM-based approaches for the automatic recognition of perceptively relevant aspects of spontaneous German speech melody. In: Proc. European Conf. on Speech Communication and Technology, Budapest, vol 2, pp 699–702

[29] Brown PF, Pietra VJD, deSouza PV, Lai JC, Mercer RL (1992) Class-based n-gram models of natural language. Computational Linguistics 18(4):467–479

[30] Bunke H, Wang PSP (eds) (1997) Handbook of Character Recognition and Document Image Analysis. World Scientific Publishing Company, Singapore

[31] Burshtein D (1996) Robust parametric modelling of durations in Hidden Markov Models. IEEE Trans on Acoustics, Speech, and Signal Processing 3(4):240–242

[32] Çarkı K, Geutner P, Schultz T (2000) Turkish LVCSR: Towards better speech recognition for agglutinative languages. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul

[33] Chen SF, Goodman J (1998) An empirical study of smoothing techniques for language modeling. Tech. Rep. TR-10-98, Center for Research in Computing Technology, Harvard University, Cambridge, MA

[34] Chen SF, Goodman J (1999) An empirical study of smoothing techniques for language modeling. Computer Speech & Language 13:359–394

[35] Chow YL (1990) Maximum Mutual Information estimation of HMM parameters for continuous speech recognition using the N-best algorithm. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, pp 701–704

[36] Chung KL, AitSahlia F (2003) Elementary Probability Theory, 4th edn. Springer, Berlin

[37] Clark J, Yallop C (1990) An Introduction to Phonetics and Phonology. Blackwell, Oxford

[38] Clarkson PR, Robinson AJ (1997) Language model adaptation using mixtures and an exponentially decaying cache. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, München, vol 2, pp 799–802

[39] D'Amato DP, Kuebert EJ, Lawson A (2000) Results from a performance evaluation of handwritten address recognition systems for the united states postal service. In: Proc. 7th Int. Workshop on Frontiers in Handwriting Recognition, Amsterdam, pp 249–260

[40] Daniels PT, Bright W (eds) (1996) The World's Writing Systems. Oxford University Press, New York

[41] Davenport J, Nguyen L, Matsoukas S, Schwartz R, Makhoul J (1999) The 1998 BBN BYBLOS 10x real time system. In: Proc. DARPA Broadcast News Workshop, Herndon, Virginia

[42] Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B 39(1):1–22

[43] Deng L (1991) The semi-relaxed algorithm for estimating parameters of hidden Markov models. Computer Speech & Language 5(3):231–236

[44] Dengel A, Hoch R, Hönes F, Jäger T, Malburg M, Weigel A (1997) Techniques for improving OCR results. In: Bunke H, Wang PSP (eds) Handbook of Character Recognition and Document Image Analysis, World Scientific Publishing Company, Singapore, pp 227–258

[45] Devijver PA, Kittler J (1982) Pattern Recognition. A Statistical Approach. Prentice Hall, London

[46] Diakoloukas VD, Digalakis VV (1997) Adaptation of Hidden Markov Models using multiple stochastic transformations. In: Proc. European Conf. on Speech Communication and Technology, pp 2063–2066

[47] Diakoloukas VD, Digalakis VV (1999) Maximum-likelihood stochastic-transformation adaptation of Hidden Markov-Models. IEEE Trans on Speech and Audio Processing 7(2):177–187

[48] Digalakis VV (1999) Online adaptation of hidden Markov models using incremental estimation algorithms. IEEE Trans on Speech and Audio Processing 7(3):253–261

[49] Duda RO, Hart PE (1973) Pattern Classification and Scene Analysis. Wiley, New York

[50] Duda RO, Hart PE, Stork DG (2001) Pattern Classification, 2nd edn. Wiley, New York

[51] Dugast C, Beyerlein P, Haeb-Umbach R (1995) Application of clustering techniques to mixture density modelling for continuous-speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Detroit, MI, vol 1, pp 524–527

[52] Durbin R, Eddy SR, Krogh A, Mitchison G (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press

[53] Eckert W, Gallwitz F, Niemann H (1996) Combining stochastic and linguistic language models for recognition of spontaneous speech. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Atlanta, vol 1, pp 423–426

[54] Eddy SR (1995) Multiple alignment using Hidden Markov Models. In: Proc. Int. Conf. on Intelligent Systems for Molecular Biology, pp 114–120

[55] Eddy SR (1998) Profile Hidden Markov Models. Bioinformatics 14(9):755–763

[56] Eddy SR (2007) HMMER: Biosequence analysis using profile hidden Markov models. URL `http://hmmer.janelia.org/`

[57] Efron B, Gong G (1983) A leisury look at the bootstrap, the jackknife, and cross-validation. The American Statistician 37(1):36–48

[58] Eickeler S, Kosmala A, Rigoll G (1998) Hidden Markov Model based continuous online gesture recognition. In: Proc. Int. Conf. on Pattern Recognition, vol 2, pp 1206–1208

[59] Eickeler S, Müller S, Rigoll G (1999) High performance face recognition using pseudo 2D-Hidden Markov Models. In: Proc. European Control Conference, Karlsruhe

[60] Elms AJ, Procter S, Illingworth J (1998) The advantage of using an HMM-based approach for faxed word recognition. Int Journal on Document Analysis and Recognition 1(1):18–36

[61] Ewens WJ, Grant GR (2001) Statistical Methods in Bioinformatics: An Introduction, 2nd edn. Statistics for Biology and Health, Springer, Berlin

[62] Federico M, Cettelo M, Brugnara F, Antoniol G (1995) Language modelling for efficient beam-search. Computer Speech & Language 9:353–379

[63] Fink GA (1999) Developing HMM-based recognizers with ESMERALDA. In: Matoušek V, Mautner P, Ocelíková J, Sojka P (eds) Text, Speech and Dialogue, Lecture Notes in Artificial Intelligence, vol 1692, Springer, Berlin Heidelberg, pp 229–234

[64] Fink GA, Plötz T (2004) Integrating speaker identification and learning with adaptive speech recognition. In: 2004: A Speaker Odyssey – The Speaker and Language Recognition Workshop, Toledo, pp 185–192

[65] Fink GA, Plötz T (2005) On appearance-based feature extraction methods for writer-independent handwritten text recognition. In: Proc. Int. Conf. on Document Analysis and Recognition, IEEE, Seoul, Korea, vol 2, pp 1070–1074

[66] Fink GA, Plötz T (2006) Unsupervised estimation of writing style models for improved unconstrained off-line handwriting recognition. In: Proc. Int. Workshop on Frontiers in Handwriting Recognition, La Baule, France, pp 429–434

[67] Fink GA, Plötz T (2007) ESMERALDA: A development environment for HMM-based pattern recognition systems. In: 7th Open German/Russian Workshop on Pattern Recognition and Image Understanding, Ettlingen, Germany

[68] Fink GA, Plötz T (2007) ESMERALDA: A development environment for HMM-based pattern recognition systems. URL `http://sourceforge.net/projects/esmeralda`

[69] Fink GA, Plötz T (2007) On the use of context-dependent modeling units for HMM-based offline handwriting recognition. In: Proc. Int. Conf. on Document Analysis and Recognition, Curitiba, Brazil

[70] Fink GA, Sagerer G (2000) Zeitsynchrone Suche mit n-Gramm-Modellen höherer Ordnung (Time-synchonous search with higher-order n-gram models). In: Konvens 2000 / Sprachkommunikation, ITG-Fachbericht 161, VDE Verlag, Berlin, pp 145–150, (in German)

[71] Fink GA, Schillo C, Kummert F, Sagerer G (1998) Incremental speech recognition for multimodal interfaces. In: Proc. Annual Conference of the IEEE Industrial Electronics Society, Aachen, vol 4, pp 2012–2017

[72] Fink GA, Wienecke M, Sagerer G (2001) Video-based on-line handwriting recognition. In: Proc. Int. Conf. on Document Analysis and Recognition, pp 226–230

[73] Fink GA, Wienecke M, Sagerer G (2005) Experiments in video-based whiteboard reading. In: First Int. Workshop on Camera-Based Document Analysis and Recognition, Seoul, Korea, pp 95–100

[74] Fischer A, Stahl V (1999) Database and online adaptation for improved speech recognition in car environments. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[75] Flach G, Hoffmann R, Rudolpy T (2000) Eine aktuelle Evaluation kommerzieller Diktiersysteme (An up to date evaluation of commercial dictation systems). In: Konvens 2000 / Sprachkommunikation, ITG-Fachbericht 161, VDE Verlag, Berlin, pp 51–55, (in German)

[76] Forney GD (1973) The Viterbi algorithm. Proceedings of the IEEE 61(3):268–278

[77] Fritsch J, Rogina I (1996) The bucket box intersection (BBI) algorithm for fast approximative evaluation of diagonal mixture Gaussians. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Atlanta, vol 1, pp 837–840

[78] Fritsch J, Finke M, Waibel A (1997) Context-dependent hybrid HME/HMM speech recognition using polyphone clustering decision trees. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, München, vol 3, pp 1759–1762

[79] Fukunaga K (1972) Introduction to Statistical Pattern Recognition. Academic Press, New York

[80] Fukunaga K (1990) Introduction to Statistical Pattern Recognition, 2nd edn. Academic Press, New York

[81] Furui S (2000) Digital Speech Processing, Synthesis, and Recognition. Signal Processing and Communications Series, Marcel Dekker, New York, Basel

[82] Gaizauskas R (1998) Evaluation in language and speech technology. Computer Speech & Language 12:249–262

[83] Gales MJF, Woodland PC (1996) Variance compensation within the MLLR framework. Tech. rep., Cambridge University Engineering Department

[84] Gauvain J, Lamel L, Adda G, Adda-Decker M (1994) The LIMSI continuous speech dictation system: Evaluation on the ARPA Wall Street Journal task. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Adelaide, vol I, pp 557–560

[85] Gersho A, Gray RM (1992) Vector Quantization and Signal Compression. Communications and Information Theory, Kluwer Academic Publishers, Boston

[86] Geutner P (1995) Using morphology towards better large-vocabulary speech recognition systems. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Detroit, vol 1, pp 445–448

[87] Haeb-Umbach R (1999) Investigations on inter-speaker variability in the feature space. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[88] Haeb-Umbach R, Ney H (1992) Linear discriminant analysis for improved large vocabulary continuous speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, San Francisco, vol 1, pp 13–16

[89] Hain T, Woodland PC, Niesler TR, Whittaker EWD (1999) The 1998 HTK system for transcription of conversational telephone speech. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[90] Haussler D, Kulp D, Reese MG, Eckmann FH (1996) A generalized Hidden Markov Model for the recognition of human genes in DNA. In: Proc. Int. Conf. on Intelligent Systems for Molecular Biology, St. Louis, pp 134–142

[91] Hayamizu S, Itou K, Takaka K (1993) Detection of unknown words in large vocabulary speech recognition. In: Proc. European Conf. on Speech Communication and Technology, Berlin, pp 2113–2116

[92] Hoey J, Little JJ (2000) Representation and recognition of complex human motion. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition, pp 752–759

[93] Hovland GE, Sikka P, McCarragher BJ (1996) Skill acquisition from human demonstration using a Hidden Markov Model. In: Proc. IEEE Int. Conf. on Robotics and Automation, Minneapolis, pp 2706–2711

[94] Huang X, Jack M (1989) Semi-continuous Hidden Markov Models for speech signals. Computer Speech & Language 3(3):239–251

[95] Huang X, Ariki Y, Jack M (1990) Hidden Markov Models for Speech Recognition. No. 7 in Information Technology Series, Edinburgh University Press, Edinburgh

[96] Huang X, Alleva F, Hon HW, Hwang MY, Lee KF, Rosenfeld R (1993) The SPHINX-II speech recognition system: An overview. Computer Speech & Language 2:127–148

[97] Huang X, Acero A, Hon HW (2001) Spoken Language Processing: A Guide to Theory, Algorithm, and System Development. Prentice Hall, Englewood Cliffs, New Jersey

[98] Hughey R, Karplus K, Krogh A (2005) SAM: Sequence alignment and modeling software system. URL `http://www.cse.ucsc.edu/research/compbio/sam.html`

[99] Human Genome Project (2003) URL `http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml`

[100] IEEE Computer Society Technical Committee on Microprocessors and Microcomputers, IEEE Standards Board (1987) IEEE Standard for Radix-Independent Floating-Point Arithmetic. ANSI/IEEE Std 854-1987. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA

[101] International Human Genome Sequencing Consortium (2001) Initial sequencing and analysis of the human genome. Nature 409:860–921

[102] International Human Genome Sequencing Consortium (2004) Fishing the euchromatic sequence of the human genome. Nature 43:931–945

[103] Iyer RM, Ostendorf M (1999) Modeling long distance dependence in language: Topic mixtures versus dynamic cache models. IEEE Trans on Speech and Audio Processing 7(1):30–39

[104] Jelinek F (1990) Self-organized language modeling for speech recognition. In: Waibel A, Lee KF (eds) Readings in Speech Recognition, Morgan Kaufmann, San Mateo, CA, pp 450–506

[105] Jelinek F (1997) Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA

[106] Jelinek F, Mercer R (1980) Interpolated estimation of Markov source parameters from sparse data. In: Gelsema E, Kanal L (eds) Pattern Recognition in Practice, North Holland, pp 381–397

[107] Jelinek F, Bahl LR, Mercer RL (1975) Design of a linguistic statistical decoder for the recognition of continuous speech. IEEE Trans on Information Theory 21(3):250–256

[108] Jelinek F, Mercer R, Bahl L (1982) Continuous speech recognition. In: Krishnaiah P, Kanal L (eds) Handbook of Statistics, vol 2, North-Holland, pp 549–573

[109] Jennings A, McKeown JJ (1992) Matrix Computation, 2nd edn. Wiley, New York

[110] Joshi D, Li J, Wang JZ (2006) A computationally efficient approach to the estimation of two- and three-dimensional hidden Markov models. IEEE Trans on Image Processing 15(7):1871–1886

[111] Juang BH, Rabiner LR (1990) The segmental k-means algorithm for estimating parameters of Hidden Markov Models. IEEE Trans on Acoustics, Speech, and Signal Processing 38(9):1639–1641

[112] Jusek A, Fink GA, Kummert F, Sagerer G (1996) Automatically generated models for unknown words. In: Proc. Australian International Conference on Speech Science and Technology, Adelaide, pp 301–306

[113] Kale A, Sundaresan A, Rajagopalan AN, Cuntoor NP, Roy-Chowdhury AK, Krüger V, Chellappa R (2004) Identification of humans using gait. IEEE Trans on Image Processing 13(9):1163–1173

[114] Kannan A, Ostendorf M, Rohlicek JR (1994) Maximum likelihood clustering of Gaussians for speech recognition. IEEE Trans on Speech and Audio Processing 2(3):453–455

[115] Kanthak S, Molau S, Sixtus A, Schlüter R, Ney H (2000) The RWTH large vocabulary speech recognition system for spontaneous speech. In: Konvens 2000 / Sprachkommunikation, ITG-Fachbericht 161, VDE Verlag, Berlin, pp 249–256

[116] Kapadia AS, Chan W, Moyé L (2005) Mathematical Statistics with Applications. Chapman & Hall/CRC, Boca Raton

[117] Katz S (1987) Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Trans on Acoustics, Speech, and Signal Processing 35(3):400–401

[118] Kingsbury NG, Rayner PJW (1971) Digital filtering using logarithmic arithmetic. Electronics Letters 7(2):56–58

[119] Kirchhoff K, Fink GA, Sagerer G (2000) Conversational speech recognition using acoustic and articulatory input. In: IEEE International Conference on Acoustics, Speech and Signal Processing, Istanbul

[120] Kirchhoff K, Fink GA, Sagerer G (2002) Combining acoustic and articulatory information for robust speech recognition. Speech Communication 37(3-4):303–319

[121] Kneser R, Ney H (1995) Improved backing-off for M-gram language modeling. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Adelaide, vol 1, pp 181–184

[122] Kneser R, Peters J (1997) Semantic clustering for adaptive language modeling. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, München, vol 2, pp 779–782

[123] Knill KM, Gales MJ, Young SJ (1996) Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs. In: International Conference on Spoken Language Processing, Philadelphia, PA, vol 1, pp 470–473

[124] Kosmala A, Rigoll G (1998) Tree-based state clustering using self-organizing principles for large vocabulary on-line handwriting recognition. In: Proc. Int. Conf. on Pattern Recognition, Brisbane, vol 3I, pp 1313–1316

[125] Kosmala A, Rottland J, Rigoll G (1997) Improved on-line handwriting recognition using context dependent hidden markov models. In: Proc. Int. Conf. on Document Analysis and Recognition, Ulm, Germany, vol 2, pp 641–644

[126] Kosmala A, Rottland J, Rigoll G (1997) Large vocabulary on-line handwriting recognition with context dependent Hidden Markov Models. In: Mustererkennung 97, 19. DAGM-Symposium Braunschweig, Informatik aktuell, Springer, Berlin, pp 254–261

[127] Krogh A (1994) Hidden Markov Models for labeled sequences. In: Proc. Int. Conf. on Pattern Recognition, Jerusalem, vol 2, pp 140–144

[128] Krogh A (1998) An introduction to Hidden Markov Models for biological sequences. In: Salzberg SL, Searls DB, Kasif S (eds) Computational Methods in Molecular Biology, Elsevier, New York, pp 45–63

[129] Krogh A, Brown M, Mian IS, Sjölander K, Haussler D (1994) Hidden Markov Models in computational biology: Applications to protein modeling. Journal of Molecular Biology 235:1501–1531

[130] Kuhn R, De Mori R (1990) A cache-based natural language model for speech recognition. IEEE Trans on Pattern Analysis and Machine Intelligence 12(6):570–583

[131] Kuhn R, Lazarides A, Normandin Y, Brousseau J, Nöth E (1994) Applications of decision tree methodology in speech recognition and understanding. In: Niemann H, de Mori R, Harnrieder G (eds) Proceedings in Artificial Intelligence, vol 1, infix, Sankt Augustin, pp 220–232

[132] Kummert F, Fink GA, Sagerer G (2000) A hybrid speech recognizer combining HMMs and polynomial classification. In: Proc. Int. Conf. on Spoken Language Processing, Beijing, China, vol 3, pp 814–817

[133] Lam L, Suen CY, Guillevic D, Strathy NW, Cheriet M, Liu K, Said JN (1995) Automatic processing of information on cheques. In: Proc. Int. Conf. on Systems, Man & Cybernetics, Vancouver, vol 3, pp 2353–2358

[134] Larsen RJ, Marx ML (2001) An Introduction to Mathematical Statistics and its Applications, 3rd edn. Prentice-Hall, Upper Saddle River, NJ

[135] Lee CH, Rabiner LR, Pieraccini R, Wilpon JG (1990) Acoustic modeling for large vocabulary speech recognition. Computer Speech & Language 4:127–165

[136] Lee KF (1989) Automatic Speech Recognition: The Development of the SPHINX System. Kluwer Academic Publishers, Boston

[137] Lee KF (1990) Context dependent phonetic hidden Markov models for continuous speech recognition. IEEE Trans on Acoustics, Speech, and Signal Processing 38(4):599–609

[138] Leggetter CJ, Woodland PC (1995) Flexible speaker adaptation using maximum likelihood linear regression. In: Workshop on Spoken Language Systems Technology, ARPA, pp 110–115

[139] Leggetter CJ, Woodland PC (1995) Maximum Likelihood Linear Regression for speaker adaptation of continuous density Hidden Markov Models. Computer Speech & Language 9:171–185

[140] Levinson S (1986) Continuously variable duration hidden Markov models for automatic speech recognition. Computer Speech & Language 1(1):29–45

[141] Li J, Gray RM, Olshen RA (2000) Multiresolution image classification by hierarchical modeling with two-dimensional hidden Markov models. IEEE Trans on Information Theory 46(5):1826–1841

[142] Li J, Najmi A, Gray RM (2000) Image classification by a two-dimensional Hidden Markov Model. IEEE Transactions on Signal Processing 48(2):517–533

[143] Lien JJ, Kanade T, Zlochower AJ, Cohn JF, Li CC (1997) Automatically recognizing facial expressions in the spatio-temporal domain. In: Proc. Workshop on Perceptual User Interfaces, Banff, Alberta, Canada, pp 94–97

[144] Linde Y, Buzo A, Gray RM (1980) An algorithm for vector quantizer design. IEEE Trans on Communications 28(1):84–95

[145] Liu Z, Sarkar S (2006) Improved gait recognition by gait dynamics normalization. IEEE Trans on Pattern Analysis and Machine Intelligence 28(6):863–876

[146] Lombard E (1911) Le signe de l'élévation de la voix. Annales des maladies de l'oreilles, du larynx, du nez et du pharynx 37:101–119

[147] Lowerre B, Reddy D (1980) The Harpy speech understanding system. In: Lea W (ed) Trends in Speech Recognition, Prentice-Hall Inc., Englewood Cliffs, New Jersey, pp 340–360

[148] Lowerre BT (1976) The HARPY speech recognition system. PhD thesis, Carnegie-Mellon University, Department of Computer Science, Pittsburg

[149] MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Cam LML, Neyman J (eds) Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol 1, pp 281–296

[150] Mahajan M, Beeferman D, Huang XD (1999) Improved topic-dependent language modeling using information retrieval techniques. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona, vol 1, pp 15–19

[151] Markov AA (1913) Примѣръ статистическаго изслѣдованія надъ текстомъ ,,Евгенія Онѣгина" иллюстрирующій связь испытаній въ цѣпь (Example of statistical investigations of the text of ,,Eugen Onegin", wich demonstrates the connection of events in a chain). In: Извѣстія Императорской Академій Наукъ (Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg), Sankt-Petersburg, pp 153–162, (in Russian)

[152] Marti UV, Bunke H (2002) The IAM-database: An English sentence database for offline handwriting recognition. Int Journal on Document Analysis and Recognition 5(1):39–46

[153] Martin S, Hamacher C, Liermann J, Wessel F, Ney H (1999) Assessment of smoothing methods and complex stochastic language modeling. In: Proc. European Conf. on Speech Communication and Technology, Budapest, pp 1939–1942

[154] Martínez F, Tapias D, Álvarez J (1998) Towards speech rate independence in large vocabulary continuous speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, pp 725–728

[155] Mayer J (ed) (1999) Dichterhandschriften, von Martin Luther bis Sarah Kirsch. Phillip Reclam jun., Stuttgart

[156] Mercer R (1988) Language modeling. In: IEEE Workshop on Speech Recognition, Arden House, Harriman, NY

[157] Merhav N, Ephraim Y (1991) Hidden Markov modeling using a dominant state sequence with application to speech recognition. Computer Speech & Language 5(4):327–339

[158] Merhav N, Lee CH (1993) On the asymptotic statistical behavior of empirical cepstral coefficients. IEEE Trans on Signal Processing 41(5):1990–1993

[159] Morgan N, Bourlard H (1995) Continuous speech recognition. IEEE Signal Processing Magazine 12(3):24–42

[160] Mori S, Nishida H (1998) Optical Character Recognition. John Wiley, New York

[161] Munich ME, Perona P (2002) Visual input for pen-based computers. IEEE Trans on Pattern Analysis and Machine Intelligence 24(3):313–328

[162] Nam Y, Wohn K (1997) Recognition of hand gestures with 3D, nonlinear arm movement. Pattern Recognition Letters 18(1):105–113

[163] Ney H, Ortmanns S (1999) Dynamic programming search for continuous speech recognition. IEEE Signal Processing Magazine 16(5):64–83

[164] Ney H, Haeb-Umbach R, Tran B, Oerder M (1992) Improvements in beam search for 10000-word continuous speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, San Francisco, vol 1, pp 9–12

[165] Ney H, Essen U, Kneser R (1994) On structuring probabilistic dependencies in stochastic language modelling. Computer Speech & Language 8:1–38

[166] Ney H, Essen U, Kneser R (1995) On the estimation of ‘small’ probabilities by leaving-one-out. IEEE Trans on Pattern Analysis and Machine Intelligence 17(12):1202–1212

[167] Ney H, Welling L, Beulen K, Wessel F (1998) The RWTH speech recognition system and spoken document retrieval. In: Proc. Annual Conference of the IEEE Industrial Electronics Society, Aachen, vol 4, pp 2022–2027

[168] Nguyen L, Schwartz R (1998) The BBN single-phonetic-tree fast-match algorithm. In: Proc. Int. Conf. on Spoken Language Processing, Sydney

[169] Nguyen L, Matsoukas S, Billa J, Schwartz R, Makhoul J (2000) The 1999 BBN BYBLOS 10xRT broadcast news transcription system. In: 2000 Speech Transcription Workshop, Maryland

[170] Niemann H (1990) Pattern Analysis and Understanding, Series in Information Sciences, vol 4, 2nd edn. Springer, Berlin Heidelberg

[171] Niesler TR, Woodland PC (1996) Combination of word-based and category-based language models. In: Proc. Int. Conf. on Spoken Language Processing, Philadelphia, vol 1, pp 220–223

[172] Niesler TR, Whittaker EWD, Woodland PC (1998) Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Seattle, vol 1, pp 177–180

[173] Nock HJ, Gales MJF, Young SJ (1997) A comparative study of methods for phonetic decision-tree state clustering. In: Proc. European Conf. on Speech Communication and Technology

[174] Ohler U, Harbeck S, Niemann H, Nöth E, Reese MG (1999) Interpolated Markov chains for eukaryotic promoter recognition. Bioinformatics 15(5):362–369

[175] Ortmanns S, Ney H (2000) Look-ahead techniques for fast beam search. Computer Speech & Language 14:15–32

[176] Ortmanns S, Ney H (2000) The time-conditioned approach in dynamic programming search for LVCSR. IEEE Trans on Speech and Audio Processing 8(6):676–687

[177] Ortmanns S, Ney H, Eiden A (1996) Language-model look-ahead for large vocabulary speech recognition. In: Proc. Int. Conf. on Spoken Language Processing, Philadelphia, pp 2095–2098

[178] Ortmanns S, Ney H, Seide F, Lindam I (1996) A comparison of time conditioned and word conditioned search techniques for large vocabulary speech recognition. In: Proc. Int. Conf. on Spoken Language Processing, Philadelphia, pp 2091–2094

[179] Ortmanns S, Eiden A, Ney H, Coenen N (1997) Look-ahead techniques for fast beam search. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, München, vol 3, pp 1783–1786

[180] Ortmanns S, Firzlaff T, Ney H (1997) Fast likelihood computation methods for continuous mixture densities in large vocabulary speech recognition. In: Proc. European Conf. on Speech Communication and Technology, Rhodes, vol 1, pp 139–142

[181] O'Shaughnessy D (2000) Speech Communications: Human and Machine, 2nd edn. Addison-Wesley, Reading, Massachusetts

[182] Paul D (1997) An investigation of Gaussian shortlists. In: Furui S, Huang BH, Chu W (eds) Proc. Workshop on Automatic Speech Recognition and Understanding, IEEE Signal Processing Society

[183] Pestman WR (1998) Mathematical Statistics. de Gruyter, Berlin

[184] Pfeiffer M (2006) Architektur eines multimodalen Forschungssystems zur iterativen inhaltsbasierten Bildsuche (Architecture of a multimodal research system for iterative interactive image retrieval). PhD thesis, Bielefeld University, Faculty of Technology, Bielefeld, Germany, (in German)

[185] Plötz T, Fink GA (2002) Robust time-synchronous environmental adaptation for continuous speech recognition systems. In: Proc. Int. Conf. on Spoken Language Processing, Denver, vol 2, pp 1409–1412

[186] Plötz T, Fink GA (2004) Feature extraction for improved Profile HMM based biological sequence analysis. In: Proc. Int. Conf. on Pattern Recognition, IEEE, 2, pp 315–318

[187] Plötz T, Fink GA (2005) A new approach for HMM based protein sequence modeling and its application to remote homology classification. In: Proc. Workshop Statistical Signal Processing, IEEE, Bordeaux, France

[188] Plötz T, Fink GA (2005) Robust remote homology detection by feature based Profile Hidden Markov Models. Statistical Applications in Genetics and Molecular Biology 4(1)

[189] Plötz T, Fink GA (2006) Pattern recognition methods for advanced stochastic protein sequence analysis using HMMs. Pattern Recognition, Special Issue on Bioinformatics 39:2267–2280

[190] Plötz T, Fink GA, Husemann P, Kanies S, Lienemann K, Marschall T, Martin M, Schillingmann L, Steinrücken M, Sudek H (2006) Automatic detection of song changes in music mixes using stochastic models. In: Proc. Int. Conf. on Pattern Recognition, IEEE, vol 3, pp 665–668

[191] Ponte JM, Croft WB (1998) A language modeling approach to information retrieval. In: Research and Development in Information Retrieval, pp 275–281

[192] Popovici C, Baggia P (1997) Specialized language models using dialogue predictions. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, München, vol 2, pp 779–782

[193] Pratt JW, Raiffa H, Schlaifer R (1996) Introduction to Statistical Decision Theory, 2nd edn. MIT Press, Cambridge, MA

[194] Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1988) Numerical Recipies in C: The Art of Scientific Computing. Cambridge University Press, Cambridge

[195] Rabiner LR (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 77(2):257–286

[196] Rabiner LR, Juang BH (1993) Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cliffs, New Jersey

[197] Rigoll G (1994) Maximum Mutual Information Neural Networks for hybrid connectionist-HMM speech recognition systems. IEEE Trans on Speech and Audio Processing 2(1):175–184

[198] Rigoll G, Kosmala A, Willett D (1998) An investigation of context-dependent and hybrid modeling techniques for very large vocabulary on-line cursive handwriting recognition. In: Proc. 6th Int. Workshop on Frontiers in Handwriting Recognition, Taejon, Korea

[199] Rosenberg AE, Lee CH, Soong FK (1994) Cepstral channel normalization techniques for HMM-based speaker verification. In: Proc. Int. Conf. on Spoken Language Processing, Yokohama, Japan, vol 4, pp 1835–1838

[200] Rosenfeld R (2000) Two decades of statistical language modeling: Where do we go from here? Proceedings of the IEEE 88(8):1270–1278

[201] Rottland J, Rigoll G (2000) Tied posteriors: An approach for effective introduction of context dependency in hybrid NN/HMM LVCSR. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul

[202] Salzberg SL, Delcher AL, Kasif S, White O (1998) Microbial gene identification using interpolated Markov models. Nucleic Acids Research 26(2):544–548

[203] Samaria F, Young S (1994) HMM-based architecture for face identification. Image and Vision Computing 12:537–543

[204] Samuelsson C, Reichl W (1999) A class-based language model for large-vocabulary speech recognition extracted from part-of-speech statistics. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[205] Schillo C, Fink GA, Kummert F (2000) Grapheme based speech recognition for large vocabularies. In: Proc. Int. Conf. on Spoken Language Processing, Beijing, China, vol 4, pp 584–587

[206] Schmid G, Schukat-Talamazzini EG, Niemann H (1993) Analyse mehrkanaliger Meßreihen im Fahrzeugbau mit Hidden Markovmodellen (Analysis of multi-channel measurement data in car manufacturing by hidden Markov models). In: Pöppl S (ed) Mustererkennung 1993, 15. DAGM Symposium, Springer, Informatik aktuell, pp 391–398, (in German)

[207] Schukat-Talamazzini EG, Bielecki M, Niemann H, Kuhn T, Rieck S (1993) A non-metrical space search algorithm for fast gaussian vector quantization. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Minneapolis, pp 688–691

[208] Schukat-Talamazzini EG, Kuhn T, Niemann H (1993) Das POLYPHON — eine neue Wortuntereinheit zur automatischen Spracherkennung (The polyphone — A new sub-word unit for automatic speech recognition). In: Fortschritte der Akustik, Frankfurt, pp 948–951, (in German)

[209] Schukat-Talamazzini EG, Niemann H, Eckert W, Kuhn T, Rieck S (1993) Automatic speech recognition without phonemes. In: Proc. European Conf. on Speech Communication and Technology, Berlin, pp 129–132

[210] Schultz T, Rogina I (1995) Acoustic and language modeling of human and nonhuman noises for human-to-human spontaneous speech recognition. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Detroit

[211] Schürmann J (1996) Pattern Classification. John Wiley & Sons, New York

[212] Schwartz R, Chow Y, Roucos S, Krasner M, Makhoul J (1984) Improved hidden Markov modelling of phonemes for continuous speech recognition. In: Int. Conf. on Acoustics, Speech and Signal Processing, San Diego, pp 35.6.1–35.6.4

[213] Schwartz R, Chow Y, Kimball O, Roucos S, Krasner M, Makhoul J (1985) Context-dependent modeling for acoustic-phonetic recognition of continuous speech. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Tampa, Florida, pp 1205–1208

[214] Schwartz R, LaPre C, Makhoul J, Raphael C, Zhao Y (1996) Language-independent OCR using a continuous speech recognition system. In: Proc. Int. Conf. on Pattern Recognition, Vienna, Austria, vol 3, pp 99–103

[215] Siegler MA, Stern RM (1995) On the effects of speech rate in large vocabulary speech recognition systems. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Detroit, vol 1, pp 612–615

[216] Siivola V, Kurimo M, Lagus K (2001) Large vocabulary statistical language modeling for continuous speech recognition in Finnish. In: Proc. European Conf. on Speech Communication and Technology, Aalborg

[217] Sixtus A, Molau S, Kanthak S, Schlüter R, Ney H (2000) Recent improvements of the RWTH large vocabulary speech recognition system on spontaneous speech. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Istanbul, pp 1671–1674

[218] Spiess T, Wrede B, Kummert F, Fink GA (2003) Data-driven pronunciation modeling for ASR using acoustic subword units. In: Proc. European Conf. on Speech Communication and Technology, Geneva, pp 2549–2552

[219] Starner T, Pentland A (1995) Visual recognition of American Sign Language using Hidden Markov Models. In: International Workshop on Automatic Face and Gesture Recognition, Zürich, pp 189–194

[220] Steinbiss V, Tran BH, Ney H (1994) Improvements in beam search. In: Proc. Int. Conf. on Spoken Language Processing, Yokohama, Japan, vol 4, pp 2143–2146

[221] Steinbiss V, Ney H, Aubert X, Besling S, Dugast C, Essen U, Haeb-Umbach R, Kneser R, Meier HG, Oerder M, Tran BH (1996) The Philips research system for continuous-speech recognition. Philips Journal of Research 49(4):317–352

[222] Sundaresan A, Chowdhury AR, Chellappa R (2003) A hidden Markov model based framework for recognition of humans from gait sequences. In: Int. Conf. on Image Processing

[223] Terrell GR (1999) Mathematical Statistics. Springer Texts in Statistics, Springer, Berlin

[224] Tokuno J, Inami N, Matsuda S, Nakai M, Shimodaira H, Sagayama S (2002) Context-dependent substroke model for HMM-based on-line handwriting

recognition. In: Proc. Int. Workshop on Frontiers in Handwriting Recognition, pp 78–83

[225] Venter JC, et al (2001) The sequence of the human genome. Science 291:1304–1351

[226] Viterbi A (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Trans on Information Theory 13:260–269

[227] Wachsmuth S, Fink GA, Sagerer G (1998) Integration of parsing and incremental speech recognition. In: Proceedings of the European Signal Processing Conference, Rhodes, vol 1, pp 371–375

[228] Wachsmuth S, Fink GA, Kummert F, Sagerer G (2000) Using speech in visual object recognition. In: Sommer G, Krüger N, Perwass C (eds) Mustererkennung 2000, 22. DAGM-Symposium Kiel, Springer, Informatik Aktuell, pp 428–435

[229] Waibel A, Geutner P, Tomokiyo LM, Schultz T, Woszczyna M (2000) Multilinguality in speech and spoken language systems. Proceedings of the IEEE 88(8):1297–1313

[230] Weitzenberg J, Posch S, Rost M (2000) Diskrete Hidden Markov Modelle zur Analyse von Meßkurven amperometrischer Biosensoren (Discrete hidden Markov models for the analysis of measurement curves of amperometric biosensors). In: Sommer G, Krüger N, Perwass C (eds) Mustererkennug 2000. Proceedings 22. DAGM-Symposium, Springer, Berlin, Informatik Aktuell, pp 317–324, (in German)

[231] Wellekens CJ (1992) Mixture density estimators in viterbi training. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, vol 1, pp 361–364

[232] Welling L, Kanthak S, Ney H (1999) Improved methods for vocal tract normalisation. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona, pp 761–764

[233] Wells J (2005) SAMPA – Computer readable phonetic alphabet. URL `http://www.phon.ucl.ac.uk/home/sampa/home.htm`

[234] Wendt S, Fink GA, Kummert F (2001) Forward masking for increased robustness in automatic speech recognition. In: Proc. European Conf. on Speech Communication and Technology, Aalborg, Dänemark, vol 1, pp 615–618

[235] Wendt S, Fink GA, Kummert F (2002) Dynamic search-space pruning for time-constrained speech recognition. In: International Conference on Spoken Language Processing, Denver, vol 1, pp 377–380

[236] Wessel F, Baader A (1999) Robust dialogue-state dependent language modeling using leaving-one-out. In: Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, Phoenix, Arizona

[237] Wessel F, Ortmanns S, Ney H (1997) Implementation of word based statistical language models. In: Proc. SQEL Workshop on Multi-Lingual Information Retrieval Dialogs, Plzen, pp 55–59

[238] Wessel F, Baader A, Ney H (1999) A comparison of dialogue-state dependent language models. In: Proc. ECSA Workshop on Interactive Dialogue in Multi-Modal Systems, Irsee, Germany, pp 93–96

[239] Westphal M (1997) The use of cepstral means in conversational speech recognition. In: Proc. European Conf. on Speech Communication and Technology, Rhodes, Greece, vol 3, pp 1143–1146

[240] White DJ (1993) Markov Decision Processes. John Wiley & Sons, New York

[241] Whittaker EWD, Woodland PC (1998) Comparison of language modelling techniques for Russian and English. In: Proc. Int. Conf. on Spoken Language Processing, Sydney

[242] Whittaker EWD, Woodland PC (2000) Particle-based language modelling. In: Proc. Int. Conf. on Spoken Language Processing, Beijing

[243] Wienecke M, Fink GA, Sagerer G (2001) A handwriting recognition system based on visual input. In: Schiele B, Sagerer G (eds) Computer Vision Systems, Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 63–72

[244] Wienecke M, Fink GA, Sagerer G (2002) Experiments in unconstrained offline handwritten text recognition. In: Proc. 8th Int. Workshop on Frontiers in Handwriting Recognition, Niagara on the Lake, Canada

[245] Wienecke M, Fink GA, Sagerer G (2003) Towards automatic video-based whiteboard reading. In: Proc. Int. Conf. on Document Analysis and Recognition, Edinburgh, pp 87–91

[246] Wienecke M, Fink GA, Sagerer G (2005) Toward automatic video-based whiteboard reading. Int Journal on Document Analysis and Recognition 7(2–3):188–200

[247] Wilson AW, Bobick AF (1995) Learning visual behavior for gesture analysis. In: Proc. IEEE Symposium on Computer Vision, Coral Gables, Florida

[248] Winograd T (1983) Language as a Cognitive Process, vol 1: Syntax. Addison-Wesley, Reading, MA

[249] Witten IH, Bell TC (1991) The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Trans on Information Theory 37(4):1085–1094

[250] Wrede B, Fink GA, Sagerer G (2000) Influence of duration on static and dynamic properties of German vowels in spontaneous speech. In: Proc. Int. Conf. on Spoken Language Processing, Beijing, China, vol 1, pp 82–85

[251] Wrede B, Fink GA, Sagerer G (2001) An investigation of modelling aspects for rate-dependent speech recognition. In: Proc. European Conf. on Speech Communication and Technology, Aalborg, vol 4, pp 2527–2530

[252] Yakowitz SJ (1970) Unsupervised learning and the identification of finite mixtures. IEEE Trans on Information Theory 16(3):330–338

[253] Yamamoto J, Ohya J, Ishii K (1992) Recognizing human action in time-sequential images using Hidden Markov Models. In: Proc. Int. Conf. on Computer Vision and Pattern Recognition, pp 379–387

[254] Young SJ, Chase LL (1998) Speech recognition evaluation: A review of the U.S. CSR and LVCSR programmes. Computer Speech & Language 12:263–279

[255] Young SJ, Woodland PC (1994) State clustering in Hidden Markov Model-based continuous speech recognition. Computer Speech & Language 8:369–383

[256] Young SR (1994) Detection of misrecognitions and out-of-vocabulary words in spontaneous speech. In: McKevitt P (ed) AAAI-94 Workshop Program: Integration of Natural Language and Speech Processing, Seattle, Washington, pp 31–36

[257] Zwicker E, Fastl H (1999) Psychoacoustics: Facts and Models, Springer Series in Information Sciences, vol 22, 2nd edn. Springer, Berlin, Heidelberg, New York

Index

absolute discounting → discounting, absolute
adaptation, 181–188
  of n-gram models, 186–188
  of hidden Markov models, 182–186
  principles, 181–182
adding one, 101, 112

backing off, 106–108, 113, 176, 213, 220
backward algorithm, *see* forward-backward algorithm
backward variable, 78, 81, 171
  see also forward-backward algorithm
Bakis model, *see* hidden Markov model, topology
Baum-Welch algorithm, **80–86**, 90, 93, 170, 183, 185, 217, 223
  segmental, 171–173
Bayes' rule, 35, 58, 74, 123, 190
beam search, **167–170**, 171, 191, 195, 198, 209
bi-gram model, 96, 105, 110, 112, 177, 192, 196
  see also n-gram model
BYBLOS, 209, 215

category, *see* n-gram model, category-based
centroid, 48
cepstrum, *11*, *148*, *207*, *209*, *211*
channel model, 189
classification and regression tree, → decision tree
cluster analysis, *see* vector quantization
codebook, 46
  see also vector quantization
correlation matrix, *141*
covariance matrix, **38**, 141
curse of dimensionality, *138*

decision tree, 156, 157, 208, 218
density → probability density
Dirichlet distribution, *222*
discounting, 102–104
  absolute, **103–104**, 107, 113, 176, 213, 220
  linear, **102–103**, 106, 113, 208
distribution function, 35
divergence, *159*
dynamic time warping, *75*
  see also Viterbi algorithm

eigenvalue matrix, 145
eigenvector matrix, 145
EM algorithm, *56*, *59*, *81*, *93*, *188*
  for mixture densities, 56–59
  see also Baum-Welch algorithm
ESMERALDA, **210**, 219, 224–225
event
  in language modeling, 96
  random, 33
expected value, 37
experiment, 33

flooring, 122–123
forward algorithm, **69–71**, 111
  see also forward-backward algorithm
forward variable, 70, 75, 81, 171
  see also forward algorithm

forward-backward algorithm, 78–79
frame, 10, 21, 208, 209, 211
frequency
  absolute, 97
  relative, 34, 97

garbage model, *see* log-odds score
Gaussian distribution → normal distribution
Gaussian selection, 166
Gaussian short-lists → Gaussian selection
grammar
  stochastic, *95*, *112*

hidden Markov model, **61–92**, 190
  compound models, 131–133
  configuration, 127–136
  continuous, 63
  decoding, 74–76
  definition, 61–63
  discrete, 63
  duration modeling, *92*
  ergodic, 127
  evaluation, 68–74
  hybrid, *92*
  linear, 212, 218
  modeling of emissions, 63–65, 136
  modularization, 129–131
  notation, 67
  parameter estimation, 76–91
  production probability, 68–71
  representation, tree-like, 174–175
  semi-continuous, 64, 84, 123, 160, 161, 173, 212, 220, 224
  topology, 127–128
    Bakis model, 127, 208, 210, 216
    bounded left-right model, *224*
    left-to-right model, 128, 224
    linear, 127, 157
  use cases, 65–67
history
  of an n-gram, 96
  see also n-gram model
HMM → hidden Markov model
HMM networks, 192–193
HMMER, 135, 222–223

independence
  statistical, 34
interpolation, 176
  non-linear, **106**, 108, 113, 208
  of n-gram models, 104–106, 112, 208

Jacobi method, *145*

k-means algorithm, **53–55**, 59, 88
  see also segmental k-means
Karhunen-Loève transform, → principal component analysis
Kingsbury-Rayner formula, 121

language model, 112
  dialog-step dependent, 187
    see also adaptation, of n-gram models
  statistical, *see* n-gram model
  topic-based, 187
    see also adaptation, of n-gram models
language-model look-ahead, **197–198**, 209
Laplacian density, 162
law of large numbers, **34**, 137
LBG algorithm, **52–53**, 59
LDA → linear discriminant analysis
leaving-one-out, *108*
likelihood function, **42**, 56
linear discounting → discounting, linear
linear discriminant analysis, **148–152**, 163, 208
linguistic matching factor, 191
Lloyd's algorithm, **50–52**, 53, 59
log-odds score, 125
logarithmic
  probability representation, 120–122
  summation → Kingsbury-Rayner formula

Mahalanobis distance, 55
MAP → maximum a posteriori
marginal distribution, 36
Markov chain, **40**, 95, 190
  see also stochastic process
Markov chain model → n-gram model
Markov property, **40**, 69, 75
maximum a posteriori estimation, 43, 183, 222
maximum likelihood estimation, **41**, 81, 183
maximum likelihood linear regression, **184–186**, 225
  see also adaptation, of hidden Markov models
maximum mutual information, *see* training, discriminative

mixture, *see* mixture density
mixture density, **39**, 45, 64
  parameter estimation, 56–59
ML → maximum likelihood
MLLR → maximum likelihood linear regression, 210
monophone, 130, 155
multi-pass search, 193–194
multiple alignment, 24, 133, 222, 223

n-gram model, **95–113**, 132, 190
  cache, 187
    see also adaptation, of n-gram models
  category-based, 109–111, 113
  definition, 95–96
  evaluation, 98–100
  notation, 97–98
  parameter estimation, 100–109
  representation, tree-like, 175–178
  use cases, 96–97
neuronal networks, *see* hidden Markov model, hybrid
non-emitting state, 132, 134
normal distribution, **38**, 64, 141
  multivariate, 39

observation, → element of an observation sequence
observation sequence, 62
  see also hidden Markov model

parameter estimation, 41–44
  for n-gram models, 100–109
  for hidden Markov models, 76–91
  for mixture densities, 56–59
  see also maximum likelihood estimation, maximum a posteriori estimation
PCA → principal component analysis
perplexity, 98–100
  see also n-gram model, evaluation
phone, 9
phoneme, 9
phonetically-tied mixture HMM, 161, 209
polyphone, 131, 155
power method, *145*
prefix tree, 174, 176, 208, 219
principal component analysis, **141–144**, **147–148**, 150, 224
probability, 34
  conditional, 34
  posterior, 34
  prior, 34
probability density, 36
production probability, *see* hidden Markov model
profile HMM, 131, **133–136**, 222, 224
pruning, 165
  see also beam search

quantization error, *see* vector quantization
quinphone, 131, 209

random variable, 35–37
  continuous, 35
  discrete, 35
  multivariate → random vector
random vector, 36
  see also random variable
regression class, *see* maximum likelihood linear regression

SAM, 223
scatter matrix, 141, 148, 149
search space copies, 194–197
segmental k-means, **88–90**, 93, 163, 185
simulated annealing, *223*
singleton, 98, 176
sparse data problem, 137
state probability, 77–79, 81, 89, 171, 173
statistical independence, → independence, statistical
stochastic process, **40–41**, 61
sub-word units, 129, **153**
  context-dependent, 130, 154
  context-independent, 129
  see also hidden Markov model, modularization

tied-mixture HMM, → hidden Markov model, semi-continuous
topic mixture, *see* language model, topic-based
training, 76
  corrective, *91*
  discriminative, *91*, *218*
  see also hidden Markov model, parameter estimation
transformation
  orthonormal, 142

tri-gram model, 96, 105
  see also n-gram model
trigraph, 218
  see also triphone
triphone, **130**, 154, 173, 208, 209, 212
  generalized, 131, 156
tying, 64, **152–163**, 174

uni-gram model, 101, 176, 187
  see also n-gram model

variance, 37
vector quantization, **45–59**, 88, 166
Viterbi algorithm, 72, **75–76**, 86, 89, 91, 93, 167, 172, 191
Viterbi training, **86–88**, 93, 185, 223
vocal tract length normalization, *208, 209*

whitening, **144–146**, 150
word network → HMM network

zero probability, *see* discounting
zero-gram model, 101, 176
  see also n-gram model