HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Acoustics and Audio Signal Processing

Jouni Pohjalainen

Methods of Automatic Audio Content Classification

Contents

Abbreviations

AR	Autoregressive
ASR	Automatic Speech Recognition
BIC	Bayesian Information Criterion
CSR	Continuous Speech Recognition
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
EM	Expectation-Maximization
FFT	Fast Fourier Transform
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
ICCR	Iterative Cluster Centroid Reduction
IDFT	Inverse Discrete Fourier Transform
IWR	Isolated Word Recognition
KKZ	the cluster initialization method of Katsavounidis, Jay Kuo and Zhang
kNN	k Nearest Neighbor
LBG	the vector quantizer training method of Linde, Buzo and Gray
LP	Linear Prediction
LPC	Linear Predictive Coding
LSF	Line Spectral Frequency
MFCC	Mel Frequency Cepstral Coefficients
MVDR	Minimum Variance Distortionless Response
PDF	Probability Density Function
PLP	Perceptual Linear Prediction
STE	Short-Time Energy
VAR	Vector Autoregressive
VQ	Vector Quantization

Chapter 1

Introduction

This thesis deals with the automatic extraction of information from audio material. More specifically, it is a review of the most important techniques of digital signal processing and pattern recognition that find application in the automatic extraction of categorical audio content descriptors. From a slightly different perspective, the problem domain is *automatic recognition* - i.e., *segmentation and classification* - of sounds. This study also provides an overview of the different practical applications in which such information extraction has been useful as well as a discussion of the ways in which the algorithms are typically combined in these applications.

Digitally stored multimedia material is currently a rapidly growing resource due to the ongoing technological advancement in data storage, communications and computing. In only recent years, it has become widely possible to transfer long audio and video files via the internet with an acceptable delay. The storage capacities of portable multimedia devices and personal computers have been rapidly increasing. The amount of available digital multimedia information is beginning to overwhelm the capacity of humans to manage and organize it. Thus, computerized solutions for automatic organization of the multimedia material are an attractive approach to access the content efficiently. Because of this, *information retrieval* is an important field of application for automatic audio recognition. Besides audio material, the indexing and retrieval of (audiovisual) video material also benefits from audio recognition. The MPEG-7 standard supports multimedia content analysis and retrieval by specifying an interface for content description by attaching metadata to multimedia content. The analysis methods introduced in this study are applicable in this context.

Besides retrieval applications, another important type of recognition application for audio is *automatic transcription*. It refers to using computer-aided methods for transcription - i.e., generating a sequence of class labels - in order to save human labor; examples include speech dictation and music transcription applications. While information retrieval uses

automatic recognition results as content descriptors to accomplish the search, transcription can directly use the recognition results as output.

Automatic audio recognition is also central in some user interfaces and surveillance applications. It is also frequently useful in application areas such as coding, enhancement and synthesis.

The methods discussed in this thesis should be regarded as general building blocks of audio recognition systems. They may find use as components of solutions in a wide array of specific applications. To mention only a few examples, such applications include:

- Audio content analysis to locate speech and music segments in broadcast audio
- Speech-to-text dictation transcription (by automatic speech recognition)
- Search of a specific speech file based on query words (by automatic speech recognition)
- Query by humming to retrieve a piece of music based on its melody
- Automatic chord transcription of a piece of music
- Real-time audio surveillance

Many of the techniques used in these problems have originally been developed for automatic speech recognition (ASR), for which an impressive amount of research efforts has been invested over decades. To cover ASR comprehensively would be largely out of the scope of this thesis, as this very challenging problem entails important aspects other than just pattern recognition of audio signals. The *domain knowledge* used in designing ASR systems necessarily includes various aspects of speech communication and linguistics, as well as the mechanisms humans use for understanding and decoding speech. However, the basic techniques of the *acoustic modeling* part of ASR, such as the auditory feature extraction methods and the hidden Markov models, are central from the perspective of other audio recognition applications as well. Many music-specific recognition applications also involve a fair amount of domain knowledge related, for example, to the theory of music. It would likewise be beyond the scope to go in too deeply into domain specifics in music analysis.

The main focus of the study is on the recognition of content classes that are clearly dominant in a mono signal when they appear, i.e., not too severely corrupted by interfering sound sources. However, simple methods for increasing robustness against stationary or slowly changing background noise are included in the discussion on signal processing. The problem of separating competing, nonstationary sound sources such as different speakers in a complex auditory scene - essentially the “cocktail-party” problem - belongs to a related field

of research often referred to as computational auditory scene analysis (CASA) [27]. While this study reviews some of the auditory foundations of CASA and presents algorithms that can find use in sound source separation, it is not intended to cover these areas of research. In particular, multichannel analysis methods are outside the scope of this thesis.

Mathematical formulas given for the signal processing and pattern recognition methods, while for the most part sufficient for computer implementation, are primarily intended to illustrate a "basic form" and the principle of each method rather than to represent the computationally most efficient implementation.

Human auditory perception is reviewed in Chapter 2, because the modeling of auditory perception allows more efficient signal representations and classification methods for audio recognition. The most common signal processing techniques used in preprocessing and feature extraction are discussed in Chapter 3. The acoustic features are given as input to pattern recognition methods discussed in Chapter 4. Chapter 5 contains a survey of some published audio classification systems that can be considered representative of different application domains and of different approaches to solve the recognition problem. Chapter 6 summarizes the main findings and presents the author's conclusions.

1.1 On Pattern Recognition

Pattern recognition is one of the central themes in this thesis. For the present purpose and scope, a *pattern* is informally defined as any combination of observable qualities (*features*) that serves either as a model for replication or an exemplar thereof, i.e., is not just an "uninteresting" random combination of such qualities. What is "interesting" is determined by the kind of pattern *classes* or categories considered; every pattern has an associated class of which the pattern is an exemplar.

Pattern *recognition* can be informally defined to mean the combined act of 1) detecting the presence of patterns in data (pattern *segmentation*) and 2) identifying the class which each detected pattern represents (pattern *classification*) [1]. In other words, pattern recognition answers two questions: *where* are there patterns in the set of observed data and *which* are the class labels of those patterns. Segmentation answers the *where* question and classification answers the *which* question.

A distinction is made between *supervised* and *unsupervised* pattern recognition. Unsupervised pattern recognition creates the pattern class specifications directly from the data "on the fly", identifies single-class segments and assigns them to the generated classes. In

[1] In case the data has already been segmented such that each observation is a single pattern which can be considered to be uniquely associated with a single class segment, pattern recognition is equivalent to pattern classification.

supervised pattern recognition, the pattern class specifications are already known and the problem is to segment and classify the observations so as to label each observation with the identity of some such predefined class.

The canonical pattern recognition system consists of three modules: preprocessing of measurement data, extraction of features from the preprocessed data and the recognition of patterns in the multidimensional space spanned by the feature variables (*feature space*) [28] [120]. These modules and their roles in audio signal recognition applications are discussed more fully below.

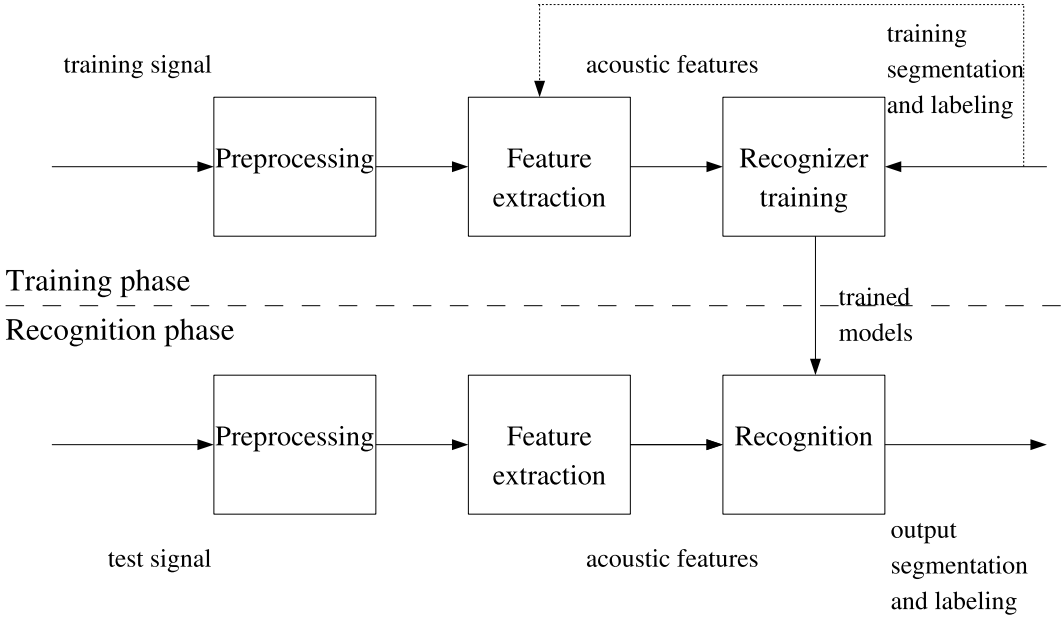


Figure 1.1: A general diagram of a supervised audio content recognition system.

Figure 1.1 shows a diagram of a supervised audio classification/recognition system. The system is first trained in the training phase, shown in the upper half of the diagram. The actual use of the system is called the recognition phase, shown in the lower half of the diagram. In unsupervised recognition, there is no separate training phase nor labeled training data. Instead, the training and recognition phase are equivalent in unsupervised pattern recognition, as the pattern class descriptions themselves are formed in the same process in which their presence in the data is detected. When “training” is mentioned below, it should be taken to refer generally to the process used to define the pattern classes, whether in the training phase of supervised pattern recognition or in the course of unsupervised pattern recognition.

Preprocessing is a set of operations performed on a "raw" digital signal. Preprocessing does not change the essential representation of the data, only some of its qualities. Such operations may include, for example, sampling rate conversion, filtering by a constant *pre-emphasis* filter, as well as *enhancement* operations.

Feature extraction is used to convert the digital signal into a sequence of acoustic *feature vectors*. This requires the feature extraction module to segment the digital signal in time into successive blocks such that each segment block can be associated with a single pattern class (while any "true" class segment can be represented by more than one feature vector; this is called *oversegmentation*). Feature vectors are typically computed from signal blocks of *constant length* short enough to make the "single class" assumption valid; the feature vectors are sampled using a constant interval (often in the order of milliseconds). In this case, oversegmentation is drastic and the task of segmentation is essentially left to the recognition module.

Feature extraction may also begin by explicit segmentation of the signal into presumable single-class segments, after which one *segmental* feature vector is produced to describe each segment. These segmental feature vectors are sampled nonuniformly in time. In this case, the recognition module can concentrate more on pattern classification, although some oversegmentation may still be present and additional *smoothing* of the class decisions may have to be performed in order to eliminate suspiciously short (and potentially erroneously labeled) class segments.

Feature extraction has the important task of reducing the dimensionality of the data in order to avoid the effects of the *curse of dimensionality* [28] which is an ubiquitous problem in pattern recognition. The curse of dimensionality refers to the fact that the more high dimensional is the feature space in which the recognition module must try to model pattern class boundaries, the more training data is required. The amount of required training data grows linearly as a function of unit volume in the feature space and thus exponentially as a function of the number of features [120]. If the curse of dimensionality would not have to be dealt with, feature extraction might not be necessary as patterns could be identified directly from the (preprocessed) digital signals.

Feature selection is an important issue which must be addressed in designing the feature extraction module. It refers to deciding which features to include in the feature vector representation. Often, the features are selected using a combination of domain knowledge and experimentation. Objective methods for feature selection exist and can be applied to a basic set of features selected by the domain expert [44]. Because of the curse of dimensionality, it is important to limit the number of features to those that help in the classification. The way to accomplish this is not to simply aim for minimal correlation among the features, however, because in such process important class-discriminatory information will typically

be lost. While it is true that in a pair of two features with a correlation coefficient of unity, the other feature is completely redundant, it is not true that high correlation between two features automatically renders one of them useless. As stated in [44], very high variable correlation or anti-correlation does not mean absence of variable complementarity in terms of pattern class discrimination. In fact, since in reality it is hard to come by features that are truly orthogonal and at the same time helpful in separating the desired pattern classes, it is actually likely that a feature that is highly correlated with another feature having good class discrimination power is itself also a good discriminant between the same classes - maybe offering some additional useful information. Moreover, it is stated in [44] that a variable that is completely useless by itself can provide a significant performance improvement when taken with others. This could not happen either if the variables were required to be completely orthogonal.

When the patterns belonging to different classes form distinct, well-separated *clusters* in the feature space defined by the selected features, pattern recognition is easy. For example, if the classes form a clear clustering in a one-dimensional space spanned by a single feature, classification reduces to simple thresholding of the feature value. On the other hand, if the classes are badly overlapping in the feature space, it is often very hard to come by with a classification solution that will separate them. Firstly, this highlights the importance of generating enough good features. The class discriminatory information that is lost in the feature representation must be substituted by prior knowledge in the pattern classification stage, if the classes are to be separated. Secondly, if a clustering structure is desired in order to make pattern recognition easier, the features should be carefully selected because including irrelevant features is likely to destroy simple clustering structures that could have been found with a more compact set of good features [68]. All the discussed factors make feature selection necessarily a compromise between information preservation and parsimony.

Pattern classification could be conceptually viewed as regression of the feature vectors on a binary vector with as many elements as there are classes in the classification problem. Each element in this binary vector is 1 if the feature vector pattern belongs to the corresponding class and 0 otherwise. In fact, many classifier structures do contain separate regression modules for each class that regress the feature vectors on a continuous-valued variable (which can be probabilities, for example). The outputs of these regressors are then compared to achieve the class decision (corresponding to the binary vector). On the other hand, the continuous-valued regression outputs can be viewed as features themselves. From yet another viewpoint, the binary vector itself could be viewed as a feature vector fed to a completely trivial pattern classifier (which just determines the location of the value 1 in the binary vector), or it could act as a feature vector for a higher-level recognizer dealing with different classes. As Duda *et al* [28] point out, there are no particular theoretical reasons

for making a strict division of processing steps into feature extraction and classification. The purpose of both tasks is to extract information. In complex solutions with several processing stages, it may be hard to draw a single boundary between the feature extraction and classification stages, as this division depends on the perspective.

1.2 Terminology and Notation

To make the discussion easier to follow, certain conventions have been adopted for both terminology and notation, with an attempt to follow these guidelines whenever possible. Table 1.1 lists and defines some of the recurring terminology. Table 1.2 lists mathematical symbols used throughout to refer to specific things, together with the meanings.

Table 1.1: Glossary of central pattern recognition terminology as used in this study.

time series	a time signal; a discretely sampled sequence of scalar or vector observations (e.g., a digital audio signal or a *sequence* of feature vectors computed thereof)
segmentation	the act or outcome of dividing a body of data into smaller segments spatially or temporally
... of a time series	*temporal* segmentation of a time series
classification	the act or outcome of assigning segments of data into categories *)
supervised cl.	classification with predefined categories
unsupervised cl.	classification without predefined categories
pattern recognition	the combined act of segmentation and classification **)
unsupervised p. r.	the combined act of segmentation and *unsupervised* classification
transcription	(supervised) time series pattern recognition whose main purpose is to produce a class label sequence that somehow describes the time series
training	a procedure for setting the parameters of models used in supervised pattern recognition
training data	data used for setting the model parameters during training
testing	the act of evaluating a pattern recognition system
test data	data used in the evaluation; in general, a test data set should be completely disjoint from the training data set
change detection	time series segmentation that is independent of any classification and is related to changes in the short-time statistics of the signal
template	a model of the time evolution of feature vectors in a feature vector sequence
smoothing	the act of eliminating abrupt changes in a continuous feature or in an inferred sequence of class labels

*) note that the classification of observations always produces a segmentation, if one does not exist

**) segmentation may be performed independently and used to constrain the classification

Table 1.2: Some constant notation used in this thesis

N	number of observations or signal samples
F_s	the sampling frequency of a digital audio signal
d	dimensionality of feature vectors in pattern recognition
K	the number of points in a discrete Fourier transform *or* the number of clusters in unsupervised classification
L	the number of retained spectral or cepstral samples
J	the number of components in a mixture data model
M	the number of classes in pattern recognition
p	order of an autoregressive model
s_n	a digital audio signal (nth sample)
$S(z)$	the z transform of s_n
S_k	a discrete Fourier transform of s_n
c_k	a cepstrum
$\boldsymbol{c}_n$	a truncated cepstral vector $\boldsymbol{c}_n = (c_{n,1} \dots c_{n,L})^T$
$\boldsymbol{x}_n$	a feature vector with d features $\boldsymbol{x}_n = (x_{n,1} \dots x_{n,d})^T$
X	a set or sequence of feature vectors, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$
$X_{n_1}^{n_2}$	a sequence of feature vectors, $X_{n_1}^{n_2} = \{\boldsymbol{x}_{n_1}, \boldsymbol{x}_{n_1+1}, \dots, \boldsymbol{x}_{n_2}\}$
$\omega_1, \dots, \omega_M$	sound class identifiers
λ	a data model; especially a model of a probability density function
q_n	the identity of the mixture component that generated the nth observation, $q_n \in \{1, \dots, J\}$
$u_n(k)$	a classification where $u_n(k) = 1$ if $\boldsymbol{x}_n$ represents class ω_k *or* a clustering where $u_n(k) = 1$ if $\boldsymbol{x}_n$ belongs to the kth cluster
y_n	a class/cluster labeling with $y_n = \mathrm{argmax}_k\, u_{n,k}$
$\boldsymbol{\mu}$	the mean vector of a multivariate Gaussian distribution
$\boldsymbol{\Sigma}$	the covariance matrix of a multivariate Gaussian distribution
$\boldsymbol{\theta}_k$	cluster prototype of the kth cluster in a cluster model
$b_i(\boldsymbol{x})$	the probability distribution associated with component i of a mixture model or a hidden Markov model

Chapter 2

Auditory Perception

The human auditory system consists of two fundamentally different processing regions: the *peripheral* and the *neural* region. Auditory processing begins in the periphery, i.e., the ear, and continues on the neural level from the cochlea (in the inner ear) through several nuclei until the *auditory cortex*. There is also feedback from the higher neural stages back to lower level nuclei and to the cochlea.

The two main approaches to modeling auditory perception, or parts thereof, are the *physiological* and the *psychoacoustical* approach. Models of the former kind are derived from explicit physiological and anatomical knowledge, while the latter approach produces *functional* models derived from subjective listening tests. The anatomy and physiology of the auditory periphery is rather well known, making detailed physiological auditory models possible. Due to lack of knowledge of the details of neural auditory processing, the principal method of modeling the behavior of the higher stages is psychoacoustics. Due to their relative simplicity, psychoacoustical models are also frequently used for modeling the periphery.

The psychoacoustical approach involves partial modeling of human perception. At higher psychoacoustical levels, the perception of sounds can be divided into different components: *loudness*, *pitch*, *timbre* and *subjective duration* [132] [105]. Of these three, fairly good physical correlates can be found, in the properties of physical sounds, for loudness (sound intensity), pitch (fundamental frequency) and subjective duration (true duration). Timbre, however, is a collective name for other perceptual aspects for which no simple one-dimensional physical correlate can be found [132] [17] [66].

2.1 Peripheral Auditory Processing

Several important psychoacoustical phenomena have their origins in the auditory periphery. These include the hearing threshold in quiet, the loudness sensation, the frequency masking phenomenon and the nonuniform frequency resolution of hearing (modeled by, e.g., the concept of critical bands). Anatomically, the auditory periphery can be partitioned into the outer ear, the middle ear, and the inner ear (which includes the cochlea). Significant nonlinear signal processing occurs already in the periphery, mostly in the cochlea. Figures 2.1 and 2.2 show a drawing and a simplified diagram, respectively, of the peripheral auditory system.

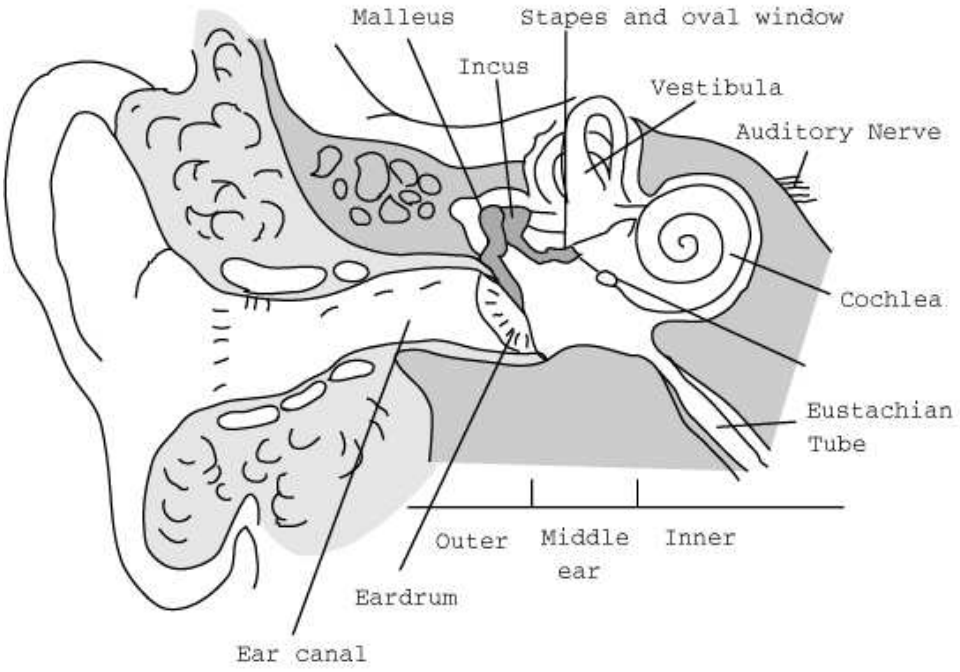


Figure 2.1: The structure of the peripheral auditory system [66].

The outer ear consists of the *pinna* and the *ear canal* and is separated from the middle ear by the *eardrum*. The pinna is relevant for localization of sound on the front-back axis; by its asymmetric shape, it makes the ear more sensitive to sounds coming from in front of the listener than to those coming from behind and thus causes these directions to be perceived differently. The ear canal can be approximated as a simple acoustic tube open at one end (pinna) and closed at the other (eardrum). The canal in an adult is between 2 and 3 cm in length and about 0.7 cm in width [90]. Acting as a quarter-wavelength resonator, it amplifies energy in the range around 4 kHz [132].

The eardrum marks the beginning of the middle ear, an air-filled cavity of about 6 cm^3

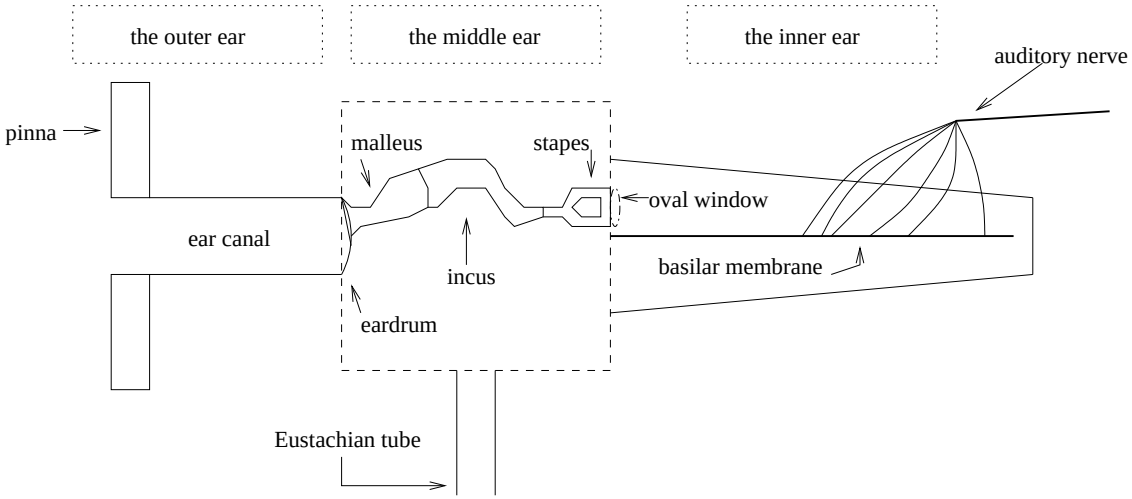


Figure 2.2: A simplified diagram of the human ear (after [66]). The cochlear part has been drawn straight instead of coiled.

in volume that contains the *ossicular bones*: *malleus* (hammer), *incus* (anvil), and *stapes* (stirrup). Their function in the hearing mechanism is to linearly transmit eardrum vibrations to the *oval window* membrane of the inner ear while also doing an acoustic impedance transformation. The liquid medium in the inner ear has about 4000 times higher an acoustic impedance than the air medium in the outer ear. The transformation is based primarily on the large area difference between the eardrum (about 65 mm^2) and the stapes (about 3 mm^2 [66]), but secondarily also on the lever action of the ossicular bones. The impedance match is almost perfect in frequencies around 1 kHz [132]. The middle ear filter contributes, together with the resonance of the ear canal and shadowing and reflection due to the head and the shoulders, to make the human hearing particularly sensitive to frequencies between approximately 1 kHz and 5 kHz. Collectively, these mechanical filtering effects are reflected in two important frequency-dependent psychoacoustical phenomena: the absolute threshold of hearing in quiet and the *equal loudness* sensation. The absolute threshold of hearing refers to the lowest sound pressure level (SPL) in which a tone is audible at each frequency. A functional approximation of the threshold for pure tones in quiet as a function of frequency, pertaining to young listeners with acute hearing [92], is given by [119]

$$L_{TH}(f) = 3.64\left(\frac{f}{1\text{ kHz}}\right)^{-0.8} - 6.5e^{-0.6(f/(1\text{ kHz})-3.3)^2} + 10^{-3}\left(\frac{f}{1\text{ kHz}}\right)^{4}\text{ dB SPL} \tag{2.1}$$

The *equal loudness level contours* for pure tones are specified in the standard ISO-226 [1]. These contours give, for each frequency of a tone and for the *loudness level* associated

with each contour, the SPL at which the tone should be heard in order to sound equally loud as a 1000 Hz tone heard at the SPL equal to the loudness level. Figure 2.3 shows the hearing threshold according to Eq. 2.1 and the equal loudness contours for eight loudness levels according to [1], implemented in [118].

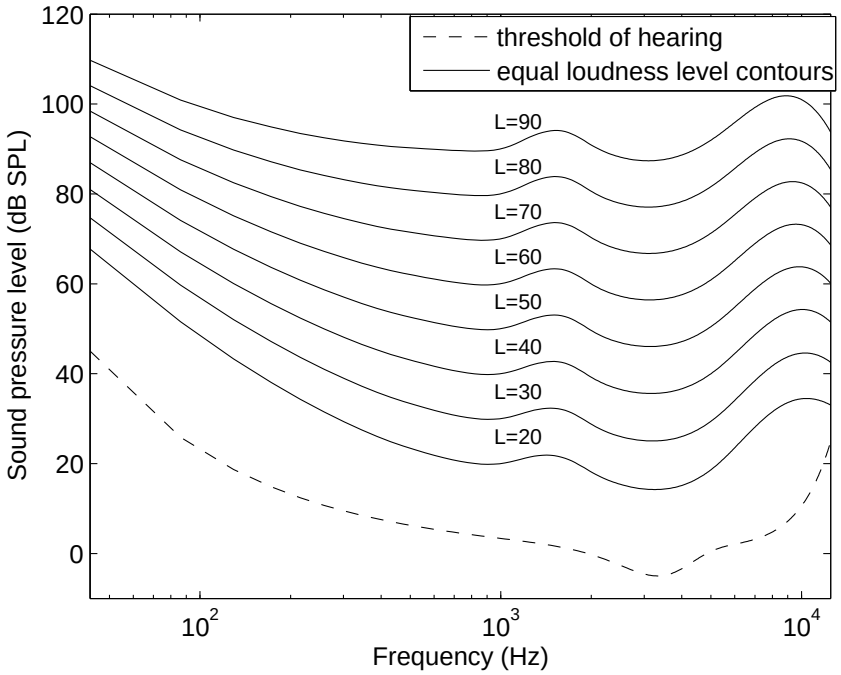


Figure 2.3: Psychoacoustical models of the absolute threshold of hearing [119] and the equal loudness level contours (for eight loudness levels between 20 and 90 dB) [1] [118].

The *cochlea*, located in the inner ear, is a coiled tube that forms two and a half turns [132]. It is filled with two different fluids and consists of three channels (or scalae) which run together from the base to the apex. Sound waves enter the cochlea from the middle ear via the oval window, causing vibrations of the fluid called *perilymph* located in the *scala vestibuli*. These vibrations are transferred through the very thin and light *Reissner's membrane* to the fluid called *endolymph* located in the *scala media*. The vibratory motion is transmitted through the endolymph to the *basilar membrane*, along which it proceeds as traveling waves of vertical displacement of the membrane. A traveling wave begins with small amplitude near the oval window, grows slowly, reaches its maximum at a certain location and then rapidly dies out in the direction of the apex. The basilar membrane varies gradually in width and density along its length of approximately 32 mm [132]. In the beginning (near the oval window and the *round window*) it is narrow and stiff but near the

apex it is compliant and massive. Each location on the basilar membrane responds differently to sounds in different frequencies. High frequency traveling waves in the inner ear resonate near the beginning of the basilar membrane, while low frequencies travel across the membrane and resonate in the apex end. The inner ear thus performs frequency separation based on resonance location on the basilar membrane. The resolution of this spectrum analysis is best at low frequencies. This nonuniform frequency resolution is reflected in psychoacoustical scales, such as the *Bark* (*critical band*), *ERB* and *mel* scales [66].

On the basilar membrane lies the *organ of Corti* which contains about 30000 sensory *hair cells*, arranged in several rows along the length of the cochlea and the basilar membrane. Basilar membrane vibration in some location causes the affected hair cells to send electrical impulses up the neural fibers of the *auditory nerve*. By measuring the electrical signals traveling in these fibers, it has been found that the voltage spikes corresponding to stimulation of the hair cells are closely correlated with the mechanical vibration pattern on the basilar membrane up to frequencies of about 4 or 5 kHz [105]. The *spreading* effect, which manifests itself in this *excitation pattern* of the basilar membrane, gives rise to the psychoacoustical phenomenon of *frequency masking*. The excitation pattern also has a clear connection to the loudness perception [132].

A classical and fundamental concept in the study of hearing and sound perception is that of a *critical band* related to the frequency resolution of hearing. A critical band defines a frequency range in psychoacoustic experiments for which perception abruptly changes as a narrowband sound stimulus is modified to have frequency components beyond the band. When two competing sound signals pass energy through a critical-band filter, the sound with higher energy within the critical band dominates the perception and *masks* the other sound. Critical bandwidths can be measured by various slightly different psychoacoustic tests [66] [90]. Below 500 Hz, critical bandwidth is roughly constant at about 100 Hz. For higher frequencies, it increases proportional to the center frequency (roughly logarithmically above 1 kHz) reaching bandwidths of 700 Hz when the center frequency is near 4 kHz. An analytic expression for mapping the frequency f into critical-band rate z, or the *Bark scale*, is [90]

$$v = 13\tan^{-1}(0.76f/\text{kHz}) + 3.5\tan^{-1}(f/(7.5\text{kHz}))^2 \tag{2.2}$$

Another, less precise approximation for the Bark scale is [66]

$$v = 7\ln\left(f/650\text{Hz} + \sqrt{1 + (f/650\text{Hz})^2}\right) \tag{2.3}$$

with inverse [66] [108]

$$f/\text{Hz} = 650\sinh(v/7). \tag{2.4}$$

Of practical importance is also the *mel scale*, which can be approximated by [66]

$$B(f) = 2595 \log_{10}(1 + f/700) \tag{2.5}$$

The inverse of the mel scale is thus given by

$$B^{-1}(b) = 700(10^{b/2595} - 1) \tag{2.6}$$

Both mappings, the Bark scale and the mel scale, are shown graphically in Figure 2.4. The frequency resolution of these auditory scales is obviously nonuniform; dv/df and db/df are much greater, meaning better frequency resolution, at low frequencies than at higher frequencies.

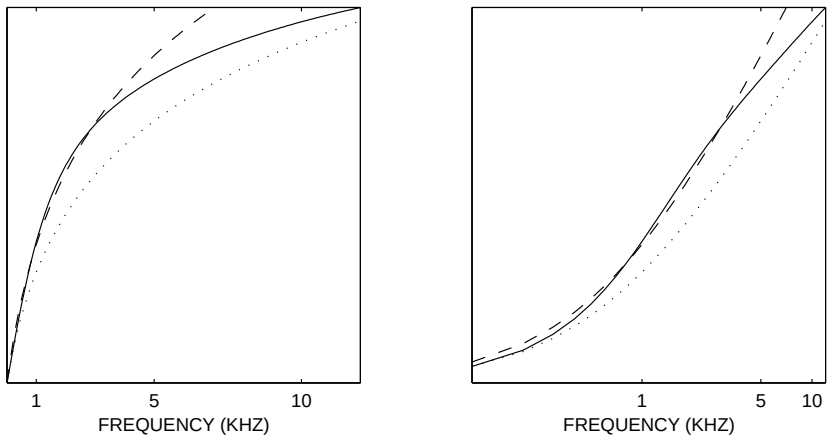


Figure 2.4: The Bark scale (solid line) and the mel scale as a function of frequency, with the mel scale mapping scaled so that one Bark corresponds to 120 mel (dashed line) and 150 mel (dotted line). The frequency axis is linear in the left panel and logarithmic in the right panel.

The spreading of excitation on the basilar membrane can be modeled by first estimating the power spectrum of the sound signal, converting it to a critical band power spectrum and convolving the result with a *spreading function*. Johnston [64] used this approach in his model for frequency masking. One popular model for the spreading of excitation across critical bands applies to intermediate SPLs. It is given by [108]

$$10 \log_{10} B(v) = 15.81 + 7.5(v + 0.474) - 17.5\sqrt{1 + (v + 0.474)^2}\text{dB}. \tag{2.7}$$

Another possible spreading function model, that depends on the narrowband masker level and center frequency, is given by [119]

$$10\log_{10} B(v) = \begin{cases} 27v & v \leq 0 \\ (-24 - 0.23\text{kHz}/f_c + 0.2L_c/\text{dB})\, v & v > 0 \end{cases} \tag{2.8}$$

In Eq. 2.8, f_c is the masker center frequency and L_c is the masker level. Figure 2.5 shows spreading functions according to Eqs. 2.7 and Eq. 2.8, the latter with three different values for SPL and a masker center frequency of 1 kHz. The SPL-dependent curves display the phenomenon known as *upward spread of masking* [85]; the spreading of excitation towards higher frequencies increases particularly strongly with increasing SPL. Thus, although the increase of the level of a narrowband masker will increase masking both below and above the center frequency of the masker, the higher frequencies will be masked more and more *in proportion to* the lower frequencies as the sound level increases.

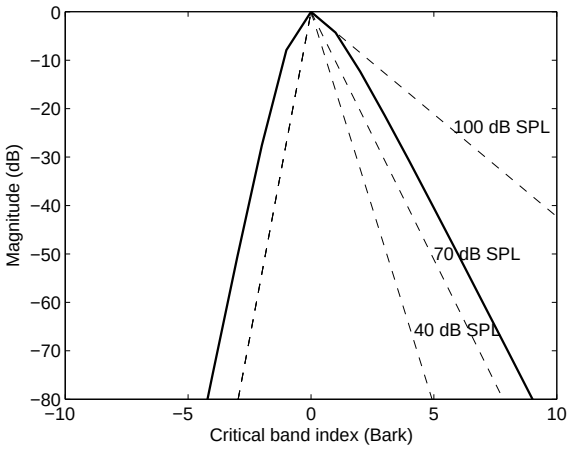


Figure 2.5: Excitation spreading function for intermediate SPL according to Eq. 2.7 (solid line) and for SPL levels 40 dB, 70 dB and 100 dB, with a center frequency of 1 kHz, according to Eq. 2.8 (dashed line).

The excitation pattern on the basilar membrane (a kind of an auditory spectrum) can be modeled as follows [66]:

1. Estimate the power spectral density $\hat{P}(f)$ of a signal using some power spectrum estimation method (such as the periodogram method; see Section 3.3.1).

2. Map the power spectral density onto the Bark scale using the formula

$$P'(v) = P(f(v))\frac{df}{dv}, \tag{2.9}$$

where $f(v)$ is the conversion function from the Bark scale to the frequency/Hertz scale, e.g. Eq. 2.4, and df/dv is its first derivative that represents the change in spectral density caused by the mapping.

3. Compute the Bark scale excitation pattern by convolving the Bark scale power spectral density with a spreading function, e.g. the one given by Eq. 2.7:

$$E(v) = P'(v) \star B(v) \tag{2.10}$$

From this representation, it is possible to further compute the *loudness* of a steady-state sound as follows [66]:

1. Compute the *loudness density* or *specific loudness* [132] by

$$L'(v) = cE(v)^{0.23}, \tag{2.11}$$

where c is a constant intended to be chosen such that a 40 dB SPL tone at 1 kHz will correspond to 1 sone (the unit of loudness).

2. Compute the loudness by integrating the loudness density over the whole range of hearing (24 Bark):

$$L = \int_0^{24\text{Bark}} L'(v)dv \tag{2.12}$$

To obtain a more accurate estimate of the excitation pattern or loudness, weighting by a suitable equal loudness function can be applied to any intermediate representation.

Using largely the same computational steps as when computing an estimate of the loudness, it is also possible to model a perceptual quality called *sharpness* [132]. Sharpness, which has been found to be an important component of the timbre perception, is a sensation which human listeners can consider separately. It is possible for listeners to subjectively compare the sharpness of one sound with the sharpness of another. A model for sharpness is obtained using the loudness density as

$$S = 0.11 \frac{\int_0^{24\text{Bark}} L'(v)g(v)vdv}{\int_0^{24\text{Bark}} L'(v)dv}, \tag{2.13}$$

where the weighting function $g(v)$ stays constant at unity up to about 16 Bark, after which it starts to rise nonlinearly with an increasing slope, reaching a value of four at 24 Bark [132].

The loudness model discussed above applies to steady-state sounds. At the onset of such a sound, however, the full loudness is not perceived instantly. Instead, the loudness reaches its maximum value at about 200 ms after the onset of the sound. This phenomenon seems to reflect some kind of integration process and is known as *temporal integration* of loudness. From the two kinds of integration associated with loudness - spectral integration modeled by Eq. 2.12 and temporal integration - spectral integration of excitation is thought to precede temporal integration, so that the latter takes place in more central areas of the auditory system [132].

Besides temporal integration, another important temporal effect is *post-stimulus masking* or *postmasking* (also known as *forward masking*). It refers to increased masking threshold after the end of a masking sound (a *masker*). Its main properties can be summarized as follows [85] [63]:

1. The amount of postmasking in dB is a decreasing linear function of $log(\Delta t)$, where Δt is the time delay between the current time and the end of the masker.

2. The rate of recovery from postmasking is greater for higher masker levels, so that regardless of the initial amount of forward masking, the masking will decay to zero after 100-200 ms.

3. The *growth of masking* for postmasking, i.e., the rate at which the amount of masking increases with increasing masker level at a certain delay, is less than unity, so that increments in masker level do not produce equal increments in the amount of postmasking.

4. The amount of postmasking increases with increasing masker duration for durations up to at least 20 ms, possibly even up to 200 ms.

The physiological mechanisms underlying postmasking are not clear. It could be explained in terms of residual neural activity or, for small intervals, by the response decay of the basilar membrane [85].

Pre-stimulus masking (also known as *backward masking*) refers to increased masking threshold *before* the start of a masker. However, in comparison to postmasking, premasking is much shorter in effect and greatly affected by training of the test subjects, so that practiced subjects often show little or no premasking [85].

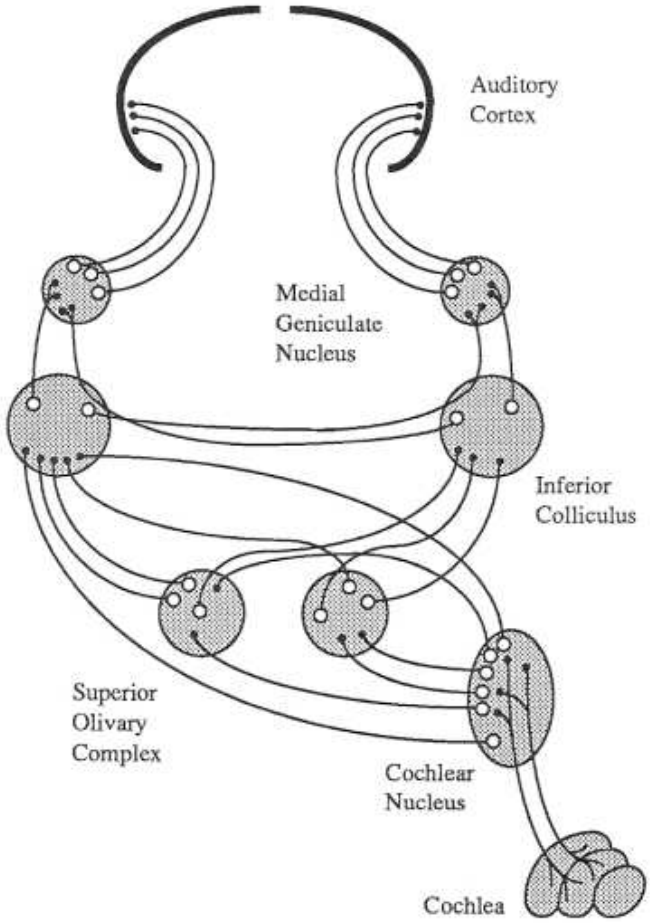


Figure 2.6: A simplified illustration of the principal nuclei and fiber pathways of the central auditory system. Only pathways from the right ear are shown. Neurons are illustrated with small open circles, while synaptic connections are displayed as small filled circles. From [86].

2.2 Neural Auditory Processing

Higher, neural stages of the auditory pathway ascend from the hair cells of the cochlea to the *auditory cortex*. Figure 2.6, taken from the book by Morgan and Scofield [86], shows a simplified illustration of the neural auditory pathway, starting from the cochlea. Compared to peripheral auditory processing, neural auditory processing is rather poorly understood. A concise overview of the neural auditory pathway in mammals is included in [86]; [85] discusses the existing knowledge of the human auditory system in more detail. In

general, higher level nuclei tend to generate progressively more sophisticated and longer-term representations of the auditory sensation while also acting as relay stations for lower-level representations produced by the periphery and the lower neural stages. *Tonotopic* or *cochleotopic* [85] organization, where the neurons are organized by their characteristic frequencies so as to preserve the innervation pattern of the basilar membrane, is found throughout the mammalian auditory system [86].

The auditory nerve communicates hair cell outputs from the cochlea to the *cochlear nucleus*, which contains neurons with different time responses: primarylike, onset, chopper, pauser and buildup. Many different kinds of abstractions of the original stimulus are generated already in specialized regions of the cochlear nucleus. The initial site of bilateral representation of the acoustic environment is found in the *superior olivary complex*, known to play a key role in the localization of sound sources. The *inferior colliculus* receives bilateral input from the superior olivary complexes and contralateral input directly from the cochlear nuchleus; the exact role of the inferior colliculus is not known, but it has been proposed that it is specialized for the representation of pitch and for localizing sound sources which consist of complex temporal variations. The cells of the inferior colliculus display *modulation frequency selectivity*; they phase-lock to amplitude modulations of the stimulus[1]. The final waystation on the way to the auditory cortex is the *medial geniculate nucleus* (MGN). Like the cells in the inferior colliculus, cells in the MGN also phase-lock to amplitude modulation, albeit with poorer temporal resolution (lower modulation frequencies being represented). Descending input from the cortex may also play an important role in the MGN. It has been suggested that in addition to its role as a relay station for an auditory pathway conveying all the information necessary to characterize acoustic events, the MGN is also involved in a second pathway that allows the auditory cortex to selectively label stimuli with perceptual qualities; it would thus play an essential role in the perception of the acoustic environment.

Psychoacoustically, high-level auditory processing is manifested by phenomena covered by Bregman in his book *Auditory Scene Analysis* [17]. According to Bregman, the auditory system organizes incoming auditory evidence (mostly mutually exclusively) into separate perceptual representations termed *auditory streams*. Distinct auditory streams tend to present distinct environmental sounds. The mechanisms of the formation of auditory streams appear to be rather well described by analogies to the grouping principles of *Gestalt psychology*, which was evolved by a group of German psychologists in the early 20th century to explain the organization of perception. According to Gestalt theory, this organization

[1]This responsiveness to long-term temporal variations is important for speech communication, among other things; reflecting the syllabic and phonetic temporal structure of speech, the modulation spectrum of speech is dominated by components between 2 and 8 Hz [54] [41].

is governed by a competition of the "forces of attraction" between sensory elements. The perceptual result of this competition is the formation of *patterns* (*Gestalt* in German) in our experience as a consequence of the net effect of all the said forces (instead of our experiencing any kind of simple "summation" of the properties of the individual elements taken in isolation). According to the Gestalt psychologists, it was impossible to perceive sensory elements without their forming an organized whole; they argued that the tendency to form such patterns was innate and an automatic tendency of the brain tissue [17].

While the grouping principles were originally meant to be common to different sense modalities [17], they are most often encountered in describing visual perception. Auditory analogies to the Gestalt grouping principles presented by Bregman include:

- Principle of proximity: the closer visual elements in a set are to one another, the more strongly we tend to group them perceptually. In auditory perception, proximity of sound elements in time and/or frequency favors their being grouped into the same auditory stream; i.e., time and frequency are auditory correlates of the spatial dimensions in vision.

- Principle of similarity: visual elements with similar quality tend to be grouped together. In audition, sounds with similar timbre are more likely to be grouped into the same stream. Although timbre is necessarily a multidimensional quality whose best characterization may be some kind of auditory spectrum, *brightness* [2], which is "roughly speaking ... the balance of high and low partials in the spectrum" and "very much like the frequency that is at the mean of the distribution" [17], has been identified as one very probable physical constituent of the timbre perception.

- Principle of closure: a mechanism of completing evidence with gaps in it, created by partial occlusion. In audition, completing evidence from a sound temporarily masked by another sound to infer its continuation during the masked segments.

- Principle of common fate: visual elements moving together are perceived as a group. In audition, this refers to the grouping of different parts of the spectrum that change in the same way at the same time. The said change in the spectral components can be frequency modulation (common change in the center frequencies) or amplitude modulation (common change in the amplitudes of the components).

According to Bregman, the Gestalt grouping principles can be viewed as a kind of heuristics that combine their effects, much like voting, to aid the auditory system in decomposing a mixture of sound into separate perceptual entities corresponding to different real-world

[2] Brightness appears to be very closely related to sharpness [132] discussed in Section 2.1.

events. Bregman divides auditory grouping in auditory stream formation into sequential (temporal) and simultaneous (spectral) grouping. He also argues that there are two ways of acquiring skills for auditory stream segregation: “primitive segregation” mechanisms are innate, while ”schema-based segregation” is based on *schemas* that are learned. Schema-based segregation probably involves the learned control of attention.

Considering the principle of similarity and the role of the different dimensions of timbre, brightness appears to be important for stream segregation. According to Bregman, it is unclear whether any of the possibly more elaborate acoustic dimensions related to the timbre perception are used in *primitive* segregation at all [17].

In reviewing the research on human speech recognition performed by Fletcher and his colleagues, Allen [4] presents certain conclusions that are in line with Bregman’s findings. In particular, compared to across-frequency processing (spectral templates), which is central in most automatic speech recognition systems, human recognition of speech is claimed to rely more on across-time processing, with only local coupling across frequency. This statement is reminiscent of auditory stream formation by the principle of proximity in time and frequency.

Chapter 3

Signal Processing for Audio Analysis

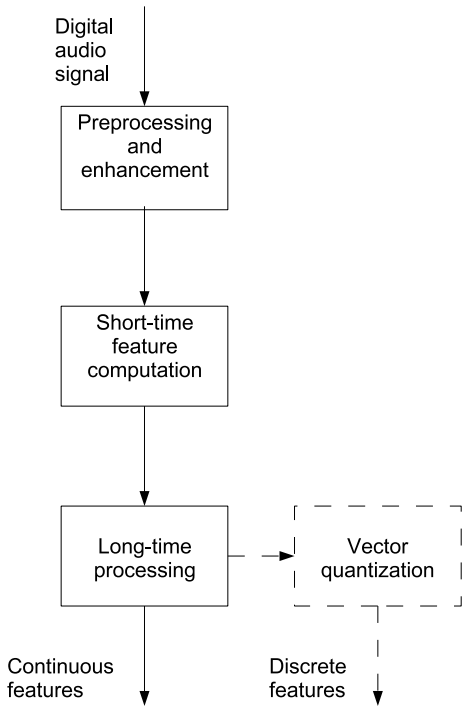


Figure 3.1: The data flow of a typical acoustic feature extractor.

Automatic recognition and classification of audio content is a pattern recognition problem. The solutions thus generally contain implementations of the three main modules of a canonical pattern recognition system [28]: 1) preprocessing, 2) feature extraction and 3) recognition (see Figure 1.1). This chapter focuses on the signal processing techniques commonly employed in the preprocessing and feature extraction modules, while the next

chapter discusses pattern recognition methods that are applicable in the recognition module.

Figure 3.1 shows the general data flow diagram of audio preprocessing and feature extraction. The digital audio signal, sampled at some sampling frequency F_s, is converted into a sequence of feature vectors via the processing stages shown in the figure.

3.1 Approaches to Short-Time Processing

Short-time processing is typically based on dividing the signal into frames, i.e., signal segments of short constant length spaced at equal time intervals. Let us denote the size of the frame (in signal samples) by N, so that the length of the frame in time is N/F_s seconds. Also, let us denote by M the amount of samples by which corresponding points of two successive frames differ. This quantity, sometimes called the *shift interval* or the *step size*, is typically smaller than the frame size N. With the frame size N given, the information contained jointly in N and M can equivalently be expressed by replacing M with one of two alternative framing parameters, namely *frame overlap* or *frame rate*. The overlap $N - M$ specifies the amount of samples that two successive frames have in common. The frame rate F_s/M specifies the number of frames per second.

Each frame of N samples is typically windowed using some window function w_n which is nonzero only when $1 \leq n \leq N$. If s'_m denotes the complete signal, a windowed frame starting at the mth sample is given by

$$s_{n,m} = \begin{cases} s'_{n+m-1} w_n & 1 \leq n \leq N \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

For notational convenience, the subscript m will later be omitted whenever discussing analyses that only have to deal with a single frame of data. The window function w_n can be, for example, a rectangular window or a Hamming window. The choice of the window function has an effect on the short-time features computed from the windowed frames [102]. In practice, some processing methods, such as the discrete Fourier transform and the autocorrelation method of linear prediction, benefit from using a frame window function whose edges taper close to zero (e.g., the Hamming or Hann window), so as to avoid strong discontinuities at the edges of the frames [102].

Frame-based processing has two distinct applications within the scope of this study: audio enhancement and feature extraction. In audio enhancement, the recorded signal r_n is considered to be corrupted according to a certain corruption model and the goal is to extract the desired audio s_n from the corrupted signal r_n. This is typically performed by dividing the signal into sufficiently short frames, transforming each frame into a spectral representation and processing each frame spectrally. Optionally, the processed spectral rep-

resentations can be transformed back into the time domain and combined using overlap add resynthesis [102]. The resynthesis is not necessary, however, if the goal is to extract information only from the magnitude spectrum. Two basic methods of audio enhancement, applicable in the spectral domain, are briefly discussed in Section 3.2.

In feature extraction, the goal is to transform the signal s'_n into a sequence of feature vectors. In this case, each frame $s_{n,m}$ is typically associated with one feature vector. The feature vector is usually computed primarily from the associated frame, but may also contain information from neighboring frames (long-term processing, see Section 3.6). Alternatives to frame-based feature extraction include sampling the outputs of a time-domain filterbank, such as an auditory spectrum analyzer, at specific time instants (e.g. at every M samples) as well as sampling the coefficients of adaptive filters (especially linear predictive filters) in a similar fashion. However, the frame-based approach, in which the signal is assumed to be locally stationary within each frame, is the most prevalent. In frame-based processing, various *short-time features* can be computed from the short frame $s_{n,m}$; these will be discussed in Sections 3.3-3.7.

3.2 Signal Preprocessing and Enhancement

Before a digital audio signal is subjected to automatic classification methods, it is often preprocessed in some manner in order to make the classification task easier. Preprocessing may include simple DSP operations such as sampling rate conversion. It is important to choose the correct sampling rate (a practical upper bound being, of course, the original recording rate). Too high a sampling rate may place unnecessarily high requirements on the feature extraction module, as a large portion of the information it receives is irrelevant for the goals of the automatic classification. Too low a sampling rate will make the antialiasing cutoff frequency so low as to discard useful information from the higher frequencies. Signal preprocessing may also include filtering with a fixed *pre-emphasis* filter; in speech applications, it is typical to use a first-order high-pass FIR filter of the form [90]

$$H_{\text{pre}}(z) = 1 - \alpha z^{-1}, \tag{3.2}$$

where $\alpha < 1$, to emphasize the higher frequency formants in speech.

The preprocessing stage may also contain *enhancement* operations which attempt to remove certain types of distortion from the signal. The distortion may be acoustic (environmental) noise or channel noise. If the noise is additive and mostly stationary, so that its statistics can be reliably estimated during a segment known to contain just the noise, two well-known methods for signal enhancement are applicable: *spectral subtraction* and *Wiener filtering*.

Spectral subtraction [14] is a straightforward method for signal enhancement. The basic assumption is that the desired clean signal s_n has been corrupted by additive noise v_n:

$$r_n = s_n + v_n \tag{3.3}$$

Taking z transforms [89] of both sides gives

$$R(z) = S(z) + V(z) \tag{3.4}$$

The basic idea of spectral subtraction is to estimate $S(z)$ (and thus s_n) so that the magnitude of the estimator is given by

$$|\hat{S}(z)| = |R(z)| - N(z) \tag{3.5}$$

where $N(z)$ is an averaged magnitude spectrum of the noise $V(z)$,

$$N(z) = E\{|V(z)|\}, \tag{3.6}$$

which is to be estimated from segments known to contain only background noise. The phase of the estimator $\hat{S}(z)$ is given directly by the phase of $R(z)$.

If the problem is cast as that of filtering r_n to obtain $\hat{s}_n$, i.e., as $\hat{S}(z) = H(z)R(z)$, a filter $H(z)$ that realizes the above specifications is given by [14]

$$H(z) = 1 - \frac{N(z)}{|R(z)|}. \tag{3.7}$$

There may be cases where $N(z) > |R(z)|$ at some frequency. Such noisy frequencies are usually assumed to be unrecoverable; it is therefore desirable to ensure that the subtraction result stays nonnegative. This can be accomplished to using half-wave rectified filtering $\hat{S}(z) = H_r(z)R(z)$, where [14]

$$H_r(z) = \frac{H(z) + |H(z)|}{2} \tag{3.8}$$

Random residual spikes that remain in non-active temporal and spectral regions after subtraction give rise to tonal noise, which can be reduced by certain methods [14].

The basic principles of *magnitude spectral subtraction* have been described above; another alternative is *power spectral subtraction*, in which case the magnitude spectra in Eq. 3.5 are replaced by power spectra; see [57] for details.

Wiener filtering is another approach to noise removal. It minimizes the mean squared error of the filtered target signal [51]. The *noncausal Wiener filter* is obtained using power

spectra of the desired signal and the noise. Such a filter is given by [57]

$$H(z) = \frac{|S(z)|^2}{|S(z)|^2 + |V(z)|^2}. \tag{3.9}$$

Unfortunately, this filter can not be realized unless the power spectra of both the desired signal s_n and the noise v_n are known; at the same time, in recognition applications, wanting a better estimate of the magnitude or power spectrum of the signal s_n is often the reason for performing enhancement filtering in the first place. However, Eq. 3.9 has applications in cases where reasonably accurate estimates of these quantities can be generated by some means.

3.3 Representations of the Short-Time Magnitude Spectrum

Very often, feature vectors in speech/audio classification are based on the short-time magnitude spectrum. This is well justified by the way in which the auditory periphery works in humans and other mammals; the basilar membrane of the inner ear is a kind of a spectrum analyzer, albeit with a frequency-dependent frequency resolution. This section reviews tools of spectral modeling that are central from the perspective of speech/audio recognition: the discrete Fourier transform (DFT), linear predictive coding (LP) and the *cepstral* representation of the magnitude spectrum.

3.3.1 Discrete Fourier Transform (DFT)

Fourier or spectral analysis is an important mathematical tool. The continuous Fourier transform of a continuous signal $h(t)$ and the corresponding inverse transform are given by

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt \tag{3.10}$$

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df \tag{3.11}$$

where t generally denotes time and f cycles per unit time, i.e., frequency [1].

For discretely sampled sequences of finite length N, indexed from 0 to $N-1$, the discrete Fourier transform (DFT) is given by

$$H_k = \sum_{n=0}^{N-1} h_n e^{2\pi i n k/N}, \qquad 0 \le k \le K-1, \tag{3.12}$$

[1] However, t may be replaced by, e.g., some position coordinate, in which case f is the number of cycles per unit length.

and is related to the continuous Fourier transform as $H(f = k/(K\Delta t)) \approx H_k \Delta t$, where Δt is the sampling interval of the discrete sequence h_n [99]. The inverse DFT (IDFT) of a K-point DFT sequence is given by

$$h_n = \frac{1}{K} \sum_{k=0}^{K-1} H_k e^{-2\pi i n k/K} \tag{3.13}$$

A DFT of length K can be said to be symmetric (even) if $H_k = H_{K-k}$ and antisymmetric (odd) if $H_k = -H_{K-k}$. Similarly, a sequence of length N is symmetric if $h_n = h_{N-n}$ and antisymmetric if $h_n = -h_{N-n}$. Some properties of the DFT (which are similar to those of the continuous Fourier transform) include [89] [99]:

1. The DFT/IDFT of a real sequence is conjugate symmetric.
2. The DFT/IDFT of a symmetric sequence is symmetric.
3. The DFT/IDFT of an antisymmetric sequence is antisymmetric.
4. The DFT/IDFT of a real and symmetric sequence is real and symmetric (follows from properties 1 and 2).
5. The magnitude spectrum of a real sequence is symmetric (follows from property 1).

The *periodogram* is a simple estimate of the power spectrum [2] of the random process that produces the observed signal h_n. It is given by [49]

$$\hat{P}(f = k/K) = \frac{1}{K}|H_k|^2, \tag{3.14}$$

where H_k is the K-point DFT of h_n.

The fast Fourier transform (FFT) algorithms are computationally efficient algorithms for computing the DFT. Their time complexity is generally $O(N \log N)$, instead of $O(N^2)$ that results from using Eqs. 3.12 and 3.13 directly [89] [99].

3.3.2 Linear Prediction

Linear prediction (LP), or linear predictive coding (LPC), is a widely used method for parametrizing the short-time magnitude spectrum of an audio signal frame as an all-pole *synthesis filter* having the z-domain transfer function

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{G}{A(z)} \tag{3.15}$$

[2] The power spectrum (or power spectral density) $P(f)$ of a wide-sense stationary random process is defined as the discrete-time Fourier transform of the autocorrelation sequence [49].

or, equivalently, as an autoregressive (AR) process having the time-domain difference equation

$$s_n = \sum_{k=1}^{p} a_k s_{n-k} + G u_n, \tag{3.16}$$

where u_n is the excitation signal fed to the synthesis filter [78] [79] [102]. In these equations, the $\{a_k\}$ are known as the *predictor coefficients*, p is the *prediction order* and G is the gain factor of the filter model. The FIR filter $A(z)$ in Eq. 3.15 is known as the *inverse filter* or *prediction error filter*.

In speech and audio coding, LP can be used to model the broad spectral shape with $|H(z)|$, while the fine detail of the magnitude spectrum as well as the phase information are coded using other means. The most natural application of LP is in speech coding. LP has a direct connection to the *source-filter model* of speech production [33] if the order p is chosen such that the LP filter depicts the vocal tract *filter* while the excitation signal u_n plays the role of the voiced or unvoiced *source*. All-pole models are particularly well suited for modeling speech signals; an all-pole model can be viewed as a digital filter representation of a lossless acoustic tube model, that is a simplified physical model of the vocal tract [79]. Two poles are needed to model one spectral peak, such as a formant. Thus, if the goal is to model the magnitude spectrum envelope, two poles should be allocated for each expected spectrum envelope peak within the signal band. Speech has, on the average, one formant per kHz of the signal band (0 to $F_s/2$). In addition, a few poles are needed for modeling the excitation source and lip radiation. Thus, a rule of thumb is to choose the prediction order for speech as $p = F_s/\mathrm{kHz} + 3$ or 4 [102]. However, by using a larger value of p, LP can model any signal with a chosen level of accuracy. Besides coding and enhancement applications, LP is also frequently applied in feature extraction in automatic recognition problems.

When p latest signal samples are known, the model of Eq. 3.16 can be used to form a prediction $\hat{s}_n$ of the next sample:

$$\hat{s}_n = \sum_{k=1}^{p} a_k s_{n-k} \tag{3.17}$$

The prediction error or *residual* e_n is the difference between the actually observed speech samples s_n and the model-based predictions $\hat{s}_n$, according to Eq. 3.17:

$$e_n = s_n - \hat{s}_n = s_n - \sum_{k=1}^{p} a_k s_{n-k} \tag{3.18}$$

From Eq. 3.18 it can be seen that e_n is the signal obtained by filtering s_n with the inverse

filter $A(z)$. Comparison of equations 3.16 and 3.18 shows that if the model is correct, then $e_n = Gu_n$.

In the following, speech signal is treated as deterministic and the method of least squares is applied for solving the model parameters a_k. On the basis of Eq. 3.18, the total squared prediction error E (with a predictor of order p) can be calculated from

$$E = \sum_n e_n^2 = \sum_n (s_n - \sum_{k=1}^{p} a_k s_{n-k})^2 \tag{3.19}$$

which is minimized with respect to the model parameters by setting

$$\frac{\partial E}{\partial a_i} = 0, \qquad 1 \leq i \leq p \tag{3.20}$$

Expanding the square in Eq. 3.19 and setting the partial derivatives to zero as in Eq. 3.20 leads to a set of equations known as the *normal equations* [78]:

$$\sum_{k=1}^{p} a_k \sum_n s_{n-k} s_{n-i} = \sum_n s_n s_{n-i}, \qquad 1 \leq i \leq p \tag{3.21}$$

These p equations in p unknowns can be solved to obtain the predictor coefficients a_k, $1 \leq k \leq p$, that minimize E. The minimum value for the total squared prediction error can be shown to be [102]

$$E_p = \sum_n e_n^2 = \sum_n s_n^2 - \sum_{k=1}^{p} a_k \sum_n s_n s_{n-k} \tag{3.22}$$

while a goodness of fit error measure, assuming values between 0 and 1, is the normalized residual energy: [78]

$$V = \frac{E_p}{\sum_n s_n^2} = \frac{\sum_n e_n^2}{\sum_n s_n^2}. \tag{3.23}$$

The range of summation over n in equations 3.19-3.23 was left unspecified. The choice of the range of summation leads to the two basic methods of linear prediction: the *autocorrelation method* and the *covariance method*.

In the autocorrelation method, the error in Eq. 3.19 is theoretically minimized over an infinite interval, $-\infty \leq n \leq \infty$. In practice - given that any realistic signal frame is of finite length - the autocorrelation method presumes that the signal s_n has already been windowed using a window function that is nonzero only during the frame of interest, according to Eq. 3.1. Moreover, to avoid large prediction errors near the boundaries of the frame, the window function w_n should taper gradually towards zero near the boundaries. This makes smooth

pulse-like window functions such as the Hamming window suitable for the autocorrelation method. Since s_n is zero outside the analysis frame, equations 3.21 and 3.22 reduce to

$$\sum_{k=1}^{p} a_k R(i-k) = R(i), \qquad 1 \leq i \leq p \tag{3.24}$$

and

$$E_p = R(0) - \sum_{k=1}^{p} a_k R(k) \tag{3.25}$$

where

$$R(i) = \sum_{n=0}^{N-1} s_n s_{n-i} = \sum_{n=i}^{N-1} s_n s_{n-i} \tag{3.26}$$

is the short-time *autocorrelation function* for the windowed signal. Taking into account the fact that $R(i)$ is an even function, $R(i) = R(-i)$, equations 3.24 can be expressed in matrix form as

$$\begin{pmatrix} R(0) & R(1) & R(2) & \dots & R(p-1) \\ R(1) & R(0) & R(1) & \dots & R(p-2) \\ R(2) & R(1) & R(0) & \dots & R(p-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R(p-1) & R(p-2) & R(p-3) & \dots & R(0) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} R(1) \\ R(2) \\ R(3) \\ \vdots \\ R(p) \end{pmatrix}$$

or in a shorter form as

$$\boldsymbol{Ra} = \boldsymbol{r} \tag{3.27}$$

The autocorrelation matrix $\boldsymbol{R}$ is a *Toeplitz* matrix: all elements along any diagonal are equal. It is also a symmetric matrix. Moreover, $p-1$ of the right hand side elements are also found in the matrix on the left hand side. This special structure allows for an efficient solution algorithm known as *Levinson-Durbin recursion* [102].

In the covariance method, the error in Eq. 3.19 is minimized only over a finite interval, $p \leq n \leq N-1$. It is assumed that the signal is available over the interval $0 \leq n \leq N-1$. Eqs. 3.21 and 3.22 reduce to

$$\sum_{k=1}^{p} a_k \varphi_{ik} = \varphi_{i0}, \qquad 1 \leq i \leq p \tag{3.28}$$

and

$$E_p = \varphi_{00} + \sum_{k=1}^{p} a_k \varphi_{0k} \tag{3.29}$$

where

$$\varphi_{ij} = \sum_{n=p}^{N-1} s_{n-i} s_{n-j} \tag{3.30}$$

are correlation-like values calculated from the data within the signal frame. In matrix form, equations 3.28 can be written as

$$\begin{pmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \cdots & \varphi_{1p} \\ \varphi_{21} & \varphi_{22} & \varphi_{23} & \cdots & \varphi_{2p} \\ \varphi_{31} & \varphi_{32} & \varphi_{33} & \cdots & \varphi_{3p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \varphi_{p1} & \varphi_{p2} & \varphi_{p3} & \cdots & \varphi_{pp} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \cdots \\ a_p \end{pmatrix} = \begin{pmatrix} \varphi_{10} \\ \varphi_{20} \\ \varphi_{30} \\ \cdots \\ \varphi_{p0} \end{pmatrix}$$

or more concisely as

$$\boldsymbol{C}\boldsymbol{a} = \boldsymbol{\varphi} \tag{3.31}$$

Also the covariance method matrix is symmetric, since $\varphi_{ij} = \varphi_{ji}$, but it is not Toeplitz and the right-hand-side terms do not appear on the left hand side. Thus, Levinson-Durbin recursion cannot be applied. The covariance method equations can be solved using Cholesky decomposition [102].

Block estimation methods, such as the autocorrelation method and the covariance method, solve the coefficient vector $\boldsymbol{a}$ (Eqs. 3.27 and 3.31) directly by assuming the signal to be stationary within each estimation frame. The coefficient vector can also be estimated in an adaptive fashion such that the coefficients of an adaptive filter are updated with each incoming sample and the coefficient values are stored at the chosen interval. For example, the *gradient adaptive lattice (GAL)* algorithm [42] [49], which involves an adaptive FIR lattice filter, can be employed for adaptive LP analysis. The simple LMS algorithm [124] can also be used for adaptive LP analysis [49], although its accuracy for speech signals is typically not as good as that of the aforementioned methods.

Using the LP direct form coefficient representation a_i, $1 \leq i \leq p$, as features in pattern recognition requires the use of special distance measures [60]. There are, however, various alternative ways of representing the LP information for which the distance computation is simpler. *Reflection coefficients* [57], also known as *partial correlation* (PARCOR)

coefficients [102], are essentially parameters of the lattice form of the LP filter. They can be computed directly or as a by-product of the Levinson-Durbin recursion for autocorrelation LP analysis. The *log area ratios* can be formed from the reflection coefficients by the formula [102]

$$g_i = \log\left(\frac{1-k_i}{1+k_i}\right) \tag{3.32}$$

The line spectral frequencies (LSFs), also known as the line spectral pair (LSP) representation [116], are another way to represent an all-pole model. First, the all-pole polynomial $A(z)$ is used to generate the symmetric polynomial

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) \tag{3.33}$$

and the antisymmetric polynomial

$$Q(z) = A(z) - z^{-p(+1)}A(z^{-1}). \tag{3.34}$$

The LSF representation is now derived from computing the roots of the polynomials $P(z)$ and $Q(z)$. A fundamental property of LSFs is that as long as the roots due to $P(z)$ and $Q(z)$ interlace on the unit circle, the corresponding all-pole model is stable. This is a very useful property in coding, which is the primary application area of LSFs. However, LSFs have also been employed as features in classification applications, e.g. in speaker segmentation [76].

The autocorrelation coefficients are computed in the first stage of the autocorrelation method of LP. It is possible to recover the $p+1$ first coefficients of the autocorrelation function from the LP filter model and they can be used as an alternative representation for the LP filter, containing equivalent information. The LP filter can also be represented in the form of a *cepstrum*, discussed in the following section.

3.3.3 Cepstral Representations

Cepstral analysis refers to a group of homomorphic signal processing methods [89] that is frequently useful in decomposing convoluted signal components, provided that these components have sufficiently different spectra. For discrete-time sequences, the *real cepstrum* can be defined as

$$\mathrm{DFT}\left(\{\log\left(|\mathrm{IDFT}(\{s_n\})|\right)\}\right) \tag{3.35}$$

where $\mathrm{DFT}(\{x_n\})$ and $\mathrm{IDFT}(\{x_n\})$ denote the discrete Fourier transform and the inverse discrete Fourier transform operations, respectively, performed on some sequence

$\{x_n\}$. From the selected properties of Fourier transforms listed in Section 3.3.1, it can be seen that the real cepstrum is real-valued and symmetric [3].

Consider two time domain signals, say s_n and h_n, that have been convoluted in the observed signal $r_n = s_n \star h_n$. The goal of the analysis is to find the magnitude spectrum $|S_k|$ of s_n. Applying the convolution theorem [89], the magnitude spectrum of the observed signal is

$$|R_k| = |S_k||H_k| \tag{3.36}$$

where $|H_k|$ is the magnitude spectrum of h_n. Now take the logarithm (of any base) of both sides. This gives

$$\log|R_k| = \log|S_k| + \log|H_k| \tag{3.37}$$

In the logarithmic magnitude spectrum of r_n, the undesired component appears as an additive component instead of a convolved component. However, the two components still cannot be separated without an estimate of $|H_k|$. Suppose, however, that it is known from experience that the logarithmic magnitude spectrum $|H_k|$ of the convolved component will always be very different from $|S_k|$, such that the other changes much more rapidly with frequency than the other. Now taking the (inverse) discrete Fourier transform results in

$$\text{IDFT}\left(\{\log|R_k|\}\right) = \text{IDFT}\left(\{\log|S_k|\}\right) + \text{IDFT}\left(\{\log|H_k|\}\right) \tag{3.38}$$

in which the components due to the two signals, s_n and h_n, are still additive because the IDFT is a linear operation. The two components that were originally convolved in the time domain are now additive *and* inhabit different "quefrency" (a term for the cepstral domain "frequency") ranges. Note that the choice of the IDFT instead of DFT only affects the scaling, not the shape, of the real cepstrum. To see this, consider the following: 1) because the logarithmic magnitude spectrum is real and symmetric, its DFT or IDFT will be real (and symmetric); 2) if DFT/IDFT is real, it is known that the phase inversion between Eqs. 3.12 and 3.13 has been canceled out in the summation and does not affect the transform; 3) thus, the only practical difference between using Eq. 3.12 or 3.13 in computing the real cepstrum can be the scaling coefficient $1/K$ in 3.13.

A frequent application of the cepstrum is in speech processing, in the separation of the excitation source from the vocal tract filter, according to the source-filter model of speech production [33]. The cepstrum is particularly suited for this task, because the spectrum of the glottal excitation of voiced speech is very different from the vocal tract filter response.

[3] However, the name "real cepstrum" does not refer to the fact that it is real-valued. The real cepstrum and the *complex cepstrum* differ in whether the real or complex logarithm is used [89].

Consider again the two signals that are now separated in the cepstral domain. The ultimate goal is to extract information pertaining to the magnitude spectrum $|S_k|$ that varies at a certain rate (for example, $|S_k|$ may correspond to the magnitude spectral envelope and vary more slowly than $|H_k|$). Because the cepstrum is a Fourier transform of the logarithmic magnitude spectrum, the convolution theorem allows bandpass filtering of the logarithmic magnitude spectrum to be performed in the cepstral domain using *multiplication* (windowing) by the bandpass transfer function. This operation is frequently called *liftering* (a play on *filtering*). In the ideal case when the liftering operation manages to completely separate the cepstra of the two signals, the logarithmic magnitude spectrum $\log|S_k|$ can be recovered by DFT if desired (and the magnitude spectrum by further exponentiation). *Low-time liftering* refers to a "low pass" type operation in which the aim is to separate the spectral envelope.

In feature extraction for recognition applications, it is often wisest to stay in the cepstral domain after the liftering. It can namely be shown that the Euclidean distance between two cepstra, that have been liftered using a rectangular low-time window and truncated to the length of the window L, lower bounds the root mean square (rms) log spectral distance, which in turn is known to be a relevant distance measure between audio signals [40]. When L is increased, i.e., more cepstral coefficients are included, the cepstral Euclidean distance approaches the rms log spectral distance from below. The fact that the simple Euclidean distance is meaningful for computing distances between cepstral vectors has consequences for audio pattern recognition: as will be seen in Chapter 4, the Euclidean distance often significantly reduces the number of parameters that have to be estimated for the models, thus alleviating the curse of dimensionality. Because of this desirable property, the cepstrum is often used simply as a feature vector representation for magnitude spectral models computed by other means, such as LP analysis. Conversion from the LP filter coefficients to cepstral coefficients can be accomplished by the recursive formula [79]:

$$c_0 = \log(G) \tag{3.39}$$

$$c_n = \frac{1}{2}\left(a_n + \sum_{m=1}^{n-1}\left(\frac{m}{n}\right)c_m a_{n-m}\right), \quad 1 \leq n \leq p \tag{3.40}$$

3.4 Perceptual Features

The term 'perceptual feature' is used in the present context to refer to features that take into account more aspects of human auditory perception than a standard logarithmic magnitude spectrum (or its equivalent representation, such as the real cepstrum). It should be noted that already the logarithmic magnitude spectrum is a perceptually motivated representation for the following reasons: as reflected by the psychoacoustical phenomenon of

intensity-loudness compression, the "amplitude response" of the peripheral auditory system is roughly logarithmic; the inner ear is known to perform spectral analysis; and the hearing has been found to be rather insensitive to the phase spectrum of the signal (meaning that the magnitude spectrum is more important). In making feature representations even more like the representations generated by the peripheral auditory system, the next step is usually the incorporation of the nonuniform frequency resolution of the cochlea.

3.4.1 Auditory Filterbanks

Auditory filterbanks refer to filterbanks, implemented in either the time domain or the frequency domain, that implement the nonuniform frequency resolution of the inner ear. While sophisticated time-domain filter models exist that attempt to duplicate the physiology of the inner ear, for practical applications in automatic sound classification it often suffices to have simple functional models whose output somewhat resembles the true auditory spectrum. Mel-scale filterbanks are one such approach which has shown success, although they are most often used in the form of the mel frequency *cepstrum*, discussed in the next section.

In the frequency domain, one can construct a triangular mel-filterbank with M filters, such as the one whose transfer functions are shown in Figure 3.2, as follows. Denote the lowest and highest frequency of the filterbank by f_l and f_h, respectively. Let F_s be the sampling frequency and let L be the DFT index corresponding to the Nyquist frequency $F_s/2$. Determine the frequency boundaries of the filter transfer functions so that they are uniformly spaced on the mel scale [57]:

$$f(m) = \left(\frac{L}{F_s}\right) B^{-1}(B(f_l) + m\frac{B(f_h) - B(f_l)}{M+1}, \qquad 0 \le m \le M+1, \tag{3.41}$$

where the mel scale function B and its inverse transform function B^{-1} are given by Eqs. 2.5 and 2.6, respectively.

The transfer functions of the type shown in Figure 3.2 are given by [57],

$$H_{m,k} = \begin{cases} 0 & k < f(m-1) \\ \frac{(k-f(m-1))}{(f(m)-f(m-1))} & f(m-1) \le k \le f(m) \\ \frac{(f(m+1)-k)}{(f(m+1)-f(m))} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases} \tag{3.42}$$

Using these filters, logarithmic mel-filterbank energies for a signal frame whose DFT is S_k are given by

$$E_m = \log\left(\sum_{k=0}^{L-1} |S_k|^2 H_{m,k}\right) \tag{3.43}$$

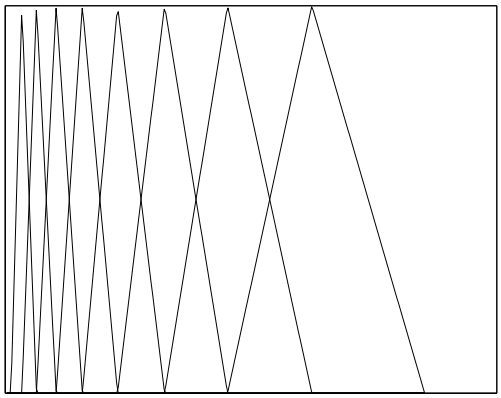


Figure 3.2: Triangular magnitude responses of filters for mel-scale auditory filterbank analysis.

3.4.2 Mel Frequency Cepstral Coefficients

The mel frequency cepstral coefficients (MFCC) representation [23] is, by far, the most popular method of feature extraction in ASR. The first three stages of MFCC computation are 1) estimation of the short-time magnitude spectrum; 2) computation of mel-filterbank energies using triangular bandpass filters in the frequency domain; 3) taking the logarithm of each filterbank output. These steps were discussed in the previous section in the context of mel-filterbank analysis. The final step in computing the MFCC representation is 4) discrete cosine transformation (DCT) of the logarithmic filtered energies. The DCT-II discrete cosine transform for M mel-filterbank outputs is given by [57]

$$c_i = \sqrt{\frac{2}{M}} \sum_{j=1}^{M} E_m \cos\left(\frac{\pi i}{M}(j-0.5)\right) \tag{3.44}$$

The MFCC is essentially a variation of the real cepstrum, in which the logarithmic magnitude spectrum is substituted by logarithmic energies from an auditory filterbank. Another modification is the use of DCT-II in place of IDFT to convert the logarithmic spectral representation to a cepstral representation.

The four processing stages are illustrated in Figure 3.3.

Basic MFCC computation uses the DFT in the first step for magnitude spectrum estimation (essentially equivalent to the periodogram method of power spectrum estimation). In order to enhance the robustness of MFCC features in the presence of environmental noise, linear prediction (LP) and minimum variance distortionless response (MVDR) spectrum

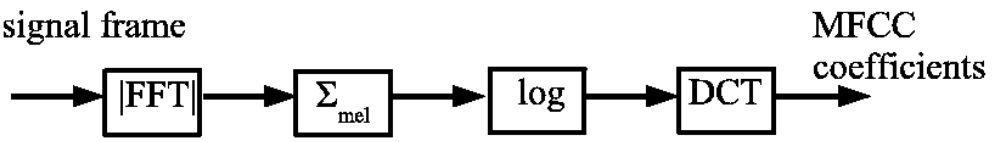


Figure 3.3: Data flow diagram of MFCC computation.

estimation methods have been proposed as alternatives to DFT [24] [26].

3.4.3 Other Perceptual Representations

A popular auditory spectral feature extraction technique is *perceptual linear prediction* (PLP) proposed by Hermansky [53]. It consists of weighting auditory filterbank outputs by an equal loudness curve, simulating the intensity-loudness compression of the weighted filter outputs by taking the cubic root (power 0.33), transforming the auditory power spectrum into an autocorrelation sequence by IDFT and finally estimating an LP model using the conventional autocorrelation method. The basic LP-to-cepstrum conversion formulas (Eqs. 3.39-3.40) can be used to convert the PLP representation to a cepstrum. With auditory frequency warping, fewer LP or cepstral coefficients are needed to represent the spectrum while still containing roughly the same information in a perceptual sense.

Other auditory feature vector representations can be obtained by, for example, the ensemble interval histogram method [100], frequency-warped linear prediction [58] and perceptual MVDR-based cepstral analysis [26].

3.5 Specialized Short-Time Features

Previous sections have introduced vector representations of either the short-time magnitude spectrum or an auditory version of the magnitude spectrum. Besides these representations, various individual features exist that measure some quality of the signal. They may be computed in different domains: e.g., time domain, frequency domain, autocorrelation domain or cepstral domain. On the other hand, they represent different perceptual aspects; most of the features can be roughly categorized into one of three categories - loudness features, timbre features and pitch features. This section introduces some common, easily computable short-time features that depict loudness or certain aspects of timbre. In general, loudness features tend to be simpler to compute than timbre or pitch features and can provide a fairly accurate characterization of the loudness perception. Also some dimensions of the timbre perception can be easily modeled. However, a more generally adequate characterization of the timbre probably requires using auditory spectral feature vectors. As for pitch, estima-

tion of the pitch period/fundamental frequency is a challenging problem for which many schemes have been developed [56]. No general-purpose solution, that would be reliable for a broad class of sounds, exists [66] [130]. However, apart from music melody recognition, explicit estimation of a single pitch is not very central in audio recognition applications, although it can be helpful [130].

It should be noted that scaling coefficients in the formulas are frequently unimportant in an automatic classification setting. As long as the same finite, non-zero scaling coefficient is used in training and in recognition for all the feature observations, its value does not normally affect the classification. Thus, two alternative formulas appearing in different literary sources and purporting to produce the same feature can be considered equivalent from the pattern classification perspective if they differ only in scaling (e.g., in whether or not to divide by the number of observations). In the present work, however, an attempt has been made to correctly include the scaling coefficients so that the features correspond to their common names.

The *short-time energy* STE is defined as the energy in a signal frame. It is given by [102]

$$\mathrm{STE} = \sum_{n=1}^{N} s_n^2 \tag{3.45}$$

Computing *short-time average energy* instead of the total energy, by applying a scaling coefficient $1/N$, would limit the magnitude of the feature and make energies computed with different frame sizes N comparable in magnitude. Usually, however, this is unimportant as only one frame size N is used.

The *logarithmic short-time energy* LGSTE is some logarithm (typically natural or base 10) of STE.

In addition to the full-band energies STE and LGSTE, any number of *filtered energies*, either pure or logarithmic, can be defined as energies of the outputs of different (typically bandpass) filters. *Energy ratios* are (usually logarithmic) ratios of some band energy to either some other band energy or the full-band energy [112]. They can be used to characterize selected aspects of the distribution of energy in the spectrum.

Loudness computed by the perceptual loudness model discussed in Chapter 2 can also be used as a feature [80]; it will be denoted by LOUD.

The *autocorrelation function* is given by

$$R(i) = \sum_{n=i+1}^{N} s_n s_{n-i} \tag{3.46}$$

such that $R(0)$ is equal to the STE. *Normalized autocorrelation coefficients* are given by $r(i) = R(i)/R(0)$. In particular, the *unit-lag normalized autocorrelation coefficient* $r(1)$,

denoted by $\mathrm{AC1}$, has often been used as a speech feature [6] [112] [96].

The *zero crossing rate* [102] is an extremely popular short-time feature that depicts the concentration of energy in the spectrum. It is defined as the average-per-sample number of times the signal changes sign (crosses zero) within the frame:

$$\mathrm{ZCR} = \frac{1}{N-1}\sum_{n=2}^{N} I\left(\mathrm{sign}(s_{n-1}) \neq \mathrm{sign}(s_n)\right) \tag{3.47}$$

The zero crossing rate can be viewed as a measure of the *dominant frequency* in the signal [69]. It assumes low values for signals with a predominantly low frequency content (such as voiced speech) and high values for noisy signals (such as unvoiced speech).

A less well known descendant of the zero crossing rate is a measure developed by Paulus [93], known by the name *gradient index* [61]:

$$\mathrm{GIN} = \frac{\sum_{n=2}^{N-1} \Delta\psi(n)|s_n - s_{n-1}|}{\sqrt{\sum_{n=1}^{N} s_n^2}}, \tag{3.48}$$

where

$$\Delta\psi(n) = \frac{|\psi(n) - \psi(n-1)|}{2} \tag{3.49}$$

and

$$\psi(n) = \frac{s_n - s_{n-1}}{|s_n - s_{n-1}|}. \tag{3.50}$$

Like the zero crossing rate, the gradient index has been found to contain information about broad classes of speech sounds (such as vowel, fricative, etc.) and has been utilized in artificial bandwidth extension of telephone speech [61].

The *spectral centroid* is defined as the "center of gravity" of the magnitude spectrum, i.e., the frequency which divides the magnitude spectrum into two portions of approximately equal "mass":

$$\mathrm{SCENT} = \frac{\sum_{k=1}^{L} k|S_k|}{\sum_{k=1}^{L} |S_k|} \tag{3.51}$$

The spectral centroid is commonly regarded as a good measure of the perceptual quality of *brightness* (e.g. [126]), which in turn is an important constituent of the perception of timbre [17].

The *spectral flatness* [79] is the ratio of the geometric mean of the magnitude spectrum

to its arithmetic mean. It can be computed as

$$\mathrm{SFLAT} = \frac{\exp\left((1/L)\sum_{k=1}^{L}\log(|S_k|+\epsilon)\right)}{\epsilon + (1/L)\sum_{k=1}^{L}|S_k|} \tag{3.52}$$

where the arbitrary small positive value ϵ is added in the numerator to prevent taking a logarithm of zero and in the denominator to prevent division by zero. Spectral flatness assumes values between 0 and 1, with low values for sounds with highly shaped spectra and high values for sounds with flat spectra (such as white noise or a unit impulse train [78]).

The normalized residual energy of linear prediction (LPNRE), given by Eq. 3.23, has also been used as a short-time feature for speech [6] [96]. It takes on low values for signal segments that are well predictable by the all-pole model (such as vowel-like speech sounds) and high values for segments that are hard to predict (such as noisy fricatives and rapid transients).

Many *pitch estimation* methods are based on first preprocessing the signal in order to emphasize the fundamental frequency components and to suppress other information, after which the pitch period is found as the autocorrelation lag (within allowed range) corresponding to the maximum peak in the short-time autocorrelation function of the processed signal [102]. A prewhitening operation by LP inverse filtering (that is, analyzing the autocorrelation function of the LP residual) is one possibility. The real cepstrum is related to the autocorrelation function [4] and can also be utilized for pitch estimation in a similar fashion as the autocorrelation function. Tolonen and Karjalainen [121] proposed a computationally efficient *multiple pitch analysis* method in which the signal is prewhitened by frequency-warped LP and divided into two channels (below and above 1000 Hz), a generalized autocorrelation is computed separately for the low-channel signal and for the envelope of the high channel signal, the two autocorrelation functions are summed and finally the summary autocorrelation function is enhanced by pruning unnecessary peaks.

Figure 3.4 illustrates the shapes of the time contours of selected short-time features during six different sound segments. The features have been computed from 20 ms windows with a 10 ms shift interval. Each sound sample is five seconds long and sampled at 16 kHz. It is apparent from the figure that these sound classes can hardly be reliably separated by direct thresholding of the short-time features. However, if a longer time segment is analyzed at a time, the short-time feature waveforms can look very different for different types of sound. This observation is one motivation for long-term feature analysis discussed in Section 3.6.

Figure 3.5 shows an absolute-valued correlation matrix of the ten features illustrated

[4] The autocorrelation function is the IDFT of the power spectrum, while the real cepstrum is the IDFT of the logarithmic magnitude spectrum.

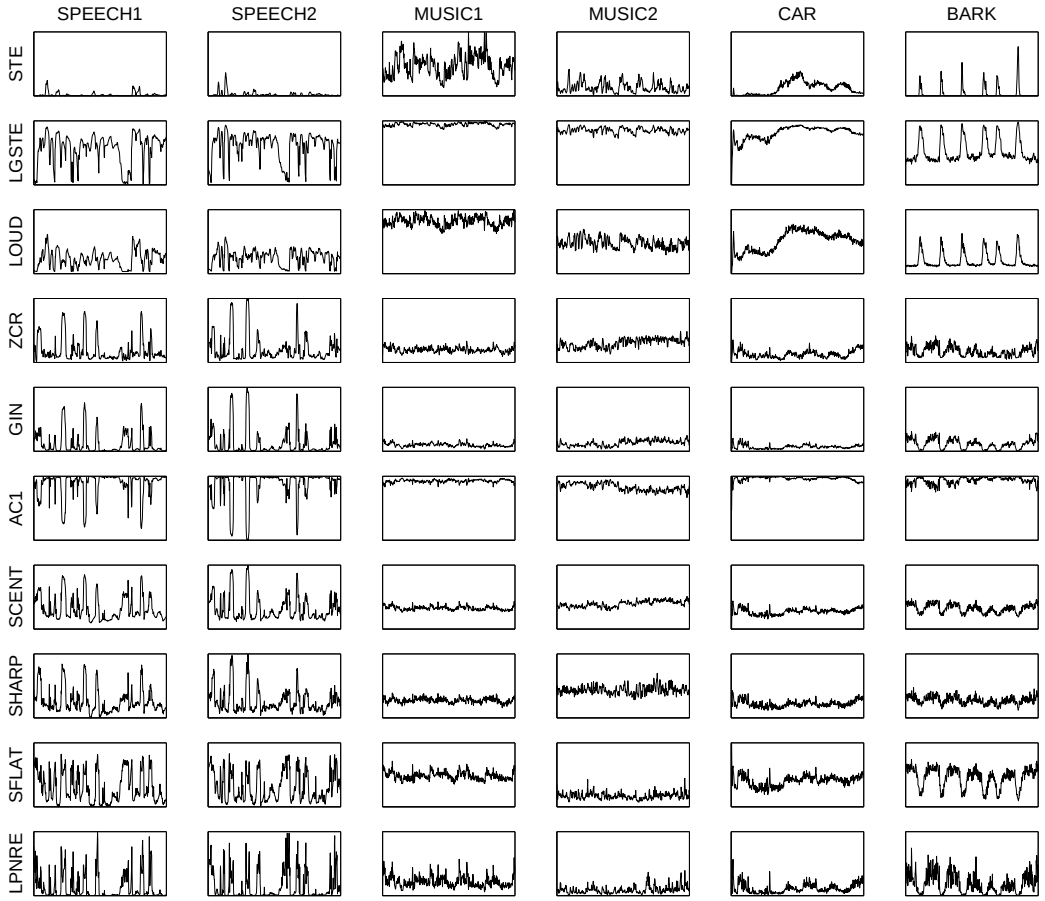


Figure 3.4: Examples of the time evolution of ten short-time acoustic features during six samples containing different types of sound. The features are short-time energy (STE), logarithmic short-time energy (LGSTE), loudness (LOUD), zero crossing rate (ZCR), gradient index (GIN), unit-lag normalized autocorrelation (AC1), spectral centroid (SCENT), sharpness (SHARP), spectral flatness (SFLAT) and LP normalized residual energy (LPNRE). The features have been computed from frames of 20 ms with 10 ms frame shift. Descriptions of the sound samples: *Speech1* is English speech by a male speaker. *Speech2* is a female speaker speaking the same sentences. *Music1* is a loud segment from a rock song. *Music2* is violin music. *Car* is the sound of a car starting engine and driving off. *Bark* is barking of a dog. The length of each sample is five seconds and the sampling rate is 16 kHz. Prior to feature computation, the maximum amplitude of each of the six sound segments was made equal. The vertical axis is the same in each plot of the same feature.

in Figure 3.4, estimated over the same six sound segments. Dark squares indicate high correlation or anti-correlation between two features while light squares indicate lack of

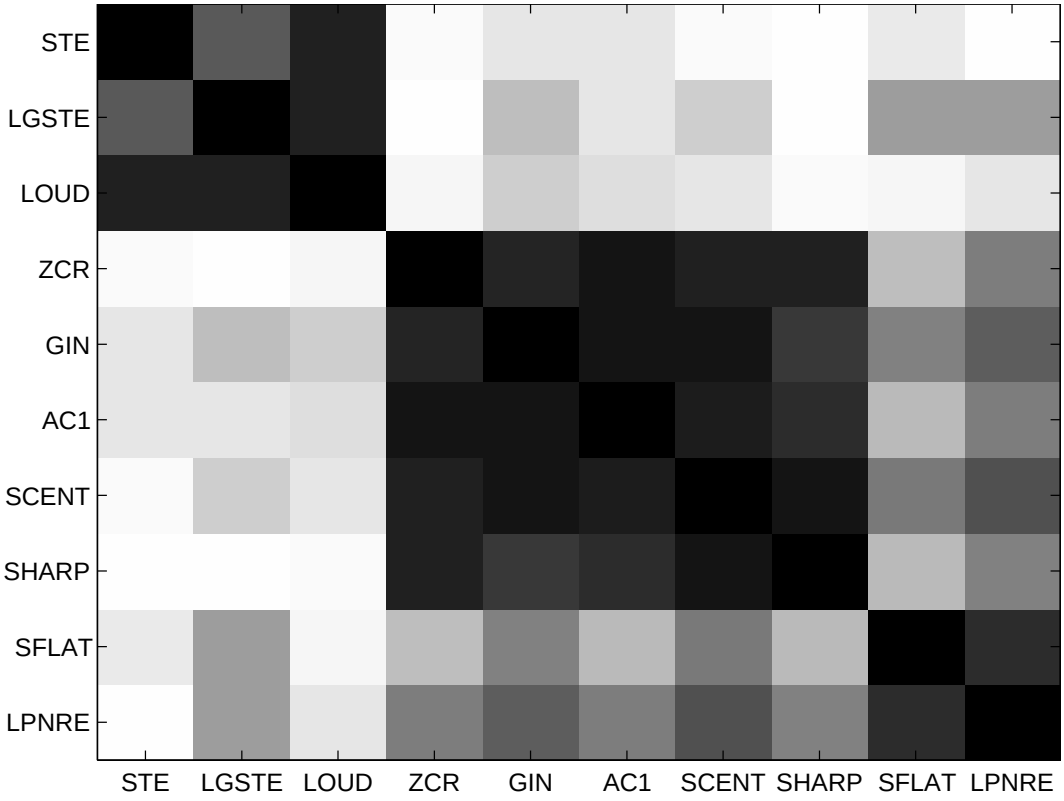


Figure 3.5: The absolute-valued correlation coefficient matrix of ten short-time acoustic features estimated over the six sound samples of Figure 3.4. Lighter areas indicate a correlation close to zero (orthogonality) and darker areas indicate high correlation (correlation coefficient close to 1) or anti-correlation (correlation coefficient close to -1).

correlation (orthogonality) between two features. Three groups of features can be identified based on correlation/anti-correlation. In terms of perceptual attributes, the features STE, LGSTE and LOUD seem to form a group of "loudness features", while ZCR, GIN, AC1, SCENT and SHARP seem to be associated with the timbral aspect of sharpness (modeled by the feature SHARP) [132] or brightness [17] (commonly associated with the spectral centroid SCENT). The two remaining features SFLAT and LPNRE form a group of their own while being more similar to the brightness features than to the loudness features.

3.6 Long-Term Processing

Considering the important role of the inner ear as a spectrum analyzer, it is natural that many of the features that have been found to be useful in the classification of audio have an interpretation related to the short-time auditory spectrum. However, as discussed in Chapter

2, it is known that the higher neural stages of the auditory pathway generate longer-term representations of the sound that are modulation frequency selective. Considering human perception in general, relative changes in the stimulus are often more relevant than the absolute values of the stimulus [86]. Temporal changes of the stimulus at certain rates appear to dominate the auditory perception. For speech communication, spectral components that change at a rate close to 4 Hz appear to carry the perceptually most relevant information [54]. This is close to the typical syllabic rate in speech. Long-term features based on 4 Hz *modulation* energies have been successful in separating speech and music [107]. More generally, modulation band energy features can be efficient representations in discriminating among different sound classes. From Figure 3.4, it is apparent that in many cases, the same short-time feature computed from two different types of sound displays very different power spectrum, when the spectra of the feature waveforms are estimated from whole segments (for example by DFT). Bandpass energies of the modulation of different short-time features have been used with success as long-term features for both general audio classification and musical genre classification in [80].

On a shorter time scale, simple measure of the instantaneous change in the spectrum is the *spectral flux*, which is defined in [107] as the 2-norm of the difference (i.e., the Euclidean distance) of two successive magnitude spectra $\boldsymbol{S}_n$ and $\boldsymbol{S}_{n-1}$:

$$\mathrm{SFLUX} = \|\boldsymbol{S}_n - \boldsymbol{S}_{n-1}\| \tag{3.53}$$

The so called *delta features*, denoted here by Δ_n and $\Delta\Delta_n$, are commonly computed for feature x_n using linear regression on an integer variable as [36]

$$\begin{aligned} \Delta_n &= \frac{\sum_{\theta=-W}^{W} \theta x_{n+\theta}}{\sum_{\theta=-W}^{W} \theta^2} \\ \Delta\Delta_n &= \frac{\sum_{\theta=-W}^{W} \theta \Delta_{n+\theta}}{\sum_{\theta=-W}^{W} \theta^2} \end{aligned} \tag{3.54}$$

where the parameter W determines the width of the window used in computing the regression coefficients. Usually the deltas are computed for cepstral features and logarithmic frame energies and then supplemented to the feature vector, i.e., they are usually not used in isolation but together with the original features from which they have been computed.

In the field of ASR, feature postprocessing techniques have been developed to reduce the effects of very slowly or occasionally varying channel and environment conditions in the short-time spectrum-based features, while retaining the relevant modulation frequencies. The most notable methods of this kind include *RASTA* filtering and *cepstral mean subtraction*.

RASTA filtering is part of the RASTA-PLP speech analysis technique proposed by Hermansky [55]. An IIR temporal filter with the transfer function

$$H(z) = 0.1z^4 \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 - 0.98z^{-1}} \tag{3.55}$$

is applied to the feature vectors across frames. In the original RASTA technique [55], filters such as Eq. 3.55 are applied to logarithmic critical-band spectra.

Cepstral mean subtraction refers to the subtraction of a long-term average cepstrum from cepstral feature vectors [5] [57]:

$$\hat{\boldsymbol{c}}_n = \boldsymbol{c}_n - \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{c}_i \tag{3.56}$$

This operation makes the resulting feature vectors invariant to changes in the long-term average that is subtracted. Thus, it serves to increase robustness against stationary or slowly changing background noise.

In feature extraction for the recognition of broad sound classes with high variability (such as speech in general, music in general, speech of a specific speaker, music from a specific genre or some class of environmental sounds), longer segments of successive short-time features are often considered. The longer analysis window, whose length is typically between 0.5 and 5 seconds, can be called a *texture window* [122]. Statistics like the sample *mean* and the sample *variance* of the short-time feature values are frequently used as the final features to characterize audio textures over the texture window frame. Moreover, the AR(1) prediction coefficient of some short-time feature x_n can be computed using least squares estimation as follows [47]. Consider the AR(1) model $x_n = c + \phi x_{n-1} + u_n$, where c is a constant intercept term, ϕ is the AR(1) coefficient and u_n is Gaussian white noise. For a sample of N values within the texture window $(x_1, \ldots, x_N)$, this can be written in matrix form as $\boldsymbol{x} = \boldsymbol{Z}\boldsymbol{\beta} + \boldsymbol{u}$, where $\boldsymbol{x} = [x_2 \quad \ldots \quad x_N]'$, $\boldsymbol{\beta} = [c \quad \phi]'$, $\boldsymbol{u} = [u_2 \quad \ldots \quad u_N]'$ and

$$\boldsymbol{Z} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_{N-1} \end{bmatrix}$$

The least squares solution for $\boldsymbol{\beta}$ is given by [47]

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\boldsymbol{x}. \tag{3.57}$$

From $\hat{\boldsymbol{\beta}}$, the estimate of ϕ can be extracted and functions as a measure of the self-similarity of the variable with short lags, or as an inverse measure of the rate of change.

The AR(1) coefficient defined and estimated as above is invariant to changes in the constant DC level of x_n. For zero-mean sequences, the estimated AR(1) coefficient is approximately equivalent to the unit delay normalized autocorrelation coefficient. Thus, an equivalent feature can also be computed by subtracting the mean and computing the normalized unit delay autocorrelation.

Specially tailored long-term statistics are also frequently used as features, such as the number of frames for which the short-time feature exceeds a certain threshold dependent on the mean of the short-time feature. In [76], the *high zero crossing rate ratio* (HZCRR) was defined as the ratio of the number of frames (within a 1 s window) whose zero crossing rate is above 1.5 times the average zero crossing rate for the same window (denoted $\overline{\mathrm{ZCR}}$), i.e.,

$$\mathrm{HZCRR} = \frac{1}{2N} \sum_{n=1}^{N} I\left(\mathrm{ZCR}_n > 1.5\overline{\mathrm{ZCR}}\right), \tag{3.58}$$

where N is the number of frames within the one-second window, ZCR_n is the zero crossing rate of the nth frame within the window and and $\overline{\mathrm{ZCR}}$ is the mean of the ZCR feature over the N frames.

Because HZCRR picks up the skewed distribution of the zero crossing rate during speech segments, it was proposed in [76] as one feature for detecting the presence of speech. A related ZCR-based statistic for detecting a skewed distribution due to speech activity was also used in [106].

As another example of specialized long-time features, the *low short-time energy ratio* feature reacts to the fact that speech contains relatively large amounts of silence (due to pauses and the closure portions of the stop consonants) [76]:

$$\mathrm{LSTER} = \frac{1}{2N} \sum_{n=1}^{N} I\left(\mathrm{STE}_n < 0.5\overline{\mathrm{STE}}\right) \tag{3.59}$$

3.7 Music-Specific Features

Different music applications may require musical signal recognition or transcription in terms of chords, key and genre, among other things. Besides instrumentation, which is well reflected in timbral features, music is also characterized by its rhythmic structure and harmonic content [122]. The short-time features discussed so far have been mostly timbral or loudness features. Such features are inadequate for representing rhythm, pitch or harmony. The conventional short-time analysis frame of roughly 20 milliseconds is too short to capture rhythmic structure, while timbral features such as the MFCCs lose pitch information and are thus not suited for representing the harmonic information in the spectrum.

To capture the harmonic content in music, *chroma-based* feature vectors are often used. A typical approach is to map each frequency into a pitch class (corresponding to one of the twelve semitones), irrespective of octave, and by summation of a magnitude spectral representation over each pitch class (i.e., over different octaves) to form a *chroma vector* whose each element corresponds to one pitch class. In one such approach proposed by Bartsch and Wakefield [10], pitch class averages are computed from the logarithmic magnitude of the DFT. In another similar feature extraction method [110], originally proposed by Fujishima [35], squared DFT magnitudes are averaged over each pitch class. Variations of chroma-based representations have been used, e.g., in structural music analysis [10], musical genre identification [122], chord segmentation [110], chord transcription and mid-level representations [12] and musical scene analysis [72]. Multiple pitch analysis methods can also be useful in generating harmonic features for music [72] [122].

In automatic musical genre identification and similar applications, the rhythm plays an important role. Many of the long-term features introduced in Section 3.6 capture rhythm information to an extent. However, several more sophisticated feature extraction methods for characterizing rhythmic content have also been developed specifically for music signals [72]. A prominent approach is to compute a *beat histogram*. One implementation of beat histogram feature extraction has been used by Tzanetakis and Cook [122] in musical genre classification. First, the signal is divided into octave frequency bands by discrete wavelet transform, after which an enhanced autocorrelation is computed for the sum of the time domain envelopes computed separately for each band. Multiple autocorrelation peaks from the appropriate beat range are picked and their values added to the beat histogram. The periods and amplitudes of peaks in beat histograms can be used as rhythmic features [122].

3.8 Vector Quantization

After short-time and long-time processing in Figure 3.1, a representation of the audio signal as a sequence of feature vectors $\{\boldsymbol{x}_n\}$ is obtained. In the general audio classification system depicted in Figure 1.1, feature extraction is followed by pattern recognition. Sometimes it is more desirable to apply pattern recognition methods to sequences of discrete nominal symbols instead of sequences of continuous-valued vectors. This is where *vector quantization* (VQ) [38] comes into play. A complete vector quantizer is specified by

- a *decision rule* that partitions the d-dimensional feature space into J nonoverlapping regions
- a *codebook* $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J\}$ containing one *codeword*, or reproduction vector, for each region

Vector quantization uses the decision rule to replace each continuous-valued vector by a symbol, essentially an element of the set $\{1, \ldots, J\}$, corresponding to the region in the data space to which the vector belongs. As VQ has its primary applications in coding and compression, an important aspect of VQ is that it is possible to reproduce the continuous-valued data - with some loss of precision - from the encoded discrete-valued data. This is accomplished by replacing the VQ symbols with corresponding codebook entries.

In some VQ approaches, the two components of VQ - the decision rule and the codebook - are very closely related and may share the same parameters. In others, they may have an entirely different parametrization. In the context of feature extraction for audio classification, reproduction is not usually important and it often suffices to have just the decision rule component of VQ. A simple method called *k-means* and its variant *LBG*, which can be used to construct VQ decision rules (and codebooks) by means of unsupervised learning, are discussed in Chapter 4.

Chapter 4

Pattern Recognition Methods

This chapter reviews some of the most central pattern recognition methods that find application in automatic classification of audio content. In supervised classification, emphasis is given to Bayesian classification methods, including Gaussian, Gaussian mixture model (GMM) and hidden Markov model (HMM) classifiers. Besides the wide popularity of Gaussian and GMM classifiers in general supervised pattern recognition, this emphasis can be justified by the importance of HMMs in time series recognition applications, by the close connection between GMMs and HMMs and by the applicability of both GMMs and HMMs to unsupervised segmentation and classification of a time series. Nearest neighbor classifiers (the kNN rule) and dynamic time warping (DTW) are also discussed in some detail, as the former is a conceptually simple general-purpose supervised classification method and the latter is an alternative to HMMs in measuring the similarity of time series segments. Other popular supervised classification methods, which are usually interchangeable with GMMs and nearest neighbor classifiers, are briefly introduced with references to the literature. In unsupervised classification, some of the most central clustering algorithms and HMM learning receive a detailed treatment. An example of the separation of speech and music by unsupervised classification methods is presented. Finally, methods of explicit segmentation are discussed.

4.1 Supervised Pattern Recognition

4.1.1 Bayesian Classification

The present discussion on pattern recognition is based on the following assumptions. Let $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ be a set of $(d \times 1)$ feature vectors that are observed; of course, if $N = 1$ then X consists of a single feature vector. Let ω be a pattern of some external categorical variables, i.e., a *state of nature* [28], that produced X. For example, in automatic

speech recognition, the state of nature ω could consist of category variables whose values equal to both the statements "a person is speaking in English" and "a person says the word "cat"". ω would thus define the "cat" class of observations. Automatic classification is required when ω can not be directly observed completely; the goal is then to infer ω from the observations X.

After observing the data X, the natural probabilistic approach to recognizing the underlying state of nature is to find $\hat{\omega}$ such that

$$\hat{\omega} = \underset{\omega}{\operatorname{argmax}}\, P(\omega|X). \tag{4.1}$$

Evaluating the conditional discrete probability distribution $P(\omega|X)$ is problematic, however, if X is a continuous-valued random vector (as is typical for low-level feature vectors derived from physical measurements using sensors such as microphones). A popular solution is to apply *Bayes' formula*:

$$P(\omega|X) = \frac{P(X|\omega)P(\omega)}{P(X)} \tag{4.2}$$

Substituting this into Eq. 4.1, it is easily seen that the term $P(X)$ in the denominator of Eq. 4.2, being independent of ω, does not affect the outcome of Eq. 4.1. Thus, Eq. 4.1 can be rewritten as

$$\hat{\omega} = \underset{\omega}{\operatorname{argmax}}\, P(X|\omega)P(\omega). \tag{4.3}$$

Eq. 4.3 is the principle of Bayesian classification. $P(\omega)$ is the *prior* probability that nature is in state ω during observation of X, i.e., that X belongs to the class specified by ω. The distribution $P(\omega)$ can be chosen independently according to the problem domain. For example, the prior distribution can be chosen to be uniform, in which case $P(\omega)$ does not effectively appear in the maximization (Eq. 4.3) at all. The distribution $P(X|\omega)$ is the conditional probability density function (PDF) of X with respect to the state of nature (or class) ω. If its value at X can be modeled accurately for each class ω, the class can with high probability be correctly identified using Eq. 4.3. In Bayesian classification, each state of nature ω is parametrized by a PDF model λ_ω such that the model parameter set λ_ω is considered to be a sufficient statistic for estimating the state of nature ω. Eq. 4.3 then becomes

$$\hat{\omega} = \underset{\omega}{\operatorname{argmax}}\, P(X|\lambda_\omega)P(\omega). \tag{4.4}$$

In practical applications, the posterior probabilities (likelihoods) $P(X|\lambda_\omega)$ of long sequences frequently grow so small as to cause numerical problems. To solve this problem,

logarithmic probabilities are frequently used instead. Eq. 4.4 can be equivalently written as

$$\hat{\omega} = \underset{\lambda_\omega}{\operatorname{argmax}} \left(\log P(X|\lambda_\omega) + \log P(\omega) \right), \tag{4.5}$$

Being a monotonic function, the logarithm completely preserves any relative ordering of PDF values.

Gaussian and Minimum Distance Classification

The normal density, or Gaussian density, is very often used in pattern classification. Its popularity can be justified by the Central Limit Theorem, which states that, under various conditions, a random variable that is the sum of independent random variables will be distributed approximately normally [28] [47].

The univariate normal density is given by

$$p(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{x-\mu}{\sigma} \right)^2 \right) \tag{4.6}$$

The multivariate Gaussian density in d dimensions is given by

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \right) \tag{4.7}$$

where $\boldsymbol{\mu}$ is the $(d \times 1)$ *mean vector* and $\boldsymbol{\Sigma}$ is the $(d \times d)$ *covariance matrix*. The univariate Gaussian density is a special case of Eq. 4.7 in which $d = 1$. For $d > 1$, two forms of the covariance matrix are commonly used: a *full* covariance matrix can have any element nonzero, while in a *diagonal* covariance matrix only the *variance* parameters on the main diagonal are nonzero and the rest of the covariance parameters are zero.

A Gaussian classifier, based on the Bayesian principle, uses a Gaussian distribution parametrized as $\lambda_\omega = \{\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega\}$ as the PDF model for each class ω. Figure 4.1 illustrates the PDFs of two classes in the one-dimensional case. As there is notable overlap in the PDFs, the expected classification error rate is greater than zero. Using Eq. 4.4 with equal priors, the classification boundary would be placed as shown by the dashed line in Figure 4.1. Integrating the total area of the shaded regions gives the minimum attainable asymptotic error rate (for these particular classes and class PDFs), also known as the *Bayes error rate*. The Bayes error is a measure of the "best possible" classification error rate using the particular features and no additional information. It can be calculated if the class PDFs are known.

A Gaussian classifier uses Eq. 4.7 to model $P(X|\omega)$ as $P(X|\lambda_\omega) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\mu}_\omega, \boldsymbol{\Sigma}_\omega)$. Because this is a product of a large number of probabilities (in the range $(0, 1)$), direct computation of the likelihood quickly leads to numerical underflow even with a relatively small

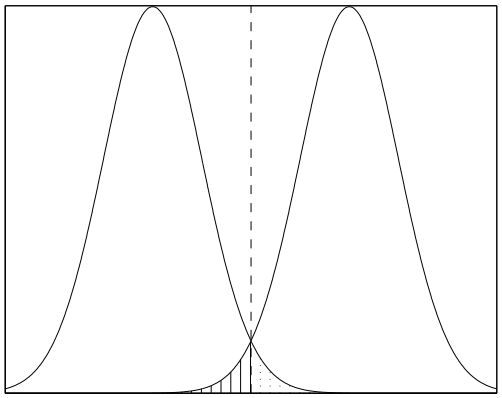


Figure 4.1: Univariate normal probability density functions (PDFs) of two classes. The vertical dashed line shows the optimal class decision boundary with equal priors. The shaded regions indicate the areas of classification error with the optimal decision boundary.

number of observations N. Thus, the *log likelihood* is a better practical measure; The log likelihood for observation sequence X with Gaussian PDF λ_ω is given by

$$\log P(X|\lambda_\omega) = \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\lambda_\omega) = N \log C - \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu}_\omega)'\boldsymbol{\Sigma}_\omega^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_\omega) \quad (4.8)$$

where the constant (with respect to X) term C is given by

$$C = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_\omega|^{1/2}} \quad (4.9)$$

and its logarithm by

$$\log C = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log(|\boldsymbol{\Sigma}_\omega|) \quad (4.10)$$

Multivariate Gaussian classification in its most general form now proceeds by substituting Eq. 4.8 into Eq. 4.5.

When the class prior probabilities are assumed equal, so that the value of Eq. 4.5 is completely determined by the likelihood in Eq. 4.8, there are certain special cases of Gaussian classification worth considering. Let us consider the case in which each class shares a common *global* covariance matrix $\boldsymbol{\Sigma}$, such that the term $N \log C$ becomes equal for all classes in Eq. 4.8. This being the case, also the constant factor $(1/2)$ of the second term on the right hand side of Eq. 4.8 becomes unimportant in view of the maximization in Eq. 4.5.

Maximizing the value of Eq. 4.8 with respect to λ now becomes equivalent to *minimizing*

$$D = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{\mu}_\omega)' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_\omega) \tag{4.11}$$

For a single observation ($N = 1$), minimization of Eq. 4.11 is equivalent to minimizing the square of the *Mahalanobis distance* [120]. The Mahalanobis distance is given by

$$d_M(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \tag{4.12}$$

and because it is always nonnegative, minimization of Eq. 4.11 with $N = 1$ is also equivalent to minimizing the Mahalanobis distance itself. This motivates the use of *minimum distance classifiers*, which decide for each test vector the class of the reference pattern that is closest in terms of the chosen distance measure. Minimum Mahalanobis distance classification has been shown above to be equivalent to Gaussian (Bayesian) classification using equal class priors and a global covariance matrix. This kind of classifier supports class distributions with shapes like arbitrarily oriented *hyperellipsoids*, as manifested by the diagonally oriented point cloud shown in Figure 4.2. It should be noted that when multiple vector observations are classified simultaneously (under the assumption that they all belong to the same class) using Gaussian classification, equivalent distance-based classification requires the use of a sum of *squared* Mahalanobis distances as the quantity to be minimized (Eq. 4.11).

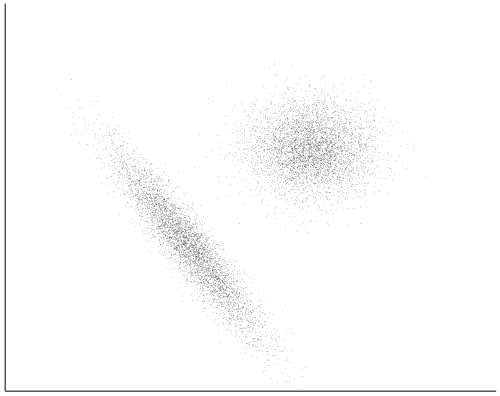


Figure 4.2: Point clouds obtained by sampling two Gaussian densities. The ellipsoidal cluster originates from a density with a full covariance matrix. The circular (spherical) cluster originates from a density with a diagonal, equal-variance covariance matrix of the form $\sigma \boldsymbol{I}$.

Now consider further constraints on the global covariance matrix: if it is made diagonal, the Mahalanobis weighting matrix is diagonal consisting of the inverses of the feature variances on the main diagonal. Such a distance is known as the *weighted Euclidean distance*. The minimum weighted Euclidean distance classifier is also commonly known as the *Naive-Bayes* classifier [28]. This classifier structure supports hyperellipsoidal class distributions whose axes align with the feature axes.

Finally, the minimum *Euclidean distance* condition follows, as another special case of the Mahalanobis distance, by assuming equal variances for each of the d features such that $\boldsymbol{\Sigma} = \sigma \boldsymbol{I}$, where $\boldsymbol{I}$ is the $(d \times d)$ identity matrix. Minimum Euclidean distance classification of a single observation thus corresponds to Gaussian classification assuming equal priors, a global diagonal covariance matrix and equal variance for each feature. The class distributions are thus implicitly assumed to be *compact* (as opposed to *elongated*) and *spherical*, according to the shape of diagonal-covariance, equal-variance Gaussian distributions (such as the circular point cloud shown in Figure 4.2). Such a classifier is rather simplistic, but should work well if a feature representation is found such that the desired classes form fairly distinct and compact clusters in the feature space.

Classification using GMMs

A unimodal multivariate distribution, such as the multivariate Gaussian distribution, is frequently an inadequate model for representing probability distributions with complex shapes. The situation can be helped by instead using a *mixture* of such distributions. A *Gaussian Mixture Model* (GMM) with J *mixture components* is given by

$$p(\boldsymbol{x}|\lambda) = \sum_{j=1}^{J} p_j b_j(\boldsymbol{x}) \tag{4.13}$$

where each component distribution

$$b_j(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)'\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j)\right) \tag{4.14}$$

is a multivariate Gaussian distribution parametrized by a $(d \times 1)$ mean vector $\boldsymbol{\mu}_j$ and a $(d \times d)$ covariance matrix $\boldsymbol{\Sigma}_j$. The p_j are the mixture *weights*, or component priors. Such a mixture model can approximate distributions with arbitrarily complex shapes, provided that the number of mixture components J is sufficiently large [104].

An important consideration in specifying a GMM is its covariance structure. The following definitions can be used to characterize covariance [104]: if a GMM has *nodal* covariance it means that every component has its own covariance matrix, *grand* covariance refers to a common covariance matrix shared by all components of the same GMM, and *global* covari-

ance refers to a single covariance matrix used in all contexts. In addition, each covariance matrix can be either full or diagonal.

In Bayesian classification, the observation PDF corresponding to each class (state of nature) ω_i is modeled by a GMM $\lambda_i = \{p_1, \ldots, p_J, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_J, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_J\}$. According to Eq. 4.4, classification of a set of feature vectors X thus requires computation of the quantities $P(X|\lambda_i)$. For any GMM λ, this likelihood is given by

$$P(X|\lambda) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\lambda), \tag{4.15}$$

where $p(\boldsymbol{x}_n|\lambda)$ is given by Eq. 4.13. The formula for the log likelihood of a data set X is thus

$$\log P(X|\lambda) = \sum_{n=1}^{N} \log \sum_{i=1}^{J} p_i b_i(\boldsymbol{x}_n) \tag{4.16}$$

The details of training GMMs are given in Section 4.2.

Time Series Segmentation and Classification using HMMs

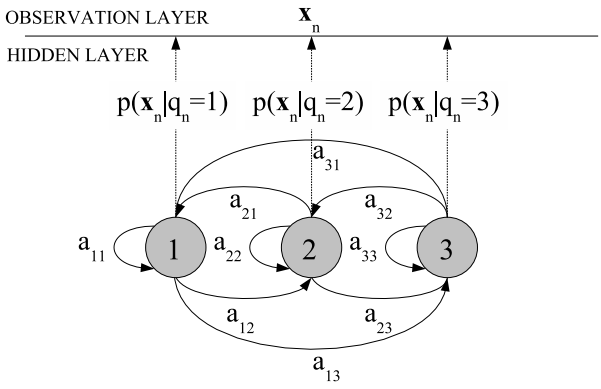


Figure 4.3: A diagram of an ergodic HMM with three states.

Hidden Markov models (HMMs) [101] are a class of stochastic models for (vector or scalar) time series. In applications involving supervised segmentation and classification of a multivariate time series, HMMs are frequently employed in a Bayesian supervised

classification framework. The most prominent such application area is automatic speech recognition (ASR), where HMM-based Bayesian classification is currently the prevalent technology.

In the Bayesian classification framework, the problem is to find the labeling (symbol sequence) $\hat{W} = \{\omega_1, \omega_2, \ldots, \omega_N\}$, that is most likely to have produced the observed feature vector sequence $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ [62]:

$$\hat{W} = \arg\max_W P(W|X) \tag{4.17}$$

Applying Bayes' formula, we have

$$\hat{W} = \arg\max_W \frac{P(W)P(X|W)}{P(X)} \tag{4.18}$$

Again, $P(X)$ has no effect on the maximization in Eq. 4.18 and we have

$$\hat{W} = \arg\max_W P(W)P(X|W) \tag{4.19}$$

In ASR terminology, $P(W)$ corresponds to the *language model* while $P(X|W)$ is the *acoustic model*. HMMs are frequently used in acoustic modeling to model the PDF $P(X|W)$.

An HMM can be thought to consist of the *hidden layer* and the *observation layer*. In the hidden layer, the model has a discrete-time state process q_n governed by the rules of a first-order Markov chain (hence the name hidden Markov model). At any time instant n, the state process is in one of K states; however, the state q_n is not directly observable. Instead, the observation layer presents the time series $\boldsymbol{x}_n$, where each observation $\boldsymbol{x}_n$ is assumed to have been drawn from a state-specific probability distribution $p(\boldsymbol{x}_n|q_n = i, X_1^{n-1})$, which can be continuous or discrete, depending on the type of the HMM.

To summarize, an HMM with K states has three main elements:

- The *state transition probabilities*, which can be expressed as a $(K \times K)$ transition probability matrix $\boldsymbol{A} = (a_{ij})$, where each element is given by

  $$a_{ij} = P(q_{n+1} = j|q_n = i), \qquad 1 \leq i, j \leq K,$$

  i.e., the conditional probability of a transition to state j at the next time instant, given that the model is currently in state i.

- The *initial state probability distribution*, which can be expressed in vector form as $\boldsymbol{\pi} = [\pi_1 \quad \pi_2 \quad \ldots \quad \pi_K]$, where

  $$\pi_i = P(q_1 = i), \qquad 1 \leq i \leq K,$$

i.e., the probability that the model is in state i at the initial time instant $n = 1$.

- The *state-specific observation probability distributions* $b_i(\boldsymbol{x}_n) = p(\boldsymbol{x}_n|q_n = i, X_1^{n-1})$, which may have various forms. For example, these distributions can be discrete distributions of vector quantization indices, continuous Gaussian distributions, Gaussian mixtures or vector autoregressive Gaussian models (see Section 4.2.3). The parameter sets of these distributions will be denoted by B_i, such that each parameter set B_i parametrizes the probability distribution $b_i(\boldsymbol{x}_n)$.

Thus, the HMM is parametrized by the parameter set $\lambda = \{\boldsymbol{\pi}, \boldsymbol{A}, B_1, \ldots, B_K\}$. Figure 4.3 illustrates an HMM with three states. The HMM represented in the figure is called *ergodic* on the condition that each $a_{ij} > 0$, meaning that every transition between two states (including self-transitions) is possible. Another commonly employed type of HMM is a *left-to-right* model, where transitions are possible only in one "direction". One way to convert the HMM in Figure 4.3 to a left-to-right model would be to enforce the condition $a_{ij} = 0$, $i > j$. Left-to-right models are best suited to modeling short time series segments of certain temporal structure, such as words or phonemes, whereas ergodic models are convenient for modeling class sequences in arbitrarily long signals (although an ergodic model can be used for both purposes).

In practice, the training of HMMs is an important issue. Because HMM training can also be used for unsupervised classification, this discussion is postponed until Section 4.2. For now, let us consider an HMM λ whose parameters have been previously set - by some training procedure - to reasonable values for modeling the desired PDF of a vector time series.

As a practical application, consider a simple example of isolated segment recognition (e.g. isolated word recognition), where it is known beforehand that the feature vector sequence X corresponds to a single class segment, and hence a single class symbol, so that $W = \{\omega_1\}$ in Eq. 4.19. If each possible class label ω is associated with an HMM λ_ω, the classification problem becomes equivalent to determining which model λ_{ω_1} gives the highest probability $P(\omega_1)P(X|\lambda_{\omega_1})$. This requires the computation of the probability $P(X|\lambda)$ for a given HMM λ, a central problem in dealing with HMMs. The most common algorithm for computing this quantity is known as the *forward algorithm*. It makes use of a variable

$$\alpha_n(i) = P(X_1^n, q_n = i|\lambda), \qquad 1 \leq i \leq K \tag{4.20}$$

i.e., the joint probability of the observation sequence until time n and the hidden state at time n given the hidden Markov model λ. The algorithm proceeds as follows [101]:

1. $\alpha_1(i) = \pi_i b_i(\boldsymbol{x}_1), \qquad 1 \leq i \leq K$

2. For $n = 1$ to $N - 1$:
   $\alpha_{n+1}(j) = \left(\sum_{i=1}^{K} \alpha_n(i)a_{ij}\right) b_j(\boldsymbol{x}_{n+1}), \qquad 1 \leq j \leq K$
3. $P(X|\lambda) = \sum_{i=1}^{K} \alpha_N(i)$

In practice, to avoid numerical underflow, the forward prodedure requires intermediate scaling of the forward variable, as discussed in [101].

An alternative to the forward computation has been presented by Hamilton in the field of econometrics [47] [45]. It uses the intermediate quantities $\gamma_{n+1|n}(i) = P(q_{n+1} = i|X_1^n, \lambda)$ and $\gamma_{n|n}(i) = P(q_n = i|X_1^n, \lambda)$. The algorithm proceeds as follows:

1. $\gamma_{1|0}(i) = \pi_i, \qquad , 1 \leq i \leq K$
2. For $n = 1$ to N:
   $P(\boldsymbol{x}_n|X_1^{n-1}, \lambda) = \sum_{i=1}^{K} \left(\gamma_{n|n-1}(i)b_i(\boldsymbol{x}_n)\right)$
   $\gamma_{n|n}(i) = \frac{\gamma_{n|n-1}(i)b_i(\boldsymbol{x}_n)}{\sum_{k=1}^{K}\left(\gamma_{n|n-1}(k)b_k(\boldsymbol{x}_n)\right)}, \qquad 1 \leq i \leq K$
   $\gamma_{n+1|n}(j) = \sum_{i=1}^{K} \left(a_{ij}\gamma_{n|n}(i)\right), \qquad 1 \leq j \leq K$

By the chain rule of probabilities, $P(X|\lambda)$ is given by

$$P(X|\lambda) = \prod_{n=1}^{N} P(\boldsymbol{x}_n|X_1^{n-1}, \lambda) \tag{4.21}$$

Thus, the desired likelihood is obtained as a product of values computed on the first line of the main loop in the above algorithm. Again, however, to avoid numerical underflow, it is better to instead compute the log likelihood as the sum of the logarithms of these values:

$$L(X, \lambda) = \log P(X|\lambda) = \sum_{n=1}^{N} \log P(\boldsymbol{x}_n|X_1^{n-1}, \lambda) \tag{4.22}$$

The ability to evaluate $P(X|\lambda)$ is sufficient for automatic classification on the condition that X is known to represent a single class, i.e., that the input has been somehow segmented beforehand. An ASR example of such a task is isolated word recognition, in which each input is assumed to be a single word (perhaps the words have been segmented apart from each other by an *endpoint detector* [57] [65]). If this is not the case and W in Eq. 4.19 is a sequence of class symbols, such as in continuous speech recognition, the HMM recognizer must also take care of the segmentation. Such a recognizer uses a composite HMM network where either individual states or smaller sub-HMMs (such as word HMMs) correspond to the different classes. The recognition problem becomes that of finding an optimal decoding of the input feature vectors to a sequence of HMM states (or a sequence of groups of

states corresponding to different sub-HMMs). This is equivalent to finding the maximum likelihood path through a *trellis* of HMM states [1]. A path through a trellis is a function, defined for each discrete time instant, that tells us the identity of the HMM state occupied at each time instant. Such a path can be denoted as $Q_1^N = \{q_1, q_2, \ldots, q_N\}$, where each $q_n \in \{1, \ldots, K\}$. For a composite HMM consisting of more than one sub-HMM, the observation labeling Q_1^N in terms of individual states can be easily converted into another labeling expressed in terms of the sub-HMM identities occupied at each time instant.

For the above explained purpose, the well-known *Viterbi algorithm*, an implementation of the *dynamic programming* optimization principle, is usually used [101]. This algorithm makes use of the quantity $\delta_n(i) = \max_{Q_1^{n-1}} P(Q_1^{n-1}, q_n = i, X_1^n|\lambda)$.

1. $\delta_1(i) = \pi_i b_i(\boldsymbol{x}_1), \qquad 1 \leq i \leq K$

   $\psi(i) = 0, \qquad 1 \leq i \leq K$

2. For n = 2 to N:

   $\delta_n(j) = \max_{1 \leq i \leq K} \left(\delta_{n-1}(i) a_{ij}\right) b_j(\boldsymbol{x}_n), \qquad 1 \leq j \leq K$

   $\psi_n(j) = \operatorname{argmax}_{1 \leq i \leq K} \left(\delta_{n-1}(i) a_{ij}\right), \qquad 1 \leq j \leq K$

3. $P^* = \max_{1 \leq i \leq K} \left(\delta_N(i)\right)$

   $q_N^* = \operatorname{argmax}_{1 \leq i \leq N} \left(\delta_N(i)\right)$

4. For n = N-1 to 1:

   $q_n^* = \psi_{n+1}(q_{n+1}^*)$

In the first loop, for each pair of observation $\boldsymbol{x}_n$ and HMM state j, corresponding to a node (n, j) in the trellis, the Viterbi algorithm finds the maximum likelihood $\delta_n(j)$ according to the best partial path $\{q_1, q_2, \ldots, q_n = j\}$ (constrained to contain node (n, j)). It also keeps track of the predecessor node along the said path (in $\psi_n(j)$). After termination, the maximum joint likelihood along the best path, $P^* = \max_{Q_1^N} P(Q_1^N, X|\lambda)$, is known. Lastly, the *backtracking* loop finds out the maximum likelihood path $\{q_1^*, q_2^*, \ldots, q_N^*\} = \operatorname{argmax}_{Q_1^N} P(Q_1^N, X|\lambda) = \operatorname{argmax}_{Q_1^N} P(Q_1^N|X, \lambda)$ itself, using the stored predecessor node information.

The Viterbi algorithm finds the maximum likelihood state sequence that best explains the observations X. It is sometimes more desirable to evaluate the likelihoods of the states

[1] A trellis is a two-dimensional point grid where the other (usually visualized as the horizontal) axis is the discrete time of the observations and the other axis is the HMM state identity.

individually at each time instant; importantly, such quantities are useful in the automatic training of HMM parameters by the EM algorithm (see Section 4.2.3). For this purpose, the forward algorithm is followed by the *backward* computation which uses the backward variable $\beta_n(i) = P(X_{n+1}^N | q_n = i, \lambda)$ [101]:

1. $\beta_N(i) = 1, \qquad 1 \leq i \leq K$
2. For n = N-1 to 1:
   $\beta_n(i) = \sum_{j=1}^K a_{ij} b_j(\boldsymbol{x}_{n+1}) \beta_{n+1}(j)$

The individual state occupancy posterior probabilities are now given by [101]

$$\gamma_n(i) = P(q_n = i | X, \lambda) = \frac{\alpha_n(i)\beta_n(i)}{\sum_{j=1}^K \alpha_n(j)\beta_n(j)}. \tag{4.23}$$

The posterior probabilities of state transitions are given by

$$\xi_n(i,j) = P(q_n = i, q_{n+1} = j | X, \lambda) = \frac{\alpha_n(i) a_{ij} b_j(\boldsymbol{x}_{n+1}) \beta_{n+1}(j)}{\sum_{i=1}^K \sum_{j=1}^K \alpha_n(i) a_{ij} b_j(\boldsymbol{x}_{n+1}) \beta_{n+1}(j)}. \tag{4.24}$$

An alternative method for the computation of the likelihoods $P(q_n = i|X, \lambda)$,$1 \leq i \leq K$, proposed by Kim [70] [47], is applicable after first running Hamilton's algorithm:

1. $\gamma_N(i) = \gamma_{N|N}(i), \qquad 1 \leq i \leq K$
2. For n = N-1 to 1:
   $\gamma_n(i) = \gamma_{n|n}(i) \sum_{j=1}^K \left(a_{ij} \frac{\gamma_{n+1}(j)}{\gamma_{n+1|n}(j)} \right), \qquad 1 \leq i \leq K$

The desired probabilities are now given by the $\gamma_n(i)$. To obtain the probabilities corresponding to the $\xi_n(i,j)$, without having to modify the above algorithm, one could use for example the approximation

$$\xi_n(i,j) \approx \gamma_n(i)\gamma_{n+1}(j) \tag{4.25}$$

in which the state variables at two successive times, q_n and q_{n+1}, are assumed conditionally independent given the knowledge of both X and λ. This assumption can be expressed as $P(q_{n+1} = j | q_n = i, X, \lambda) \approx P(q_{n+1} = j | X, \lambda) = \gamma_{n+1}(j)$. The intuitive idea behind this approximation is that X and λ should contain sufficient information for reliably estimating q_{n+1}, so that even additional knowledge of q_n does not particularly affect the posterior probability distribution of q_{n+1}. Whether this is a valid argument may depend on whether the goal is indeed to infer the values of the hidden state variable q_n, which might be

the case in, e.g., unsupervised classification but requires that the observation distributions in the feature space are sufficiently separable among different HMM states. Alternatively, the HMM may have been designed for just modeling a probability density function of a time series. In the latter case, only the HMM likelihood has been considered to be of interest and the specific state inferences may be unreliable or meaningless from a classification perspective.

4.1.2 Nearest Neighbor Classification

A difficulty in Bayesian classification is that it requires reliable estimates λ_ω of the class PDFs to evaluate $P(X|\omega) \approx P(X|\lambda_\omega)$ for Eq. 4.3. Several alternative pattern classification methods exist that do not require explicit PDF modeling. One popular alternative is the *k-nearest-neighbor* (kNN) rule, which is a conceptually simple, yet powerful, supervised classification method. The procedure is the following:

1. *Training:* Select a training set of N patterns and store their parameters as well as the associated class labels
2. *Classification:* For each test vector $\boldsymbol{x}$, find the k nearest training patterns (irrespective of category label) using the chosen distance measure
3. Determine the class label that appears most often in the set of the k nearest training patterns and assign it to the test vector $\boldsymbol{x}$

To avoid ties, the value of k should not be a multiple of the number of classes M [120].

When $k = 1$, the resulting kNN classifier is sometimes termed the nearest-neighbor (NN) classifier. For this kind of classifier, it can be shown [28] that the classification error probability P_{NN} is bounded by

$$P_B \leq P_{NN} \leq P_B \left(2 - \frac{M}{M-1} P_B\right) \leq 2P_B \tag{4.26}$$

where P_B is the optimal Bayesian error probability.

In a two-class problem ($M = 2$), it can be shown for the general kNN classifier that

$$P_B \leq P_{kNN} \leq P_B + \frac{1}{\sqrt{ke}}, \tag{4.27}$$

suggesting that the performance of kNN tends to the optimal Bayes error as $k \to \infty$ [120].

An intuitive understanding of why kNN classification works can be gained by considering the fact that the k nearest neighbor training patterns are contained, in the metric space

defined by the used distance metric (e.g., the Euclidean distance), within a hypersphere of radius r, where r is the distance between the test vector $\boldsymbol{x}$ and the kth nearest training pattern. As the number of training patterns N is increased, the space becomes more crowded and the expected value of r decreases, i.e., the hypersphere becomes smaller. With a sufficiently large training set, r becomes small enough so that the PDFs of all classes can well be approximated to be constant within the hypersphere. When k samples are drawn from within such a hypersphere, the majority of these k samples are expected to belong to the category whose PDF within the hypersphere is dominant compared to the other categories. Moreover, the larger the number of samples k drawn from within the hypersphere, the more likely it is that the correct category (the one with the dominant PDF inside the hypersphere) is selected by the majority voting. At the same time, of course, the value of k should be much smaller than N.

The above discussion highlights two important parameters of kNN classification: the size of the training set N and the value of k. In general, both should be as large as possible (while $k \ll N$) in order to achieve a performance close to the Bayes error P_B. However, the size of the training set is directly related to the computational and storage complexity of the kNN algorithm, as every training pattern must be memorized and the nearest k patterns must be found for each test vector. These factors limit the size of the training set. Efficient search algorithms have been developed to limit the computational burden, while optimal methods for selecting the training patterns have been developed to limit the storage requirements. Some references for these methods are listed in [120].

Dynamic Time Warping

Dynamic time warping (DTW) is a method for computing a meaningful, time-aligned distance between two templates that are sequences of feature vectors and can depict temporal events. Time alignment is needed because the duration and temporal organization of units belonging to each event may vary. DTW is a special case of dynamic programming and thus related to the Viterbi algorithm which achieves a similar goal of time alignment with HMMs.

DTW as such is well suited to isolated word recognition (IWR) and similar applications in which both the training and testing data has either been readily segmented into events, or can be easily segmented apart by automatic methods. The latter case is true in IWR when the words are spoken separately, with clear pauses between the words. Connected word recognition (for, e.g., digit strings with a known number of digits) can be achieved via DTW by incorporating *level building* [87]. DTW was widely researched in ASR prior to the advent of HMMs. Because HMMs are much better suited to continuous speech recognition, the central research topic of modern ASR, HMM methods have since succeeded DTW-

based templates as the dominant method of acoustic modeling in ASR. The relatively simple DTW continues, however, to be a useful classification method in recognition tasks in which the segmentation can be performed by other means.

The idea of DTW is to align two templates, a test template $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_T}\}$ consisting of N_T feature vectors and a reference template $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{N_R}\}$ consisting of N_R feature vectors, by warping their time axes in order to synchronize similar units in the templates. To accomplish the warping, template matching is cast as a dynamic programming optimization problem where one finds the minimum cost path through a two-dimensional grid of $N_T \times N_R$ nodes. Each node (i, j) corresponds to a pair of feature vectors $(\boldsymbol{x}_i, \boldsymbol{y}_j)$ and has associated node cost $d(\boldsymbol{x}_i, \boldsymbol{y}_j)$. The total cost, and the DTW distance, is the sum of the node costs along the best path:

$$D(X, Y) = \sum_{k=1}^{K} d(\boldsymbol{x}_{I(k)}, \boldsymbol{y}_{J(k)}) \tag{4.28}$$

In Eq. 4.28, the summation is along the length of the optimum path found by DTW, $I(k)$ is the mapping from the path index to the test template index and $J(k)$ is the mapping from the path index to the reference template index. The node cost is a distance between two feature vectors and is computed according to the chosen distance measure. The choice of this distance measure depends on the type of feature vectors used; the squared Euclidean distance may, in general, be the most appropriate choice [2].

In the most conventional *constrained endpoints* version of DTW, the path is required to start from grid node $(1, 1)$ and end in node (N_T, N_R) [90]. The *local continuity constraints* are an essential part of DTW that determine which transitions are allowed in moving from one node to another. In the following, the local continuity constraints are specified by defining a set-valued function $c(j)$, $1 \leq j \leq N_R$, such that transitions from node $(i-1, k)$ to node (i, j) are only possible if $k \in c(j)$. A fairly general constrained endpoints DTW algorithm can now be stated as follows [57]:

1. Initialization: set $D(1, 1) = d(\boldsymbol{x}_1, \boldsymbol{y}_1)$, $B(1, 1) = 1$ and $D(1, j) = \infty$, $2 \leq j \leq N_R$.
2. For $i = 2$ to N_T:

$$D(i, j) = \min_{k \in c(j)} (D(i-1, k) + d(\boldsymbol{x}_i, \boldsymbol{y}_j)), \qquad 1 \leq j \leq N_R$$

$$B(i, j) = \operatorname*{argmin}_{k \in c(j)} (D(i-1, k) + d(\boldsymbol{x}_i, \boldsymbol{y}_j)), \qquad 1 \leq j \leq N_R$$

[2] The sum of squared Euclidean distances reflects the negative log likelihood of a probabilistic template that consists of spherical Gaussian PDF models; see Section 4.1.1.

After running the algorithm, the DTW distance between the two templates is given by $D(N_T, N_R)$. The storing of the $B(i, j)$ values is optional, as their only purpose is to make backtracking possible in order to find the warping path, if required. It should be noted that the above algorithm can not handle the type of local continuity constraints according to which one test template vector could be matched with two or more different reference template vectors. However, many sets of local continuity constraints presented in the literature do not allow such transitions [100].

The training templates are typically clustered [100] to limit computation in the recognition phase. This involves computing pairwise DTW distances between all training templates corresponding to the same class. For each class, some number of clusters are generated and one reference template is chosen from each cluster. The representative template for each cluster is chosen, e.g, as the one with the minimum average distance between it and every other template in the same cluster [100]. During the recognition phase, the representative templates are used in some variation of nearest neighbor classification.

4.1.3 Other Forms of Supervised Pattern Recognition

Artificial Neural Networks

Artificial neural networks (ANNs) [50] [86] [120] [28] are a simplification of real, biological neural networks. They can perform signal processing and pattern recognition tasks and can be used for both regression and classification. Here, they are briefly discussed from the pattern classification perspective.

The fundamental processing element of an ANN is the *cell*, whose function resembles that of a real neuron: it must integrate input from other cells and communicate the integrated signal (*cell activity*). The physiological analogue to ANN cell activity is the firing rate of a real neuron. The activity of cell i can be expressed as $F(\boldsymbol{x}, \boldsymbol{w}_i, z_i)$, where $\boldsymbol{x} = [x_1 \quad \dots \quad x_M]'$ is the vector of inputs from M other cells of the network, $\boldsymbol{w}_i = [w_1 \quad \dots \quad w_M]$ is the set of weights characterizing the strengths of synaptic connections, and z_i is an intrinsic threshold that determines the range of values for which the cell will be driven from low to high activity [86]. The activity $F(\boldsymbol{x}, \boldsymbol{w}_i, z_i)$ can be decomposed into two parts: A scalar quantity d_i which is a *measure of dissimilarity* of the pattern of input activity $\boldsymbol{x}$ to the set of weights $\boldsymbol{w}_i$ connecting cell i to M other cells in the network, and the *activation function* $F(d_i - z_i)$. For example, the dissimilarity measure can be based on an inner product measure $d_i = \boldsymbol{w}_i \boldsymbol{x}_i$ or the Euclidean distance measure $d_i = \|\boldsymbol{w}' - \boldsymbol{x}_i\|$. Common activation functions include the linear, threshold linear, step, sigmoid, and Gaussian activation functions. Unlike the threshold linear and step activation functions, the Gaussian and sigmoid functions are differentiable. This permits gradient descent type learning algo-

rithms. These nonlinear activation functions are preferred over linear ones (which are also differentiable) because ANNs with nonlinear activation functions are able to compute more complex mappings than ANNs with linear activation functions.

The simplest neural network is the *perceptron*, which is a binary classifier consisting of one ANN cell only. Two fundamental learning algorithms exist for the perceptron. The *perceptron learning rule* assumes linearly separable classes and performs poorly when class regions overlap or share a nonlinear boundary. The *LMS algorithm*, originally introduced in [124], is quite similar to the perceptron rule but tries to minimize the mean squared error instead of the number of misclassifications. It allows the perceptron to converge to a stable state even when the class regions are not linearly separable. For more complex network topologies, the main categories are [86]:

- *recurrent*; arbitrary connections between cells in different layers are allowed, enabling feedback loops
- *non-recurrent*; recurrent connectivity (feedback) is not permitted, i.e. information propagates only in the "forward" direction (from the input towards the output)
- *feedforward*; a special type of non-recurrent networks where recurrent connectivity is not permitted and forward connectivity is permitted only between neighboring layers

In general, *supervised* learning in ANNs follows these general steps:

1. a stimulus pattern is presented at the ANN input
2. a desired "target" response pattern is presented at the ANN output
3. if the response pattern of the network does not match the target response pattern, the network is corrected by modifying the network weights (the ω_i's) to reduce the difference between the observed and target patterns

Multilayer perceptrons (MLPs) are (usually feedforward) networks that map the input pattern (obtained from the previous layer) to a new space at each layer. In classification, the goal is to have an eventual representation in which the classes are separable by linear decision boundaries. Using this approach, MLPs can model complex class regions and nonlinear decision boundaries in the original feature space. Different learning algorithms give rise to subtypes of MLPs. The backward error propagation (backpropagation) MLP network is one of the most widely used nonlinear ANN classifiers.

Support Vector Machines

Support vector machines (SVMs) [19] [120] [28] are fundamentally binary classifiers, but any number of classes can be accommodated by combining binary SVM classifiers. The principle of SVM classification can be described by first considering linearly separable classes, i.e., two classes which can be perfectly separated using a linear hyperplane as a decision boundary. SVM training is based on the idea of maximizing the *margin* between any decision boundary and the closest observations at each side of the hyperplane, i.e., the goal is to maximize the distance from the closest class representative points to the decision boundary. These representatives are called *support vectors*. The optimization problem of designing a maximum margin hyperplane can be solved using Lagrange multipliers.

In the general case in which the classes are not separable even with a nonlinear decision boundary, the nonlinear SVM classifier effectively maps the feature vectors into a higher-dimensional space in which linear separation of the training set is possible. The margin is then maximized in the higher-dimensional space during the training procedure. Maximization of the margin in SVM training aims for improved *generalization* performance of the classifier when presented with previously unseen data. The specifics of SVM-based classification can be found, for example, in [19].

Decision trees

A decision tree [18] [28] [120] is a hierarchical classification scheme in which each test pattern is subjected to a sequence of questions, each of which corresponds to a node in the tree. The first question corresponds to the root node of the tree, usually displayed at the top of a tree graph visualization. The answer to each question determines the branch along which the sequence of questions proceeds. Upon reaching a *leaf node*, the test pattern receives the category label associated with the leaf node that was reached.

Every non-leaf node is said to perform a *split* of the feature vectors, as some population of feature vectors is effectively split into different sub-populations going down different branches. In the simplest form, the questions may be of the form, "is feature $x_i \leq \alpha$?" where α is some threshold value. Such a decision tree is said to be *binary*, as there are only two possible answers to each question. Furthermore, since the questions involve only one feature at a time, a tree classifier of this kind will produce decision boundaries that consist of portions perpendicular to the feature axes. CART (Classification and Regression Trees) is a systematic methodology for training decision trees, described in the book by Breiman *et al* [18].

Heuristic rule-based systems

Many practical classifiers are not based on any theory of pattern recognition, but are instead constructed manually based on knowledge and intuition. They are based on explicit, heuristic rules and are often hierarchical in nature. Some of these rule-based systems are like decision trees. The rule-based systems are important in practice as they provide a natural way of incorporating expert knowledge into the classification method. Their construction can be guided by expert knowledge and refined by experimentation and visual analysis of the features.

4.2 Unsupervised Classification

In the present work, a distinction is made between the concepts of *unsupervised classification* and *clustering*. The former is used as a general term for classification without the help of labeled training data. The latter is a special case of the former and means organizing patterns into groups or *clusters* based solely on their interrelations, i.e., similarities and differences between the patterns. This distinction is important when dealing with vector time series data. Ordinary clustering algorithms deal with the data as simply a *set* of observations and do not consider their temporal order. At the same time, there exist other unsupervised classification methods that do take the temporal order into account, thus considering the data as a *sequence* of observations instead.

Applications of ordinary cluster analysis in time series recognition typically occur after an explicit segmentation algorithm has detected segment boundaries, and the feature vectors for clustering have been computed from complete segments [3]. Some clustering algorithms can also be used for generating vector quantization (VQ) codebooks. In the latter case of *data modeling*, the mean squared error (MSE) is a relevant performance measure, given by

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{x}_n - \boldsymbol{\theta}_{y_n}\|^2. \tag{4.29}$$

where $\boldsymbol{\theta}_i$ is the codeword for the ith VQ codebook entry and y_n is the labeling of observations with VQ codewords. MSE measures the accuracy with which the actual data vectors can be reproduced using the obtained VQ codebook, i.e. the data modeling performance. In the context of cluster analysis, if a ground truth class labeling is available, another performance measure can be used. It could be called minimum misclassification

[3] Change detection algorithms applicable to such explicit segmentation are introduced in Section 4.3.

rate (MMR) and is given by

$$\mathrm{MMR} = 1 - \frac{1}{N}\sum_{i=1}^{K}\left(\max_{j}\sum_{n=1}^{N} I(y_n = i \text{ and } z_n = j)\right), \tag{4.30}$$

where y_n is the clustering result, z_n is an underlying "true" reference labeling, and $I()$ is an indicator function that assumes a value of 1 if the inside statement is true and a value of 0 otherwise. MMR assigns each found cluster to the true class with which it agrees most often and thus describes the agreement between the unsupervised classification and the ground truth classification. This simple measure, in effect, measures the *impurity* of the found clusters in terms of the true classes. It is appropriate for cases in which the number of automatically found classes is greater than or equal to the number of true reference classes.

The two arguably most popular families of clustering algorithms are *hierarchical clustering* and clustering based on *iterative function optimization*. A brief review of the most important methods belonging to these families is followed by discussion on model initialization for the iterative optimization methods. Next, the attention is turned to direct unsupervised classification of a time series based on HMM modeling. In the end of this section, the unsupervised methods are applied to speech/music discrimination.

4.2.1 Hierarchical Clustering

Instead of producing a single partition of the data, hierarchical clustering algorithms always produce a hierarchy of nested clusterings. Hierarchical clustering solutions can be visualized by a tree graph known as a *dendrogram* such as the one shown in Figure 4.4. The vertical axis in this graph corresponds, in general, to a measure of distance between two clusters. The joining of two clusters in the dendrogram graph is indicated at the corresponding threshold level. Cutting the dendrogram at any level results in some clustering solution.

There are two main approaches to constructing hierarchical clusterings. The *agglomerative* method starts with each observation in its own cluster (at the leaf branches of the dendrogram) and proceeds to successively merge clusters according to some criterion. The *divisive* method starts with all the observations in one cluster (at the root of the dendrogram) and proceeds to successively split the clusters. This brief presentation will consider only the agglomerative algorithms, as they are more popular in practice.

In principle, the general hierarchical agglomerative clustering scheme proceeds as follows [120]:

1. Initially put each observation into its own cluster (start at the leaf branches of the dendrogram).

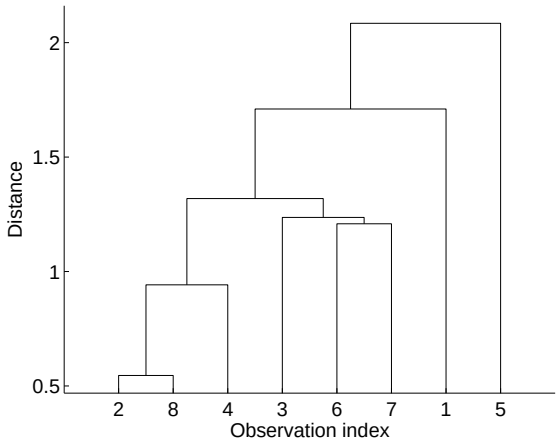


Figure 4.4: An example of a threshold dendrogram for a dataset of eight observations.

2. Among all possible pairs of clusters at the present level of the dendrogram, choose the one that minimizes a certain cluster dissimilarity function.

3. Go one level up the dendrogram.

4. For the current level of the dendrogram, produce a clustering which is equivalent to the clustering of the previous level with the exception that the two clusters for which the dissimilarity in step 2 was minimized are replaced by their union.

5. If all observations are not yet in the same cluster, go to 2. Otherwise, exit.

There are several variations of the hierarchical agglomerative scheme, which differ by the cluster dissimilarity criterion used in step 2 to determine *which clusters to merge* at each level. These cluster dissimilarities are computed by considering pairwise distances between observations belonging to different clusters. Because of this, hierarchical clustering is begun by computing a pairwise distance (triangular) matrix between all the observations, where the observation distance may be any distance measure that is considered relevant for the type of feature vectors used.

Two popular alternatives for the cluster dissimilarity criterion are *single linkage* and *complete linkage*. For merging, the former chooses the two clusters with smallest *minimum* distance between a pair of observations, while the latter chooses the two clusters with the smallest *maximum* pairwise distance. Because using single linkage, two clusters can be merged if they have *just one pair of points* close to each other, single linkage tends to produce elongated, chain-like clusters [120]. Complete linkage behaves differently; because it requires relative closeness of *every pair of points* in two clusters for the two clusters to be

merged, it produces *compact* clusters [120].

4.2.2 Iterative Optimization Clustering

Certain learning algorithms are based on the optimization of some cost function in order to *learn a cluster-based model for the data*. This makes them suitable to data modeling (vector quantizer design or PDF model estimation) as well as cluster analysis. Most often, the optimization is either a direct application of or an analogous procedure to the *EM algorithm* (from expectation/maximization). EM is a general principle that can be used to estimate PDF models from incomplete data in a maximum likelihood fashion [25]. "Incomplete data" means that not all relevant variables are observable, making closed form parameter estimation impossible. Although named "EM algorithm", it is not any specific algorithm but actually a class of algorithms based on the same principle. EM is an iterative hill-climbing optimization procedure whose each iteration increases the likelihood of the observed data (by updating the model parameters) until converging to a local maximum of the likelihood function. Whether the reached local maximum is also the global maximum, depends on the initialization of the model parameters.

Each iteration of EM estimation consists of two steps: the expectation (E) step and the maximization (M) step. During the E step, the current model is used to infer probability distributions for the unobserved variables. During the M step, these distributions are used in updating the current model parameters. Estimation procedures for both GMMs and HMMs are well-known applications of the EM principle. In addition, the very popular *k-means* algorithm and its variants can be considered *EM-style* classification algorithms [15].

K-means

K-means, also known as the isodata algorithm [120] [28], is an iterative algorithm applicable to both cluster analysis and VQ codebook generation. Define the variables $u_n(i)$, $1 \leq n \leq N$, $1 \leq i \leq K$, as $u_n(i) = 1$ if observation vector n belongs to cluster i and $u_n(i) = 0$ otherwise. In addition, define K cluster mean vectors (cluster *prototypes*) $\boldsymbol{\theta}_i$, $1 \leq i \leq K$. The algorithm makes use of a vector distance measure d, which is usually the Euclidean distance. This way, k-means produces compact, spherical clusters [120]. The k-means algorithm proceeds as follows:

1. Either: initialize the $\boldsymbol{\theta}_i$ and start from step 2 or initialize the $u_n(i)$ and start from step 3.

2. Update cluster assignment:

$$u_n(i) = \begin{cases} 1 & \text{if} i = \text{argmin}_j \, d(\boldsymbol{x}_n, \boldsymbol{\theta}_j), \qquad 1 \leq n \leq N, 1 \leq i \leq K \\ 0 & \text{otherwise} \end{cases}$$

3. Recompute cluster means:

$$\boldsymbol{\theta}_i = \frac{\sum_{n=1}^{N} u_n(i) \boldsymbol{x}_n}{\sum_{n=1}^{N} u_n(i)}, \qquad 1 \leq i \leq K$$

4. If convergence criterion is met or a specified number of iterations has been run, exit. Otherwise, go to step 2.

Step 2 assigns each observation to its closest prototype, thus reforming the clusters. Step 3 recomputes the prototypes as the mean vectors of each cluster. These steps are iterated until some convergence criterion is met. A sure convergence criterion for k-means is that any $u_n(i)$ does not change between two iterations, in other words, that the assignment of points to clusters has not changed. If this is the case, the prototypes can not change either, because they are simply the mean vectors of the clusters. As a consequence, neither the hitherto obtained clustering solution

$$y_n = \underset{i}{\text{argmax}} \, u_n(i), \qquad 1 \leq n \leq N, \tag{4.31}$$

nor the hitherto obtained VQ data model $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ can change anymore, regardless of the number of iterations computed.

K-means using the Euclidean distance is essentially a hill-climbing algorithm that can be shown to converge on a local minimum of the mean squared error (MSE) as defined in Eq. 4.29 [120] [15].

Whether the local optimum of the MSE is also the global optimum depends on the initialization of the parameters $\{\boldsymbol{\theta}_i\}$ in step 1 of the algorithm. In a later section, the initialization issue will be discussed jointly for two related algorithms: k-means and its probabilistic equivalent, EM re-estimation of a GMM.

Learning GMM parameters by the EM algorithm

The EM re-estimation of a GMM can be viewed as a probabilistic version of k-means, although k-means does not use mixture weights or covariance matrices. The analogy is visible in the descriptions of the two algorithms given in this and the previous section.

Denote by $q_n \in \{1, \ldots, J\}$ the identity of the mixture component that has produced observation $\boldsymbol{x}_n$. In the E step of the EM algorithm for GMM learning, the quantities

$\gamma_n(i) = P(q_n = i|\boldsymbol{x}_n, \lambda)$ are computed for $1 \leq n \leq N$, $1 \leq i \leq J$. These are used in the M step in updating the model parameters. The EM re-estimation is accomplished by the following algorithm:

1. Either: initialize λ and start from step 2 or initialize the $\gamma_n(i)$ and start from step 3.

2. E step:
$$\gamma_n(i) = \frac{p_i b_i(\boldsymbol{x}_n)}{\sum_{j=1}^{K} p_j b_j(\boldsymbol{x}_n)}, \qquad 1 \leq n \leq N, 1 \leq i \leq J$$

3. M step: Re-estimate mixture weights:
$$p_i = \frac{1}{N} \sum_{n=1}^{N} \gamma_n(i), \qquad 1 \leq i \leq J$$

   Re-estimate component mean vectors:
$$\boldsymbol{\mu}_i = \frac{\sum_{n=1}^{N} \gamma_n(i) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_n(i)}, \qquad 1 \leq i \leq J$$

   For each component i, compute the weighted residual matrix

$$\boldsymbol{W}_i = \left[(\boldsymbol{x}_1 - \boldsymbol{\mu}_i)\sqrt{\gamma_1(i)} \quad (\boldsymbol{x}_2 - \boldsymbol{\mu}_i)\sqrt{\gamma_2(i)} \quad \ldots \quad (\boldsymbol{x}_N - \boldsymbol{\mu}_i)\sqrt{\gamma_N(i)} \right] \tag{4.32}$$

   Re-estimate nodal covariance matrices:
$$\boldsymbol{\Sigma}_i = \frac{\boldsymbol{W}_i \boldsymbol{W}_i{}'}{\sum_{n=1}^{N} \gamma_n(i)}, \qquad 1 \leq i \leq J \tag{4.33}$$

   or a grand covariance matrix:
$$\boldsymbol{\Sigma} = \frac{\sum_{i=1}^{J} \boldsymbol{W}_i \boldsymbol{W}_i{}'}{N} \tag{4.34}$$

   In the case of diagonal covariance, only the variance parameters need to be re-estimated and the above matrix equations can be simplified.

4. If convergence criterion is met or a specified number of iterations has been run, exit. Otherwise, go to step 2.

For example, the convergence criterion could be one that compares some difference measure between parameter sets of two successive iterations to a threshold. GMM training can be used for clustering. In the mixture decomposition clustering scheme [120], the observations are assigned to clusters based on the quantities $\gamma_n(i)$.

Initialization of parameters

The initialization of the parameters is a relevant issue in iterative EM-style optimization, as the initial values determine the local optimum of the cost/likelihood function that will be reached. For data modeling purposes (PDF modeling or vector quantization), EM-style algorithms often produce an usable solution with almost any "sensible" initialization [127] [104]. In cluster analysis, the initialization may be more important. When clustering data by estimating the parameters of a cluster model consisting of compact Gaussian-like clusters, the initialization may be particularly important if the data distribution does not quite correspond to the assumptions of the cluster model (compactness and normality of the individual clusters) or when the clusters are not well separated. In such a case, different initializations can lead the re-estimation to converge on very different clustering solutions. Several popular initialization approaches have been used with the k-means algorithm. Because GMM learning can be viewed as just a probabilistic, soft-threshold version of k-means, most of the initialization methods are applicable to GMM learning also. Although a GMM has additional parameters to initialize compared to the VQ model estimated by k-means, the initialization of the mean vectors is frequently more important than the initialization of the GMM mixture weights and covariances. The weights can be initialized uniformly as $p_i = 1/J$, while most reasonably small non-zero covariance values typically work well in initialization.

Fisher's Iris dataset [34] will be used to illustrate the properties of different clustering methods. Figure 4.5 shows two dimensions of this originally four-dimensional dataset, i.e., the measured petal length and petal width of 150 flowers belonging to three different species of the genus *Iris*. The ground truth class labeling of the two-dimensional observations represented by the points is indicated by using three different markers.

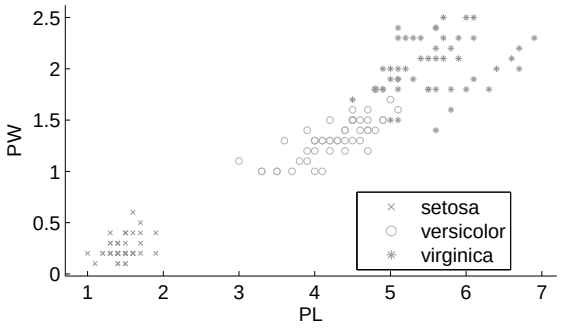


Figure 4.5: The petal length and petal width dimensions of Fisher's Iris dataset.

Arguably the most typical way of initializing the cluster centers in k-means is *random selection*. It is performed by choosing a set of K prototypes $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ at random among

the training vectors $\boldsymbol{X}$.

Another approach to random initialization is *random labeling* using a randomized observation labeling $y_n \in \{1, \ldots, K\}, 1 \leq n \leq N$. The cluster labels $u_n(k)$ are randomly initialized so that, for all n, $u_n(k) = 1$ when $k = y_n$ and $u_n(k) = 0$ otherwise.

With large datasets (which may be encountered in audio pattern clustering), an initial point refinement solution has been proposed which consists of first clustering several smaller random sub-samples of the data and then clustering the sub-sample cluster mean vectors to obtain refined initial points for k-means [16].

The *binary split LBG* algorithm [73] is a popular method for generating VQ codebooks. It is based on k-means iteration, but instead of requiring initial values for the cluster mean vectors, it generates them by successive *splitting* of the cluster prototypes. The algorithm can be stated as follows:

1. Set $K = 1$ and compute the mean of all training vectors

$$\boldsymbol{\theta}_1 = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n$$

2. Split each of the existing K codewords by replacing the current codebook $[\boldsymbol{\theta}_1 \quad \ldots \quad \boldsymbol{\theta}_K]$ with a new codebook $[\boldsymbol{\theta}_1 + \boldsymbol{\epsilon} \quad \boldsymbol{\theta}_1 - \boldsymbol{\epsilon} \quad \ldots \quad \boldsymbol{\theta}_K + \boldsymbol{\epsilon} \quad \boldsymbol{\theta}_K - \boldsymbol{\epsilon}]$, where $\boldsymbol{\epsilon}$ is a small perturbation vector.

3. Replace K by $2K$.

4. Use the splitted codebook as an initialization to k-means iteration.

5. If K is less than the desired number of vectors J, go back to step 2. Otherwise, exit.

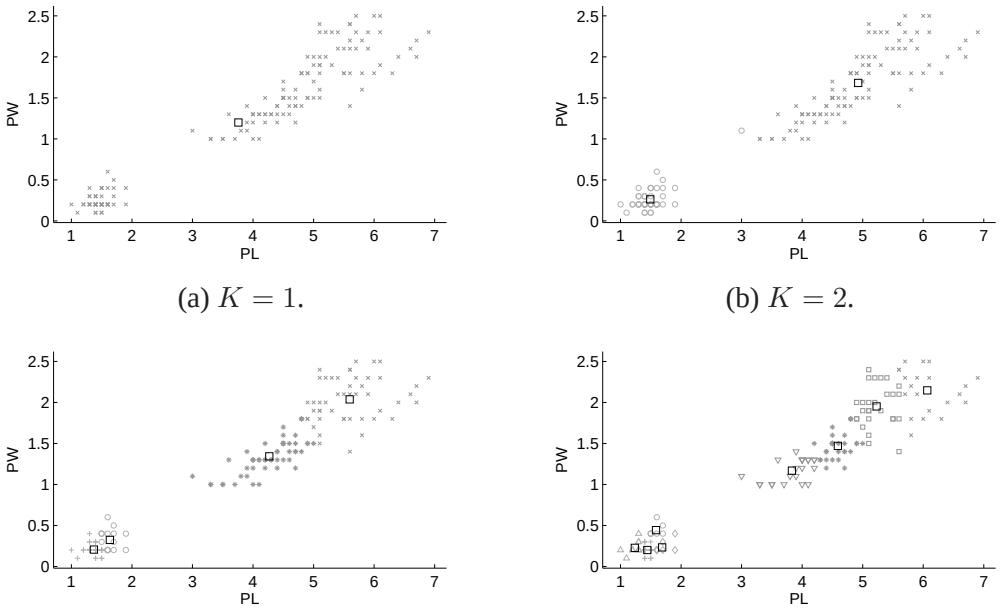
Figure 4.6 shows the application of LBG to the Iris data.

If randomized initialization or the LBG do not produce satisfactory results, there are alternative methods based on different heuristics. For example, Katsavounidis, Jay Kuo and Zhang [67] proposed the following algorithm for initializing k-means iteration in VQ codebook generation, when J codewords are required:

1. Choose the vector with the maximum Euclidean norm as the first codeword:

$$\boldsymbol{\theta}_1 = \underset{\boldsymbol{x}_n}{\operatorname{argmax}} \|\boldsymbol{x}_n\|$$

2. For $i = 2$ to J:



(a) $K = 1$. (b) $K = 2$.

(c) $K = 4$. (d) $K = 8$.

Figure 4.6: Application of the LBG clustering method for clustering two-dimensional Iris data.

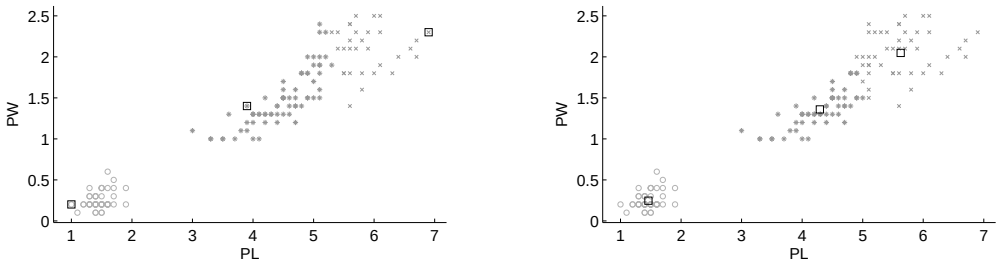
Choose the vector with the maximum distance from the current codebook as the new prototype:

$$\boldsymbol{\theta}_i = \underset{\boldsymbol{x}_n}{\operatorname{argmax}} \| \boldsymbol{x}_n - \underset{\boldsymbol{\theta}_j, 1 \leq j \leq i-1}{\operatorname{argmin}} \left(\| \boldsymbol{x}_n - \boldsymbol{\theta}_j \| \right) \|$$

This algorithm, which has been referred to as the KKZ algorithm in some previous work [3] [52], produces initial cluster centers near the boundaries of the sample data distribution. Figure 4.7 a) illustrates the initialization provided by this algorithm for the Iris data. Figure 4.7 b) shows the clustering result after k-means convergence.

Yet another alternative to initializing the mean vectors of mixture/cluster/VQ models is a method based on iterative elimination of the cluster prototypes, previously applied to clustering speech sounds [97]. The algorithm will be referred to as ICCR (iterative cluster centroid reduction) in the sequel. This algorithm uses a parameter $\alpha \in (0, 1)$ to control the amount of approximation permitted when discarding, at each iteration, the prototypes associated with the least populated clusters:

1. Choose K initial cluster prototypes $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ so that they approximately cover most of the feature space inhabited by the training data $\boldsymbol{X}$. Initialize cluster assign-



(a) Cluster initialization. (b) K-means result.

Figure 4.7: Application of the KKZ method to initialize k-means clustering of two-dimensional Iris data.

ment $u_n(k) = 0$ for $1 \leq n \leq N$, $1 \leq k \leq K$. Initialize cluster order $s(k) = k$, $1 \leq k \leq K$.

2. Set $K_p = K$.

3. For points that do not belong to any of the K clusters, i.e., $\sum_{k=1}^{K} u_n(k) = 0$, assign them to the cluster with the nearest prototype:

$$u_n(\operatorname*{argmin}_{k \in \{1,\ldots,K\}} d(\boldsymbol{x}_n, \boldsymbol{\theta}_{s(k)})) = 1$$

4. Update $s(i)$ to be the descending sort indexing of cluster point counts $S_k = \sum_{n=1}^{N} u_n(k)$, such that $\{S_{s(1)}, S_{s(2)}, \ldots, S_{s(K)}\}$ is a sequence sorted in descending order.

5. Replace K by finding lowest K such that

$$\frac{1}{N} \sum_{i=1}^{K} \sum_{n=1}^{N} u_n(s(i)) \geq \alpha$$

6. For all k such that $K + 1 \leq k \leq K_p$, set $u_n(s(k)) = 0$.

7. if $K = K_p$, exit. Otherwise, go to step 2.

The parameter α can be used to control the maximum number of clusters and their minimum size, because the iteration will not stop as long as the size of the smallest remaining cluster is less than $\lfloor (1 - \alpha)N \rfloor + 1$, where $\lfloor \ldots \rfloor$ denotes rounding towards negative infinity. This property follows directly from the elimination condition in step 5, as any smaller

cluster would be eliminated before convergence. With this lower bound for the size of the final clusters, it follows that an upper bound for the number of final clusters is

$$M_{\max} = \left\lfloor \frac{N}{\lfloor 1 - \alpha + (1/N) \rfloor} \right\rfloor . \tag{4.35}$$

To see this, consider the extreme case in which all the final clusters are of the same minimal size. The number of clusters in this case is given by dividing N by the cluster size, and non-integer values should be rounded downward if the cluster size is not a factor of N.

During the iteration, surviving prototypes can never lose points assigned to them; they can only gain points by inheriting them from eliminated prototypes. One consequence of this is that, using parameter α to reduce clusters, a prototype whose cluster size exceeds $(1 - \alpha)N$ can no longer be eliminated.

If it is necessary to be able to directly choose the number of clusters, it is possible to replace step 5 with simply $K = \max(K - 1, J)$, where J is the desired number of clusters given as a parameter to the method (in place of the approximation level α). However, a better solution may be to first iterate the algorithm until convergence with the α parameter having a high enough value so that after convergence, $K > J$. Then, step 5 of the algorithm can be modified as mentioned and the iteration can continue from step 2. This procedure produces the desired number of clusters.

Automatic selection of the number of clusters can be approached by using the above modification (reducing one cluster at a time) with parameter $J = 1$ (or the minimum allowed number of final clusters). If, at each iteration after step 5, the number of points belonging to the newly eliminated cluster, $\sum_{n=1}^{N} u_n(s(K_p))$, is plotted against the number of remaining clusters before elimination, K_p, a monotonically decreasing curve is obtained. A reasonable number of clusters can be automatically chosen by locating a turning point of this curve. Automatic selection of the number of clusters using similar turning-point criteria is a common approach in pattern recognition [120].

The initial prototypes in step 1 can be generated by simple random selection among the data vectors, or they can be the points defined by a conceptual *hypergrid* in d dimensions; the latter approach was adopted in [97] and will be used in the present work also. Denote the *resolution* of the grid, or number of point hyperplanes along each dimension, by R. As the number of the points in the grid is R^d, it is not wise to try to construct the grid explicitly when dealing with high-dimensional data or even large R. Instead, the following approach is suggested:

1. Compute grid node interval for each dimension ($x_{n,i}$ is the ith element of $\boldsymbol{x}_n$):

$$r_i = (\max_n(x_{n,i}) - \min_n(x_{n,i}))/(R - 1), \qquad 1 \leq i \leq d$$

2. Generate a set of nonnegative integer coordinate vectors:

$$\boldsymbol{z}_n = \left[\text{INT} \left(\frac{x_{n,1} - \min_k(x_{k,1})}{r_1} \right) \quad \ldots \quad \text{INT} \left(\frac{x_{n,d} - \min_k(x_{k,d})}{r_d} \right) \right]$$

where INT $(\ldots)$ denotes rounding to the nearest integer.

3. Generate a set $Z = \{\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_K\}$ in which the integer vectors $\boldsymbol{z}_n$ appear *uniquely*, without duplicates.

4. Form prototypes from the integer vectors $\boldsymbol{z}'_i$ by inverting the transformation done in step 2.

Let it be noted that by further counting the points associated with each prototype, the described grid approximation algorithm can be used to construct a *multidimensional histogram* of the feature data.

Figure 4.8 illustrates cluster approximation by the ICCR method. The initial prototypes are found using a grid with resolution $R = 10$; the result is shown in Figure 4.8 a). The parameter $\alpha = 0.90$ is used in the iteration. After the algorithm converges, i.e., no more prototypes can be eliminated using the parameter α, six prototypes and clusters remain as shown in Figure 4.8 b). The algorithm now switches to eliminating one prototype at a time until three prototypes remain. The end result is shown in Figure 4.8 c). By using these prototypes to initialize k-means, the clustering solution of Figure 4.8 d) is obtained.

In addition to the described cluster initialization methods, several others have been proposed [21] [3] [68] and evaluated [95] [52] in the literature.

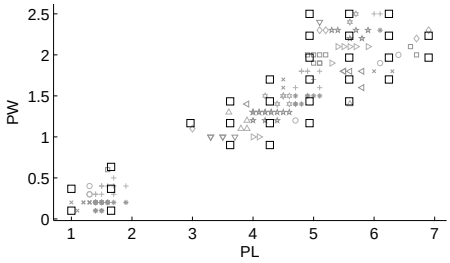
4.2.3 Parameter Learning for HMMs

This section reviews a form of the EM / Baum-Welch re-estimation for continuous observation densities. The re-estimation principle has been originally developed by Baum and his colleagues [11] and later extended (e.g., [74]). For the simpler case of discrete observation distributions, the details can be found in the tutorial [101].

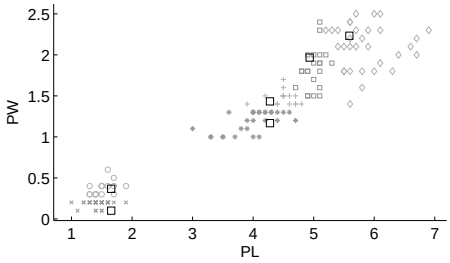
Gaussian Vector Autoregressive HMMs

In this section, the estimation formulas for a quite general class of continuous observation density HMMs are discussed. Here, each state-specific observation density is modeled as a Gaussian *vector autoregressive* (VAR) process [47] [77] such as

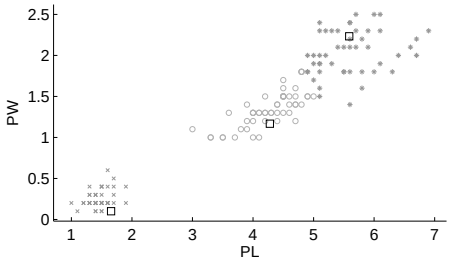
$$\boldsymbol{x}_n = \boldsymbol{\mu} + \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{x}_{n-i} + \boldsymbol{e}_n \tag{4.36}$$



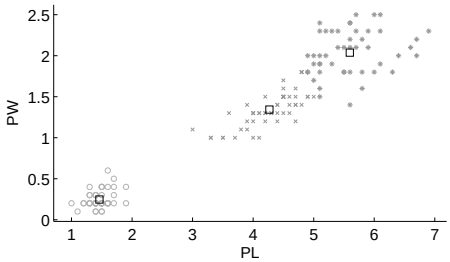
(a) Initial grid prototypes and clustering.



(b) Convergence with parameter $\alpha = 0.9$.



(c) Convergence with parameter $J = 3$.



(d) K-means result.

Figure 4.8: A sequence of clusterings in the application of the ICCR algorithm to approximate two-dimensional Iris data and the result of the subsequent k-means clustering initialized with the ICCR result.

where the $\boldsymbol{x}_n$ are $(d \times 1)$ observation vectors, $\boldsymbol{\mu}$ is the $(d \times 1)$ constant intercept vector, p is the order of the vector autoregression, the $\boldsymbol{\Phi}_i$ are $(d \times d)$ VAR coefficient matrices and $\boldsymbol{e}_n$ is multivariate Gaussian white noise with zero mean. Eq. 4.36 can be written in a more compact matrix notation

$$\boldsymbol{x}_n = \boldsymbol{B}\boldsymbol{z}_n + \boldsymbol{e}_n \tag{4.37}$$

where

$$\boldsymbol{B} = [\boldsymbol{\mu} \quad \boldsymbol{\Phi}_1 \quad \ldots \quad \boldsymbol{\Phi}_p], \tag{4.38}$$

and

$$\boldsymbol{z}_n = [1 \quad \boldsymbol{x}_{n-1}{}' \quad \ldots \quad \boldsymbol{x}_{n-p}{}']' \tag{4.39}$$

This representation allows several special types of observation densities in HMMs: a Gaussian density (with mean $\boldsymbol{\mu}$) when $p = 0$; a univariate AR(p) model when $d = 1$ and $p > 0$; and a true VAR(p) model when $d > 1$ and $p > 0$. However, an important type

of continuous observation density not yet covered by this form is the mixture density, for example, a Gaussian mixture model to parametrize the observation density associated with each state. As will be shown in the end of this section, the "M step" of the EM estimation can be easily modified to accommodate mixture densities.

The parameters of a vector autoregressive model in Eq. 4.37 can be estimated by ordinary least squares in the following way [77]. Let the data vectors be arranged as columns in a $(d \times N)$ matrix

$$\boldsymbol{X} = [\boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \dots \quad \boldsymbol{x}_N]\,. \tag{4.40}$$

Assuming that p pre-sample vectors $\{\boldsymbol{x}_{-p+1}, \dots, \boldsymbol{x}_0\}$ are available, form another $((dp+1) \times N)$ matrix

$$\boldsymbol{Z} = [\boldsymbol{z}_1 \quad \boldsymbol{z}_2 \quad \dots \quad \boldsymbol{z}_N] \tag{4.41}$$

The maximum likelihood estimator for the VAR parameter matrix, which is also the least squares estimator [47], is [77]

$$\hat{\boldsymbol{B}} = \boldsymbol{X}\boldsymbol{Z}'(\boldsymbol{Z}\boldsymbol{Z}')^{-1} \tag{4.42}$$

The maximum likelihood estimator for the covariance matrix of the white noise $\boldsymbol{e}_n$ is given by [47] [77]

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}(X - \hat{\boldsymbol{B}}\boldsymbol{Z})(X - \hat{\boldsymbol{B}}\boldsymbol{Z})' \tag{4.43}$$

A Gaussian VAR HMM is completely parametrized by $\lambda = \{\boldsymbol{\pi}, \boldsymbol{A}, B_1, \dots, B_K\}$, where each state distribution B_i is either $B_i = (\boldsymbol{B}_i, \boldsymbol{\Sigma}_i)$ (state-specific covariance matrices for the noise $\boldsymbol{e}_n$) or $B_i = (\boldsymbol{B}_i, \boldsymbol{\Sigma})$ (a shared covariance matrix for the noise $\boldsymbol{e}_n$). $\boldsymbol{\pi}$ and $\boldsymbol{A}$ denote the initial state probability vector and the state transition probability matrix, respectively. The EM estimation for such a model proceeds as follows:

1. Either: initialize λ and start from step 2 or initialize the $\gamma_n(i)$ as well as $\xi_n(i,j)$ and start from step 3.

2. E step:

   Re-estimate $\gamma_n(i)$ for $1 \leq n \leq N$ and $1 \leq i \leq K$ using either the forward-backward procedure or the successive application of Hamilton's and Kim's algorithms, as explained in Section 4.1.1. Estimate $\xi_n(i,j)$ for $1 \leq n \leq N$, $1 \leq i \leq K$, $1 \leq j \leq K$ using either Eq. 4.24 or Eq. 4.25.

3. M step:

   Re-estimate the initial state prior probabilities for $\boldsymbol{\pi} = [\pi_1 \quad \dots \quad \pi_K]$:

   $$\pi_i = \gamma_1(i), \qquad 1 \leq i \leq K \tag{4.44}$$

   Re-estimate the state transition probabilities for $\boldsymbol{A} = (a_{ij})$:

   $$a_{ij} = \frac{\sum_{n=1}^{N-1} \xi_n(i,j)}{\sum_{n=1}^{N-1} \gamma_n(i)}, \qquad 1 \leq i,j \leq K \tag{4.45}$$

   For each state i, form weighted observation matrices, in which the observation vectors are weighted by the square roots of the state probabilities $\gamma_n(i)$ [46]:

   $$\boldsymbol{X}_i = \left[\boldsymbol{x}_1\sqrt{\gamma_1(i)} \quad \boldsymbol{x}_2\sqrt{\gamma_2(i)} \quad \dots \quad \boldsymbol{x}_N\sqrt{\gamma_N(i)}\right] \tag{4.46}$$

   and

   $$\boldsymbol{Z}_i = \left[\boldsymbol{z}_1\sqrt{\gamma_1(i)} \quad \boldsymbol{z}_2\sqrt{\gamma_2(i)} \quad \dots \quad \boldsymbol{z}_N\sqrt{\gamma_N(i)}\right] \tag{4.47}$$

   Re-estimate the state specific VAR coefficient matrices:

   $$\boldsymbol{B}_i = \boldsymbol{X}_i\boldsymbol{Z}_i'(\boldsymbol{Z}_i\boldsymbol{Z}_i')^{-1}, \qquad 1 \leq i \leq K \tag{4.48}$$

   Re-estimate the state specific covariance matrices:

   $$\hat{\boldsymbol{\Sigma}}_i = \frac{(\boldsymbol{X}_i - \boldsymbol{B}_i\boldsymbol{Z}_i)(\boldsymbol{X}_i - \boldsymbol{B}_i\boldsymbol{Z}_i)'}{\sum_{n=1}^{N} \gamma_n(i)}, \qquad 1 \leq i \leq K, \tag{4.49}$$

   or a shared covariance matrix:

   $$\hat{\boldsymbol{\Sigma}} = \frac{\sum_{i=1}^{K}(\boldsymbol{X}_i - \boldsymbol{B}_i\boldsymbol{Z}_i)(\boldsymbol{X}_i - \boldsymbol{B}_i\boldsymbol{Z}_i)'}{N} \tag{4.50}$$

4. If the convergence criterion is met or a specified number of iterations has been run, exit. Otherwise, go to step 2.

An HMM with K states and J mixture components per state can be realized by a HMM with KJ states, in which each state corresponds to a mixture component of one of the K "virtual" states. For notational convenience, one could decide that the mixture components of the mth virtual state are presented by the true states $(m-1)J+1, (m-1)J+2, \dots, mJ$. The true HMM states can be made to act like mixture components with a simple modification of Eq. 4.45 in the M step of the EM estimation algorithm given above. What is

required is that within each set of J true states acting as mixture components of the same virtual state, the state transition probabilities should be made independent of the source state, i.e., $a_{ij} = a_{kj}$ if true states i and k are components of the same virtual state. This makes these transition probabilities behave like mixture weights in that the true state process is essentially i.i.d. (within the subset of true states that belongs to the same virtual state) and the ordering of the true state sequence is not meaningful. The described modification can be accomplished by replacing Eq. 4.45 by

$$a_{ij} = \frac{\sum_{k=(m-1)J+1}^{mJ} \sum_{n=1}^{N-1} \xi_n(k,j)}{\sum_{k=(m-1)J+1}^{mJ} \sum_{n=1}^{N-1} \gamma_n(k)}, \qquad (m-1)J+1 \leq i \leq mJ \tag{4.51}$$

When $J = 1$, Eq. 4.51 reduces to Eq. 4.45.

The arguably most popular form of continuous density HMMs, the Gaussian mixture HMM, is obtained by using the specifications given above with $p = 0$ and with the substitution of Eq. 4.51 in the re-estimation algorithm.

Initialization issues

HMMs are generally found to be not particularly sensitive to the initial values of the state transition probabilities $\boldsymbol{A}$ or the initial state probabilities $\boldsymbol{\pi}$, as long as they form reasonable probability distributions [101]. Care must be taken, however, not to initialize any state probability parameter to zero, because such parameter would stay zero through the course of the iteration. With discrete VQ observation distributions, the initialization of the discrete VQ symbol distributions is usually not a critical issue either [101]. With continuous observation distributions, however, the initialization is more important. The initialization of the distribution parameters in continuous density HMMs can be assisted by k-means or some other suitable clustering algorithm (which in turn may use some intelligent initialization procedure).

On a higher level, the simultaneous training of multiple HMMs can be incorporated in an iterative resegmentation scheme that uses a segmentation of the training data in terms of the different HMMs. Iterative resegmentation of HMMs, used for unsupervised classification in [71] [125], consists of the following steps:

1. Either use an initial segmentation and start from step 2 or use initial HMMs and start from step 3.

2. Train each HMM on the data associated with it in the current segmentation.

3. Segment the audio using Viterbi decoding with a composite HMM constructed from the current HMMs.

4. If convergence criterion is not satisfied, go back to step 2. Otherwise, exit.

Importantly, the iterative resegmentation training algorithm can start with an initial segmentation, without giving specific initial values for the HMM observation distribution parameters. The above algorithm also supports supervised embedded training of multiple HMMs, if the composite HMM network in step 3 is constructed according to a known training label sequence [128] [62].

4.2.4 Application to Speech/Music Discrimination

This section describes an experiment in which unsupervised learning was applied to audio feature data with the goal of separating speech and music from each other.

The audio material contained the following parts, sampled at 16 kHz:

1. S1: 160 utterances from the TIMIT database of American English [37], comprising ten utterances from one female and one male speaker from each of the eight dialect regions
2. M1: A heavy metal song performed live (Iron Maiden: Fear of the Dark)
3. M2: A slow rock ballad (Metallica: Nothing Else Matters)
4. M3: A pop song (Texas: Summer Son)
5. M4: Classical violin music (The Four Seasons by Vivaldi, Concerto 1 - Spring)
6. S2: A list of 25 Finnish words read aloud in turn by eight native Finnish speakers, four male and four female

The total length of the music material (M1-M4) was 21 minutes and 30 seconds. The length of the TIMIT speech material S1 was 7 minutes and 59 seconds and the length of the Finnish speech material S2 was 3 minutes and 51 seconds. Each of these three parts of the test material was separately normalized so as to have equivalent *average* (per sample) energy. This was done in order to eliminate the change of the recognition being assisted by signal energy, which is a rather arbitrary property of "found" audio signals but can still be, on the average, greater for music signals than for speech signals [4].

The feature extraction phase used a long-term feature frame of one second which was shifted with one second shift interval (zero overlap). Within each one-second segment, logarithmic short-time energy (LGSTE), loudness (LOUD) and gradient index (GIN) (see

[4]Nevertheless, measures of signal energy were not included in the feature set. Instead, dynamic long-time features of short-time measurements were used.

Section 3.5) were computed from short-time analysis frames of 20 milliseconds, shifted with a 10 millisecond shift interval at a time. The variance of the gradient index and the AR(1) prediction coefficients of LGSTE and LOUD were included in the feature vector representation. This procedure resulted in a data set with $N = 1999$ observations and $d = 3$ features. Figure 4.9 shows the two-dimensional scatter plots for each pair of features, plotted with different markers for speech and music.

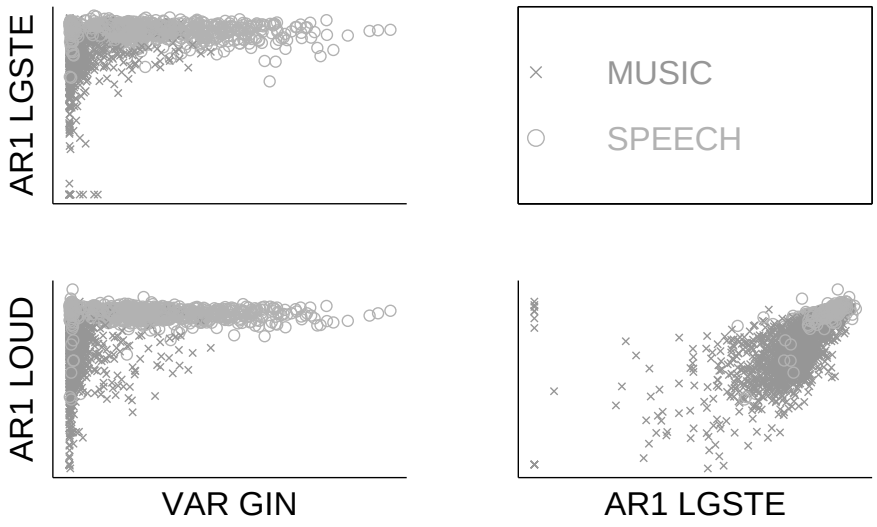


Figure 4.9: Scatter plots for three features, with speech and music categories denoted by different markers.

Mixture decomposition of a GMM with eight components and nodal, diagonal covariance was used to cluster the three-dimensional feature vectors. Before EM re-estimation, the mean vectors of the GMM components were initialized by three different methods: LBG ($J = 8$), KKZ ($J = 8$) and ICCR ($\alpha = 0.95$,$J = 8$). The same experiment was repeated by replacing the GMM with a HMM having eight states and the observation density of each state being a multivariate normal distribution with diagonal covariance. In HMM re-estimation, the sounds were presented in the given order.

The performance was measured by computing the minimum misclassification rate (MMR) as defined in Eq. 4.30. Table 4.1 shows the results. It can be observed that different initialization methods do have an effect on the final class separation. Moreover, the temporal context taken into account by HMM-based unsupervised classification improves the class separation. To apply these results in automatic supervised classification would require that the eight clusters could somehow be automatically mapped into the two classes in an optimal fashion (the MMR statistic always reports the "best case" misclassification rate). Figure

Table 4.1: Minimum misclassification rate (MMR), or the best-case misclassification percentage of one second frames, using GMM and HMM unsupervised classification with three different initialization methods for the mean vectors of the Gaussian component distributions.

Initialization method	GMM, $J = 8$ MMR %	HMM,$K = 8$ MMR %
LBG	7.9	4.8
KKZ	6.9	4.8
ICCR	7.5	4.6

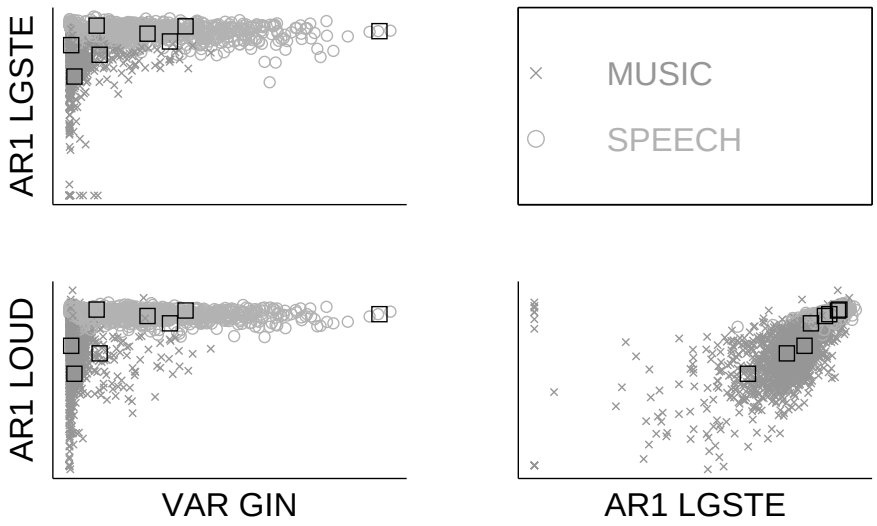


Figure 4.10: A semiautomatic speech/music classification produced by assigning eight clusters found by GMM unsupervised learning, initialized by the KKZ method, optimally to the speech and music classes in terms of the known true class labeling.

4.10 shows the class labeling corresponding to Table 4.1, with optimal assignment of the eight Gaussian components to the two classes, for a GMM initialized by the KKZ method. Figure 4.11 contains the same visualization for a HMM initialized by the ICCR method. It is evident that in both cases the two groups of Gaussian component means vectors - those associated with speech and those associated with music - are linearly separable, and perhaps better separable in the HMM/ICCR case. This could have positive implications for the goal of developing an automatic supervised classification method using these mostly unsupervised methods.

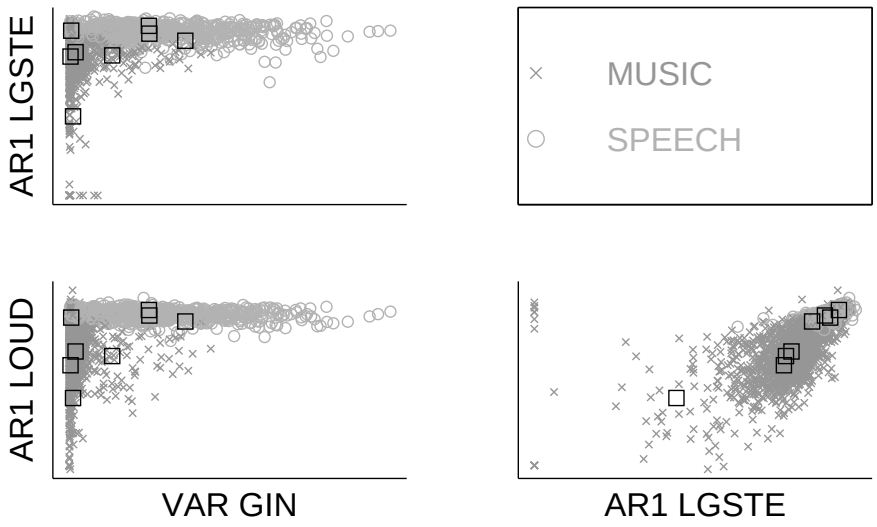


Figure 4.11: A semiautomatic speech/music classification produced by assigning eight clusters found by HMM unsupervised learning, initialized by the ICCR method, optimally to the speech and music classes in terms of the known true class labeling.

4.3 Time Series Segmentation

Pattern recognition is easiest when the class of each feature vector can be assumed independent of the classes of other feature vectors. Whenever the observations to be recognized have associated temporal or spatial information, pattern recognition can be viewed as consisting of two phases: segmentation and classification. This is the case in audio and image classification, if the data has not been previously segmented. Segmentation can be explicit and followed by classification, or the two phases can be performed simultaneously. Chapter 5 discusses different approaches to the combined task of segmentation and classification. Explicit segmentation can be performed by using change detection methods, which are usually either probabilistic or heuristic.

4.3.1 The Bayesian Information Criterion (BIC)

A successful example of probabilistic change detection methods is time series segmentation based on the *Bayesian information criterion (BIC)*. BIC itself is a general-purpose criterion for statistical model selection. Given a likelihood $L(X, M)$ for data set X based on model

M, the Bayesian information criterion is computed as [20]

$$BIC(M) = \log L(X, M) - \frac{1}{2} K_M \log(N) \tag{4.52}$$

where K_M is the number of parameters in the model M and N is the number of observations in X. Thus, BIC consists of the log likelihood from which a penalty term is subtracted. The more parameters there are in the model, the greater is the penalty.

In segmenting audio signals using BIC, the segmentation problem is often formulated as a hypothesis test involving multivariate Gaussian models. According to the null hypothesis H_0, the complete data has been generated by a single multivariate Gaussian distribution $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, whose parameters are estimated from the data by maximum likelihood. According to hypothesis H_1, there occurs a change at point n such that $X_1^n \sim N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $X_{n+1}^N \sim N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. The parameters of these distributions are similarly estimated from the respective windows. When d is the dimension of the feature space, the numbers of parameters in models corresponding to hypotheses H_0 and H_1 are given by $d + (1/2)d(d+1)$ and $2d + d(d+1)$, respectively. The difference of two BIC scores evaluated with change point candidate n is [2]

$$d_{\mathrm{BIC}}(n) = \log L(X, H_1) - \log L(X, H_0) - \frac{1}{2} \Delta K \log N \tag{4.53}$$

where the unpenalized, logarithmic likelihood ratio is [20]

$$\log L(X, H_1) - \log L(X, H_0) = N \log |\boldsymbol{\Sigma}| - N_1 \log |\Sigma_1| - N_2 \log |\Sigma_2| \tag{4.54}$$

and the difference in the number of parameters in the models corresponding to the two hypotheses is

$$\Delta K = d + \frac{1}{2} d(d+1). \tag{4.55}$$

A change is detected at time n when $d_{\mathrm{BIC}}(n) > 0$. An algorithm like the following has been used in employing BIC in time series segmentation [20] [2]:

1. Initialize $a = 1$, $b = 2$.
2. Detect if there is a change point in the interval $[a, b]$ using BIC.
3. If no change was detected, let $b = b + 1$. Otherwise, if a change at i was detected, set $a = i + 1$, $b = a + 1$.
4. If b has not reached the end of the time series, go back to step 2. Otherwise, exit.

In many applications of BIC to audio segmentation, an additional multiplier generally greater than 1 has been applied on the penalty term in order to improve the results. It has been pointed out that in such a case, the penalty multiplication factor acts as a built-in threshold that must be explicitly tuned [2]. A modified version of BIC segmentation [2] replaces the multivariate Gaussian model associated with H_0 by a GMM with two components. A log likelihood ratio test is subsequently used for segmentation.

4.3.2 Other Segmentation Methods

A wide variety of explicit segmentation methods can be conceived, all of which are based on certain assumptions on the behavior of the time series at class segment boundaries and during steady-state class segments. The BIC method discussed in the previous section is well justified by probabilistic theory and the assumptions made are quite explicit. This is not always the case in many "heuristic" segmentation methods. A common approach in such methods is to slide two adjacent analysis windows across the time series and at each location, compute a measure of change that somehow describes the difference of the time series behavior in the two windows. A segment boundary is placed at the boundary between the two windows if, for example, the computed change measure exceeds a certain threshold. Perhaps the simplest alternative for the measure of change is a distance between the window averages of the time series from the two windows (e.g., [130] [8]).

A different approach to segmentation is the convex hull algorithm proposed by Mermelstein for syllable-level segmentation of speech [82]. The segmentation is based on thresholding the difference between an instantaneous energy measure and the convex hull of the time contour of the energy.

Change detection can also be approached by employing the so called CUSUM statistical test from the field of control engineering, introduced by Page [91] as a sequential detection method. Consider a sequence z_n which is normally negative and for which a change is detected if its values suddenly tend to be too strongly positive. The cumulative sum of the first n observations is given by $S_n = \sum_{k=1}^{n} z_k$. A change in the process statistics is detected when

$$S_n - \min_{1 \leq i < n} S_i \geq h \tag{4.56}$$

or, equivalently, when

$$S'_n = \max(S'_{n-1} + z_n, 0) \geq h, \qquad n \geq 2 \tag{4.57}$$

where $S'_1 = z_1$ and in both formulas h is some detection threshold value. This principle has been applied - in an off-line, non-sequential fashion - to probabilistic audio segmen-

tation as an alternative to BIC segmentation [88]. The problem is formulated as that of determining whether a change in the time series of feature vectors occurs within a given window of N vector observations. A change is detected if

$$S'_N = \max_{r \in \{1,\ldots,N\}} \sum_{k=r}^{N} L_k \geq h \tag{4.58}$$

where L_k is the log likelihood ratio of two multivariate Gaussian models with respect to the kth observation. The two Gaussian models are obtained by re-estimating a GMM with two components from the vector sequence, initializing the Gaussian component mean vectors such that the other is a mean of a few observations in the beginning of the vector sequence and the other is a mean of a few observations in the end of the vector sequence. L_k is positive if the kth observation is better explained by the Gaussian component associated with the beginning of the sequence.

Chapter 5

Applications of Audio Recognition

This chapter discusses both the practical problems, for which automatic audio signal recognition is useful, and the ways in which the techniques introduced in the two previous chapters can be combined to achieve the goals. The practical applications can be roughly divided into three domains: speech, music and general audio, as listed in the left column in Table 5.1. Speech and music are the two most generally important types of utility signals, which has justified many applications targeting the extraction of information from these signals. General audio content analysis may be an outer layer that retrieves segments of speech and music for further speech- or music-specific analysis, or it can be of interest in itself. The middle column in Table 5.1 lists some important practical application areas from each of the three application domains, while the rightmost column lists examples of the more specific pattern recognition problems that may arise within each domain.

For each audio pattern recognition problem, such as those shown in the right column of Table 5.1, many alternative combinations of signal processing and pattern classification techniques are typically viable. The first section of this chapter discusses the general "patterns" that these time series pattern recognition solutions typically follow, while the second section offers a glimpse on the literature to show how such schemes are typically employed in the practical applications.

5.1 Basic Recognition Strategies

Despite the high variability in practical applications and in the employed techniques, it appears that a major part of the published speech or audio pattern recognition solutions can be at least roughly categorized into one of four categories based on the fundamental recognition control strategy adopted: *change detection*, *sliding window*, *template matching* and *unsupervised classification*. These will be loosely and informally defined below.

Table 5.1: Left column: the major application domains of audio pattern recognition. Middle column: some central practical application areas from each domain. Right column: specific audio pattern recognition problems arising in the different application domains.

Application domain	Practical application areas	Pattern recognition problems
Speech	transcription retrieval control	isolated word recognition connected speech recognition continuous speech recognition keyword spotting speaker recognition etc.
Music	transcription retrieval	chord recognition key identification genre identification artist identification structural analysis etc.
General audio	transcription retrieval surveillance	speech/music/background discrimination speaker recognition environmental sound recognition unsupervised audio classification etc.

One popular approach to audio segmentation and classification consists of the following steps: 1) locating segment boundaries explicitly (possibly with moderate oversegmentation); 2) preparing feature representations for each segment (either short-time feature vector sequences or one segmental feature vector per segment); 3) applying supervised or unsupervised classification methods on the found segments under the assumption that each segment belongs to a single class; 4) possible smoothing of the class decisions (e.g., by eliminating too short class segments). This will be called the *change detection* strategy in this study, as explicit change detection (segmentation) in the first stage is a fundamental operation. Change detection methods were discussed in Section 4.3. The subsequent feature extraction step is free to produce any type of features; alternatively, this step can be omitted and the classification step can use the same features that were used for the initial change detection. The classification step can use supervised or unsupervised pattern classification.

Because applications in the "general audio" domain have to deal with great acoustic variability and do not necessarily have access to a meaningful set of class definitions, they must often resort to unsupervised classification of the patterns at least in the early stages of system development. In this process the class definitions are created and refined. The change detection approach is conceptually simple and can accommodate both supervised and unsupervised pattern classification. The majority of recently published pioneering solutions to general audio content analysis use this recognition strategy, e.g.: [20] [129] [130] [59] [103] [123]. The approach has also been employed in blind phonetic segmentation of speech, e.g., in [8].

Another approach is to determine a classification window of suitable length and shift it across the signal representation with some shift interval, which does not exceed the length of the window. Each location of the window extracts a segment, which is classified in a supervised fashion. Subsequently, the class decisions must typically be smoothed by eliminating too short class segments and perhaps using additional information as well. This will be called the *sliding window* strategy. The sliding of the window is a trivial operation; the classification of the windowed segments can use any *supervised* pattern classification technique. The sliding window technique is straightforward and simple to implement. In the past, it has been adopted in the development of statistical pattern recognition solutions for audio signals, including voiced/unvoiced/silence classification of speech using a multivariate Gaussian classifier [6] and speaker identification using a GMM classifier [104]. It has also been used in general audio content classification with a rule-based classifier [76].

The third strategy, called *template matching*, is based on a *template* that describes the time evolution of the feature vectors. The template consists of different sub-models for the observed data (i.e., the feature vectors), each corresponding to an audio class, and some kind of rules for how the different classes can be sequenced in time. Segmentation and classification is formulated as an optimization problem of finding the optimal path through the template. A path through a template is a function of the observation time that gives, for each observation time instant, the identity of the sub-model that is thought to have produced the observation at that time. The optimal path is the path that best explains the observations using some criterion. The optimal path is found by search algorithms, e.g., implementations of dynamic programming. Dynamic programming is an optimal search technique. Also sub-optimal techniques such as A* search or possibly even greedy search can be used. From Chapter 4, it is readily seen that both dynamic time warping and Viterbi decoding of hidden Markov models can conform to the above definition *if* each component of the respective templates (an example feature vector in DTW, a state-specific observation probability distribution in a HMM) is taken as a model of an individual *class*. In the case of HMMs, it is easy to construct a composite HMM consisting of individual class HMMs,

then obtain the class labeling as the state path through the composite HMM trellis, and finally to express the path in terms of the class-specific HMM identities. DTW as such is less general than the HMM framework and is rarely used as the outmost layer in template matching. Reasons for this include the strict restrictions that the DTW local constraints impose on the possible sequence of states as well as the simplistic class model inherent in DTW. However, *level building dynamic programming* (LBDP) provides a somewhat more general dynamic programming framework [87]. The current mainstream ASR solutions to continuous speech recognition are fundamentally based on template matching with HMM templates [62] [57] [128].

In the fourth approach, an unsupervised classification is produced for the feature vectors in some manner. This will be called simply the *unsupervised classification* approach. In some simple cases, conventional clustering techniques such as k-means can lead to an acceptable segmentation/classification, although they do not take the temporal order of the observations into account. However, in many practical problems, it is necessary to use time context information in the classification. This means treating the feature vector sequence as a vector time series, not as a data set consisting of independent observations. HMMs are an attractive solution for this purpose due to being widely used and well understood. To perform temporal context-aware unsupervised classification with HMMs, a HMM is trained on the data being analyzed. The HMM in question may be a small model with a few states or a large composite HMM network consisting of sub-HMMs. When the goal is to simultaneously segment and cluster the data for exploratory purposes, the unsupervised classification approach is a natural choice. HMM training by the EM algorithm has been used for this purpose in, e.g., speaker segmentation [125], acoustic environment segmentation of personal recordings [22] and structural analysis of music [7]. An interesting question is also the following: can a suitably made "unsupervised" classification produce a meaningful *supervised* classification? The current author has experimented with this particular approach in the problem of voiced/unvoiced/silence segmentation and classification of speech [96]. Promising results were obtained by initializing HMM state densities to correspond to the three different classes in an exaggerated fashion, then running EM training and finally decoding the state sequence from the newly trained HMM.

The *bottom-up* and *top-down* strategies of information processing are the two basic choices in implementing audio recognition solutions. They have also been long employed in the analogous field of machine vision [115]. Interesting parallels can be drawn with these two general information processing strategies and the listed four audio recognition strategies. The *bottom-up* control strategy for image understanding [115] consists of first preprocessing the image, then performing segmentation, generating descriptions for the segmented regions (either separately or in conjunction with segmentation) and finally us-

ing pattern classification techniques to map the detected objects with real objects that are present in the solution domain. If the region descriptions are generated separately after segmentation, the above procedure is directly analogous to the change detection approach for time series data; if the region descriptions are formed in the course of segmentation (in an unsupervised fashion), the procedure is analogous to the unsupervised classification approach. The other main control strategy, the *top-down* strategy, is model-based and controlled by prior knowledge, so that the general mechanism of top-down control is hypothesis generation and its testing [115]. This is reminiscent of the sliding window and template matching approaches to audio recognition, which are fundamentally supervised classification strategies and utilize previously trained models.

5.2 Applications

In the following sections, audio recognition applications are categorized into three broad categories: speech, music and general audio. From each category, several applications and problems are highlighted. Examples from the literature are listed focusing on the signal processing and pattern recognition methods they use. This is by no means a comprehensive survey, but rather a representative sample of various practical problems and their solutions, each of which has as a core component an audio recognition module utilizing familiar signal processing and pattern recognition techniques.

5.2.1 Speech Applications

In comparison to music recognition applications and especially to general audio recognition applications, there is a vast amount of literature on automatic speech recognition (ASR). Even a roughly comprehensive sampling of published work would quickly be outside the scope of this thesis. Therefore, the number of references in the following discussion will be rather limited and the attention is focused to presenting a very concise, broad overview of the field.

The most central speech applications involve ASR. Although ASR is often viewed as simply the speech-to-text end application from the viewpoint of low-level speech signal processing, it is an enabling technology and not a full practical application as such. This distinction is important due to the extreme difficulty of the general unconstrained ASR problem and the amount by which a more constrained problem domain - with the constraints dependent on the end application - can alleviate this difficulty. Practical applications, in which ASR is central, include dictation [57], transcription of recorded speech [9], telephony applications [57], other control applications and speech user interfaces [57] and various speech-related search and retrieval applications, e.g., [48].

A common subdivision of ASR problems from the *technological* point of view is threefold: *isolated word recognition* (IWR), *connected word recognition* (CWR) and *continuous speech recognition* (CSR). IWR refers to automatic recognition of words spoken in isolation, with pauses in between. IWR as such may find use in simple and constrained control applications and telephone-based dialog systems. CWR assumes the speech to consist of continuously spoken strings of the vocabulary words, often with the assumption that each spoken utterance contains a known number of words. It is a rather specific problem but may be useful in, e.g., recognizing digit strings such as phone numbers and number codes in telephony applications. Continuous speech recognition typically has a large vocabulary and deals with the recognition of "normal" speech. In practice, the speech may still have to be constrained somehow; in order to achieve acceptable performance with limited training data, the current state-of-the-art continuous speech recognition systems still have to impose notable restrictions on the size of the speaker population, speaking style, vocabulary and/or grammar. Large vocabulary continuous speech recognition (LVCSR) has been applied in dictation, transcription and retrieval. A subproblem of continuous speech recognition is *keyword spotting*, which refers to detecting certain words from a speech signal. Keyword spotting can have applications in both control and retrieval.

The most common feature representation for any speech or speaker recognition application is the MFCC. A very common choice is to compute 13 MFCC coefficients for each frame, possibly replacing c_0 with logarithmic short-time energy containing roughly equivalent information, and to compute the Δ and $\Delta\Delta$ features for these coefficients. The 39-dimensional feature vector is then formed by concatenating the 13 MFCC coefficients (or log energy and 12 MFCC coefficients) with the Δ's and the $\Delta\Delta$'s.

Practical solutions for continuous and connected speech ASR are almost invariably based on the template matching approach such that the "template" is a large composite HMM consisting of sub-HMMs (e.g., word HMMs), which have been trained in the training phase. Details of the special considerations of HMM recognizer training in practical CSR applications can be found in [62] [57] [128]. The structure of the composite HMM and the transition probabilities between the sub-HMMs are determined by the language model, which is an important aspect in CSR [62] [57]. Recognition is a problem of searching for the most likely path through the composite network of trained sub-HMMs. The use of the optimal Viterbi search is computationally too heavy in general LVCSR problems; therefore, a heuristic suboptimal search strategy must be employed. The two main alternatives are the Viterbi beam search and A* stack decoding [62]. A number of more specialized techniques have been developed to make the CSR search more efficient [62] [57].

Simple IWR systems, which assume distinct pauses between words (or more generally, discrete utterances), are often based on a preliminary endpoint detection [57] after which

a time series pattern recognizer (typically based on DTW or HMMs) evaluates the test word segment against the reference models. If HMMs are used, each word class may be sufficiently represented by a single HMM. Once the word boundaries have been discovered by the endpoint detector, HMM-based Bayesian classification is straightforward. If DTW is used instead, several DTW templates are typically needed to represent each word class in kNN-style classification [100].

Some speech analysis methods define broader, less specific phonetic classes than the typically phoneme- or word-related classes of ASR. They are normally not useful in isolation, but as intermediate analyses in some end application. Such auxiliary applications include voiced/unvoiced/silence and related classification tasks, for which specialized features combined with a Gaussian classifier [6] or a rule-based classifier [112] have been used, and syllable segmentation, for which the change-detecting convex hull segmentation method has been proposed by Mermelstein [82]. Phonetic-level blind speech segmentation methods, using a change detection approach [8] or a pure unsupervised classification approach such as unsupervised level building dynamic programming [117] [109], generate segments and classes whose usefulness depends on the end application; acoustic-phonetic ASR [100] could be such an application.

Besides applications involving ASR or phonetically oriented speech analysis, automatic recognition of speech signals is involved in *speaker recognition* applications. Speaker recognition can be subdivided into *speaker identification*, in which the goal is to determine the identity of the speaker among a known set of speakers, and *speaker verification*, in which the goal is to detect a known speaker apart from any impostors. In either task, the speech can be constrained to be a known phrase (text-dependent) or anything (text-independent) [104]. Speaker identification can be further subdivided into supervised speaker identification and unsupervised *speaker segmentation*. In supervised speaker identification, Reynolds and Rose [104] achieved good results using sliding window GMM classification with MFCC features. Kimber and Wilcox [71] used unsupervised HMM training and resegmentation, with cepstral feature vectors, for the same purpose. For supervised speaker identification, their initial models were obtained by training an HMM for each speaker in the population. Unsupervised speaker segmentation, in which no prior training is available for the speakers, can be used to find single-speaker segments and speaker changes, for example as an aid to ASR. This task is often handled by a change detection approach [20] [131] [2] [48], typically employing BIC segmentation with MFCC features, followed by hierarchical segment clustering which also uses BIC as the segment distance measure [20] [131]. Also unsupervised classification using HMM learning has been applied for the purpose of speaker segmentation [125] [71].

In speech coding applications, the problem of *voice activity detection* (VAD), i.e., speech

detection, is also handled using pattern recognition methods [13] [114]. In coding applications, VAD typically has special requirements. First, it must work in real time. Second, because the goal is to suppress transmission of silent portions in order to conserve bandwidth, it is generally a larger error for a VAD algorithm not to detect speech when it is present than to erroneously detect speech when no speech is present.

5.2.2 Music Applications

In the context of music signals, various different recognition problems have been investigated. These include different subtasks of music transcription [72], query by humming [39], identification of the musical genre [122] and audio thumbnailing by identification of recurring segments [10].

Music transcription refers to the complete process of writing down musical notation based on an acoustic musical signal [72]. Automatic systems have been developed for accomplishing different subtasks of music transcription. Chord transcription refers to automatic segmentation and classification with respect to chords. Sheh and Ellis [110] used a HMM-based recognizer with chroma-based features for both segmentation (by forced Viterbi alignment) and transcription. Bello and Pickens [12] employed unsupervised learning of HMMs on chroma feature vectors of the test data, with HMM parameters initialized according to musical theoretic knowledge; e.g., the state transition probabilities were initially made to reflect some very general patterns of chord progression. *Key extraction*, or automatic identification of the musical key, has been handled using averaged chroma vectors [94]. Automatic *melody transcription* is strongly connected with the problem of pitch tracking. An overview of melody transcription methods can be found in [98].

Automatic identification of musical genres is a relatively novel research problem. With the number of genre classes ranging from six to twenty, automatic genre classification methods have achieved correct identification rates of over 60 % [122] [80] [75], while human performance was between 76 % and 90 % in [75]. Tzanetakis and Cook [122] used Gaussian, GMM and KNN classifiers with a feature vector comprising various long-term features (depicting timbre, pitch and rhythm content) to classify complete audio files into categories corresponding to different musical genres. Using a Gaussian classifier structure, McKinney and Breebaart [80] evaluated four different modulation-based long-term feature vectors in musical genre recognition and general audio classification.

Retrieval of a specific song based on its melody, as represented by the user, has been researched by several groups. A popular solution for query presentation is *query by humming*. It refers to example-based detection in which the goal is to retrieve from a database the pieces of music corresponding to the recorded tune hummed by the user. The unconstrained form of the sound examples increases the difficulty of the task and these systems

need to be robust to both poor humming and environmental noise. Because the melody must be recognized, pitch analysis is central in feature extraction. Many published solutions first use pitch analysis and note segmentation in extracting a string representation that depicts the relative changes in pitch between successive notes [39] [81] [111]. This category-valued feature extraction is performed for both the query and the pieces of music in the database. For subsequent detection of melodic similarity, approximate string matching techniques [39] [81] or HMM-based methods [111] can be utilized.

Structural analysis of music may be useful in search and retrieval applications. Automatic identification of certain broad regions of musical pieces, such as the chorus, verse and different instrumental sections [7] can be helpful in browsing musical audio files and in *audio thumbnailing*. Thumbnailing refers to the representation of an audio file with a representative segment; for example, a song could be represented by its chorus [10]. Typically, structural analysis is handled using unsupervised learning. Aucouturier and Sandler [7] used EM training of mixture HMMs to perform unsupervised segmentation and classification. Bartsch and Wakefield [10] computed chroma vectors from short frames and located recurring segments from a similarity matrix created by correlating the chroma vectors occurring at different times in the audio signal.

5.2.3 General Audio Applications

In large-scale audio content analysis, ordinary speech and music may be regarded as just individual sound classes, like any other type of sound. The accumulation of large bodies of broadcast and digitally stored audio material, containing various types of sound, calls for automatic methods to accomplish efficient search and retrieval of desired speech and music content (e.g., [106] [107]). Automatic sound recognition is also often used as an aid in the analysis of *video* content, i.e., using the soundtrack to locate certain video segments (e.g., [83] [130]). Another field of application deals with *personal archiving* (transcription and retrieval) of recordings performed on mobile or wearable equipment (e.g., [22] [32]). Finally, audio-based *surveillance* solutions require automatic identification of interesting or suspicious sound events (e.g., [123] [103]).

Broadcast audio is usually a relatively constrained domain, typically consisting mostly of speech and music, the two of which are also often the subjects of primary interest in such content. Thus, speech detection, music detection and speech/music discrimination are central research problems in the first stage of analyzing audio broadcasts; once speech and music segments have been identified, the analysis can be continued and refined by employing speech- or music-specific methods (Sections 5.2.1 and 5.2.2). An early study on speech/music discrimination was published by Saunders [106]. His approach is based on using the bimodality of the short-time (16 ms frame and frame shift) zero crossing rate,

as seen in a larger window of 2.4 seconds, to detect speech segments apart from music. Another well-known study on speech/music discrimination is that of Scheirer and Slaney [107], who evaluated various combinations of features, computed from a window of one second, using different statistical classifiers (minimum Mahalanobis distance, GMM and kNN classifiers). The choice of the feature set was found to be more important than the choice of the classifier structure, although the latter did affect the balance between speech and music detection performance. In general, music was somewhat more difficult to detect correctly than speech. A 4 Hz modulation energy (computed by separately filtering 40 mel frequency channels with a 4 Hz bandpass filter and integrating the result) and the variance of the spectral flux (variance of Eq. 3.53) were found to be among the best features. El-Maleh *et al* [30] approached speech/music discrimination using Gaussian classification of feature vectors based on the LSF representation of LP models.

Content analysis of video material, such as movies and television broadcasts, can also be helped by analysis of the soundtrack. It has been speculated that, in content parsing of audiovisual data, the audio content could even play a primary role in comparison to the image content [130]. Different classification schemes for detecting speech, music and environmental sounds have been used for random access of video material in e.g. [83] [130] [76].

Example-based retrieval of general audio has been discussed by Wold *et al* [126], who described their system for the retrieval of short sound segments. They used minimum weighted Euclidean distance classification (minimum Mahalanobis distance with diagonal covariance), on a feature vector consisting of four perceptually motivated features: logarithmic energy (representing loudness), logarithm of a fundamental frequency estimate (representing pitch), logarithm of the spectral centroid and a measure of bandwidth (the latter two representing timbre). Guo and Li [43] applied SVM classification to example-based retrieval of the same segments as in [126]. Their feature representations consists of the means and standard deviations of different short-time features, computed over the complete sound files.

In the general case of analyzing unconstrained audio, meaningful or reliable class specifications may not be available beforehand. Analysis of such recordings can be based on unsupervised classification whose results are either used as such or corrected by the user. As one example of such an approach, Misra *et al* [84] proposed a system that performs unsupervised detection of acoustic events in a sound scene using a rule-based approach that emulates the psychoacoustic rules of auditory stream segregation. The final goal is the resynthesis of a similar sound scene using the detected events. As another example, retrieval and transcription of personal recordings using *wearable* equipment is a new and emerging field of study [22] [31] [32]. A wearable system could record a large portion of the acoustic

environments which the user visits during the day. Some pioneering work on the subject has been done. Clarkson and Pentland [22] analyzed audio and video recorded using wearable equipment in everyday life. Their feature vector consists of both video features and auditory filterbank acoustic features. Using iterative resegmentation for unsupervised learning of left-to-right HMMs, they arrive at a clustering of acoustic events, each being represented by a HMM. A further layer of analysis constructs scene HMMs from the likelihoods of the event HMMs in order to separate acoustic scenes. For a similar problem, but analyzing only audio signals, Ellis and Lee [31] [32] used the change detection strategy. They used the means and standard deviations of different short-time features as the feature representation and applied BIC segmentation, followed by segment clustering using the spectral clustering method [32].

Audio surveillance is a relatively new field. In [29], a surveillance-related system was proposed for detecting door slams, glass breaks, human screams, explosions, gun shots and stationary noises from background noise. Energy-based change detection is followed by GMM or HMM classification of filterbank energy features. In [103], the application is surveillance in elevators by detecting certain types of suspicious audio events, such as banging and non-neutral speech. The method is based on change detection followed by GMM-based supervised classification using MFCC feature vectors. In [123], audio surveillance is used as part of an audiovisual surveillance system for public transport vehicles. The audio event detection module focuses on detecting shouting segments so that these detections can be combined with data from video-based surveillance. The method involves change detection (using a heuristic approach) followed by GMM-based classification of the detected stationary segments using MFCC/delta-MFCC feature vectors. The use of audio event detection to supplement video surveillance is also proposed in [113], where very simple detectors (due to computational requirements of audio sensor networks) based on thresholding five-second energy and zero crossing rate are used. In [59], experiments on environmental sound detection and classification in an office environment are reported. The method is based on change detection by thresholding spectrum-based distances to an adaptive noise floor. Both unsupervised and supervised classification is applied to segmental feature vectors consisting of ten selected features. The unsupervised approach uses k-means while the supervised approach uses minimum Euclidean distance classification with manually selected class centers.

Chapter 6

Conclusions

An audio pattern is a feature vector template associated with a certain "hidden" state of nature. The state of nature consists of category variables that may, depending on the application, define the broad sound type (e.g., speech, music), the identity of the sound source as well as semantic information associated with the sound content. Audio pattern recognition refers to the combined task of identification of the temporal locations of audio patterns (segmentation) and the association of the audio pattern segments with the hidden states of nature (classification). Below, the main conclusions and observations of this study are highlighted.

Feature extraction is important. To reliably detect and identify audio patterns, it is important to limit the size of the acoustic data set via feature extraction. As in all pattern recognition, the curse of dimensionality causes the required training data set size to grow exponentially as a function of the number of features in the feature vector. In the limits imposed by the curse of dimensionality and the available amount of training data, however, no important class-discriminating features should be excluded from the feature set. Often, good separation between some given classes can not be achieved by a single feature but requires a combination of features. The features that discriminate between the same classes may even be highly correlated and yet greatly supplement each other in terms of class discrimination [44] - thus, feature orthogonality alone is not a good goal in feature selection.

Because class separability in the feature space is generally desirable in pattern classification, and because clustering is about identifying separable classes in an unsupervised fashion, it is desirable to have some kind of a clustering structure in the feature space so that a natural clustering of the feature vectors corresponds highly to the ground truth pattern class labeling. This structure can consist of arbitrarily shaped clusters and does not have to be obvious to any given clustering algorithm. Intuitively, the better the patterns belonging

to different classes are separable by unsupervised classification, the less the system must rely on learning complex class boundaries in the training phase, and the easier the classification problem can be expected to become. The desirability of natural clusterings provides another motivation for careful feature selection, because a clustering structure in some selected feature subspace may not be evident in a higher-dimensional feature space in which more features are included [68].

Striving to obtain a class-clustering structure in feature extraction leads to another viewpoint: almost everything in pattern recognition is actually feature extraction. A complex feature extractor might indeed produce feature vectors that form a very distinct class-clustering structure, so that the actual pattern classification of these feature vectors is a trivial task using a simple linear classifier (such as a minimum Euclidean distance classifier). On the other hand, as discussed in Section 1.1, practically any intermediate representation in a pattern classifier can be interpreted as the output of a feature extraction module, followed by a simpler pattern classifier. From yet another viewpoint, pattern recognition systems could be arranged in cascade such that the class probability outputs from a Bayesian classifier stage are used as features for some classifier in the second stage (which may deal with different classes). In this case, where does feature extraction end and pattern classification begin? In some sense, designing audio pattern recognition systems is all about designing audio feature extractors. In an audio pattern recognition system, if the audio classes are to be separable by a general-purpose nonlinear pattern classification method (Chapter 4), the audio features produced by the feature extraction module must be of sufficiently high level.

Attention should be paid to auditory knowledge. A good starting point for reasonable feature exctraction for audio is knowledge of the human/mammalian auditory system (Chapter 2). The auditory perception capabilities in animals have evolved over millions of years and can reasonably be assumed to be computationally efficient so as to preserve maximal brain capacity for other functions as well. In terms of auditory perception, most computable acoustic features can be classified as either loudness, pitch, timbral or rhythmic features (Sections 3.5-3.6). Loudness features have a strong connection to the short-time acoustic energy, pitch features are obtained by pitch estimation and tracking methods, timbral features characterize the shape of the short-time (auditory) spectrum and rhythm features use long-term processing to capture modulation properties. In selecting features for a specific recognition application, it may be useful to consider what perceptual aspects best differentiate between the target classes.

Discrimination of broad sound classes, such as speech, music and environmental sounds, often requires the use of long-term rhythmic/modulation information, because these classes can not be reliably discriminated based on the instantaneous short-time auditory spectra. Carefully observing the behavior of short-time features during different sound classes (e.g.,

Figure 3.4 in Section 3.5) may lead to discoveries of useful long-term features that capture just the correct class-discriminating rhythmic/modulation information. For example, AR(1) prediction coefficients estimated from short-time loudness features over a one second window were found in Section 4.2.4 to be promising candidates for detecting speech apart from music. This is because the modulation spectrum of short-time loudness features is generally dominated by lower frequencies for speech than for music (see, for example, the features LGSTE and LOUD in Figure 3.4). As another related example, the 4 Hz modulation energy has been found to be an effective feature for speech detection [107].

Unsupervised classification methods are promising. The psychoacoustical phenomenon of auditory stream segregation, which is fundamental to all hearing and which has its origins in the higher neural stages of the auditory system, has been shown by Bregman [17] to be well explained by the Gestalt grouping rules (Section 2.2). Because the Gestalt grouping rules describe the formation of perceived patterns from sensory elements by forces of mutual attraction, they seem to require the presence of an unsupervised, bottom-up control mechanism in pattern recognition. This makes unsupervised pattern recognition methods attractive. A completely unsupervised approach is a natural choice in problems with unconstrained general audio, in which the classes may be very hard to define beforehand. However, in a supervised classification setting, in which the classes are known in advance, it is often not clear how the found unsupervised pattern classes should be associated with the supervised, predefined classes. Nevertheless, the author holds the view that unsupervised classification methods can be useful in a central role even in a supervised classification setting (as demonstrated in [96], where the supervised class association was handled by means of parameter initialization for EM re-estimation of a HMM). The author sees this kind of approach as an interesting direction of research and has attempted to highlight this possibility also in the unsupervised speech/music discrimination example (Section 4.2.4).

As further justification for unsupervised classification methods it can be noted that, rather surprisingly, *the Bayesian classification principle, Eq. 4.4 in Section 4.1.1, can also accommodate unsupervised learning*. When each state of nature ω is modeled by a corresponding PDF model λ_ω, the distinction between supervised Bayesian classification and unsupervised probabilistic classification is determined by whether λ_ω is previously trained on labeled training data or whether it is obtained from the testing data in an unsupervised manner (e.g., as a set of decomposed Gaussian mixture components after EM re-estimation of a GMM). In the latter case, the discussion of Section 4.1.1 leading to Eq. 4.4 still applies as long as the correspondence between ω and λ_ω (the *supervision*) can somehow be established.

The author also presents a meaningful performance measure for evaluating unsupervised classification performance in a supervised classification setting, the minimum misclassifi-

cation rate MMR (Eq. 4.30 in Section 4.2).

Because the author considers unsupervised pattern classification and clustering methods to be important in audio recognition, several *cluster initialization* methods for the iterative optimization or EM-style unsupervised classification algorithms are presented in Section 4.2.2 (initialization is central in [96], where the unsupervised EM algorithm is used in a supervised classification task). Among others, the iterative cluster centroid reduction (ICCR) method developed by the author is presented for clustering/cluster initialization.

For future research, it would also be very interesting to look for general pattern recognition methods that can combine *automatic on-line feature selection* with unsupervised classification. In comparison to unsupervised classification using a fixed set of features, the on-line feature selection approach would be even better justified by the psychoacoustical mechanisms of auditory stream segregation, because stream segregation is considered to gather competing evidence and combine it in a mechanism that is like voting [17] [4] (Section 2.2); i.e., human auditory scene analysis does not use a fixed feature set.

While the building blocks for low-level feature extraction and subsequent pattern recognition are presented in Chapters 3 and 4, respectively, the author also presents a categorization of recognition strategies that combine these building blocks in different ways (Section 5.1). The four basic strategies have been termed 1) change detection, 2) sliding window, 3) template matching and 4) unsupervised classification. Most of today's automatic speech recognition is based on strategy 3), template matching (Viterbi decoding for a composite HMM template) with a great deal of domain knowledge included in the classifier (most notably in the form of sophisticated language models). In contrast, strategies 1) and 4) lend themselves to the largely unsupervised, largely bottom-up approach which has justifications in human auditory scene analysis. Recently, both strategies have become popular in applications involving recognition of general, unconstrained audio material (Chapter 5).

In terms of the end application domain, the end applications can be conveniently categorized into three categories: speech applications, music applications and general audio applications. This categorization highlights the importance of speech and music as utility signals. Of these three, recognition of general audio typically has to deal with the broadest class specifications and often includes speech and music as separate categories. Applications of automatic speech recognition are central in the speech application category. Both automatic speech recognition and phonetic analysis of speech have been studied for decades. The categories of music and general audio include newer recognition problems that have only recently attracted attention. For many of these recognition problems, widely accepted state of the art solutions have not yet emerged. The research in all audio information extraction is further driven by the ever increasing storage and transmission capacity for digital multimedia material (the need for automatic organization) as well as by the in-

creasing computational power (the means for automatic organization). Thus, the field of automatic audio recognition can be expected to contain many important research topics in the future.

Bibliography

[1] *ISO 226:2003 Acoustics - Normal Equal-Loudness-Level Contours*, second edition.

[2] J. Ajmera, I. McCowan, and H. Bourlard. Robust speaker change detection. *IEEE Signal Processing Lett.*, 11(8):649–651, August 2004.

[3] M. B. Al-Daoud and S. A. Roberts. New methods for the initialisation of clusters. *Pattern Recognition Letters*, 17(5):451–455, May 1996.

[4] J. Allen. How do humans process and recognize speech? *IEEE Trans. Speech Audio Processing*, 2(4):567–577, 1994.

[5] B. S. Atal. Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *Journal of the Acoustical Society of America*, 55(6):1304–1312, 1974.

[6] B. S. Atal and L. R. Rabiner. A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 24(3):201–212, June 1976.

[7] J.-J. Aucouturier and M. Sandler. Segmentation of musical signals using hidden Markov models. In *Proc. Audio Engineering Society 110th Convention*, Amsterdam, The Netherlands, May 2001.

[8] G. Aversano, A. Esposito, A. Esposito, and M. Marinaro. A new text-independent method for phoneme segmentation. In *Proc. 44th IEEE 2001 Midwest Symposium on Circuits and Systems (MWSCAS 2001)*, pages 516–519, Dayton, USA, 2001.

[9] R. Bakis, S. Chen, P. Gopalakrishnan, R. Gopinath, S. Maes, L. Polymenakos, and M. Franz. Transcription of broadcast news shows with the IBM large vocabulary speech recognition system. In *Proc. 1997 DARPA Speech Recognition Workshop*, Chantilly, USA, February 1997.

[10] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proc. Int. Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY, October 2001.

[11] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[12] J. P. Bello and J. Pickens. A robust mid-level representation for harmonic content in music signals. In *Proc. 6th Int. Conf. Music Information Retrieval (ISMIR 2005)*, London, UK, September 2005.

[13] A. Benyassine, E. Shlomot, H.-Y. Su, D. Massaloux, C. Lamblin, and J.-P. Petit. ITU-T recommendation G.729 annex B: A silence compression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data applications. *IEEE Commun. Mag.*, pages 64–73, September 1997.

[14] S. F. Boll. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Trans. Acoust., Speech, Signal Processing*, 27(2):113–120, April 1979.

[15] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems 7*. MIT Press, 1995.

[16] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *Proc. 15th International Conference on Machine Learning (ICML '98)*, pages 91–99, Madison, Wisconsin, USA, July 1998.

[17] A. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International, 1984.

[19] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[20] S. S. Chen and P. S. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the Bayesian information criterion. In *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, USA, February 1998.

[21] S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of Intelligent and Fuzzy Systems*, 2(3):267–278, September 1994.

[22] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. In *Proc. 1999 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'99)*, pages 3037–3040, Phoenix, USA, March 1999.

[23] S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. Acoust., Speech, Signal Processing*, 28(4):357–366, August 1980.

[24] F. de Wet, B. Cranen, J. de Veth, and L. Boves. A comparison of LPC and FFT-based acoustic features for noise robust ASR. In *Proceedings of Eurospeech 2001*, Aalborg, Denmark, 2001.

[25] A. P. Dempster, N. M. Laird, and D. B.Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B*, 39:1–38, 1977.

[26] S. Dharanipragada, U. H. Yapanel, and B. D. Rao. Robust feature extraction for continuous speech recognition using the MVDR spectrum estimation method. *IEEE Trans. Audio, Speech, and Language Proc.*, 15(1):224–234, January 2007.

[27] P. Divenyi, editor. *Speech Separation by Humans and Machines*. Kluwer Academic Publishers, 2005.

[28] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons Inc., 2001.

[29] A. Dufaux, L. Besacier, M. Ansorge, and F. Pellandini. Automatic sound detection and recognition for noisy environment. In *Proc. EUSIPCO 2000, European Signal Processing Conference*, Tampere, Finland, September 2000.

[30] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal. Speech/music discrimination for multimedia applications. In *Proc. 2000 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'00)*, pages 2445–2448, Istanbul, Turkey, June 2000.

[31] D. Ellis and K. S. Lee. Features for segmenting and classifying long-duration recordings of personal audio. In *Proc. ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA-04)*, Jeju Island, Korea, October 2004.

[32] D. Ellis and K. S. Lee. Minimal-impact audio-based personal archives. In *1st ACM workshop on Continuous archival and retrieval of personal experiences (CARPE'04)*, pages 39–47, New York, USA, October 2004.

[33] G. Fant. *Acoustic Theory of Speech Production*. Mouton, The Hague, 1960.

[34] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[35] T. Fujishima. Real-time chord recognition of musical sound: a system using common lisp music. In *Proc. International Computer Music Conference (ICMC'99)*, Beijing, China, 1999.

[36] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Trans. Acoust., Speech, Signal Processing*, 34(1):52–59, February 1986.

[37] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fisc us, D. S. Pallett, and N. L. Dahlgren. DARPA TIMIT acoustic-phonetic continuous speech corpus. CD-ROM, 1993. NIST.

[38] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[39] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming - musical information retrieval in an audio database. In *Proc. ACM Multimedia 95*, San Francisco, USA, November 1995.

[40] A. H. Gray and J. D. Markel. Distance measures for speech processing. *IEEE Trans. Acoust., Speech, Signal Processing*, 24(5):380–391, October 1976.

[41] S. Greenberg. On the origins of speech intelligibility in the real world. In *Proc. ESCA-NATO Tutorial and Research Workshop on Robust Speech Recognition for Unknown Communication Channels*, pages 23–32, Pont-a-Mousson, France, 1997.

[42] L. J. Griffiths. A continuously-adaptive filter implemented as a lattice structure. In *Proc. Int. Conf. Acoust. Speech, and Signal Proc.*, pages 683–686, Hartford, USA, May 1977. IEEE.

[43] G. Guo and S. Z. Li. Content-based audio classification and retrieval by support vector machines. *IEEE Trans. Neural Networks*, 14(1):209–215, January 2003.

[44] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, (3):1157–1182, 2003.

[45] J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57:357–384, 1989.

[46] J. D. Hamilton. Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45:39–70, 1990.

[47] J. D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[48] J. H. L. Hansen, R. Huang, B. Zhou, M. Seadle, J. R. Deller, A. R. Gurijala, M. Kurimo, and P. Angkititrakul. SpeechFind: Advances in spoken document retrieval for a national gallery of the spoken word. *IEEE Trans. Speech Audio Processing*, 13(5):712–730, 2005.

[49] M. O. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley and Sons Inc., 1996.

[50] S. Haykin. *Neural Networks: A Comprehensive Foundation*. IEEE Press, 1994.

[51] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, third edition, 1996.

[52] J. He, M. Lan, C.-L. Tan, S.-Y. Sung, and H.-B. Low. Initialization of cluster refinement algorithms: A review and comparative study. In *Proc. International Joint Conference on Neural Networks (IJCNN04)*, Budapest, Hungary, July 2004.

[53] H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738–1752, 1990.

[54] H. Hermansky. Should recognizers have ears? *Speech Communication*, 25(1-3):3–27, August 1998.

[55] H. Hermansky and N. Morgan. RASTA processing of speech. *IEEE Trans. Speech Audio Processing*, 2(4):578–589, 1994.

[56] W. Hess. *Pitch Determination of Speech Signals*. Springer-Verlag, 1983.

[57] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing*. Prentice Hall PTR, 2001.

[58] A. Härmä, M. Karjalainen, L. Savioja, V. Välimäki, U. K. Laine, and J. Huopaniemi. Frequency-warped signal processing for audio applications. *J. Audio Eng. Soc.*, 48(11):1011–1029, November 2000.

[59] A. Härmä, M. F. McKinney, and J. Skowronek. Automatic surveillance of the acoustic activity in our living environment. In *Proc. IEEE Int. Conf. Multimedia and Expo (ICME 2005)*, Amsterdam, The Netherlands, July 2005.

[60] F. Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 23(1):67–72, February 1975.

[61] P. Jax and P. Vary. On artificial bandwidth extension of telephone speech. *Signal Processing*, 83:1707–1719, 2003.

[62] F. Jelinek. *Statistical Methods for Speech Recognition*. The MIT Press, 1997.

[63] W. Jesteadt, S. P. Bacon, and J. R. Lehman. Forward masking as a function of frequency, masker level, and signal delay. *Journal of the Acoustical Society of America*, 71(4):950–962, April 1982.

[64] J. D. Johnston. Transform coding of audio signals using perceptual noise criteria. *IEEE J. Select. Areas Commun.*, 6(2):314–323, February 1988.

[65] J.-C. Junqua, B. Mak, and B. Reaves. A robust algorithm for word boundary detection in the presence of noise. *IEEE Trans. Speech Audio Processing*, 2(3):406–412, 1994.

[66] M. Karjalainen. Kommunikaatioakustiikka. Technical report, Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing, 1999.

[67] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Lett.*, 1(10):144–146, October 1994.

[68] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[69] B. Kedem. Spectral analysis and discrimination by zero-crossings. *Proceedings of the IEEE*, 74(11):1477–1493, November 1986.

[70] C.-J. Kim. Dynamic linear models with Markov-switching. *Journal of Econometrics*, 60:1–22, 1994.

[71] D. Kimber and L. Wilcox. Acoustic segmentation for audio browsers. In *Proc. Interface Conference*, Sydney, Australia, 1996.

[72] A. Klapuri and M. Davy, editors. *Signal Processing Methods for Music Transcription*. Springer, 2006.

[73] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Trans. Commun.*, 28(1):84–95, January 1980.

[74] L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Trans. Inform. Theory*, 28(5):729–734, 2000.

[75] S. Lippens, J. P. Martens, T. De Mulder, and G. Tzanetakis. A comparison of human and automatical musical genre classification. In *Proc. 2004 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'04)*, pages 233–236, 2004.

[76] L. Lu, H.-J. Zhang, and H. Jiang. Content analysis for audio classification and segmentation. *IEEE Trans. Speech Audio Processing*, 10(7):504–516, October 2002.

[77] H. Lütkepohl. *Introduction to Multiple Time Series Analysis*. Springer-Verlag, second edition, 1993.

[78] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[79] J. D. Markel and A. H. Gray. *Linear Prediction of Speech*. Communication and Cybernetics 12. Springer-Verlag, 1976.

[80] M. F. McKinney and J. Breebaart. Features for audio and music classification. In *Proc. 4th Int. Symp. Music Information Retrieval (ISMIR-03)*, pages 151–158, Baltimore, USA, October 2003.

[81] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proc. First ACM International Conference on Digital Libraries*, 1996.

[82] P. Mermelstein. Automatic segmentation of speech into syllabic units. *Journal of the Acoustical Society of America*, 58(4):880–883, 1975.

[83] K. Minami, A. Akutsu, H. Hamada, and Y. Tonomura. Video handling with music and speech detection. *IEEE Trans. Multimedia*, 5(3):17–25, 1998.

[84] A. Misra, P. R. Cook, and G. Wang. A new paradigm for sound design. In *Proc. 9th International Conference on Digital Audio Effects (DAFx-06)*, Montreal, Canada, September 2006.

[85] B. C. J. Moore, editor. *Hearing*. Academic Press, 1995.

[86] D. P. Morgan and C. L. Scofield. *Neural Networks and Speech Processing*. Kluwer Academic Publishers, 1991.

[87] C. S. Myers and L. R. Rabiner. A level building dynamic time warping algorithm for connected word recognition. *IEEE Trans. Acoust., Speech, Signal Processing*, 29(2):284–297, April 1981.

[88] M. K. Omar, U. Chaudhari, and G. Ramaswamy. Blind change detection for audio segmentation. In *Proc. 2005 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'05)*, pages 501–504, Philadelphia, USA, March 2005.

[89] A. V. Oppenheim and R. W. Schafer. *Digital Signal Processing*. Prentice-Hall, 1975.

[90] D. O'Shaughnessy. *Speech Communications: Human and Machine*. IEEE Press, second edition, 2000.

[91] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[92] T. Painter and A. Spanias. Perceptual coding of digital audio. *Proceedings of the IEEE*, 88(4):451–513, April 2000.

[93] J. W. Paulus. Variable bitrate wideband speech coding using perceptually motivated thresholds. In *Proceedings of IEEE Workshop on Speech Coding*, pages 35–36, Annapolis, USA, 1995.

[94] S. Pauws. Musical key extraction from audio. In *Proc. 5th Int. Conf. Music Information Retrieval (ISMIR 2004)*, Barcelona, Spain, October 2004.

[95] J. M. Peña, J. A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, October 1999.

[96] J. Pohjalainen. A new HMM-based approach to broad phonetic classification of speech. In *Proc. of Eurospeech 2003*, pages 2921–2924, September 2003.

[97] J. Pohjalainen. Clustering techniques for acoustic-phonetic speech classification. In *Proceedings of the 6th Nordic Signal Processing Symposium (NORSIG 2004)*, pages 348–351, Espoo, Finland, June 2004.

[98] G. E. Poliner, D. P. W. Ellis, A. F. Ehmann, E. Gomez, S. Streich, and B. Ong. Melody transcription from musical audio: Approaches and evaluation. *IEEE Trans. Audio, Speech, and Language Proc.*, 15(4):1247–1256, May 2007.

[99] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[100] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.

[101] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[102] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice-Hall, 1978.

[103] R. Radhakrishnan, A. Divakaran, and P. Smaragdis. Audio analysis for surveillance applications. In *Proc. 2005 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, USA, October 2005.

[104] D. A. Reynolds and R. C. Rose. Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Trans. Speech Audio Processing*, 3(1):72–83, January 1995.

[105] T. D. Rossing, F. R. Moore, and P. A. Wheeler. *The Science of Sound*. Addison Wesley, third edition, 2002.

[106] J. Saunders. Real-time discrimination of broadcast speech/music. In *Proc. 1996 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'96)*, pages 993–996, Atlanta, USA, May 1996.

[107] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proc. 1997 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'97)*, pages 1331–1334, Munich, Germany, April 1997.

[108] M. R. Schroeder, B. S. Atal, and J. L. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66(6):1647–1652, December 1979.

[109] M. Sharma and R. Mammone. 'blind' speech segmentation: Automatic segmentation of speech without linguistic knowledge. In *Proc. Int. Conf. Spoken Language Processing (ICSLP-1996)*, pages 1237–1240, Philadelphia, USA, October 1996.

[110] A. Sheh and D. Ellis. Chord segmentation and recognition using EM-trained hidden Markov models. In *Proc. 4th Int. Symp. Music Information Retrieval (ISMIR-03)*, pages 185–191, Baltimore, USA, October 2003.

[111] J. Shifrin, B. Pardo, C. Meek, and W. Birmingham. HMM-based musical query retrieval. In *Proc. Second ACM/IEEE-CS Joint Conference on Digital Libraries*, 2002.

[112] L. J. Siegel and A. C. Bessey. Voiced/unvoiced/mixed excitation classification of speech. *IEEE Trans. Acoust., Speech, Signal Processing*, 30(3):451–460, June 1982.

[113] A. F. Smeaton and M. McHugh. Towards event detection in an audio-based sensor network. In *Proc. third ACM Int. Workshop on Video Surveillance and Sensor Networks*, Hilton, Singapore, November 2005.

[114] J. Sohn, N. S. Kim, and W. Sung. A statistical model-based voice activity detection. *IEEE Signal Processing Lett.*, 6(1):1–3, January 1999.

[115] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. PWS Publishing, 2nd edition, 1999.

[116] F. K. Soong and B.-H. Juang. Line spectrum pair (LSP) and speech data compression. In *Proc. 1984 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'84)*, pages 37–40, San Diego, USA, 1984.

[117] T. Svendsen and F. Soong. On the automatic segmentation of speech signals. In *Proc. 1987 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'87)*, pages 77–80, Dallas, USA, 1987.

[118] J. Tackett. ISO 226 equal-loudness-level contour signal, Matlab function. http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=7028. Accessed November 2007.

[119] E. Terhardt, G. Stoll, and M. Seewann. Algorithm for extraction of pitch and pitch salience for complex tonal signals. *Journal of the Acoustical Society of America*, 71(3):679–688, March 1982.

[120] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, second edition, 2003.

[121] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE Trans. Speech Audio Processing*, 8(6):708–716, 2000.

[122] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Trans. Speech Audio Processing*, 10(5):293–302, 2002.

[123] V.-T. Vu, Q.-C. Pham, and J.-L. Rouas. Audio-video event recognition system for public transport security. In *The Institution of Engineering and Technology Conference on Crime and Security*, London, UK, June 2006.

[124] B. Widrow and M. E. Hoff Jr. Adaptive switching circuits. In *IRE WESCON Conv. Rec.*, pages 96–104, 1960.

[125] L. Wilcox, F. Chen, D. Kimber, and V. Balasubramanian. Segmentation of speech using speaker identification. In *Proc. 1994 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'94)*, pages 161–164, Adelaide, Australia, April 1994.

[126] E. Wold, T. Blum, D. Keislar, and J. Wheaten. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3):27–36, 1996.

[127] L. Xu and M. I. Jordan. On convergence properties of the EM algorithm for Gaussian mixtures. *Neural Computation*, 8(1):129–151, January 1996.

[128] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book (for HTK Version 3.4)*. Cambridge University Engineering Department, 2006.

[129] T. Zhang and C.-C. Jay Kuo. Hierarchical classification of audio data for archiving and retrieving. In *Proc. 1999 IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP'99)*, pages 3001–3004, Phoenix, USA, March 1999.

[130] T. Zhang and C.-C. Jay Kuo. Audio content analysis for online audiovisual data segmentation and classification. *IEEE Trans. Speech Audio Processing*, 9(4):441–457, May 2001.

[131] B. Zhou and J. H. L. Hansen. Unsupervised audio stream segmentation and clustering via the Bayesian information criterion. In *Proc. Int. Conf. Spoken Language Processing (ICSLP-2000)*, pages 714–717, Beijing, China, October 2000.

[132] E. Zwicker and H. Fastl. *Psychoacoustics, Facts and Models*. Springer-Verlag, 1990.