David Weenink

Speaker-adaptive vowel identification

Speaker-adaptive vowel identification

Academisch proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. mr. P.F. van der Heijden
ten overstaan van
een door het college voor promoties ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit

op dinsdag 14 november 2006, te 12.00 uur

door

David Joseph Maria Weenink

geboren te Lichtenvoorde

Promotiecommissie:

Promotor:	Prof. dr. ir. L.C.W. Pols
Overige leden:	Prof. dr. P.P.G. Boersma
	Prof. dr. ir. F.C.A. Groen
	Prof. dr. ir. R.J.H. Scha
	Prof. dr. V.J.J.P. van Heuven (RUL)
	Prof. dr. T.M. Nearey (University of Alberta, Canada)

Faculteit der Geesteswetenschappen

Published by

SpeechMinded
Sloterweg 1229-B
1066 CH Amsterdam
The Netherlands
http: //www.speechminded.com

ISBN-10: 90-9021207-8
ISBN-13: 978-90-9021207-4

NUR: 616

Typeset by the author with the LaTeX Documentation System
Printed in The Netherlands by ACI Offsetdrukkerij b.v. - Amsterdam

Acknowledgements

Writing is no hobby of mine. I enjoy only the writing of software code. That is the reason why this thesis took many years to finish and the reason why I will be brief here. My heartfelt thanks go to my professors Louis Pols and Paul Boersma, my past and present colleagues at IFA, my colleagues around the world, my relatives, my friends and my family Thomas, Louise and Jenny.

Contents

Chapter 1

Introduction

Abstract

This thesis is about speaker-adaptive vowel identification by humans and by machines. In this first chapter we introduce the research topics and our approaches. We start with a short description of the vowel material that we have used. This is followed by an overview of various learning models used to model speaker adaptive behaviour.

This thesis is about speaker-adaptive vowel identification by humans and by machines. The vowel sets that we have studied are Dutch and American-English vowels. Several sources of vowel data have been used in this study. The first Dutch vowel database originates from our own recordings of sentences "*V* van p*V*t" (Weenink, 1986a). In these sentences *V* is one of the twelve Dutch monophthongal vowels /u, ɔ, o, ɑ, a, ʏ, ø, y, i, ɪ, e, ɛ/ as they occur in the Dutch "words" *poet*, *pot*, *poot*, *pat*, *paat*, *put*, *peut*, *puut*, *piet*, *pit*, *peet* and *pet*, respectively. To have variability in speaker context, we made recordings of ten men, ten women and ten children. This resulted in a database with 360 sentences (30 speakers × 12 vowels). This material will be more extensively described in chapter 2. Two other sources of Dutch vowel data that we have used are those from Pols, Tromp & Plomp (1973) and those from Van Nierop, Pols & Plomp (1973). We used their data on the first three formant frequencies and levels from 50 males and 25 females, respectively.

The acoustic properties of Dutch vowels have been described very well in the literature, and besides the already mentioned data above, many other studies exist. Koopmans-van Beinum (1980) measured formant frequencies of four speakers in eight different speech conditions. Weenink (1985) published besides the formant frequencies of ten men and ten women, the formant frequencies of ten children because data about Dutch children were not yet available. Van Heuven & Van Houten (1985) measured Dutch vowel characteristics produced by immigrants. Van Son & Pols (1990) measured formant frequencies of Dutch vowels in a text, read at normal and fast rates. Recently, in a Dutch-Flemish co-operation, a large database with recorded speech from 160 teachers of standard Dutch was set up and the first three formant frequencies and durations of a subset, 20 males and 20 females, were measured (Adank, Van Hout & Smits, 2004; Adank, 2003). The speech material in Adank's study consisted of read monosyllabic utterances in a neutral consonantal /s*V*s/ context.

Besides the descriptions of Dutch vowels in terms of formant frequencies, descriptions in terms of bandfilter values have been given too. Both Pols et al. (1973) and Van Nierop et al. (1973) present vowel spectra in terms of 18 one-third octave filter levels for the data sets with the 50 males and the 25 females, respectively.[1] They showed that in the plane formed by the first two principal components of these bandfilter values, the relative positions of the vowels are very similar to their orientations in the plane formed by the logarithms of the first two formant frequencies.

The other important non-Dutch source of vowels in our study is TIMIT, an acoustic-phonetic American-English speech corpus (Lamel, Kassel & Seneff, 1986). This corpus consists of recordings from 630 speakers, 438 males and 192 females, each of whom pronounced exactly ten sentences. We have used this database because it provides easy access to all realizations of the vowels since all these sentences have been hand-labelled at the word level and at the phoneme level. This corpus is often used in the literature as a standard. In chapter 8 we discuss extensively how we have made the audio files and the label files in this database accessible to ourselves and to the general public. To give an indication of its size: it contains 241,225 hand-labelled segments.

[1] Their actual analysis was with 21 filters. To cope with fundamental frequency variations they sum the energies in the first three filters, and the energies in filters four and five.

As was said before, the theme of this thesis is speaker-adaptive vowel identification. The perceptual aspects of speaker-adaptive behaviour with respect to vowel identification will be described in the next chapter, where we show results of listening experiments that we performed in the nineteen eighties (Weenink, 1986a). These experiments were about identification of short vowel segments in mixed and blocked speaker contexts. In a blocked speaker context the listener knows that the stimulus he will hear next is from the same speaker as the previous stimulus. In a mixed speaker context the stimulus he will hear next, will most probably not be from the same speaker as the stimulus he just heard and no stable guess about speaker characteristics can be made. The results of these experiments were in line with results obtained by others at that time and showed that listeners are better able to identify vowels in a blocked speaker context than in a mixed speaker context (see Strange (1989) for an overview). Although this speaker context effect is not a large effect, at most a few percent better identification, it is always present. At that time we could only present this mixed-blocked effect as such without being able to present a proper model for description that could be implemented on a computer. We could only, as many others, offer the following explanation: in the blocked speaker context the listener has ample time to adjust to the idiosyncrasies of a speaker and therefore is able to identify these stimuli somewhat better.

We have decided to include our data about these experiments in this thesis, despite their antiquity, because of two reasons.The first reason is that not all results of these experiments have been published before. The second reason is that in the penultimate chapter of this thesis we introduce a simple speaker-adaptive vowel classification model that is able to reproduce, at least qualitatively, the different classification results for these blocked and mixed speaker context experiments. In this way the thesis will present an overview of our "march" from these first experiments to the final model that simulates an adaptive listener. During this march several excursions have been made into different machine learning models in which we tried to incorporate some of the adaptive behaviour of subjects in the listening experiments.

One of these excursions concerns the development of the computer program PRAAT (Boersma & Weenink, 2006) in which Weenink has tried to incorporate all these learning models in a user-friendly way.

We will, for now, use the terms "learning model", "classification model" and "classifier" indistinguishably. We may look at a classifier as a machine that has inputs and outputs. In the statistical literature the input and output variables are often referred to as either *independent* and *dependent* variables, or as *predictors* and *responses*, respectively. We will use these sets of terms interchangeably. When a data item is presented at the input, it is processed by this machine and, after processing, an output appears. The output is then an indication of the class to which the input most probably belongs.

Before a classifier can do any classification at all, it has to learn the association between inputs and outputs. Depending on whether we refer from the inside or the outside of the model, we speak of *learning* or *training*, respectively. Only after the model has learned the regularities from the data, i.e. the relation between its inputs and outputs, may it be able to classify data. Sometimes training and classification are two separate phases in a model and either phase has to be activated explicitly; we

then speak of *supervised* learning because some external knowledge has to be used to switch phases. For the training phase a combination of three items is needed: the input data, the corresponding output data and the classifier to connect these two data sources by means of a particular learning algorithm, i.e. a procedure that tells the classifier how to connect inputs and outputs. In the classification phase only two items are needed, the classifier and the input data, and the third item, the output data, will be generated by the classifier. In *unsupervised* learning the two phases cannot be distinguished: the model is permanently being trained, i.e learning while classifying.

The various learning models that we describe in this thesis and have incorporated in PRAAT are:

- Discriminant analysis and principal component analysis. Principal component analysis (PCA) learns from *unlabelled* multivariate data how to represent these data in a very efficient way. Discriminant analysis (DA) learns from *labelled* multivariate data. It can be used both as a data reduction tool and as a data classification tool. As a method for data reduction it is comparable to PCA as it tries to find a "rotated" co-ordinate system that more optimally describes the data. The criteria to find the new co-ordinate systems has to differ for DA and PCA. DA finds directions where classification is a maximum, whereas PCA, having no access to classes, finds directions where the variance is a maximum. The measure used for classification is the quotient of variance-between-classes and variance-within-classes, hence the requirement for labelled data. The success in data reduction for both methods depends on the distribution of the eigenvalues of (subsets of) the data matrices.

  For the training and the classification stages of the DA we do not need to explicitely calculate the rotated co-ordinate system as the classification can be solely based on the Mahalanobis distance metric which only makes use of covariance matrices and class means. In the training phase, the class means and covariances are extracted from labelled data, therefore we can categorize DA as a supervised learning model. In Linear Discriminant Analysis (LDA) we use pooled class-specific covariance matrices in the distance calculation, i.e. only one covariance matrix is involved in the distance calculation and this matrix is the same for all the classes. In Quadratic Discriminant Analysis (QDA), the class-specific covariance matrices themselves are used in the distance calculation. The result is that now the boundaries between the classes, when projected on any two-dimensional space, are quadratic curve segments, while they are straight line segments for LDA. A data point on the boundary between two classes has an equal probability of belonging to both.

  QDA has as a side effect that "distance" is no longer symmetric: the distance from cluster A to cluster B need not be equal to the distance from cluster B to cluster A because these distances depend on possibly different covariance matrices. As with many classical tools, DA gives best results if data is normally distributed. In chapter 3 we discuss the models and their implementations in the PRAAT program with some examples. Since the LDA model is so fundamental, all other learning and classification models will be tested against it.

- Feedforward neural networks. In feedforward neural networks we model parts of the brain with algoritms that, in a very rudimentary way, mimic the interaction of neurons. Contrary to DA, these neural net models do not make any assumptions on the distribution of the data, i.e. the data do not have to be normally distributed. Just as DA, feedforward neural networks have separate training and classification phases and therefore belong to the class of supervised models. In the training phase, stimuli are presented one by one and the network adapts to the individual stimulus. This contrasts with discriminant learning which is batch-oriented, i.e. a statistic like the mean is derived from all data taken together. A theoretical introduction to these networks can be found in chapter 4. In chapter 5 we describe aspects of the training of a neural net, with particular emphasis on the cost functions being used. Two further chapters, 6 and 9, are dedicated to a supervised model of speaker adaptation with feedforward neural nets. In these models speaker adaptation is simulated by varying only the biases of the hidden layer or the output layer in an already trained neural net.

- Canonical correlation analysis (CCA). In CCA correlations between two groups of multivariate data are determined. Most people do not think about CCA as a learning model although it can be used as such: if we correlate vowel data with vowel categories we have a learning model similar to LDA. The model is also batch-oriented and performs best when the data are normally distributed. Chapter 7 extensively discusses this model and gives some examples of its use.

- Adaptive Resonance Theory networks (ART). These neural network models are based on the perception theories of Grossberg. An overview of his ART theory can be found in Grossberg (1998). ART is based on unsupervised real-time learning, i.e. learning to structure data without having a teacher around that has already pre-labelled the input. The important problem to be solved in this context is the *stability-plasticity* dilemma: how can we learn new things without gradually forgetting old things. In chapter 10 we will discuss our own algorithmic variant of one of these ART models that, although supervised, is based on a combination of two unsupervised modules.

The inputs for all these learning models will consist of either scaled formant frequency values when they are provided in the literature, or bandfilter values if the audio is available. Formant frequency values were available from Pols et al. (1973), Van Nierop et al. (1973) and Weenink (1985). For TIMIT we have performed a bandfilter analysis because this can be performed objectively, without operator intervention. Formant frequency analysis always involves subjectivity. For example, Pols et al. (1973) state that in a number of cases a priori knowledge of where the formant should be located played a significant role in their decision on formant frequency values for 50 male speakers. When the fundamental frequency increases, the problems aggregate. For the 25 female speakers, Van Nierop et al. (1973) had difficulties in determining formant frequencies in 40% of the cases if no a priori knowledge was used. One could argue that these interventions are solely based on their measurement procedure, i.e. spectral interpolation. However, recently, also Adank et al. (2004) had to manually alter 20–25% of their automatically determined formant contours. Of course,

with bandfilter analysis we also get our share of analysis troubles: the levels of the low-frequency filters can vary considerably with fundamental frequency. This effect has to be considered when we compare bandfilter measurements at different fundamental frequencies. Pols et al. (1973) and Van Nierop et al. (1973) sum the first three filter outputs and the next two in their one-third octave bandpass filter bank, while Van Alphen (1992) interpolates filter values, to compensate for these filter-level variations due to varying fundamental frequency. In our own processing of bandfilter values we never mix male and female data but we separate data by speaker gender to avoid this complication.

The learning models incorporated in this thesis do not have any explicit notion of time. Hidden Markov models or, for example, recurrent neural nets will not be discussed. The learning models that we describe function very well with vowel data, although they do not explicitly model vowel dynamics. Dynamics in these models can be simulated by grouping several analysis frames into one data item, as can be seen for example in figure 8.9. This proved to be sufficient for the scope of this study since most vowels do not behave in a highly dynamic fashion. And as the title of this thesis indicates: we were not interested in modelling the dynamic aspects of vowels per se, but in modelling the adaptive behaviour that tries to cope with variations in these vowels due to different speaker contexts. In the final chapter 12 we will come back to this theme and explore some of the results.

Chapter 2

The identification of vowel stimuli from men, women, and children[‡]

Abstract

In this chapter we present a series of listening experiments in which short vowel-like stimuli were manipulated in terms of speaker context, consonantal context, duration, and fundamental frequency. The results indicate that isolated vowels are not by their nature impoverished stimuli as some authors tended to believe. When the duration of the stimuli is limited to 50 ms, the number of confusions increases with respect to full duration vowels, especially the number of confusions between long and short vowels. Using resynthesized stimuli, in which fundamental frequency and spectral characteristics of speaker categories (men, women and children) were crossed, showed that the number of confusions increases with the difference between the fundamental frequency of the resynthesized and the original signal. Two factors were responsible for this increase: (1) the spectral ambiguity increases when the fundamental frequency increases and (2) the perceived *misfit* between the fundamental frequency and the spectral envelope. All experiments showed more confusion errors for the mixed than for the blocked speaker condition. In the following chapters we introduce several models that try to simulate this blocked/mixed effect. Finally, in chapter 11 we will show an adaptive model that simulates this blocked/mixed behaviour.

[‡]This chapter is an extended version of Weenink (1986a).

2.1 Introduction

In this chapter we present results of a series of listening experiments on speaker normalization. These experiments were performed in the mid 1980's and were in line with experiments by other authors at that time who also tried to investigate aspects of speaker normalization by means of manipulating vowels, vowel context and speaker context. Overviews of these kinds of experiments have been given by Strange (1989) and Nearey (1989). Despite the antiquity, we present our data here because they were never published completely, and because they were the starting point for the ideas that finally culminated in a model for speaker-adaptive vowel identification that will be described in chapter 11.

Of all the possible aspects of normalization in the present chapter we would like to consider especially the perceptual aspects which concern speaker variation. In a series of eight listening experiments we have investigated how well listeners can recognize vowels from different speakers when these vowels are presented in a mixed and in a blocked condition. In the mixed condition the listeners, on each vowel, encounter a voice that is unfamiliar and unpredictable, while in the blocked condition the listener hears a series of vowels produced by the same speaker. In this latter condition there is ample opportunity to become familiar with the voice because the speaker is fully predictable from one vowel to the next.

In order to gain better insight in this speaker variation, the vowel stimuli we used were manipulated in terms of duration, consonantal context, and fundamental frequency. Several authors have directed their attention to the speaker's context effect in the recognition of vowels a.o. Strange, Verbrugge, Shankweiler & Edman (1976), Verbrugge, Strange, Shankweiler & Edman (1976), Macchi (1980), and Assmann, Nearey & Hogan (1982). The experimental scheme is generally such that vowel stimuli are presented in two conditions, mixed and blocked, to subjects who are asked to identify the vowel. Although there are great differences in the absolute error rates in these experiments, they all reach the same conclusion: uncertainty about a speaker as is the case of the mixed condition, leads to more confusion errors than when the speaker is 'known', or at least does not change, as in the blocked condition. This effect is persistent even if the vowels are gated to a duration of 100 ms.

The influence of consonantal context on the perception of vowels is still under debate (see Nearey (1997) and Van Son & Pols (2001)). Some experimenters (Strange et al. (1976); Verbrugge et al. (1976); Rakerd, Verbrugge & Shankweiler (1984)) report significantly fewer confusion errors made by listeners for vowels presented in a CVC context than for vowels in isolation. According to them the consonantal context aids vowel identification. Other investigators, like Macchi (1980) and Assmann et al. (1982), do not endorse this hypothesis. On the contrary, they state that consonant coarticulation is not a necessary condition for accurate identification of naturally produced vowels, and that the consonantal advantage found by the other groups is not a genuine perceptual effect but a mere methodological artifact. Diehl, McCusker & Chapman (1981), using speech synthesis, did not find superior performance of listeners on CVC stimuli either. At first sight one could think that because the formant trajectories of consonant-bounded vowels often fail to reach the frequencies characteristic of

vowels produced in isolation (Lindblom, 1963; Stevens & House, 1963; Koopmans-van Beinum, 1980), consonant-bounded vowels would appear to be acoustically less distinctive than isolated vowels. Experimental evidence shows that vowels in CVC context are recognized just as well as vowels in isolation, which means that dynamic spectral features must compensate for the loss in static distinction.

There is additional evidence that the human auditory system can predict spectral targets on the basis of the transitional information (Furui, 1986). This means that, when we gate short segments out of the cental parts of vowels produced in isolation and in /p-t/ context, there will be a difference in listeners' performance because the vowel segments taken from the CVC context are acoustically less distinctive: they lack the transitional information.

As for our final point, the influence of fundamental frequency and timbre on the quality of vowels, we can say that vowel quality is largely independent of naturally varying fundamental frequency because the spectral envelope is determined by the shape and length of the vocal tract rather than by the vocal cords. This envelope does not shift when a vowel is produced at a different fundamental frequency.

Slawson (1968) studied what effect changing the fundamental frequency and/or the formants has upon vowel quality. He showed that the perceptual distance between two vowels whose fundamental frequencies differ by an octave, could be minimized by raising the formants of the vowel with the highest pitch by approximately 10%. Because higher formants as compared to lower ones show smaller variations from vowel to vowel (e.g. Weenink (1985, table I)), Fujisaki & Kawashima (1968) tested whether a normalization process could be based on higher formant frequencies. They showed that neither fundamental frequency nor higher formants by themselves are sufficient for perceptual normalization but that both are necessary in any succesful normalization theory. Van Bergem (1986), in his study on vowels in /p-t/ context, reports that it is the joint effect of acoustic context, pitch, and timbre that is important in the normalization process, in such a way that pitch and timbre determine speaker category (men, women and children); after this precategorization the reference set of each category can be used for further classification.

The global design for all eight experiments we will describe, was alike: the recordings of the speech material and its further processing were done once and served as a basis for all experiments. The preparation procedure of the stimulus tapes, the listening conditions and the subjects were the same; only the stimuli differed for each experiment. In the following sections we will give a description of these parts.

2.2 Speech material

Recordings were made from ten male, ten female and ten children's voices. All were native speakers of Dutch and they were carefully selected on their ability to speak the standard Dutch language without dialect influences. The recordings were made in an anechoic room with a Sennheiser MD421N microphone and a Revox A77 taperecorder. The recordings consisted of series of sentences "V van pVt" (V from pVt), where V is one of the twelve Dutch vowels /u, ɔ, o, ɑ, a, ʏ, ø, y, i, ɪ, e, ɛ/. These

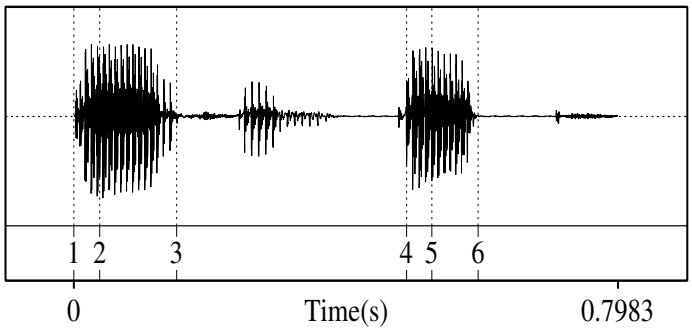

Figure 2.1. The positions of marks in the sentence "ɑ vɑn pɑt".

sentences were read from paper with normal intonation; each sentence was repeated at least twice. For Dutch the orthography is unambiguous and causes no confusions in pronunciation, at least not in closed monosyllabic words. During the recordings of the children's voices a person familiar to the child was always present in the anechoic room for reassurance. The sentences on paper were repeated until they were correctly spoken, but in general the children made few mistakes and hardly any repetitions were necessary.

2.3 Further processing of the speech material

The sentences on tape were digitized with a sampling frequency of 10 kHz and a precision of 12 bits/sample.[1] For each of the twelve vowels the best recording of each sentence from every speaker was stored on disk and used for further processing. After selection and digitization our speech database consisted of 360 utterances of the type "V van pVt" (30 speakers × 12 vowels) on disk. With the help of a speech editing program (Buiting, 1986), sentences were marked in a way shown by figure 2.1. Marks 1 and 3 isolate the first vowel, while marks 4 and 6 isolate the vowel produced in a /p-t/ context. Mark 2, which is always within a stable part within the first 100 ms of the vowel, functions as a starting point for subsequent physical analysis, resynthesis and selection. Mark 5 has this function for the vowel in /p-t/ context and is placed somewhat more toward the middle of this vowel, at a position where the amplitude is most stable. Marks 2 and 5 were set "by eye". Table 2.1 on the facing page displays the mean durations in milliseconds of the intermark durations $3-1$ and $2-1$, for the vowel produced in isolation, and $6-4$ and $5-4$, for the vowel produced in /p-t/ context.

[1] Nowadays, of course, we would have used a better precision and a higher sampling frequency.

Table 2.1. Intermark durations in ms. See figure 2.1 for the positions of marks. ($N = 30$ per data point)

	3 – 1		2 – 1		6 – 4		5 – 4	
Vowel	Mean	σ	Mean	σ	Mean	σ	Mean	σ
o	238	73	52	15	218	37	75	31
a	225	55	58	21	218	33	71	28
ø	245	70	58	14	215	43	68	31
e	234	66	53	14	199	43	60	23
y	186	49	60	16	134	31	57	26
u	174	48	47	14	133	32	54	34
ɔ	165	45	52	17	131	36	55	31
ɑ	162	45	44	18	131	23	50	19
ʏ	166	48	53	17	130	29	53	21
ɛ	167	34	50	17	130	23	53	18
i	175	46	48	12	126	32	48	27
ɪ	164	46	48	16	125	33	47	21

The table shows that the duration of the vowel in /p-t/ context, column $6-4$, is always less than the duration of the vowel produced in isolation, column $3-1$. The table also shows very clearly the differences in duration between the four 'long' vowels /o/, /a/, /ø/, /e/ and the other eight vowels. Formant frequencies were measured in a 25.6 ms segment around mark 2 of all the vowels from our 30 speakers by means of a special interactive computer program (Weenink, 1986b). The measuring procedure used an interactive linear predictive coding analysis with varying number of coefficients to get the best formant frequency estimates. Details of the analysis can be found in Weenink (1985). These frequencies are listed in tables A.1, A.2 and A.3 in appendix A and are also available as a standard table in the PRAAT program.[2] Mean values are displayed in figure 2.2.

In the following experiments we wanted to use all the vowels of a speaker twice in two listening conditions, mixed and blocked. Using all the speakers of our database, the total amount of stimuli would have been 1440 (30 speakers × 12 vowels × 2 conditions × 2 repetitions), far too many for any practical listening experiment. We decided to select five male, five female and five children speakers out of the 30 speakers. This selection was made on the basis of a bandfilter analysis and the results of a pilot listening experiment with resynthesized vowels from all 30 speakers. From the categories man, woman and child we selected some 'extreme' and some 'mean' speakers and these 15 selected speakers were used in all the experiments we decribe in this chapter.

[2] This formant table can be found under the `Tables` option in the `New` menu as `Create formant table (Weenink 1985)`.

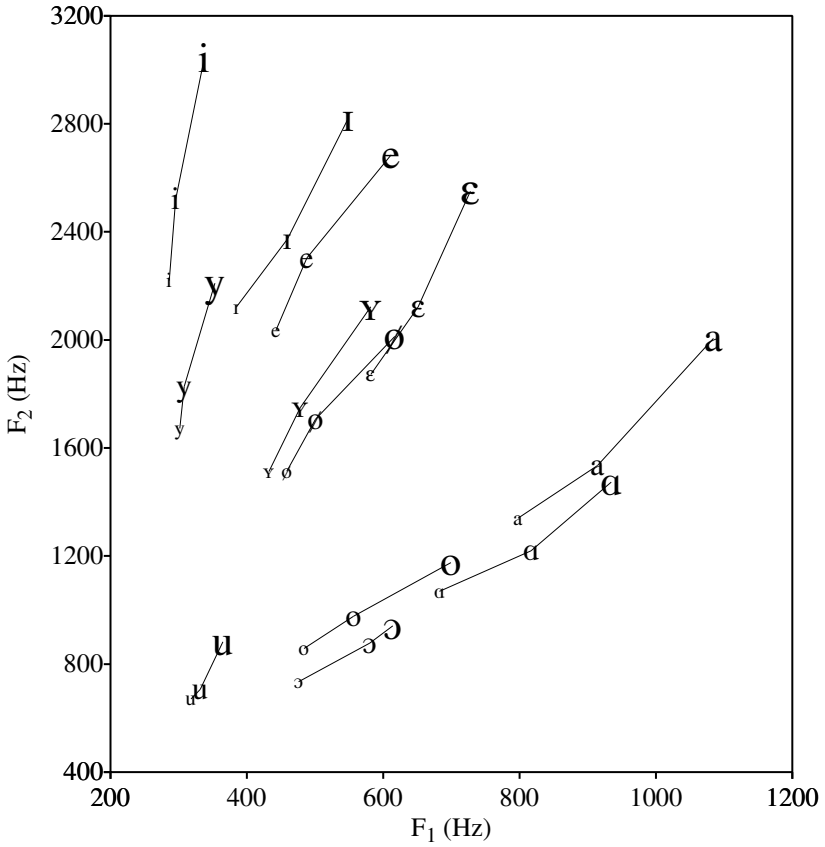

Figure 2.2. Relative positions of the twelve Dutch vowels from male, female and child speakers. Each symbol is the average of the formant frequencies from the production of the vowel in isolation by ten males (smallest size), ten females (intermediate), and ten children (largest).

2.4 Preparation of stimulus tapes

For each of the eight experiments we describe, five audiotapes were prepared and each tape contained a different random order of the stimuli. Each tape consisted of two parts: the first part with the stimuli recorded in the mixed condition and the second part with the stimuli recorded in the blocked condition. The randomization procedure we used was as follows: in the mixed condition 360 stimuli (15 × speakers × 12 vowels × 2 repetitions) were quasi-randomized under the constraint that maximally two adjacent stimuli came from the same speaker. The last 20 stimuli of this series

of 360 were also put at the beginning of the tape and served as dummies to let the subjects get accustomed to this kind of stimuli. In the mixed condition we thus have a total of 380 stimuli. In the blocked condition the 15 speakers were randomized first, then for each speaker 24 stimuli (12 vowels × 2 repetitions) were randomized under the constraint that no two adjacent vowels were the same; the last six stimuli of this series were repeated at the beginning, summing to 30 stimuli for each speaker and 450 in this condition (15 speakers × 30 stimuli). Both in the mixed and in the blocked conditions we used a 2.5 s interstimulus interval. Between every ten stimuli a double beep was played as a separation marker with the same 2.5 s time interval. Addionally, in the blocked condition after every 30 stimuli a triple beep tone was played to separate different speakers.

2.5 Subjects

The listeners were ten male and ten female, phonetically untrained, paid volunteers. Most of them were students at the Faculty of Arts of the University of Amsterdam. All were native speakers of Dutch, with no hearing deficiencies and ranging in age from 20 to 30 years.

2.6 Listening conditions

The identification tests were run in a special acoustically isolated studio room at the Language Department (ITT) of the Faculty of Arts of the University of Amsterdam. In each session four subjects at a time could be handled, so there were five sessions in every experiment. Test tapes were presented via a Revox A77 tape recorder, Sansui AU-22 amplifier, and a set of Sennheiser HD22 headphones at a comfortable listening level. Subjects were seated in front of a specially developed response unit which consisted of a monitor and a standard QWERTY keyboard, and they responded by pushing a key on the keyboard (see figure 2.3).

Twelve keys on the keyboard were marked with stickers, that showed the orthographic symbols ‘pVt’; a thirteenth was labelled ‘FOUT’ (error). The remaining keys of the keyboard were covered with a special protection plate. The layout is shown in figure 2.4.

Although we did not expect as much orthographic interference as in English, vowels that were expected to get confused orthographically such as /y/ and /ʏ/ (pUUt and pUt) were placed as closely as possible to each other, in order to attract special attention of the subject when responding: long vowels were placed on the top row and the corresponding short vowel was placed just underneath it on the next row. A subject’s response was immediately displayed on his monitor, together with the response number. In case of a typing error or an incorrect response, subjects were able to correct their last given response by using the “FOUT” button and then give their intended response. This corrected response was displayed with the same response number as the previous one. The response units of the four subjects were connected to a central Apple IIe computer (Weenink & Wempe, 1986). The responses of the four subjects were

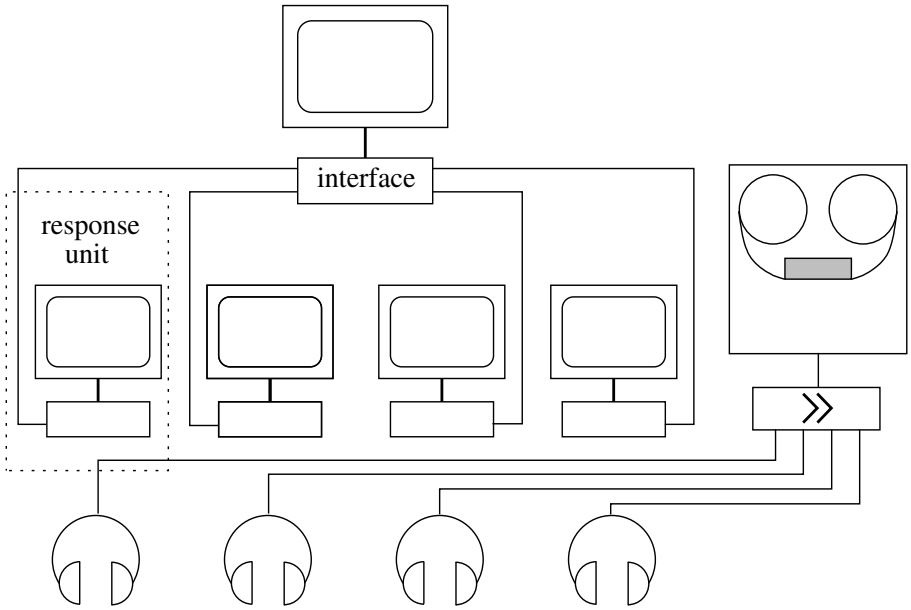

Figure 2.3. Listening configuration. Four response units are connected to a central unit (Apple IIe). A Revox A77 taperecorder and Sansui AU-22 amplifier provide the audio signals to the earphones.

displayed on the Apple IIe's monitor together with the stimulus number and type. In this way the experimenter had full control over the experiment and could intervene if necessary. The experimenter would stop the audio tape when a subject either forgot to respond or gave a double response to a stimulus; the experimenter would then ask the subject who was in error to perform the necessary corrections. The double beep between a series of ten successive stimuli served as a timer. Subjects made few mistakes: approximately once in every session the experimenter had to stop the tape to make a correction. Before a session started the subjects were instructed that the experiment was on vowels and that different vowels of different speakers were mixed in the first part of the experiment. After the 380 stimuli in the mixed condition had passed, the stimulus tape was stopped and the subject was granted a short break. Then subjects were told that in the next part they would hear the stimuli blocked for each speaker and that every new speaker would be announced by a triple beep. In general a full session, which consisted of stimuli presented in mixed and in blocked condition, took about 50 minutes. The responses of all four subjects were stored on the floppy disk of the Apple IIe computer and served as input for further data processing such as cumulative results and confusion matrices.

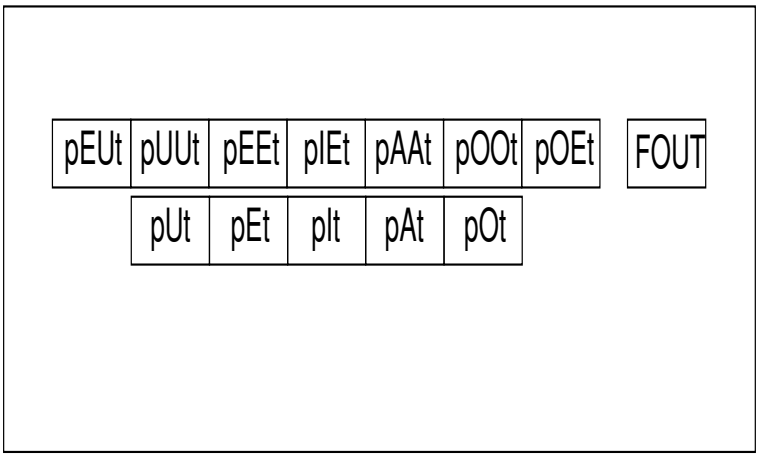

Figure 2.4. Layout of the keybord part of the response unit.

2.7 Stimuli for the experiments

In this section a description of the stimuli is given for each of the eight experiments.

Experiment 1

The vowels as produced in isolation were selected with their original length (in figure 2.1 this is the part of the sentence between marks 1 and 3). This experiment is a replication of the experiments performed for American-English vowels by a.o. Strange et al. (1976), Macchi (1980) and Assmann et al. (1982) and investigates how well natural, isolated, Dutch vowels are recognized when they are presented in mixed and blocked conditions to listeners. We expect listeners to make few mistakes, in accordance with the experiments of Assman et al. and Macchi.

Experiment 2

Stimuli were 50-ms segments around mark 2 of the vowel produced in isolation were selected. The initial half of a cosine window was used to smooth the onset of the first 5-ms portion of the selected signal; this was followed by 40 ms at the original amplitude; the last 5 ms of the signal was smoothed by the second half of the cosine window. We chose this 50-ms length to have a duration which comes close to the duration of short vowels in conversational speech. A second reason was that we wanted to increase the number of confusions. Because all segments are equalized in duration we introduce extra confusions between vowels where duration is the main cue for separating them, namely between the pairs (/ø∼ʏ/), (/o∼ɔ/), (/e∼ɪ/) and (/a∼ɑ/) (Pols et al., 1973; Van Nierop et al., 1973; Nooteboom & Doodeman, 1980). We name confusions

of this type 'long/short confusions'. Assmann et al. (1982) find a mixed/blocked effect in their experiment with vowel durations gated to 100 ms; we too expect this effect to happen despite our shorter duration of 50 ms because if speaker information is still present in the vowel, listeners can take advantage of this fact when the vowels are presented in a blocked condition.

Experiment 3

Stimuli were 50-ms segments around mark 5 of the vowel produced in /p-t/ context were selected and smoothed as described above. The importance of dynamic spectral information has been reported for vowel perception. In continuous speech, when vowels can be coarticulated with consonants, the spectral pattern of the speech signal varies in such a way that the acoustic targets found in isolated vowels may not be attained (Stevens & House, 1963; Koopmans-van Beinum, 1980). This phenomenon is referred to as target undershoot and it is determined by speaking rate, sentence and word stress and individual style of speech (Lindblom, 1963). Because of this possible undershoot we expect our vowels taken out of their /p-t/ context, to be acoustically less distinctive than their counterparts that were produced in isolation when both are gated to a fixed short duration and are presented in isolation.

Experiment 4

Stimuli were 50-ms segments, each of which was resynthesized as a stationary signal from the linear prediction analysis of order 12 that was done on a 25.6-ms segment around mark 2 of the vowel produced in isolation. The pitch in these 50-ms resynthesized segments was constant and was taken equal to the mean pitch of the corresponding analyzed segment. From pilot studies we got the impression that listeners made a precategorization of stimuli, mainly on the basis of pitch, into male-like, female-like and/or child-like. In order to manipulate the fundamental frequency in a well-defined way we had to use resynthesis. Because of the inherent smoothing performed by any analysis-resynthesis system we expect more confusion errors in this experiment than in the preceding ones. Although the spectral envelope of the resynthesized signal is smoothed we still expect enough speaker-specific information to be present in this signal to be of help in the blocked condition, which means that there should be a difference in the listeners' performance in the mixed and blocked conditions.

Experiment 5

Stimuli were 50-ms segments, resynthesized with a fundamental frequency of 135 Hz from linear prediction coefficients. Resynthesis was performed using the 12th order linear prediction coefficients from experiment 4. The chosen frequency is approximately the mean male fundamental frequency as it was measured from the voices of our ten male speakers. In resynthesizing all the analysed vowels of our five male, five female and five children speakers with the same 'male-like' fundamental frequency of 135 Hz we present to the listener partly conflicting vowel information: on the one hand

a spectral envelope belonging to a certain speaker category and on the other hand a fundamental frequency that did not 'fit' (in this experiment this was the case for children and female voices). On the basis of investigations by Fujisaki & Kawashima (1968) and Wendahl (1959) we know that there is an interaction between fundamental frequency and spectral envelope. Therefore our expectation is that especially in the categories women and children the amount of confusion will rise.

Experiment 6

This experiment contained 50-ms segments, resynthesized with a fundamental frequency of 235 Hz from linear prediction coefficients. The prediction coefficients from experiment 4 were used. 235 Hz is approximately the mean female fundamental frequency as was measured from our ten female voices. Again, as in experiment 5, conflicting information is present in the resynthesized vowels, but this time it should interact mainly with the vowels from the male and the children speakers.

Experiment 7

Stimuli were 50-ms segments, resynthesized with a fundamental frequency of 335 Hz from linear prediction coefficients. This frequency is approximately the mean children's fundamental frequency as was measured from our children's voices. The same prediction coefficients were used as in experiments 4, 5 and 6. This time we expect the male and female vowels to show the largest effects because their vowels are resynthesized with the greatest shift in fundamental frequency with respect to their 'normal' fundamental frequency.

Experiment 8

In this experiment we presented 50-ms segments, resynthesized with a white noise source as the glottal signal from the linear prediction coefficients. The same prediction coefficients were used as in all the above resynthesis experiments. Because of the fact that a very important indication of speaker category, the fundamental frequency, is absent we expect more confusion errors in this experiment than in experiment 4 where the vowels are resynthesized with their 'own' fundamental frequency. If, on the other hand, the information about speaker category is still present in the spectral envelope in another way, listeners' performance should be comparable.

2.8 Results and discussion

In table 2.2 on the next page results of the eight listening experiments are presented. This table contains the mean error percentages for each experiment, in the mixed and the blocked condition averaged over all subjects and vowels, both for all speakers as well as for the separate speaker categories of men, women and children. Table 2.3 on the following page presents the data corrected for long/short confusions. This correction implies that a short vowel response given to its long counterpart stimulus

Table 2.2. Error percentages for each experiment, averaged over subjects (20), vowels (12) and speakers (15). The speakers have also been split up into the categories Men, Women and Children for both the Mixed (M) and the Blocked (B) condition. See section 2.7 for a further specification of the experiments.

Exp.	Description	Averaged		Men		Women		Children	
		M	B	M	B	M	B	M	B
1	vowel V	10.9	4.4	9.9	4.2	11.5	4.0	11.4	5.0
2	50 ms from V	35.6	31.2	30.7	26.4	34.0	30.0	42.0	36.8
3	50 ms from pVt	40.6	33.6	39.2	30.1	38.5	34.0	44.0	36.6
4	50 ms, mean F_0	44.6	40.3	40.3	36.3	41.8	36.2	52.7	48.5
5	50 ms, $F_0 = 135$	49.9	42.8	44.0	35.3	42.9	38.9	62.8	54.0
6	50 ms, $F_0 = 235$	49.5	43.3	50.1	43.0	41.8	38.1	56.6	48.9
7	50 ms, $F_0 = 335$	59.9	57.0	70.3	68.0	55.4	52.7	53.9	50.5
8	50 ms, noise	46.7	39.2	45.1	33.7	39.5	36.1	55.5	47.7

Table 2.3. Same data as table 2.2 but all data have been corrected for long/short confusions.

Exp.	Description	Averaged		Men		Women		Children	
		M	B	M	B	M	B	M	B
1	vowel V	9.6	3.8	8.5	3.8	10.6	3.6	9.8	4.0
2	50 ms from V	18.7	15.1	12.9	11.1	16.1	12.7	27.1	21.6
3	50 ms from pVt	24.2	18.1	22.0	15.1	21.0	17.7	29.6	21.5
4	50 ms, mean F_0	29.0	25.8	22.6	20.8	24.9	20.7	39.3	35.9
5	50 ms, $F_0 = 135$	36.7	28.2	25.0	17.3	28.0	23.3	57.0	44.0
6	50 ms, $F_0 = 235$	36.7	29.0	36.3	28.0	26.1	22.0	47.9	37.0
7	50 ms, $F_0 = 335$	49.2	45.8	62.8	59.3	42.5	40.4	42.4	37.8
8	50 ms, noise	34.5	26.0	30.5	19.6	25.3	21.2	47.8	37.4

is considered to be a correct response. The reverse, a long vowel responded to its short counterpart stimulus, is considered an error response. In figure 2.5, and parts (a) and (b) of figure 2.6 the data from these tables have been visualized with histograms. From experiment 1 we can conclude that vowels produced in isolation and presented in a mixed condition, can be recognized rather well by listeners: only 10.9% errors. This result is even significantly better in the blocked condition: only 4.4% errors. These percentages are close to the percentages that Macchi (1980) and Assmann et al. (1982) report. See table 2.4 for an overview. We want to emphasize that the differences in error percentages between the mixed and the blocked condition were statistically significant ($p < 0.01$) in all eight experiments. Reducing the duration of the stimuli to 50 ms (experiment 2) has considerably increased the number of incorrect responses: 35.6 and 31.2%, respectively, for the mixed and blocked conditions.

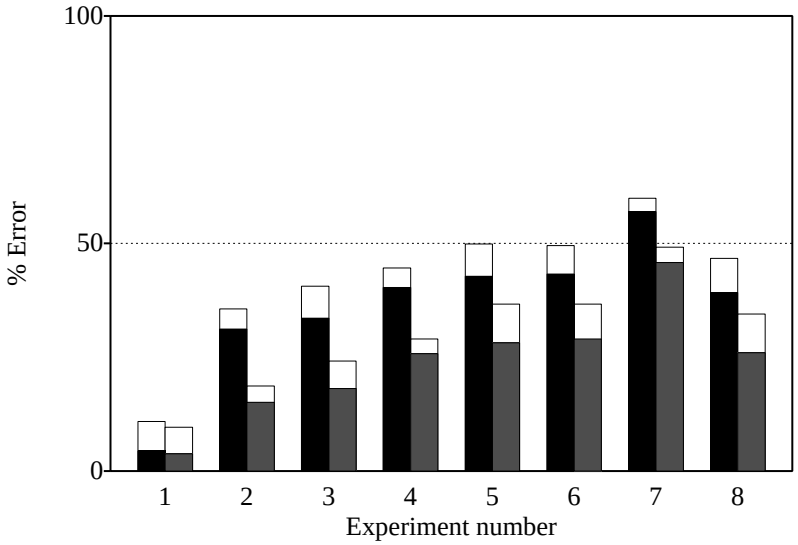

Figure 2.5. Error percentages averaged over subjects (20) and vowels (12) for experiments one to eight (see text). Each column contains the error percentages in the mixed (open) and in the blocked condition (shaded). In each pair of columns the left column contains the uncorrected data while in the right column these data have been corrected for long/short confusions.

When we correct our data for long/short confusions, results become much better: 18.7 and 15.1% confusion errors for mixed and blocked condition, respectively. These error scores are somewhat higher than the percentages that Assman et al. report for their experiment on gated vowels, probably because the duration of our gated vowels is half the duration of theirs. The number of confusion errors in experiment 3 (segments from vowels produced in /p-t/ context) has increased as compared to experiment 2 (segments from vowels produced in isolation). This difference in percentage confusion errors proved to be statistically significant, which confirms the hypothesis that the centre part of vowels in /p-t/ context is acoustically less clearly defined than without that /p-t/ context present. The only difference between the stimuli of experiments from four through eight is the fundamental frequency of the source used for the resynthesis. Because of the fact that the error percentages in figure 2.5 are not the same for all these experiments, we can conclude that indeed there is an interaction between the source and spectral envelopes. This effect is strongest in experiment 7 where we resynthesized with a fundamental frequency of 335 Hz. This impression of the interaction becomes even stronger if we look at figures 2.6a and 2.6b, where the speakers were split up into the separate categories men, women and children. We see that the error percentages in these experiments differ considerably for these cate-

Table 2.4. Comparison of error percentages in the mixed and blocked condition for different experiments reported in the literature.

Author(s)	speakers	#V	Length	Type	Mixed(%)	Blocked (%)
Verbrugge et al. (1976)	5,5,5	9	full	pVp	17.0	9.5
Strange et al. (1976)	5,5,5	9	full	pVp	17.0	9.0
	5,5,5	9	full	V	42.6	31.2
	4,4,4	9	full	CVC	23.0	22.0
Macchi (1980)	5,5,5	11	full	V	7.8	1.5
	5,5,5	11	full	tVt	8.6	2.0
Assmann et al. (1982)	5,5,–	10	full	V	5.4	4.1
	5,5,–	10	100	V	13.8	9.5
Weenink, this study	5,5,5	12	full	V	9.6	3.8
	5,5,5	12	50	V	18.7	15.1
	5,5,5	12	50	pVt	24.2	18.1

gories. In general one could say that the error percentages are lowest when a category is resynthesized with its 'proper' fundamental frequency (in experiment 5: men; in experiment 6: women; in experiment 7: children).

We further note that the children's stimuli, according to the performances of the listeners, are not as well defined as the stimuli of the men and women. This is already clear in experiment 2 where we see that the 50-ms male and female vowel stimuli are recognized much better than the children's stimuli. Because we use the analysis of the vowel segments for further processing, this effect percolates to the resynthesis experiments. There are several explanations why the children's segments are not as clearly defined:

- the limitation of the bandwidth to 5000 Hz can have a greater degrading effect on the children's vowels. Spectral analysis indicates that the high frequency components of the children's voices seem to be stronger than the corresponding components of the female and male voices.
- in the children's vowels there are more amplitude variations than in the vowels of the men and women, probably because children have less control over their voices. These amplitude variations can, in the subsequent linear prediction analysis, be the cause of some more spectral smoothing.
- the high fundamental frequency of the children's vowels makes their spectral envelope less clearly defined. This also has a degrading effect on the linear prediction analysis because the 'effective' time interval for the analysis becomes shorter.

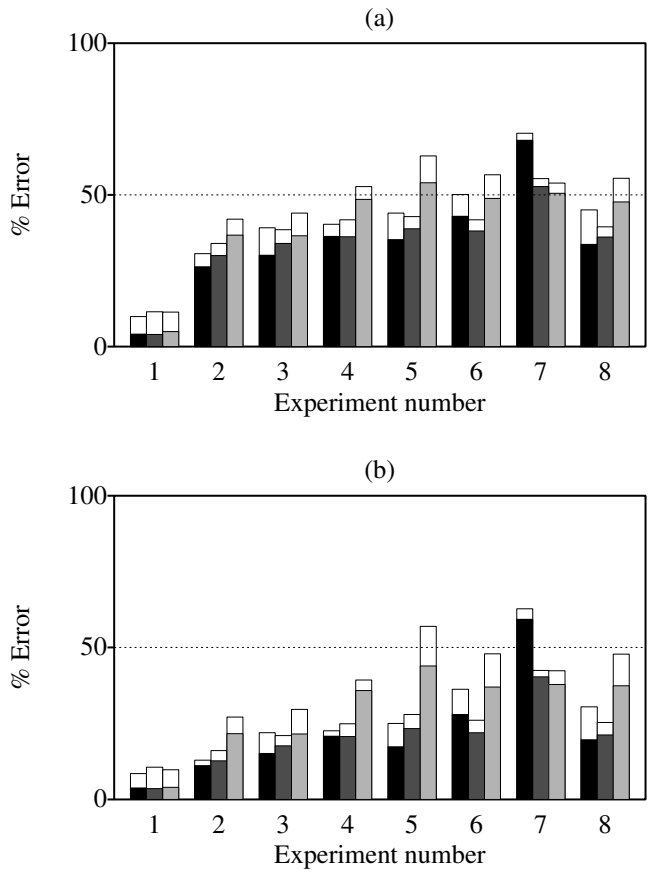

Figure 2.6. (a) Error percentages averaged over subjects (20) and vowels (12) for experiments 1 to 8, split up into the speaker categories Men (left column), Women (middle column) and Children (right column). Each column contains the error percentages in the mixed (open) and in the blocked condition (shaded). (b) Same as figure (a) with all data corrected for long/short confusions.

Further we note the especially good identification of the stimuli when resynthesized with noise: the error percentages in experiment 8, where the vowels were resynthesized with noise, are approximately the same as the percentages in experiment 4, where the stimuli were resynthesized with their original fundamental frequencies. Because of the fact that no direct fundamental frequency information is present in the stimuli from experiment 8, a major cue for speaker precategorization is not present. This means that besides pitch there must be spectral cues in the signal from which the listener can extract relevant normalization information.

In some of the next chapters physical analysis and data representation methods

are presented that try to shed some light on this matter and give us better insight in the perceptual and physical process of normalization. Finally, in chapter 11 we will present a model that simulates the classification behaviour of subjects in the mixed and blocked speaker contexts.

Chapter 3

Accurate algorithms for performing principal component analysis and discriminant analysis[‡]

Abstract

We discuss two algorithms for performing principal component analysis and discriminant analysis. These algorithms will be used extensively in subsequent chapters to perform data reduction, visualization and classification on multidimensional vowel data. Both algorithms are based on singular value decomposition (SVD). We calculate principal components and discriminants directly from the data matrix without forming the intermediate covariance matrices. In this way we do not lose numerical accuracy. The methods described here have been implemented in the speech analysis program PRAAT to obtain maximum flexibility in the analysis. Examples are presented for formant data from 50 male Dutch speakers as reported by Pols et al. (1973).

[‡]This chapter is a modified version of Weenink (1999).

3.1 Introduction

Principal component analysis (PCA) and discriminant analysis (DA) belong to the basic repertoire of multivariate data analysis. From a mathematical standpoint both methods try to construct an optimal orthogonal basis for multidimensional data. They only differ in the choice of the optimality criterion.

For *principal component analysis* the data are not labelled and one tries to find orthogonal directions in which the variance is maximal. These directions are called the principal directions and are unique up to a reflection.[1] One says that the first principal direction *explains* most of the variance. This means that when one projects the multidimensional data onto the first principal direction, the variance along this direction is a maximum, i.e. from all the possible directions in the multidimensional space no other direction shows this much variance for projected data. In mathematical terms: solve the eigensystem

$$\Sigma\mathbf{x} - \lambda\mathbf{x} = \mathbf{0} \tag{3.1}$$

for the eigenvectors $\mathbf{x}$ and eigenvalues λ. The matrix Σ is the data covariance matrix, which is a symmetrical matrix.

For *discriminant analysis* one has labelled data, i.e. each row belongs to a certain group or category, and one tries to find orthogonal directions among which discrimination between the different groups is optimal. In other words, one looks for directions in space where the ratio of the between-group variance $\mathbf{B}$ and the within-group variance $\mathbf{W}$ forms a maximum. This translates to the following eigensystem:

$$\mathbf{B}\mathbf{x} - \lambda\mathbf{W}\mathbf{x} = \mathbf{0} \tag{3.2}$$

In this chapter we will try to explain the origin of these equations and how we can solve them in an efficient and numerically stable way, without presenting too many mathematical details. As was explained in chapter 1, the terms Linear (LDA) and Quadratic Discriminant Analysis (QDA) refer to the classification stage: LDA calculates all distances with the same length measures, i.e. only one covariance matrix is used in all the distance calculations, while QDA uses a class-specific covariance matrix in the distance calculations. For now, we use the general term discriminant analysis for both types.

3.2 Principal component analysis

In figure 3.1 we have drawn 200 points in the xy-plane following a binormal distribution. These points were generated as follows.

- Both columns, x and y, of a data matrix $\mathbf{A}$ with 200 rows are filled with Gaussian random deviates, centered at the origin, with $\sigma_x = 1.0$ and $\sigma_y = 0.2$. The

[1] However, when the data are *isotropically* distributed in a subspace then there will be no preferred direction in that subspace.

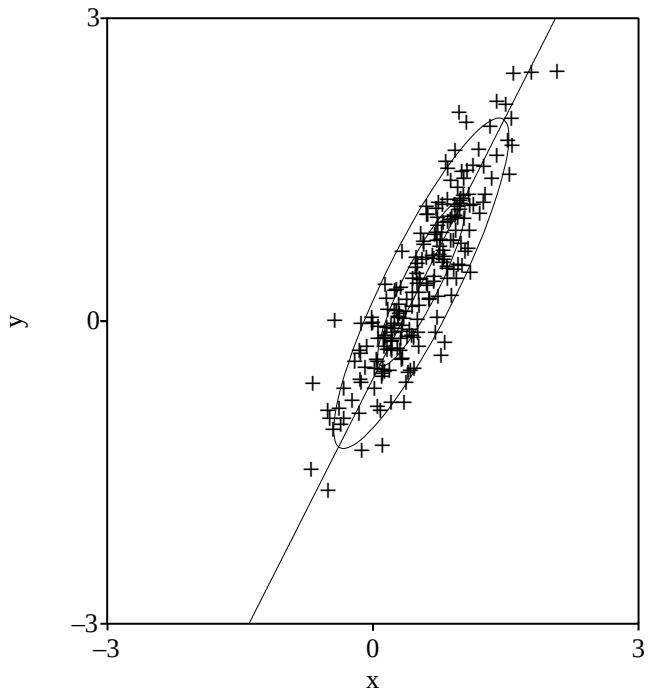

Figure 3.1. Bivariate normally distributed random data centered at (0.5, 0.3) in the xy-plane with principal direction at an angle of 60 degrees with respect to the horizontal x-axis. The ellipses are the 1σ and 2σ ellipses that include approximately 39.3% and 86.5% of the data, respectively.

variances of the x and y columns will then aproximately be 1.0 ($= \sigma_x^2$) and 0.04 ($= \sigma_y^2$), respectively. The x column contains 96% ($= 1.0/1.04$) of the total variance and the y column the remaining 4% of the total variance.

- The points are rotated counterclockwise with an angle α of 60 degrees.
- The points are translated along the vector (0.5, 0.3).

When we calculate the new variances in the x and y columns of $\mathbf{A}$ they will approximately be 0.28 ($= \sigma_x^2 \cos^2 \alpha + \sigma_y^2 \sin^2 \alpha$) and 0.76 ($= \sigma_x^2 \sin^2 \alpha + \sigma_y^2 \cos^2 \alpha$), i.e., the x column now only contains 26.9% ($= (0.28/1.04)100\%$) of the total variance and the y column contains the remaining 73.1%.

As can be seen from the figure, the direction with maximum variance no longer lies along the x axis, but now lies along a vector that makes an angle of approximately 60 degrees with the horizontal x-axis. If we created a new variable by projecting the two-dimensional points onto this direction then the new one-dimensional variable

would explain approximately 96% of the total variance, just as the original x column did.

This offers a possibility for data reduction. If we were satisfied with a description that explains 96% of the variance then we would only need one variable, the first principal component, instead of the two variabless x and y. A PCA-algorithm finds the orthogonal directions that have maximum variance.

Equation (3.1) can be derived as follows: we start with a general $m \times n$ data matrix $\mathbf{A}$, where the m rows are the n-dimensional data points. Let the first principal component $\mathbf{x}$ be the direction that maximizes the variance between the rows. The dimensionality of $\mathbf{x}$ equals n, the number of columns in $\mathbf{A}$. We project our data onto this direction and get a new vector $\mathbf{y}$ of projected data points $\mathbf{y} = \mathbf{A}\mathbf{x}$. The number of elements in $\mathbf{y}$ now equals m, the number of rows in $\mathbf{A}$. The variance (squared length) of this vector $\mathbf{y}$ should be the maximum of all lengths possible, i.e.

$$\mathbf{y}'\mathbf{y} = (\mathbf{A}\mathbf{x})'(\mathbf{A}\mathbf{x}) = \mathbf{x}'\mathbf{A}'\mathbf{A}\mathbf{x} = \mathbf{x}'\Sigma\mathbf{x} \tag{3.3}$$

should be a maximum, where $\mathbf{A}'$ means the transpose of $\mathbf{A}$. In order to obtain meaningful solutions we have to constrain the length of the vector $\mathbf{x}$ we are looking for. If no constraints were imposed on $\mathbf{x}$, any $\mathbf{x}$ of infinite length would satisfy. One normally adds the constraint that $\mathbf{x}$ be a unit vector, $\mathbf{x}'\mathbf{x} = 1$. With the help of the Lagrange multiplier λ this constraint can be included and the equation to maximize can be written as follows:

$$\mathbf{x}'\Sigma\mathbf{x} - \lambda(\mathbf{x}'\mathbf{x} - \mathbf{1}). \tag{3.4}$$

Taking the derivative with respect to $\mathbf{x}$ and setting this derivative equal to zero we end up with:

$$\Sigma\mathbf{x} - \lambda\mathbf{x} = \mathbf{0} \tag{3.5}$$

which is the desired equation (3.1). There are many ways to solve eigensystems of this type where the matrix Σ is symmetrical. We can use a method due to Jacobi or we might first reduce Σ to tridiagonal form with a Householder reduction and then solve the resulting system with QR-transformations (Golub & Van Loan, 1996; Press, Teukolsky, Vetterling & Flannery, 1996). However, when we have the original data matrix $\mathbf{A}$ at our disposal, explicitly forming the product matrix, $\mathbf{A}'\mathbf{A} = \Sigma$ is inadvisable because of a potential loss of information in finite precision arithmetic (Golub & Reinsch, 1970). We can see this as follows: when we define the condition number c of a matrix $\mathbf{A}$ as the ratio of the largest eigenvalue to the smallest eigenvalue, then the condition number of the matrix product $\mathbf{A}'\mathbf{A}$ will be c^2. When $1/c^2$ is smaller than the machine precision ϵ, the corresponding eigenvalue will be lost. This is where the singular value decomposition (SVD) enters. The SVD of an $m \times n$ matrix $\mathbf{A}$ has the form

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}', \tag{3.6}$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices of order m and n, respectively, and $\mathbf{D}$ is an $m \times n$ nonnegative diagonal matrix. (For convenience we consider $m \geq n$, i.e. more pieces of data than dimensions. We also assume that our matrices are real. See Golub & Van Loan (1996) for more details.) The diagonal elements d_i of $\mathbf{D}$ are called the

singular values of $\mathbf{A}$ and by convention are ordered so that $d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are orthonormal eigenvectors of $\mathbf{AA}'$ and $\mathbf{A}'\mathbf{A}(= \Sigma)$, respectively. We now use the SVD of $\mathbf{A}$ to calculate Σ:

$$\Sigma = \mathbf{A}'\mathbf{A} = (\mathbf{UDV}')'(\mathbf{UDV}') = \mathbf{VDU}'\mathbf{UDV}' = \mathbf{VD}^2\mathbf{V}'. \tag{3.7}$$

This is a familiar result; it shows that any real symmetric matrix Σ can brought into diagonal form by a rotation. Now $\mathbf{D}^2$ is a matrix whose diagonal values are d_i^2, the squares of the singular values of $\mathbf{A}$. When we substitute the result from equation (3.7) into equation (3.1) we obtain

$$\mathbf{VD}^2\mathbf{V}'\mathbf{x} - \lambda\mathbf{x} = \mathbf{0}, \tag{3.8}$$

now multiplying with $\mathbf{V}'$ and using orthonormality of $\mathbf{V}$ results in

$$\mathbf{D}^2\mathbf{V}'\mathbf{x} - \lambda\mathbf{V}'\mathbf{x} = \mathbf{0}. \tag{3.9}$$

The solution of equation (3.9) is now obvious. The vectors $\mathbf{x}$ that satisfy equation (3.1) are the column vectors of $\mathbf{V}$. The corresponding eigenvalues are the squares of the singular values in the diagonal matrix $\mathbf{D}$. This result shows that by taking the SVD of $\mathbf{A}$ we can easily obtain the eigenvalues and eigenvectors of the matrix $\Sigma = \mathbf{A}'\mathbf{A}$. Algorithms for determining the SVD of a rectangular matrix can be found in Press et al. (1996) and Golub & Van Loan (1996). We do not lose any numerical precision by taking the SVD of $\mathbf{A}$. Every rectangular matrix can be decomposed according to equation (3.6). The advantage of the SVD-algorithm is its numerical stability. An additional advantage of the SVD algorithms is that they are robust against $\mathbf{A}$ not being of maximum rank, or, which is the same, robust against Σ being singular. The singular values d_i show how well-behaved the matrix is. When the matrix is not of full rank the quotient of the largest and the smallest singular values will be very large.

The complete procedure for performing a PCA analysis, starting from the $m \times n$ data matrix $\mathbf{A}$ is now the following:

- Centralize the data in $\mathbf{A}$. For each element a_{ij} in $\mathbf{A}$ subtract the column mean $a_{\cdot j} = \sum_{i=1}^{m} a_{ij}/m$. This gives a new *centralized* data matrix $\mathbf{C}$ with elements $c_{ij} = a_{ij} - a_{\cdot j}$.

- Calculate the singular value decomposition $\mathbf{C} = \mathbf{UDV}'$. Sort the singular values d_i and corresponding columns of $\mathbf{V}$. Now d_1 will be the largest singular value and d_n the smallest. Save only those singular values d_i that satisfy $d_i/d_1 > \max(m, n)\epsilon$, where $\epsilon \approx 2.2\times10^{-16}$ is the machine precision for double precision floating point arithmetic.

- Store the eigenvalues, d_j^2, with their corresponding eigenvector, $\mathbf{e}_j$. The most important direction will be the first eigenvector $\mathbf{e}_1$. The projection of the data matrix on $\mathbf{e}_1$ equals $\mathbf{y} = \mathbf{Ae}_1$, and is called the first principal component.

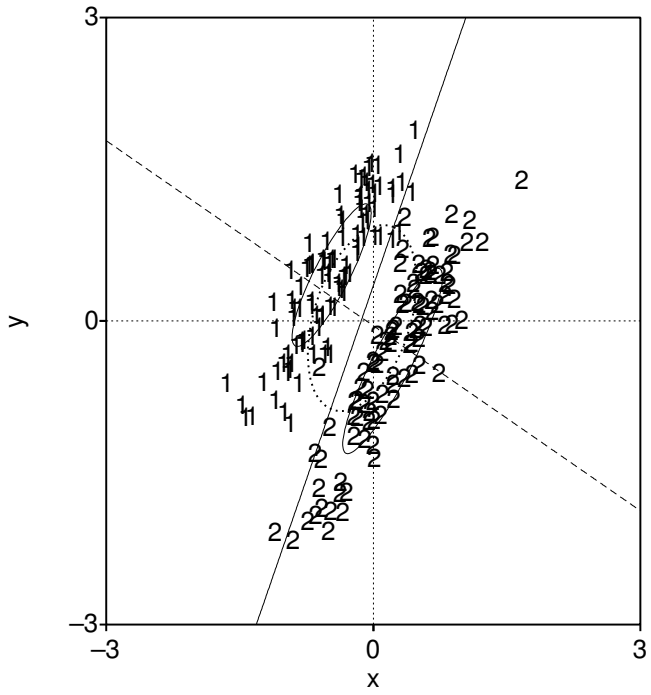

Figure 3.2. Two bivariate normally distributed random data sets in the xy-plane. Each of the sets has its principal direction at an angle of approximately 60 degrees with respect to the horizontal x-axis. The data labelled "1" are centered at $(-0.5, 0.5)$ while the data labelled "2" are centered at $(0.2, -0.5)$. Both sets have $\sigma_1 = 1$ along the first principal axis and $\sigma_2 = 0.2$ along the second principal axis. The two small ellipses are the individual 1σ ellipses of the two data sets that cover approximately 39.3% of the data. The larger ellipse is the 1σ ellipse computed from the two data sets combined. The solid line bottom-left-to-top-right that approximately parallels the long axes of the ellipses corresponds to the direction of maximum variance. The dashed line is the direction along which discrimination is maximal.

3.3 Discriminant analysis

As PCA, discriminant analysis starts with an $m \times n$ data matrix $\mathbf{A}$. Now, however, each row in the matrix is labelled as belonging to a certain group. There are g different groups, each of which has n_i elements. Clearly $\sum_{i=1}^{g} n_i = m$. When we plot each row of $\mathbf{A}$, after centralization, as a point in a space of dimension n and label the point with its corresponding label, we will see points spread around the origin. Normally points belonging to the same group lie in each others neighborhood. The purpose of

discriminant analysis is to find orthogonal directions in space such that along these directions the separation of the groups is optimal. In general, directions of maximum variance and directions of maximal separation need not have anything in common. To illustrate this we have drawn 200 points in figure 3.2 that were generated according to two different distributions. These points are labelled "1" or "2", depending on the distribution they belong to. Both distributions consist of 100 points that were generated as bivariate normally distributed around the origin with $\sigma_x = 1$ and $\sigma_y = 0.2$ and subsequently rotated 60 degrees counterclockwise, in analogy with the data from figure 3.1. Next the points labelled "1" were translated along the vector $(-0.5, 0.5)$ and the points labelled "2" were translated along the vector $(0.2, -0.5)$. The 1σ ellipses for the points labelled "1" and "2" have also been drawn as well as the 1σ ellipse for the combined data set. The direction of maximum variance is the solid line that goes from the lower left to the upper right, along the long axis of the ellipse of the combined data. It is obvious that in this figure the direction in which the best separation of the points labelled "1" and "2" is achieved, is along the dashed line, which is almost perpendicular to the previous line.

The quantification of *best separation* is the following. Suppose that the n-dimensional vector $\mathbf{x}$ is the direction that gives maximum separation. When we project the data onto this vector $\mathbf{x}$ all points now lie on a line. The points that belong to the same group hopefully cluster and lie close to each other. Good separation between the groups has been achieved when the clusters lie as far apart from each other as possible and the spread within each cluster is minimal. This amounts to saying that we want the variance of the group means along this direction to be as large as possible and at the same time the variances within the groups to be as small as possible. It is mathematically more tractable if we express this as: find the maximum value for the F-ratio, i.e. the ratio of the variance of the group means and the variance within the groups.

In the language of matrices: we first calculate from $\mathbf{A}$ the g group means and subsequently centralize $\mathbf{A}$. Call $\mathbf{C}$ the $m \times n$ matrix that results when we subtract from each row vector in $\mathbf{A}$ its corresponding group mean. When we plot these points all clusters will be centered at the origin and separate cluster would be hardly noticeable, unless the variances along the axes are very different.(For the data in figure 3.2 centralization would mean that all data then lie in one elliptic region.) Now project these data onto any vector $\mathbf{x}$ and form the m-dimensional vector $\mathbf{y}$ as $\mathbf{y} = \mathbf{C}\mathbf{x}$. The variance of $\mathbf{y}$ is a measure for the spread:

$$\mathbf{y}'\mathbf{y} = (\mathbf{C}\mathbf{x})'\mathbf{C}\mathbf{x} = \mathbf{x}'\mathbf{C}'\mathbf{C}\mathbf{x} = \mathbf{x}'\mathbf{W}\mathbf{x}. \tag{3.10}$$

$\mathbf{W}$ is the matrix with the so called within-group sums of squares and cross products (SSCP). We now form the $g \times n$ matrix $\mathbf{M}$ with group means. The projection of the group means on $\mathbf{x}$ leads to a new vector $\mathbf{z}$ defined as $\mathbf{z} = \mathbf{M}\mathbf{x}$. The variance of $\mathbf{z}$ is a measure for how far the groups lie from each other:

$$\mathbf{z}'\mathbf{z} = (\mathbf{M}\mathbf{x})'\mathbf{M}\mathbf{x} = \mathbf{x}'\mathbf{M}'\mathbf{M}\mathbf{x} = \mathbf{x}'\mathbf{B}\mathbf{x}. \tag{3.11}$$

$\mathbf{B}$ is the matrix with the between-group sums of squares and cross products. Now we

have to find the vector $\mathbf{x}$ that maximizes the ratio $\phi(\mathbf{x})$ defined as[2]

$$\phi(\mathbf{x}) = \frac{x'Bx}{x'Wx} \tag{3.12}$$

However, to obtain meaningful solutions, we first put in the constraint $\mathbf{x}'\mathbf{x} = 1$ to obtain:

$$\phi(\mathbf{x}) = \frac{x'Bx}{x'Wx} - \lambda(\mathbf{x}'\mathbf{x} - \mathbf{1}) \tag{3.13}$$

The $\mathbf{x}$ that maximizes this equation can be found by differentiation of $\phi(\mathbf{x})$ with respect to $\mathbf{x}$ and putting the result equal to zero. We then find the following *generalized eigenproblem*:

$$\mathbf{Bx} - \lambda\mathbf{Wx} = \mathbf{0} \tag{3.14}$$

To solve for the eigenvalues and eigenvectors of this equation in a numerically stable way is not a trivial matter, because either the $\mathbf{B}$ or the $\mathbf{W}$ matrix might be singular. Golub & Van Loan (1996) discuss methods for solving this type of equation. In general the eigenvectors of this equation are not orthogonal and we have to perform a subsequent orthogonalization step.

When $\mathbf{W}$ is not singular we can multiply by its inverse and obtain

$$\mathbf{W}^{-1}\mathbf{Bx} - \lambda\mathbf{x} = \mathbf{0}. \tag{3.15}$$

Although this equation has the same form as equation (3.1) there is also an important difference. In general, the product $\mathbf{W}^{-1}\mathbf{B}$ of two symmetric matrices, does not result in a symmetric matrix and, therefore, the methods used before to solve equation (3.1) for the eigenvalues and eigenvectors are not valid here (apart from the fact that it would be numerically very unwise to explicitly form $\mathbf{W}^{-1}\mathbf{B}$).

In equation (3.14) both matrices $\mathbf{B}$ and $\mathbf{W}$ are symmetric and it can be written as

$$\mathbf{M}'\mathbf{Mx} - \lambda\mathbf{C}'\mathbf{Cx} = \mathbf{0}. \tag{3.16}$$

An elegant solution is possible without forming explicitly the matrix products $\mathbf{M}'\mathbf{M}$ and $\mathbf{C}'\mathbf{C}$ that could ruin numerical precision. This solution is the generalization of the method we used with PCA analysis and is called the *generalized singular value decomposition* (GSVD).

3.4 The generalized singular value decomposition

The GSVD decomposes two matrices at the same time into a common row basis. In the following we borrow the notation from Bai & Demmel (1993). The GSVD of an $m \times n$ matrix $\mathbf{A}$ and a $p \times n$ matrix $\mathbf{B}$ is given by

$$\mathbf{A} = \mathbf{U}\Sigma_1\mathbf{R}\mathbf{Q}' \quad \text{and} \quad \mathbf{B} = \mathbf{V}\Sigma_2\mathbf{R}\mathbf{Q}', \tag{3.17}$$

[2] Many problems in multivariate data analysis can be formulated in the way of equation (3.12): find $\mathbf{x}$ where the ratio $\phi(\mathbf{x}) = \mathbf{x}'\mathbf{E}\mathbf{x}/\mathbf{x}'\mathbf{F}\mathbf{x}$ is optimal. The square matrices $\mathbf{E}$ and $\mathbf{F}$ are specific summaries of the data.

where $\mathbf{R}$ is an $n \times n$ upper triangular and nonsingular matrix, the $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{Q}$ are orthonormal matrices, and

$$\Sigma_1 = \begin{array}{c} \\ l \\ k \\ m-l-k \end{array} \begin{array}{c} \begin{array}{ccc} l & k & n-l-k \end{array} \\ \begin{bmatrix} \mathbf{I}_1 & & \\ & \mathbf{D}_1 & \\ & & \mathbf{O}_1 \end{bmatrix} \end{array}, \tag{3.18}$$

$$\Sigma_2 = \begin{array}{c} \\ p-n+l \\ k \\ n-l-k \end{array} \begin{array}{c} \begin{array}{ccc} l & k & n-l-k \end{array} \\ \begin{bmatrix} \mathbf{O}_2 & & \\ & \mathbf{D}_2 & \\ & & \mathbf{I}_2 \end{bmatrix} \end{array}. \tag{3.19}$$

The $l \times l$ matrix $\mathbf{I}_1$ and the $(n-l-k) \times (n-l-k)$ matrix $\mathbf{I}_2$ are identity matrices. The $(m-l-k) \times (n-l-k)$ matrix $\mathbf{O}_1$ and the $(p-n+l) \times l$ matrix $\mathbf{O}_2$ are zero matrices. $\mathbf{D}_1 = \text{diag}(\alpha_{l+1}, \cdots, \alpha_{l+k})$ and $\mathbf{D}_2 = \text{diag}(\beta_{l+1}, \cdots, \beta_{l+k})$ are diagonal matrices, where

$$1 > \alpha_{l+1} \geq \cdots \geq \alpha_{l+k} > 0, \quad 0 < \beta_{l+1} \leq \cdots \leq \beta_{l+k} < 1, \quad \alpha_i^2 + \beta_i^2 = 1. \tag{3.20}$$

The GSVD is a generalization of the SVD in the sense that if $\mathbf{B}$ is the identity matrix then the GSVD reduces to the SVD of $\mathbf{A}$. The pairs (α_i, β_i) are called the singular value pairs. The quotient $\lambda_i = \alpha_i/\beta_i$ is called a generalized singular value. If $\beta_i = 0$ then the generalized singular value α_i/β_i is infinite. If $\mathbf{B}$ is square and nonsingular, then the GSVD of $\mathbf{A}$ and $\mathbf{B}$ reduces to the SVD of $\mathbf{AB}^{-1}$:

$$\mathbf{AB}^{-1} = (\mathbf{U}\Sigma_1\mathbf{RQ}')(\mathbf{V}\Sigma_2\mathbf{RQ}')^{-1} = \mathbf{U}(\Sigma_1\Sigma_2^{-1})\mathbf{V}'. \tag{3.21}$$

Now we will show that the eigenvalues and eigenvectors of $\mathbf{A}'\mathbf{A} - \lambda\mathbf{B}'\mathbf{B}$ can be expressed in terms of the GSVD. We define $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{QR}^{-1}. \tag{3.22}$$

Then using equation (3.17) we find that

$$\mathbf{X}'\mathbf{A}'\mathbf{AX} = \Sigma_1'\Sigma_1 \quad \text{and} \quad \mathbf{X}'\mathbf{B}'\mathbf{BX} = \Sigma_2'\Sigma_2. \tag{3.23}$$

Therefore the columns of $\mathbf{X}$ are the eigenvectors of $\mathbf{A}'\mathbf{A} - \lambda\mathbf{B}'\mathbf{B}$, and the nontrivial eigenvalues are the squares of the generalized singular values. The matrix $\mathbf{X}$ is not an orthogonal matrix anymore. The columns of $\mathbf{Q}$ form the orthonormal set of eigenvectors.

The GSVD of two matrices $\mathbf{A}$ and $\mathbf{B}$ can be calculated with the help of the LAPACK routines `dtgsja` and `dggsvp` (Anderson et al., 1999). Routine `dtgsja` calculates the GSVD from two upper-triangle matrices $\mathbf{A}$ and $\mathbf{B}$ into the form of equation (3.17). The program `dggsvp` preprocesses two matrices $\mathbf{A}$ and $\mathbf{B}$ and brings them in upper-triangle form.

The complete procedure for performing a linear discriminant analysis, starting from the $m \times n$ data matrix $\mathbf{A}$ with g different groups is now the following:

- Centralize the data in **A** per group. This results in a new centralized matrix **C** with elements: $c_{ij} = a_{ij} - a^k_{\cdot j}$, where $a^k_{\cdot j}$ is the average value of the elements in column j that belong to group k.
- Collect the averages $a^k_{\cdot j}$ in a $k \times n$ matrix **M**.
- Calculate the generalized singular value decomposition of matrices **C** and **M**.
  - Use `dggsvp` to bring **C** and **M** into upper-triangular form.[3]
  - Use the output of `dggsvp` as input for `dtgsja` to calculate $\mathbf{C} = \mathbf{U}\Sigma_1\mathbf{R}\mathbf{Q}'$ and $\mathbf{M} = \mathbf{V}\Sigma_2\mathbf{R}\mathbf{Q}'$.
- Calculate the eigenvalues λ_i^2 from the α_i's and β_i's of equation (3.20).

This procedure results in the class centroids as well as the decomposition of the n-dimensional space into directions that are optimal with respect to classification.

3.5 Discriminant analysis in the PRAAT program

3.5.1 Introduction

In this section we will demonstrate how one performs a discriminant analysis in the PRAAT program by Boersma & Weenink (2006). This section is also available as a tutorial on discriminant analysis within the PRAAT program itself (under "Help/Tutorials").

We will use the multivariate data set from Pols et al. (1973) with the first three formant frequency values in Hertz and levels in dB of the 12 Dutch monophthong vowels as spoken in /h_t/ context by 50 male speakers. This data set has been incorporated into the PRAAT program and can be called into play with the `Create TableOfReal (Pols 1973)...` command that can be found in the "New / Tables" menu. In the list of objects a new `TableOfReal` object will appear, named `pols_50males`. After pressing the `Info` button, the "Info window" will show that this table has 6 columns and 600 rows (50 speakers × 12 vowels). The first three columns contain the formant frequencies in Hz. The last three columns contain the levels of the first three formants given in decibels *below* the overall sound pressure level of the measured vowel segment, i.e. these numbers are all positive and the larger a number is the lower the corresponding formant level was. Each row is labelled with a vowel label. Pols et al. use logarithms of

[3] In the PRAAT program we have used CLAPACK, the version of LAPACK for the C programming languangue. This version was automatically translated from the original FORTRAN sources into the C language and is supplied by the LAPACK team. Using these C-versions requires some precautions because the CLAPACK routines internally still use the FORTRAN column-wise storage of matrices while the C language uses row-wise storage. Other small nuisances are the differences in calling conventions, where C uses call-by-value while FORTRAN uses call-by-reference, and the indexing of arrays which starts at 1 in FORTRAN and at 0 in C.

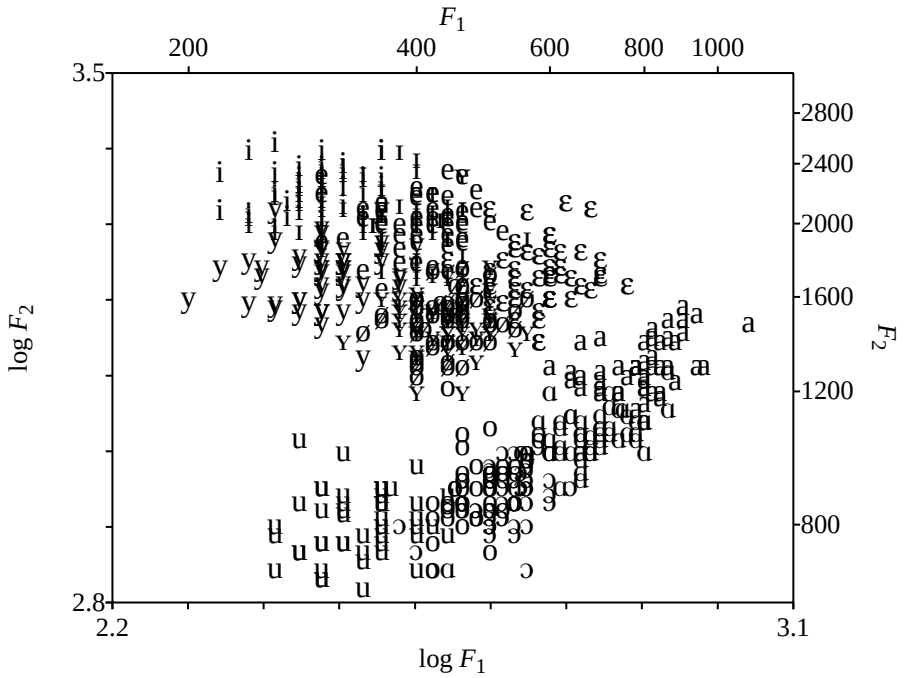

Figure 3.3. First formant frequency versus second formant frequency of the Pols et al. data set on a logarithmic scale.

formant frequency values; we will do the same.[4] Script 3.1 summarizes our achievements up till now.[5] To get an indication of what these data look like, we make a scatter

```
Create TableOfReal (Pols 1973)...  yes
Formula...  if col < 4 then log10 (self) else self endif
```

Script 3.1. Logarithmic scaling of the formant frequency values.

plot of the first log-formant-frequency against the second log-formant-frequency.

With the next script fragment 3.2 on the next page, one can reproduce part of figure 3.3. This plot equals the data points in figure 3 in the Pols et al. study.

[4] The measurement units in the first three columns and in the last three columns differ. For a discriminant analysis it is not necessary to standardize the columns, for a PCA one would normally standardize all columns first.

[5] In this script and the following ones, the essential PRAAT commands are `displayed in another type family`.

```
Select outer viewport...  0 5 0 5                    ▷ Drawing area
select TableOfReal pols_50males
Draw scatter plot...  1 2 0 0 2.2 3.1 2.8 3.5 12 + yes
One logarithmic mark top...  200 y y n
#...other logarithmic marks at top
One logarithmic mark right...  800 y y n
#...other logarithmic marks at right>
```

Script 3.2. Draw formant scatter plot.

```
select TableOfReal pols_50males
To Discriminant
```

Script 3.3. Discriminant analysis.

3.5.2 How to perform a discriminant analysis

To perform a discriminant analysis we select the `TableOfReal` object from the list of objects and choose from the dynamic menu the option `To Discriminant`. This command is available in the "Multivariate statistics –" submenu in the dynamic menu. The resulting `Discriminant` object will bear the same name as the `TableOfReal` object. Script 3.3 summarizes. Text that starts with a ▷ symbol is a comment and not part of the script language. Note that these scripts only summarize the most important parts of the analyses.[6]

3.5.3 Measuring the correlation between the variables

To measure the correlation between the variables, we select the `TableOfReal` object and choose `To Correlation`.[7] When we now choose `Draw as numbers...`, the numbers in the `Picture` window reproduce the numbers shown in the lower-left part of table III in the Pols et al. study; they are reproduced here as table 3.1. To calculate the numbers in the upper-right part of their table III, we first have to get the vowel centroids. Because the row labels of the `TableOfReal` object are vowel labels, we select this object and choose the action `To TableOfReal (means by row labels)...  no`.[8] This results in a `TableOfReal` object in which each row represents the group centroid indicated by the row label. We then generate the correlation matrix based on these centroids in the same way as was depicted above. The following script 3.4 summarizes the procedure to obtain table 3.1. The correlation matrix shows that high correlations exist between

[6] Complete scripts that reproduce all analyses, drawings and tables in this thesis, can be obtained from the author's website http://www.fon.hum.uva.nl/david/.

[7] Correlations between the columns of a `TableOfReal` object are calculated. Internally this happens as a two-step process: in the first step an `SSCP` object is produced that contains the sums of squares and cross-products of the column data in the `TableOfReal` object. In the second step we scale the `SSCP` object to obtain the correlation matrix.

[8] If only the overall centroid were needed, i.e. the average value in each column, we could perform the following steps: make an `SSCP` object from the `TableOfReal` object followed by the command `Extract group centroid`.

```
select TableOfReal pols_50males
To Correlation
Draw as numbers if...  1 0 free 3 col < row          ▷ Draw lower-left part.
select TableOfReal pols_50males
To TableOfReal (means by row labels)...  no
To Correlation
Draw as numbers if...  1 0 free 3 row < col          ▷ Draw upper-right part.
```

Script 3.4. Correlations.

Table 3.1. Correlation matrix of the six formant variables from 50 male speakers. The part above the diagonal gives the correlation coefficients of the vowel centroids, the part below the diagonal the correlation coefficients of all the data. The correlations in the table have been measured by the PRAAT program and this table exactly reproduces table III from the Pols et al. (1973) study. For better visual separability the diagonal values, which are all 1, have been left out.

	$\log F_1$	$\log F_2$	$\log F_3$	L_1	L_2	L_3
$\log F_1$		−0.359	0.275	0.840	−0.806	0.032
$\log F_2$	−0.302		0.063	−0.278	0.796	−0.927
$\log F_3$	0.195	0.120		0.392	−0.241	−0.161
L_1	0.370	−0.900	0.116		−0.692	0.057
L_2	−0.533	0.512	−0.044	−0.042		−0.547
L_3	−0.021	−0.605	0.017	0.085	0.127	

some formant frequencies and some levels. According to the source-filter model of speech production vowel spectra have approximately a declination of −6 dB/octave which indicates that a strong negative linear correlation between the logarithm of the formant frequency and the formant level in decibel should exist. This is most clearly reflected by the correlations for the vowel centroids in the upper triangular part of the table. The correlation between the $\log F_1$ and L_1 is 0.840 which means that when $\log F_1$ increases L_1 increases too. Because a higher value for L_1 means a lower level, the positive value for the correlation in the table actually shows that the formant level decreases if $\log F_1$ increases. The same reasoning can be used for 0.796 correlation value between $\log F_2$ and L_2. The −0.806 correlation between $\log F_1$ and L_2 shows that if the first formant frequency increases, the level of the second formant increases too.

Table 3.2 gives similar results for formant data of 25 Dutch female speakers in the

[9] In the data file that Van Nierop et al. used to calculate their table IV, a typing error was present: they reported $F_3 = 300$ Hz for the /ɑ/ from female speaker 2. We have noticed this error and have changed the incorrect value to the correct value $F_3 = 3000$ Hz. Because of this single wrong value, the correlations of the other variables with $\log F_3$ all turned out to be wrong in the table they published. In the present table 3.2, the correct values for all the correlations can be found.

Table 3.2. Correlation matrix of the six formant variables from 25 female speakers. The part above the diagonal gives the correlation coefficients of the vowel centroids, the part below the diagonal the correlation coefficients of all the data. The correlations in the table have been measured by the PRAAT program and reproduce table IV from the Van Nierop et al. (1973) study except for the correlations with log F_3.[9]

	log F_1	log F_2	log F_3	L_1	L_2	L_3
log F_1		−0.230	0.103	0.973	−0.670	−0.286
log F_2	−0.196		0.208	−0.304	0.841	−0.828
log F_3	0.055	0.158		0.119	0.038	−0.082
L_1	0.593	−0.207	0.091		−0.741	−0.197
L_2	−0.511	0.656	0.076	−0.386		−0.449
L_3	−0.227	−0.576	0.116	−0.061	0.019	

study of Van Nierop et al. (1973).

In section 7.3.1 we elaborate more on correlations, as we will discuss not only correlations between columns in the data matrix, as were presented here, but also correlations between linear combinations of columns.

3.5.4 Projecting data on the discriminant space

To project the data on the discriminant space we select from the list of objects the `TableOfReal` object and the `Discriminant` object together and choose: `To Configuration...`. The axes in the `Configuration` object are the eigenvectors from the `Discriminant` object. Figure 3.4 on the next page shows the data when projected onto the plane spanned by the first two dimensions of the `Configuration` object. The plot on the left looks very similar to the log(F_1) vs. log(F_2) plot of figure 3.3 on page 33. The eigenvectors, on the right hand side, show that indeed the log(F_1) and the log(F_2) variables have the largest weight because the first eigenvector is dominated by log(F_2) and the second eigenvector is dominated by log(F_1). Script 3.5 summarizes the procedure.

```
select TableOfReal pols_50males
plus Discriminant pols_50males
To Configuration...  0
Viewport...  0 5 0 5
Draw...  1 2 -1.1 -0.3 -3.6 -2.8 yes
```

Script 3.5. Draw LDA space.

In case one is only interested in the projection, there also is a shortcut in the "Multivariate statistics –" submenu that deletes the intermediate `Discriminant` object:

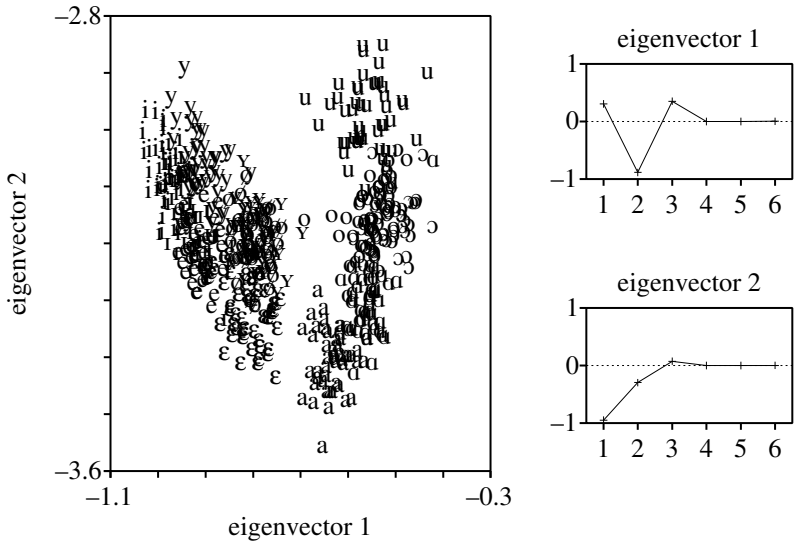

Figure 3.4. Projection of the six-dimensional Pols et al. data set on the plane spanned by the first two eigenvectors from the discriminant analysis.

```
select TableOfReal pols_50males
To Configuration (lda)...  2
```

Script 3.6. Draw LDA space (shortcut).

3.5.5 Drawing concentration ellipses

In order to draw concentration ellipses for the different groups, we select from the list of objects the `Discriminant` object and choose `Draw sigma ellipses...`. In the form one can fill out the desired coverage of the ellipse by way of the parameter *numberOfSigmas*. One can also select the projection plane. Figure 3.5 shows the 1σ concentration ellipses in the standardized log F_1 versus log F_2 plane. If the data are multinormally distributed and projected onto a plane, an ellipse whose axes have a length of $2 \cdot \mathit{numberOfSigmas}$ covers approximately $(1 - e^{-\mathit{numberOfSigmas}^2/2}) \cdot 100\%$ of the data, being 39.4% for $\sigma = 1$. It should of course be no surprise that the corresponding long/short vowel pairs /e, ɪ/, /ø, ʏ/ and /o, ɔ/ strongly overlap, since duration is not represented here. Apparently the formants of /a/ and /ɑ/ are rather different. Script 3.7 summarizes.

```
select Discriminant pols_50males
Draw sigma ellipses...  1.0 no 1 2 2.3 3.0 2.8 3.5 yes
```

Script 3.7. Draw sigma ellipses

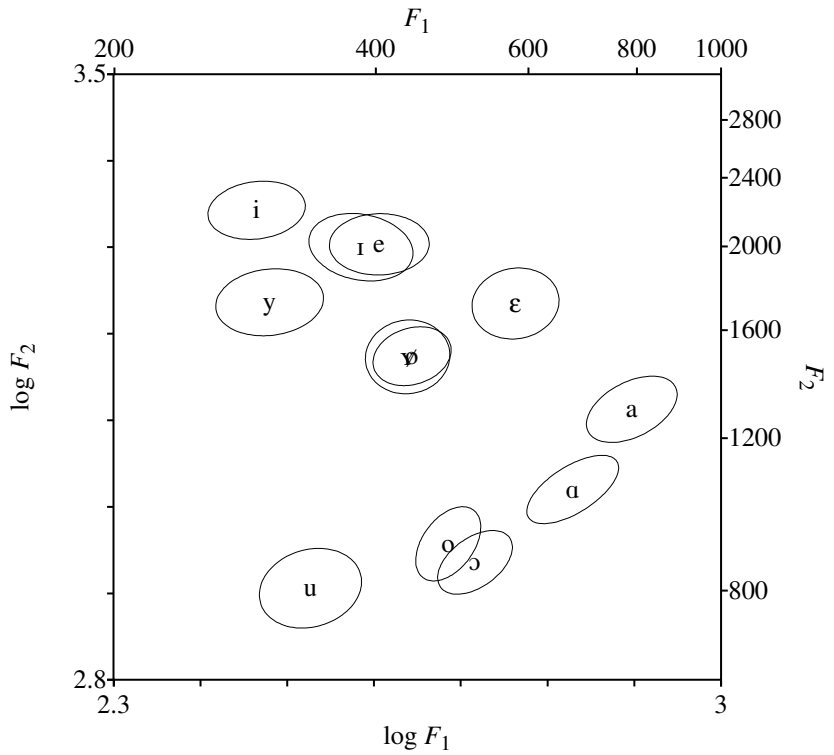

Figure 3.5. Concentration ellipses of the Pols et al. data set in the plane spanned by the first and second formant frequency on a logarithmic scale.

3.5.6 Classifying the data

Script 3.8 summarizes how to classify these and other data.

```
select Discriminant pols_50males
plus TableOfReal pols_50males
To ClassificationTable... no yes
To Confusion
Get percentage correct
```

Script 3.8. Classification and confusion matrix.

First we select both the classifier, the `Discriminant` object, and the data to be classified, the `TableOfReal` object, as is shown by the first two lines in the script. In this example, the test data set equals the training data set.[10] Next we make a `Classification-`

[10]The data set to test the classifier may be any data set whose data format conforms to the data set that

`Table` object that will contain the posterior probabilities of group membership p_j. The p_j are defined as:

$$p_j = p(j|\mathbf{x}) = \frac{\exp\left(-d_j^2(\mathbf{x})/2\right)}{\sum_{k=1}^{g} \exp(-d_k^2(\mathbf{x})/2)}, \tag{3.24}$$

where $d_i^2(\mathbf{x})$ is the generalized squared "distance" function

$$d_i^2(\mathbf{x}) = (\mathbf{x}-\mu_i)'\Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln|\Sigma_i| - 2\ln(apriori_i). \tag{3.25}$$

The first term in this "distance" function is the Mahalanobis distance; it is based on the individual covariance matrix Σ_i. The following two terms are the logarithm of the determinant of Σ_i, which may be negative, and the a priori probability $apriori_i$ of $\mathbf{x}$ belonging to group i, respectively. For each input vector $\mathbf{x}$ (each row in $\mathbf{A}$) we can calculate the p_j. In equation (3.25) as its stands, we use a group-specific covariance matrix in the distance calculation. This analysis is often called Quadratic Discriminant Analysis because the boundaries between the groups, when projected on the two-dimensional space, are quadratic curve segments. A data point on the boundary between two groups has equal probabilities of belonging to both. In QDA "distance" is not symmetric between different groups; it might even turn out to be negative. A simpler analysis, called Linear Discriminant Analysis, results if we use the same covariance matrix for all groups in the distance calculation. The distance function in this case is

$$d_i^2(\mathbf{x}) = (\mathbf{x}-\mu_i)'\Sigma^{-1}(\mathbf{x}-\mu_i) + \ln|\Sigma| - 2\ln(apriori_i), \tag{3.26}$$

where Σ is the pooled covariance matrix that can be obtained as

$$\Sigma = \frac{\sum_{i=1}^{g} n_i S_i}{\sum_{i=1}^{g} n_i}, \tag{3.27}$$

where n_i is the number of elements in group i. LDA has the advantage above QDA that we need less data for training because we only have to learn one covariance matrix instead of g matrices.

Performing a QDA classification yields a percentage correct of 79.2%. This is 0.3% less than the 79.5% that can be found in table IV of the Pols et al. (1973) study. This minor difference, which accounts for 2 extra items being misclassified in our study, may be the result of variations in the implementation of the discriminant classifiers and differences in numerical precision.

3.6 Conclusion

We have described recent algorithms for performing principal components analysis and discriminant analysis. Both algorithms are based on singular value decomposition

was used to train the discriminant classifier, i.e. it must contain the same variables in the same columns.

of the data matrix. There is no need for covariance matrices to be formed first. It was shown that leaving out this operation has numerical advantages. Both algorithms were implemented in the computer program PRAAT and from an example it was shown how the latter algorithm can be used effectively. The algorithms are designed for data sets that fit into the memory of the computer and are relatively fast on modern computers. Say, on a year 2006 computer with 1 GB of memory approximately 320 MB can be used for the data matrix (we can not use all the memory because we first have to make a copy of the data matrix and subsequently transform the copy). We can then handle a data matrix with forty million cells with double precision numbers. This could for example be a matrix with 2 000 000 rows and 20 columns. By temporarily saving the original matrix we could even double the available storage space. Nevertheless there will always be data sets too large to handle in this way and then we have to fall back on other techniques like the one displayed in equation (3.2) where we explicitly calculate the between-group variance and the within-group variance matrices **B** and **W**.

To learn the association between the labels and the data, a discriminant classifier needs all the data at once in order to calculate its parameters. In the next chapter we will first explore a feedforward neural net classifier that does not need to process all the data at once but can learn the associations between labels and data by presenting the data successively, one item at a time. In the last chapters we will then return to the techniques described in this chapter.

Chapter 4

Aspects of neural nets[‡]

Abstract

In this chapter we investigate the relation between the topology of linear supervised feedforward neural nets and their classification capabilities. Starting with the classification capabilities of one single node (the building block of neural nets) we will describe the effect of adding nodes and layers of nodes. It will be shown that the classification capabilities of one-layer nets are restricted to a problem space that is linearly separable. Two-layer nets can combine the hyperplanes formed by the first layer to form cells in the input space. However, not all possible combinations of cells can be selected by a two-layer net. Only nets with three or more layers can make all possible combinations of cells in the input space. Criteria are given in order to optimize the topology of neural nets for specific tasks because the *training* of a neural net is a very computationally intensive task. However, the *recognition* performance of a neural net is always very fast. Some aspects of the effects of varying weights on the decision regions will also be discussed. The application of feedforward neural nets to vowel classification is presented in chapter 5.

[‡]This chapter is a modified version of Weenink (1991).

4.1 Introduction

4.1.1 Context and outline

In recent years there has been much interest in the use of neural nets in speech research. Neural nets have been used a.o. for vowel-classification tasks (McCulloch & Ainsworth, 1988; Hult, 1989; Kamm, Kane-Esrig & Burr, 1989; Morin & Nusbaum, 1990; Watrous, 1991), speaker-dependent classification of consonants (Waibel, Hanazawa, Hinton, Shikano & Lang, 1989), isolated-word recognition (Kämmerer & Küpper, 1990) and speech analysis (Elman & Zipser, 1988). In the studies cited above no explicit motivations were given concerning the topology of the neural nets used. Topology in this context refers to the architecture of the net in terms of number of layers and number of nodes in each layer. Most of the time, topologies were determined by 'trial and error'. In the present study, one of the main topics will be to give some theoretical background on the link between the topology of a neural net and its 'classification capabilities'. One of the questions we want to answer is the following: Given a certain classification problem, what are the minimum bounds on the number of layers and the number of nodes in each layer, in order to be able to perform this task? When a neural net is used for classification it has to specify to which of M classes a given input belongs. We will concentrate on the classification capabilities of supervised auto- and hetero-associative neural nets with analog inputs, i.e. the inputs are real-valued (for a definition see section 4.1.2).

The general outline of this chapter will be as follows. After a brief introduction, explaining some of the terminology used in the field, we will start by examining the classification capabilities of a single node, the building block of a neural net. Next we will investigate the capabilities of one layer of these building blocks: one-layer nets. We will show that when we add another layer and make a two-layer net, the classification possibilities increase significantly. Adding yet another layer gives us three-layer nets. It will be shown that a three-layer net can meet all 'possible' classification tasks. The last section will cope with some other aspects of neural nets such as the function of the nonlinearity. Discussion and conclusions end the chapter. The application of feedforward neural nets for vowel classification will be discussed in chapter 5.

4.1.2 Terminology

In the literature, several other names for neural nets are commonly used: connectionist models, parallel distributed processing models (PDP), artificial neural nets (ANN), multi-layer perceptrons (MLP) and Boltzmann machines. We prefer the term neural net. In a neural net, the basic building block is called a node. Other denominations are processing element, processing unit and sometimes McCulloch-Pitts unit. In a neural net many of these nonlinear nodes are connected and work in parallel. The architecture resembles the patterns that neurons make in the nervous system, hence the name neural net. The nodes are usually organized into a sequence of layers and connected to each other with connections of variable strength. These connections contain the 'knowledge' or the 'memory' of the net. In general, it is not possible to

decide which specific connection is responsible for what classification decision: the 'knowledge' is *distributed* over the net. In the operation of a neural net two phases can be distinguished: the *learning* or *training* phase and the *regeneration* or *recall* phase. During the regeneration phase the net has to show what it has learned, it processes a given input to generate an output. During the learning phase a node has the ability to modify its connection strengths, i.e. its *weights*, depending on the input signals which it receives and the associated teacher signals or error signals; this is called *supervised* learning. A neural net is called *hetero-associative* when teacher signal and input signal are different, and *auto-associative* when they are the same. The teacher or error signal is not provided in some cases where a node modifies its weights depending only on its state and input signal. This is the case of *unsupervised* learning, and such a learning scheme is sometimes called *self-organization*. For further details on these types of learning, we refer to the papers of Lippmann (1987), Amari (1990) and Grossberg (1991).

4.1.3 Topology

First some remarks concerning notation. We will write down the topology of a network, the number of layers and the number of nodes per layer, in the following way:

$$(X_{n_1}, ..., X_{n_j}, ..., X_{n_k}), \tag{4.1}$$

where X is a representation of the information at a particular layer. X is either the symbol A or B, meaning that the representation is analog or binary, respectively. n_1 is the number of inputs, n_k is the number of outputs and n_j is the number of nodes in hidden layer $j-1$ (for $j > 1$). Hidden units are units that are neither input nor output. We do not count the inputs as a separate layer, so the number of layers of a neural net is always one less than the number of elements between parentheses, i.e. $k-1$. The term 'inputs' can refer to two things: the input for the net or the input for a particular node. It will be obvious from the context which input is meant. Examples of some topologies:

(A_2, B_4) one-layer net with two analog inputs followed by a layer with four binary outputs.

(A_2, A_2, B_3) two-layer net with two analog inputs followed by a layer with two hidden analog nodes and a layer with three binary outputs.

(B_4, B_2, B_3, B_4) three-layer net with four binary inputs, followed by a layer with two binary hidden nodes, followed by a layer with three binary hidden nodes and a layer with four binary outputs.

This notation is only suitable for feedforward nets. In a feedforward net there are no feedback connections; information travels in one direction. Only during the learning phase does information flow from the output of the net to the input. In this chapter we will restrict ourselves to feedforward neural nets since these nets are the most popular nowadays and mathematically the simplest to start with.

4.2 Capabilities of one node

The basic building block of a neural net is the node. A node is an element that receives a number of inputs, weighs them and then calculates an output. This output, y, is defined by:

$$y = f\left(\sum_{i=1}^{N} w_i x_i - \theta\right) \tag{4.2}$$

Here, the x_i $(i = 1, \ldots, N)$ denote the N input values, the w_i $(i = 1, \ldots, N)$ are real numbers that weigh each input x_i, the θ is the threshold or bias term and f is an arbitrary function. In equation (4.2), the inputs are linearly weighed. The argument of the function f is a linear combination of the inputs x_i. When the argument is equated to a constant, e.g., zero, it forms the equation of a hyperplane in a N-dimensional space. This hyperplane separates the input space into two regions, one on each side of the hyperplane. Essential of f is that it can be chosen nonlinear in such a way that different values on either side of the hyperplane result. Two popular forms of this function are the sigmoid function or logistic function, and the hard limiting nonlinearity, the Heaviside function. In the sequel, unless explicitly mentioned, we limit the discussion to a nonlinearity f of the Heaviside form. This means that the value of the function f attains distinct values on both sides of the hyperplane, e.g., 1 whenever its argument is greater than zero, and 0 whenever its argument is less than or equal to zero. Sometimes, a node with the Heaviside nonlinearity is called a McCulloch-Pitts unit. We will give an example of the two-dimensional case with topology (A_2, B_1).

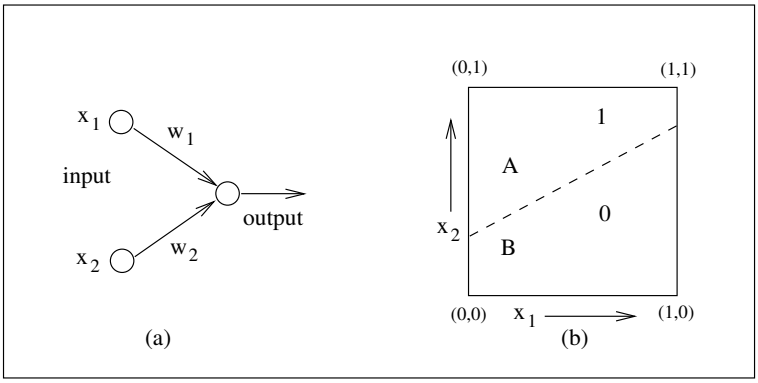

Figure 4.1. (a) Example of a net with topology (A_2, B_1). (b) The dotted line separates the input space into two regions A and B.

In figure 4.1a the full topology is drawn and in figure 4.1b, the output of the (binary) node as a function of both inputs. The equation for the hyperplane dividing the space follows from the argument of equation (4.2) and boils down to the equation of a line in the two-dimensional case:

$$w_1 x_1 + w_2 x_2 - \theta = 0. \tag{4.3}$$

This line divides the plane in two parts A and B. The exact partitioning depends on the values of the weights w_1, w_2 and the bias θ. We can see that in the example w_1 must be negative, w_2 must be positive, and θ must be positive.

4.3 Capabilities of one-layer nets

In this section we will discuss the classification capabilities of one-layer neural nets. The one-layer neural net represents the most simple neural net. Its topology is (A_N, B_p), N analog inputs and p binary outputs, each input being connected to p outputs. We consider the N-dimensional input space divided into p subspaces by p hyperplanes. Classifying in this context means that whenever the input is in subspace k, the output of node k should give a 1 and the output of the other nodes should give a 0. A single node, as we have seen in the previous paragraph, separates the input space into two regions by means of a hyperplane. Each node in the layer can classify according to one distinct hyperplane. Consequently, the classifying capabilities of a one-layer net are restricted to problems which are linearly separable. A set S, with N elements A_i, is linearly separable if for all A_i there exists a hyperplane that separates A_i from $S - A_i$.

As an example, consider a set S with three elements A, B and C. Linear separability means that A can be separated from $B \cup C$ (the union of B and C), B can be separated from $A \cup C$ and C can be separated from $A \cup B$. In figure 4.2 some examples are given of sets in a two-dimensional space that are linearly separable (a and b) and some sets that are not (c and d).

A question related to linearly separable sets is the following: Given a set S of p points uniformly distributed in N-dimensional space. What is the probability that S is linearly separable? First we note that when $p \leq N + 1$ the set is in general linearly separable. This question can be solved in a recursive manner: the probability that p points are linearly separable is equal to the probability that $p - 1$ points are linearly separable times the probability that the p-th point is situated such that it is linearly separable from the other $p - 1$ points. What is the probability that, given a set S' of $p - 1$ points that is linearly separable, the p-th point, randomly chosen from a uniform distribution, is linearly separable? We will try to make it plausible that when $p/N \to \infty$ the probability that the set S is linearly separable tends to zero.

In the one-dimensional case it is obvious that when $p > 2$, S is not linearly separable. In a two-dimensional space spanned by $[0, 1] \times [0, 1]$, three randomly chosen points in this space are linearly separable with probability 1. What is the probability that a fourth, randomly chosen, point is linearly separable? In figure 4.2e we have drawn three points A, B and C. We can see from this figure that when a fourth point E is situated in the shaded region, the set of four points is not linearly separable. When four points A, B, C and E are not linearly separable in two-dimensional space, one can always draw a triangle with corners on three of the four points such that the point not used to form the triangle lies in its interior (in the figure the triangle AEB encloses point C). The probability that the four points form a linearly separable set is equal to the ratio of the nonshaded area to the total area. It will be approximately 0.5 in figure 4.2e. The situation for $p = 5$ in two dimensions is depicted in figure 4.2f. The prob-

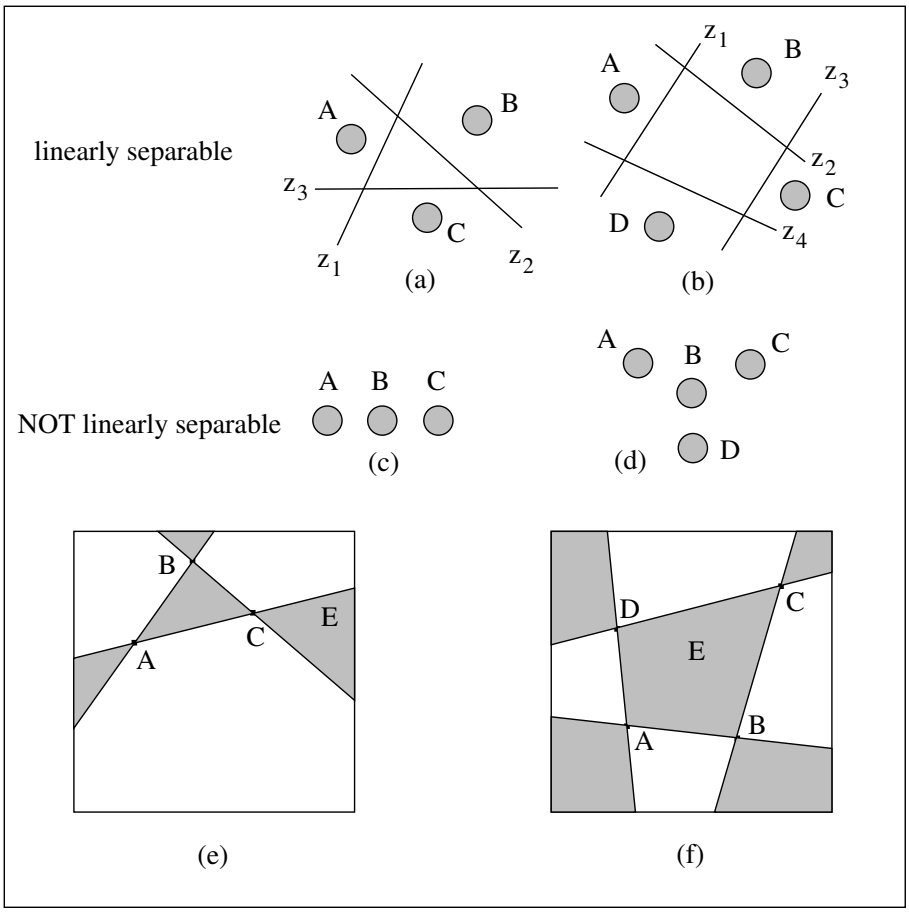

Figure 4.2. Linear separability in two dimensions. The elements in panels (a) and (b) are linearly separable with a one-layer net, the elements in panels (c) and (d) are not. In (e) and (f) the elements (A, B, C) and (A, B, C, D) are linearly separable. Adding a new element E, in the shaded area, renders the new element not linearly separable from the others.

ability for linear separability in two dimensions of a set with $p = 5$ is the product of the probabilities for the sets with $p = 3$ and $p = 4$. Every time we add a new point this probability decreases. This argument makes it plausible that the probability that a set of p randomly chosen points is linearly separable goes to zero when p/N goes to infinity.

We will show in the next section that whenever a set S is not linearly separable, at least one additional layer is needed in order to be able to classify its elements.

4.4 Capabilities of two-layer nets

4.4.1 Introduction

In the preceding sections we have seen that each node in the first layer is capable of separating the input space into two regions by means of a hyperplane. Addition of a second layer turns out to enrich the classifying potential enormously. The first layer forms hyperplanes in the input space. The second layer makes logical combinations with these hyperplanes from the first layer. A logical combination of hyperplanes boils down to forming so-called cells in the input space. A cell is the smallest region in the input space that is bounded by one or more hyperplanes. Before going into the classification capabilities of two-layer neural nets, which is a question about the possible combinations of cells in the input space that can be selected by a node in the second layer, we first want to answer another question. How many nodes in the first hidden layer do we need to be able to classify M cells in the input space?

4.4.2 Number of cells formed by a two-layer net

The question stated above can be translated into the following problem: Given an N-dimensional space S, how many hyperplanes, of dimension $N-1$, do we need to divide space S into M subspaces? Mirchandani, Cao & Bosworth (1989) and Makhoul, Schwartz & El-Jaroudi (1989) have found the answer to a related question: What is the maximum number of subspaces in which p hyperplanes can divide an N-dimensional space? Let $C(p, N)$ be the number of cells, formed by p hyperplanes in N-dimensional space. In general the following recursive relation holds (Schläfli, 1950):

$$C(p, N) \leq C(p-1, N) + C(p-1, N-1), \tag{4.4}$$

with equality if the hyperplanes are in a general position, i.e. no more than N planes intersect at the same point and no hyperplanes are parallel to each other. Starting with initial conditions:

$$C(0, N) = 1 \qquad \text{and} \qquad C(p, 0) = 1 \tag{4.5}$$

it follows that:

$$C(p, N) = \sum_{i=0}^{N} \binom{p}{i}. \tag{4.6}$$

Whenever $p \leq N$, equation (4.6) simplifies to:

$$C(p, N) = 2^p. \tag{4.7}$$

Equation (4.6) gives the theoretical upper bound on the number of cells that can be formed by p hyperplanes in N-dimensional space. The number of cells grows exponentially whenever the number of hyperplanes, p, does not exceed the dimension of the space, N. If $p > N$ the exponential growth changes into polynomial growth; however, the degree of the polynomial increases with the dimension N. Table 4.1 shows $C(p, N)$ for a number of values for p and N. For example, $C(3, 2) = \binom{3}{0} + \binom{3}{1} + \binom{3}{2} = 1 + 3 + 3 = 7$.

Table 4.1. Values of $C(p, N)$, the maximum number of cells in a N-dimensional space formed by p hyperplanes of dimension $N-1$ in general position. The upper row shows the dimension N, while the first column shows the number of hyperplanes p. The relation $C(p, N) = C(p-1, N)+C(p-1, N-1)$, is illustrated for $p = 4$ and $N = 3$ (bold font).

$p \backslash N$	1	2	3	4	5	6	7	8	9	10
1	2	2	2	2	2	2	2	2	2	2
2	3	4	4	4	4	4	4	4	4	4
3	4	**7**	**8**	8	8	8	8	8	8	8
4	5	11	**15**	16	16	16	16	16	16	16
5	6	16	26	31	32	32	32	32	32	32
6	7	22	42	57	63	64	64	64	64	64
7	8	29	64	99	120	127	128	128	128	128
8	9	37	93	163	219	247	255	256	256	256
9	10	46	130	256	382	466	502	511	512	512
10	11	56	176	386	638	848	968	1013	1023	1024
15	16	121	576	1941	4944	9949	16384	22819	27825	30827
20	21	211	1351	6196	21700	60460	137980	263950	431910	616666
50	51	1276	20876	251176	$2.4{\cdot}10^{6}$	$1.8{\cdot}10^{7}$	$1.2{\cdot}10^{8}$	$6.5{\cdot}10^{8}$	$3.2{\cdot}10^{9}$	$1.3{\cdot}10^{10}$
100	101	5051	166751	$4.1{\cdot}10^{6}$	$7.9{\cdot}10^{7}$	$1.3{\cdot}10^{9}$	$1.7{\cdot}10^{10}$	$2.0{\cdot}10^{11}$	$2.1{\cdot}10^{12}$	$1.9{\cdot}10^{13}$
1000	1001	500610	$1.7{\cdot}10^{8}$	$4.2{\cdot}10^{10}$	$8.3{\cdot}10^{12}$	$1.4{\cdot}10^{15}$	$2.0{\cdot}10^{17}$	$2.4{\cdot}10^{19}$	$2.7{\cdot}10^{21}$	$2.7{\cdot}10^{23}$

$C(p, N)$ may serve as an indication to find the minimum number of binary nodes in the first layer that is necessary to classify M sets in N-dimensional input space. We consider two cases:

1. $M \leq 2^N$
   The minimum number of nodes, p, is given by:
   $$p = \lceil {}^2\log M \rceil , \tag{4.8}$$
   where the brackets mean rounding to the nearest higher integer. For example if $M = 3$ and $N = 2$ we have $p = 2$, because $1 < {}^2\log 3 < 2$.

2. $M > 2^N$
   The problem is to find for given M a minimal p such that for a given dimension N:
   $$C(p, N) = \sum_{i=0}^{N} \binom{p}{i} \geq M. \tag{4.9}$$
   Equation (4.9) is polynomial in p of order N and hard to solve. Generally it may be impossible to find an analytical solution for $N > 5$. For small p and N a table lookup gives us the answer we need. However, if $p \gg N$ we can make the following estimation for the polynomial $C(p, N)$:
   $$C(p, N) \geq \binom{p}{N} \geq \frac{(p - N + 1)^N}{N!} \tag{4.10}$$
   which yields:
   $$p \geq \sqrt[N]{MN!} + N - 1. \tag{4.11}$$

In practice, we have to be careful what number to take for N, the dimension of the input space. When the inputs are mutually 'orthogonal', the dimension of the input space is simply equal to the number of inputs. However, in many occasions the 'effective' dimension of the input space can be substantially smaller than the number of input units. This occurs when the inputs are correlated. An indication of the actual dimension of the input space can then be given for example by a principal component analysis. A concrete example: We want to classify the twelve Dutch vowels and each vowel is specified by sixteen bandfilter values. The minimum two-layer net topology that could perform this task is (A_{16}, B_h, B_{12}), sixteen analog inputs, twelve binary outputs and h hidden binary nodes, where h depends on the real dimensionality of the vowel space. From table 4.1 we observe that, in order to make twelve cells, we need $h = 4$ when the dimension of the input space is greater than two. Should the real dimension of the problem be $N = 2$, we look up: $h = 5$. This example (and in general the table) shows that to separate the same number of classes we need more nodes if the dimensionality is low. This can also be deduced from the table, which shows that with the same number of nodes, we can create more cells if the dimensionality is high.

4.4.3 Permissible logical combinations of two-layer nets

Let us consider two-layer nets with topology (A_i, B_j, B_k): i analog inputs, j binary hidden nodes, and k binary outputs. Each of the k outputs can form logical combinations with its inputs, i.e. the output of the hidden nodes j that form hyperplanes in the input space. Potentially each output k can form 2^j distinct logical combinations with the outputs of j hidden nodes. A logical combination of hyperplanes forms cells in the input space. The question that presents itself now is: which cells, or, which combination of cells, can be selected by a particular node in the second layer, the output? Such a combination of cells will be called a decision region. Before we can answer this question we first of all have to know what logical combinations of hyperplanes are permissible with a two-layer net. In this paragraph we will prove that of all possible logical combinations only certain subsets of a restricted form are permissible with a two-layer net.

All possible logical combinations can be formed with the operators $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT) and $\oplus$ (XOR: eXclusive OR). Table 4.2 shows the outputs of these operators on two binary inputs z_1 and z_2. We can interpret this table graphically by representing z_1 and z_2 in the two-dimensional plane. In figure 4.3a we notice that one line separates the point $(1, 1)$ from the three points $(0, 0)$, $(0, 1)$ and $(1, 0)$. This line functions as the decision boundary for the Boolean AND. Likewise the line in figure 4.3b functions as the decision boundary for the Boolean OR. However, there exists no single line that could function as the decision boundary for the Boolean XOR, i.e. there is no line that could separate the points $(0, 1)$ and $(1, 0)$ from the two other points. Therefore, a two-layer neural net cannot have decision regions selected by an XOR. On the basis of this conclusion (and at that time, the lack of a suitable training algorithm) it was once suggested that neural nets were not very interesting objects to investigate (Minsky & Papert, 1969). The only logical operators that can be used in forming logical expressions, learnable with a two-layer net, therefore are: $\wedge$ (AND),

$\vee$ (OR) and $\neg$ (NOT). We will prove that the only permissible logical combinations of a node with N binary inputs are of the following two forms:

$$(z_1 \vee \ldots \vee z_p) \wedge (z_{p+1} \wedge \ldots \wedge z_{p+k}) \quad \text{for} \quad p, k \geq 0 \quad \text{and} \quad p + k \leq N \tag{4.12}$$

$$(z_1 \vee \ldots \vee z_p) \vee (z_{p+1} \wedge \ldots \wedge z_{p+k}) \quad \text{for} \quad p, k \geq 0 \quad \text{and} \quad p + k \leq N. \tag{4.13}$$

Table 4.2. Truth table of the Boolean operators $\neg$ (NOT), $\wedge$ (AND), $\vee$ (OR), en $\oplus$ (XOR)

z_1	z_2	$\neg z_1$	$z_1 \wedge z_2$	$z_1 \vee z_2$	$z_1 \oplus z_2$
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

In these logical expressions the z_i are either 0, 'FALSE' or 'off', or 1, 'TRUE' or 'on'. We implicitly assume that each z_j, is used only once in the particular expression. We will prove these formulas by constructing an explicit decision function for each one of them. A general decision function of a node, with N inputs z_i, has the form:

$$\alpha_1 z_1 + \alpha_2 z_2 + \ldots \alpha_N z_N > c, \tag{4.14}$$

where the α_i are the weights and c is the bias. If coefficients α_i and bias c can be found for expressions (4.12) and (4.13) then these forms are learnable. We first note that, without loss of generality, we can choose the decision function for these logical expressions in such a way that only two different coefficients α_i are needed:

$$1 \cdot (z_1 + \ldots + z_p) + \alpha \cdot (z_{p+1} + \ldots + z_{p+k}) > c. \tag{4.15}$$

The logical expression (4.12) imposes the following four conditions on the decision function (4.15):

1. If at least one of the z_i $(i \leq p)$ is on, and all the $z_j (j > p)$ are on, then (4.15) holds.

2. All the $z_i (i \leq p)$ may be on, but, whenever one of the $z_j (j > p)$ is off, (4.15) does not hold.

3. If all k nodes $z_j (j > p)$ are on but none of the $z_i (i \leq p)$, (4.15) does not hold.

4. If none of the $z_i (i \leq p)$ and none of the $z_j (j > p)$ are on, (4.15) does not hold.

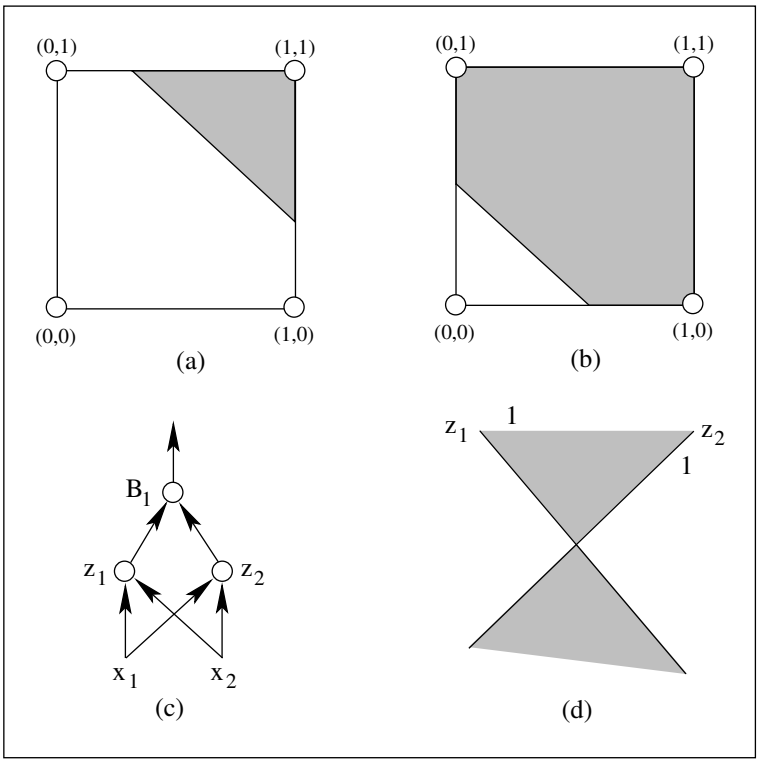

Figure 4.3. Graphical representation of boolean functions AND (a) and OR (b) of a neural net (c) with topology (A_2, B_2, B_1). The XOR function cannot be made: there is no single straight line that separates both $(0, 1)$ and $(1, 0)$ from the other two points. In (d) the XOR-decision region in the input space is shown shaded.

These conditions can be mathematically stated as:

$$\begin{aligned} 1 + \alpha k &> c \qquad k \geq 1 \\ p + \alpha(k-1) &\leq c \qquad p \geq 1 \\ \alpha k &\leq c \\ 0 &\leq c. \end{aligned} \tag{4.16}$$

From the first and the third expression in (4.16) it follows that

$$\alpha k \leq c < \alpha k + 1, \tag{4.17}$$

while all three combined give:

$$\frac{c-1}{k} < \alpha \leq \min\left(\frac{c-p}{k-1}, \frac{c}{k}\right), \tag{4.18}$$

which gives rise to the following condition:

$$\frac{c-1}{k} < \min\left(\frac{c-p}{k-1}, \frac{c}{k}\right). \tag{4.19}$$

This gives rise to two possibilities for c:

$$\begin{aligned} c &> k(p-1)+1 \\ c &> kp. \end{aligned}$$

The first equation includes the second one if $k \geq 1$. Equation 4.17 limits the range of c to an interval of size smaller than 1 and this allows us to write:

$$c = k(p-1)+1+\epsilon \quad \text{for} \quad 0 < \epsilon < 1. \tag{4.20}$$

The value for α can be taken halfway between the values on the right-hand and left-hand side of equation (4.18) and it becomes:

$$\alpha = p - 1 + \frac{\epsilon(2k-1)}{2k(k-1)} \qquad \text{for} \qquad 0 < \epsilon < 1. \tag{4.21}$$

Equations (4.20) and (4.21) show that values for the bias and the weights can be found that satisfy decision function (4.12). In an analogous manner the conditions imposed by logical expression (4.13) on the decision function (4.15) are the following:

$$\begin{aligned} 1 &> c \\ 0 &\leq c \\ \alpha k &> c \\ \alpha(k-1) &\leq c. \end{aligned}$$

From these four equations we deduce:

$$\begin{aligned} 0 &\leq c < 1 \\ \alpha &= \tfrac{c(2k-1)}{2k(k-1)}. \end{aligned} \tag{4.22}$$

These expressions for the weights α and the bias c show that decision function (4.13) can be satisfied. We now have shown that it is possible to form decision functions for the logical expressions (4.12) and (4.13). To show that these expressions are the only ones permissible we further have to show that no decision function can be constructed for any other possible expressions. Exactly two other 'basic' expressions exist that have a form analogous to (4.12) or (4.13): the *and-of-ors* (4.23) and the *or-of-ands* (4.24). These forms are given by:

$$(z_1 \vee \ldots \vee z_p) \wedge (z_{p+1} \vee \ldots \vee z_{p+k}) \quad \text{for} \quad p, k \geq 2 \quad \text{and} \quad p+k \leq N \tag{4.23}$$

$$(z_1 \wedge \ldots \wedge z_p) \vee (z_{p+1} \wedge \ldots \wedge z_{p+k}) \quad \text{for} \quad p, k \geq 2 \quad \text{and} \quad p+k \leq N. \tag{4.24}$$

In these expressions the minimum values of p and k are 2 since smaller values would reduce them to one of the forms (4.12) or (4.13). The following conditions can be imposed by the logical and-of-ors expression on the decision function (4.15):

$$\begin{aligned} 1+\alpha &> c \\ 2 &\leq c \\ 2\alpha &\leq c. \end{aligned}$$

Combining the last two expressions in (4.25) and dividing by 2 results in

$$1+\alpha \leq c. \tag{4.25}$$

This expression contradicts the first expression of (4.16). This proves that expression (4.23) is not a permissible one in our context. The following conditions on p and α lead to a contradiction for expression (4.24):

$$\begin{aligned} p &> c \\ k\alpha &> c \\ p-1 &\leq c \\ p-1+(k-1)\alpha &\leq c. \end{aligned}$$

We have now proved that expressions (4.12) and (4.13) are valid expressions and (4.23) and (4.24) are not. To complete the proof that (4.12) and (4.13) are the only valid expressions we still have to show that all expressions that have (4.23) or (4.24) as a subexpression are not permissible. This proof is trivial. We can summarize the permissible logical expressions of a node in the following way: all permissible logical expressions are composed of either a series of simple ANDs combined with possibly one series of simple ORs, or a series of simple ORs combined with possibly one series of simple ANDs.

4.4.4 Decision regions of two-layer nets

In section 4.4.2 we discussed the number of cells formed in the input space by a logical combination of the hyperplanes of the hidden layer and in the previous section the permissible logical combinations that a two-layer net can make. Now we are ready to tackle the classifying capabilities of two-layer neural nets. The decision regions of an output node in a two-layer net with topology (A_N, B_p, B_1), i.e. N analog inputs, p hidden binary nodes and 1 binary output, constitute those combinations of cells in the input space that are permissible. The number of possible decision regions is $2^{C(p,N)}-1$, where $C(p, N)$ is the number of cells in the input space. Because of the form of the permissible expressions (4.12) and (4.13), a combination of maximally p simple terms which each can have the value 0 or 1, the output node can maximally form 2^p different combinations. The number of different combinations at the output layer is in general much smaller than the number of possible decision regions in the input space.

In figure 4.3d an example is given of a decision region that could not be formed with a two-layer net, one that amounts to an XOR. We will show that, despite the fact that not all possible logical combinations are permissible, very many interesting decision regions are possible with two-layer neural nets. Lippmann (1987, page 16) states that the decision regions of two-layer neural nets are either bounded or unbounded convex regions in the input space. We will demonstrate that this statement is not correct and too conservative. Decision regions do not need to be convex: non-convex and 'hollow' regions are possible too. The definition of a convex region is that a straight line segment between any two points in the region lies totally within the region.

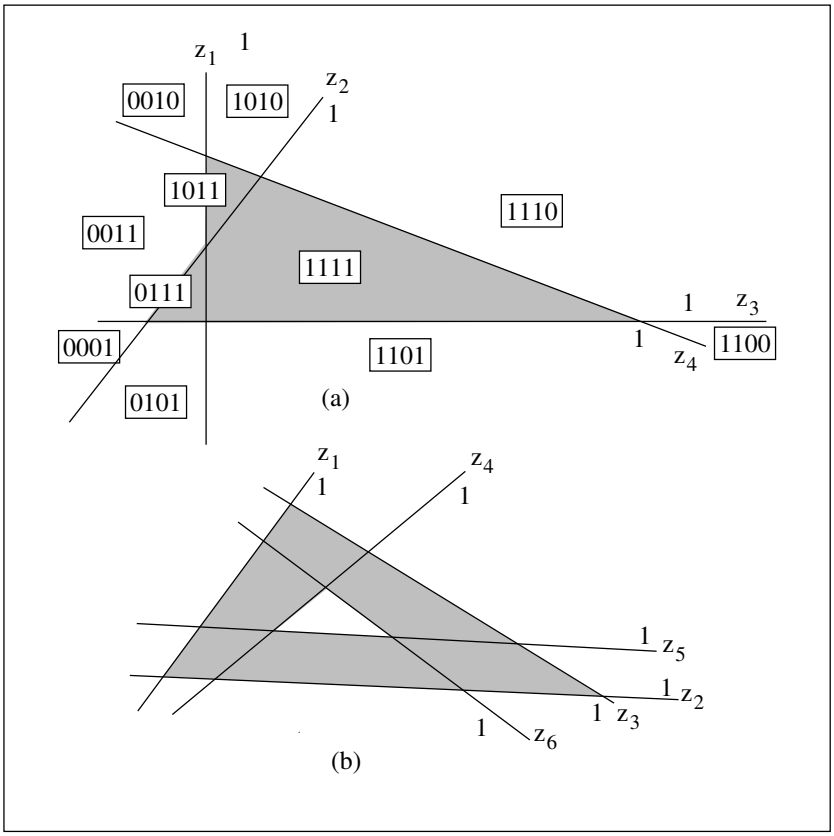

Figure 4.4. Two possible decision regions for two-layer nets. (a) Most simple concave decision region of net with topology (A_2, B_4, B_1). In figure (b) the most simple hollow decision region of a net, with the topology (A_2, B_6, B_1) is shown. That side of the line where the output of the hidden node is 1 is marked.

Figure 4.4a shows four possible decision lines, z_1, z_2, z_3 and z_4 from the four hidden nodes from a net with topology (A_2, B_4, B_1). We will show that the output node can form the non-convex decision region that is shown shaded in this figure,

by a specific combination of the four decision lines. A 1 denotes that side of the line where the hidden node generates a 1, the other side of the line gets the 0. It is not essential which side of the line is chosen. All eleven cells are labelled with the quadruple (z_1, z_2, z_3, z_4), giving the relative location with respect to the four decision lines. We have to prove that we can find a linear combination of z_1, z_2, z_3 and z_4 such that when the value of this linear combination exceeds the bias, precisely the shaded area, a combination of three cells in the input space is selected. The Boolean function characterizing the desired decision region is:

$$(z_1 \vee z_2) \wedge z_3 \wedge z_4 \tag{4.26}$$

The decision function for this logical expression can easily be found with the help of the rules we gave in the preceding paragraph. The weights and bias can be chosen according to formulas (4.20) and (4.21). Because (4.26) is a simple expression we will deduce the decision function in yet another way. Column *B1* in table 4.3 contains the value of this Boolean expression; it returns a 1 for each cell in the desired region and a 0 for all others. The table as well as the figure show that the combinations of z_1, z_2, z_3 and z_4 that select the desired decision region all contain at least three 1's. Therefore the sum function (column *Sum* in the table) $z_1 + z_2 + z_3 + z_4$ with a bias of 3 almost selects the region. A small modification of the sum function is needed to weigh z_3 and z_4 somewhat more than z_1 and z_2 to deselect two unwanted regions with a sum of 3. It appears that the following function, implemented in the output node B_1, selects the correct region:

$$z_1 + z_2 + (1 + \epsilon)(z_3 + z_4) > 3 + 1.5\epsilon \tag{4.27}$$

The exact value of ϵ is not important in this context; it can be any value between zero and one. We have now shown that the function (4.27) can select the shaded region of figure 4.4a.

A two-layer neural net is also capable of making 'hollow' decision regions as illustrated in figure 4.4b which shows the simplest 'hollow' decision region that can be constructed in two dimensions. We will prove that the output node of a neural net with topology (A_2, B_6, B_1) can already generate this decision region. As before, $z_1, \ldots, z_6$ are decision lines corresponding to the six hidden nodes. The shaded area is given by the Boolean expression:

$$z_1 \wedge z_2 \wedge z_3 \wedge \neg(z_4 \wedge z_5 \wedge z_6) \tag{4.28}$$

which can be brought into the standard form (4.12) as:

$$(\neg z_4 \vee \neg z_5 \vee \neg z_6) \wedge z_1 \wedge z_2 \wedge z_3 \tag{4.29}$$

Disconnected decision regions are possible too with a two-layer net. We will give an example in one dimension. Given that $a < b$ and the decision boundaries $z_1 = x < a$ and $z_2 = x > b$, then $z_1 \vee z_2$ selects the simplest disconnected decision region possible.

Table 4.3. Column *B1* shows values of the Boolean expression $(z_1 \vee z_2) \wedge z_3 \wedge z_4$ for all possible combinations of z_1, z_2, z_3 and z_4. Column *Sum* shows the value of $z_1 + z_2 + z_3 + z_4$. Because of the chosen topology not all possible combinations of z_1, z_2, z_3 and z_4 are associated with real cells in figure 4.4a. The combinations that are not present in the figure are associated with virtual cells and are shown in column *Present* with a minus sign.

z_1	z_2	z_3	z_4	B1	Sum	Present
0	0	0	0	0	0	−
0	0	0	1	0	1	+
0	0	1	0	0	1	+
0	0	1	1	0	2	+
0	1	0	0	0	1	−
0	1	0	1	0	2	+
0	1	1	0	0	2	−
0	1	1	1	1	3	+
1	0	0	0	0	1	−
1	0	0	1	0	2	−
1	0	1	0	0	2	+
1	0	1	1	1	3	+
1	1	0	0	0	2	+
1	1	0	1	0	3	+
1	1	1	0	0	3	+
1	1	1	1	1	4	+

4.5 Three-layer nets

Three-layer nets are capable of implementing the whole set of Boolean functions, AND, OR, XOR and NOT. We have seen that the Boolean expressions that a two-layer net could form were of the restricted forms (4.12) and (4.13). A three-layer net imposes no restrictions on Boolean expressions. To prove this, we first use the fact that the Boolean operators $\wedge$, $\vee$ and $\neg$ form a complete set, i.e. every possible Boolean expression can be constructed with only these operators. It then suffices to show that with a three-layer net, for every possible combination of these operators, a decision function can be constructed. Every Boolean expression can be broken down into simple expressions that can be handled by a two-layer net, combined with $\wedge$ or $\vee$. These notions conclude the 'proof'. We will just give a simple example how combinations of expressions can be implemented. The most simple Boolean expression that cannot be implemented in a two-layer net is the XOR. We can write $z_1 \oplus z_2$ in two possible ways either as an *or-of-ands* or as an *and-of-ors*:

$$(\neg z_1 \wedge z_2) \vee (z_1 \wedge \neg z_2) \tag{4.30}$$

$$(z_1 \wedge z_2) \vee (\neg z_1 \wedge \neg z_2) \tag{4.31}$$

The two forms above suggest two possible implementations for a net of topology (B_2, B_2, B_1), i.e. two binary inputs, two hidden binary units and one binary output. (Beware: the given topology is that of a binary two-layer net and not that of a three-layer net. However, in general, we are not interested in binary inputs but in analog inputs because we want to combine hyperplanes in the input space. Therefore the inputs should be considered as a layer that receives input from a preceding analog layer.)

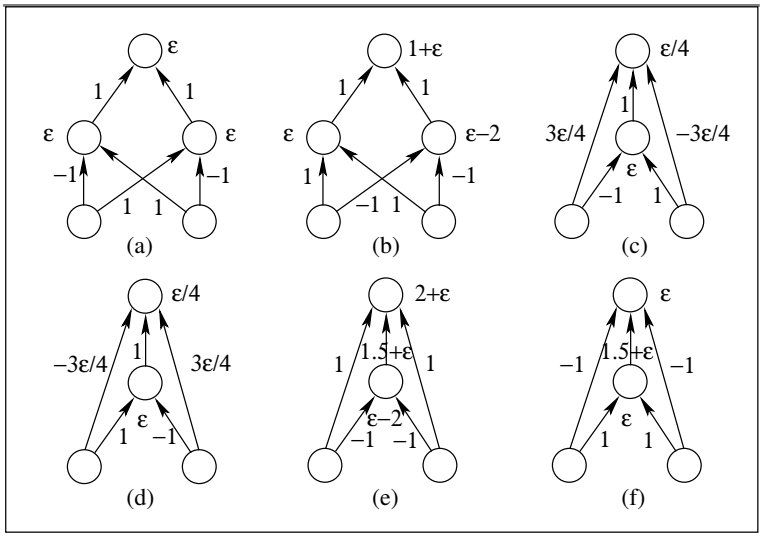

Figure 4.5. Implementations of the XOR-function with two different topologies. In all figures $0 < \epsilon < 1$.

The first implementation is suggested by (4.30): the two expressions within parentheses are implemented in either hidden node and the output combines them with $\vee$. The biases and weights can be calculated with the help of the formulas of section 4.4.3. The second implementation can be derived in an analogous manner from (4.31). These two possible implementations of the $\oplus$ are shown in figure 4.5a and 4.5b. A careful look at equations (4.30) and (4.31) suggests another topology which also implements the $\oplus$: one output node with three input connections, two directly connected to the inputs and the third to the output of a hidden node which is also connected to these inputs. The four possible combinations of weights and biases correspond to the four possible choices for the Boolean function of the one hidden node: each one of the expressions between brackets in (4.30) and (4.31). Figure 4.5c, e, and f show these combinations. Note that 4.5c and d are trivially related by a 'reflection' of the weights. Of course, the possible ways to implement the $\oplus$ are not exhausted yet: with only four constraints on nine parameters for a net with a topology like the one in figure 4.5a (six weights and three biases) there is a lot of freedom. Likewise there are seven parameters to be calculated for a net with a topology as is shown in fig. 4.5c (five weights and two biases).

4.6 Other aspects of neural nets

4.6.1 The nonlinearity

The nonlinearity f in equation (4.2) guarantees that the capabilities of multi-layer neural nets are essentially different from those of one-layer nets. Without the non-linearity the function of a multi-layer net could be reduced to that of an equivalent one-layer net. Different nonlinearities have been used in the literature, depending on the problem. Up till now, in all the derivations of the preceding paragraphs, the non-linearity was a Heaviside function. Possible outputs of nodes were limited to either 0 or 1. Jones et al. (1990) use locally tuned nonlinearities leading to Connectionist Normalized Linear Spline (CNLS) networks. In what follows, we will take the function f of equation (4.2) to be a sigmoid function, σ, because it is monotonous, continuous and differentiable. The function σ reads:

$$\sigma(X) = \frac{1}{1 + e^{-X}}. \tag{4.32}$$

X is a sum of weighted inputs x_k to the node (which can be either outputs of the preceding layer or inputs) minus a bias θ:

$$X = \sum_k w_k x_k - \theta. \tag{4.33}$$

The sigmoid function (4.32) has domain $(-\infty, +\infty)$ and range $(0, 1)$, which means that the output of a node can take all possible values between 0 and 1. From (4.32) we can see that if X is constant, the output of the sigmoid function is constant. The equation $X = \textit{constant}$ defines a hyperplane parallel to the hyperplane defined by $X = 0$, the decision boundary of the hard limiting threshold function. Contrary to the latter function, which discontinuously jumps from 0 to 1, the sigmoid changes smoothly from 0 to 1. Both these extremes are approximated when the argument tends to infinity. As shown in figure 4.6, we can define a transitional area where:

$$\delta < \sigma(X) < 1 - \delta \qquad \text{for} \qquad 0 \leq \delta \leq \frac{1}{2}.$$

On the input side this defines a symmetrical region around $X = 0$. The input range ΔX that corresponds with it is:

$$\Delta X(\delta) = 2 \ln \frac{1 - \delta}{\delta}. \tag{4.34}$$

We can interpret $\Delta X(\delta)$ as that part of the domain of X where no clear decisions are yet made, a transitional area. 'Domain' in this context must be understood as the actual domain, i.e. the range of values of X in the problem at hand and not the domain in the mathematical sense which is always $(-\infty, +\infty)$. The ratio R between this transitional area and the actual 'domain' of X is very important because it gives us an indication of the relative sharpness of the decision boundary. This ratio is:

$$0 < R = \frac{\Delta X(\delta)}{\text{domain}(X)} < 1. \tag{4.35}$$

This ratio R is a measure of the resolution we wish, it determines the width of a decision boundary. If we want sharp boundaries this ratio has to be much smaller than one. Given a certain decision boundary, the only way in which a node can 'sharpen' it is by enlarging all the weights and the bias with the same factor:

$$X' = \beta X = \sum_k \beta w_k x_k - \beta\theta = \sum_k w'_k x_k - \theta', \quad \text{for} \quad \beta > 1. \tag{4.36}$$

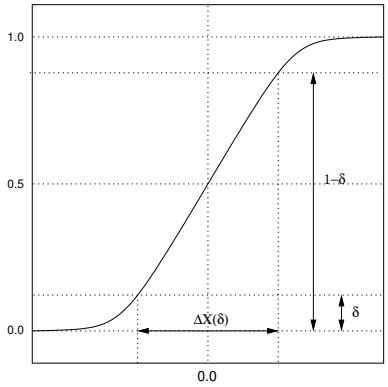

Figure 4.6. The sigmoid function.

From equations (4.33) and (4.36) we deduce that the equations $X' = 0$ and $X = 0$ describe the same hyperplane. However, the ratio R which determines the sharpness of the transition in a node is determined by the on the training data. To separate two training data items that lie close together in the input space but belong to different output classes, the sigmoid has to be steeper than when the items lie further apart in the input space. Therefore we may expect that weights and bias increase if we use training data that lie close together in the input space and but belong to different classes i.e. decision regions. This increase corresponds to an increase in β (note that when β goes to infinity the sigmoid approximates the Heaviside function). This immediately suggests a possible way to train a net: use data that lie close together in the input space but belong to different decision regions; if possible, train with data near decision region boundaries. From a theoretical point of view this may be all right but in practice it turns out to be nearly impossible with feedforward nets. We will return to this point in the discussion.

Because the hyperplanes $X = constant$ are parallel to the plane $X = 0$ (the hyperplane for the hard-limiting threshold function), we conclude that the decision regions for a neural net with sigmoids as nonlinearities can approximate the decision regions limited by hyperplanes as closely as necessary, by increasing β. However, the decision regions can substantially change when the ratio R changes. In general, when R decreases, hypersurfaces gradually degenerate to hyperplanes. As a first example of this matter, we have plotted in figure 4.7 some of the possible decision regions for

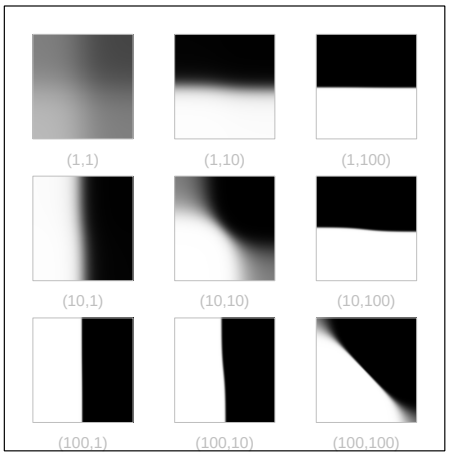

Figure 4.7. Some of the possible decision regions of a net with topology (A_2, A_2, A_1). The decision regions are given by the formula: $\sigma(\beta_1(\sigma(10(x - 0.5)) - 0.5) + \beta_2(\sigma(10(y - 0.5)) - 0.5))$. The values of the weights β_1 and β_2 that were used, are indicated underneath each figure as the pair (β_1, β_2). Black=1 and white=0.

the output of a net with topology (A_2, A_2, A_1). For simplicity, we have taken the two decision regions of the two nodes in the first layer to be the right half and the upper half of the plane. The decision regions of the output were determined by the formula:

$$\sigma(\beta_1(\sigma(10(x - 0.5)) - 0.5) + \beta_2(\sigma(10(y - 0.5)) - 0.5)), \tag{4.37}$$

where the values of the weights β_1 and β_2 have been taken from the set $\{1, 10, 100\}$. For each of the nine possible combinations of β_1 and β_2, the output of this equation was calculated with the input in the domain $[0, 1] \times [0, 1]$. Grey levels were used to represent the output. Underneath each figure the particular values of β_1 and β_2 are indicated as (β_1, β_2). Only positive values of β_1 and β_2 were considered here because different signs merely rotate the forms in the figures in multiples of 90 degrees. We note that all decision regions are unbounded, i.e. all decision regions start and end at the borders. No bounded decision regions are possible with the chosen topology. The figure clearly shows that if the values of β_1 or β_2 increase then the transitional area decreases. This makes the transitional area approach the hard-limiting threshold function.

In figure 4.8 we give an example of what happens to the form of a decision region when the only variation is in the scale factor β. We consider a two-layer net with topology (A_2, A_4, A_1). The two inputs, x_1 and x_2, are confined to the interval $[0, 1]$. With four decision lines, z_1, z_2, z_3 and z_4, we can construct a non-convex decision

region such as in figure 4.4a. The parametrizations chosen are:

$$\begin{aligned} z_1 &: \sigma(\beta_1(x_1 - 0.5)) \\ z_2 &: \sigma(\beta_2(x_1 - x_2 - 0.1)) \\ z_3 &: \sigma(\beta_3(-0.3x_1 + x_2 - 0.1)) \\ z_4 &: \sigma(\beta_4(-1.14x_1 - x_2 + 1.34)) \end{aligned} \tag{4.38}$$

The decision region of the output node, a function of x_1 and x_2, can be calculated with the help of equation (4.27) in which $\epsilon = 0.5$ was chosen:

$$z_5 : \sigma(\beta_5(z_1 + z_2 + 1.5z_3 + 1.5z_4 - 3.75))$$

The β's can help us to influence the steepness of the sigmoids because the equations of the lines in functions (4.38) are invariant under multiplication with a scale factor. All β_i's were all chosen equal, i.e. $\beta_i = \beta$ for $i = 1, \ldots, 5$. In figures 4.8a, b,c and d we have represented (4.39) as a function of the two input variables x_1 and x_2 for $\beta = 1$, 10, 100 and 1000, respectively. They clearly show that the decision region

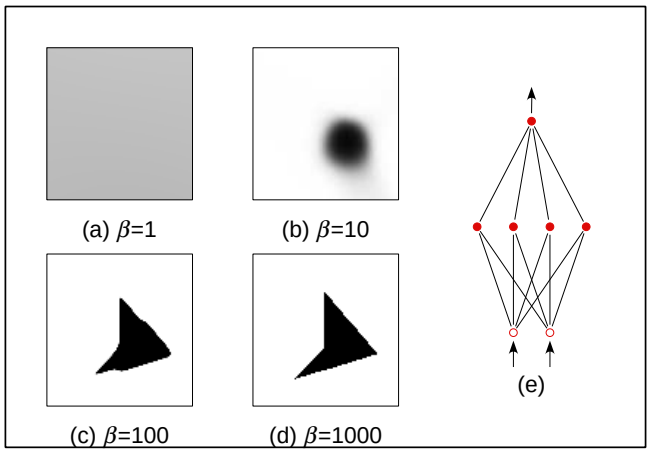

Figure 4.8. The output of net (e), with topology (A_2, A_4, A_1) as a function of the steepness of its sigmoids: (a) $\beta = 1$, (b) $\beta = 10$, (c) $\beta = 100$, (d) $\beta = 1000$. Black=1 and white=0. Further explanation is given in the text.

possesses sharp decision boundaries only if the β's are sufficiently large. For moderate values of β, e.g., $\beta = 10$, the transition area of the sigmoid plays an important role and the decision boundary diverges from the one intended. For $\beta = 1$ the decision region is nowhere near the intended one. The figures 4.8a, b, c, and d illustrate that by varying the weights and the biases, the decision regions of the sigmoid function can be made very diverse and that these decision regions are not limited to those that are bounded by hyperplanes, as was the case for the hard-limiting threshold function, but by hypersurfaces. The corners of the cell can become rounded.

Table 4.4. Inputs and outputs for a multiplexer neural net with two inputs and 4 outputs.

Input	Output
00	1000
01	0100
10	0010
11	0001

4.6.2 Level coding

The output of the sigmoid function can take all values between zero and one. This means that, in principle, more information than just 'on' and 'off' can be coded. Whenever this is the case we speak of *level coding*.

An example of level coding is the following problem. We want to solve the input-output relations of table 4.4 with a multiplexer network of topology (B_2, A_1, A_x, B_4), where x should be the smallest value possible. To solve the input-output relations of this table, the net first has to represent the four possible input values as four different output values of the first hidden node. It makes the problem a one-dimensional one: find a separation of four classes in one dimension. This problem is not linearly separable which means that after the first hidden node we need one extra level of hidden nodes. Rumelhart, Hinton & Williams (1986, their figure 8) give an example with $x = 4$. We have found a solution to this multiplexer problem with $x = 2$. The decision regions of this net are displayed in figure 4.9. The inputs are taken as analog nodes for drawing purposes. The figure demonstrates that the desired function of the net, multiplexing, is performed correctly: each decision region encloses a separate corner of the square. We note that for two possible input combinations, $(0, 0)$ and $(1, 1)$, the functioning of this multiplexer is sensitive to noise: a small deviation from 0 or 1 of the inputs activates the wrong output.

We can go further to show that $x = 2$ solves any multiplexer problem of topology (B_n, A_1, A_2, B_{2^n}): n inputs coding 2^n possible outputs can be multiplexed over one hidden node followed by a layer with two hidden nodes. It suffices to show that the two nodes in the second layer can map the 2^n different outputs of the single node in the first layer to a curved line in two dimensions. Any point on a line with curvature in the same 'direction' can be linearly separated from all other points on that curved line. Let y be the output of the single node in the first layer and z_1 and z_2 the outputs of the two nodes in the second layer. It follows from (4.32) and (4.33) that:

$$\begin{aligned} z_1 &= (1 + e^{-w_1 y + \theta_1})^{-1} \\ z_2 &= (1 + e^{-w_2 y + \theta_2})^{-1} \end{aligned} \tag{4.39}$$

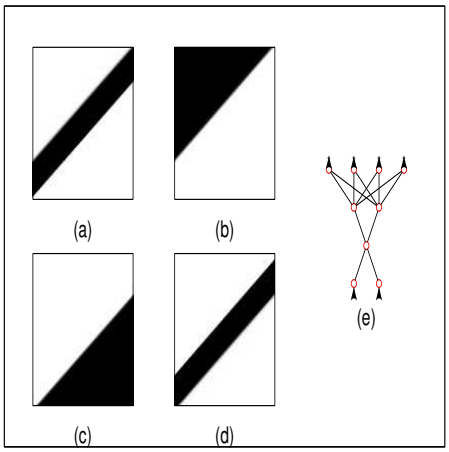

Figure 4.9. Decision regions for each of the outputs of a multiplexer net with topology (A_2, A_1, A_2, B_4). Lower left corner of each square is $(0, 0)$, upper right corner is $(1, 1)$.

Let the weight and bias of the second node be twice those of the first, it follows that:

$$\begin{aligned} z_1 &= (1 + e^{-t})^{-1} \\ z_2 &= (1 + e^{-2t})^{-1} \end{aligned} \tag{4.40}$$

Solving for t and expressing z_2 as a function of z_1 yields:

$$z_2 = \frac{2z_1 - 1}{z_1^2} \tag{4.41}$$

According to (4.41) the value for z_1 is: $0 < z_1 < 1$ and (4.41) describes a curve. By arranging the weights and biases in (4.40) the points (z_1, z_2) can always be positioned on that part of the curve described by equation (4.41) where the curvature has the same direction. This concludes the 'proof' that the two are sufficient to solve the multiplexer problem. Probably it will be very difficult to train a multiplexer like this when the number of inputs is larger than two because ever smaller differences among the output values of the singleton node must be distinguished.

4.6.3 Training a neural net

The purpose of training is to obtain a set of weights and biases that minimizes a certain cost function E. Normally during training a desired output is compared with the actual output of the net. The size of the difference, the error, is an indication for the amount of change necessary for the weights and biases. During the training of a neural net, weights and biases are iteratively updated until the error is smaller than some threshold.

Training a one-layer net is very simple since the desired output is known and the only weights are those between the outputs and the inputs. This means that weights can be gradually updated until the error is sufficiently small. However, this procedure is not directly applicable to a net with hidden nodes because most of the time one does not know what the hidden nodes should represent: there is no desired output for the hidden nodes. Despite this objection, a great many different training procedures for nets with hidden layers have been developed. The number of methods and types of cost functions is too extensive to be described here and this will be deferred to the next chapter.

4.7 Discussion

4.7.1 Possible decision regions and topology

In section 4.5 we have demonstrated that a two-layer neural net could, in principle, make non-convex decision regions: weights and biases were explicitly given for the simplest non-convex region bounded by four lines. However, when we tried to teach this decision region to a net with topology (A_2, A_4, A_1) it turned out that many more iterations were necessary than when we used the topology (A_2, A_4, A_2, A_1). Investigating the weights and biases of the latter net showed that it only had to make decisions based on ANDs. This suggests that although non-convex and 'hollow' decision regions can be formed with a two-layer net, adding an extra layer simplifies the decision making. This would indicate that in each particular layer, conclusions should only be based on a combination of ANDs since this simplifies the decision making process: all nodes make simple decisions. The benefit of adding nodes could involve a decrease in training time. This is in accordance with a notion by Rumelhart et al. (1986) that occurrences of the net getting stuck in local minima always involved networks that had just enough connections to perform the task and that adding a few more connections created extra dimensions in weight space to provide paths around the barriers that would create local minima in lower-dimensional subspaces.

4.7.2 Coding the inputs

In most cases input consists of real numbers. We note that, theoretically, it makes no sense to scale the inputs to a certain range if the scaling algorithm is linear. In principle, the weights and biases of the next layer could perform this scaling. In practical situations, however, scaling the inputs beforehand to the range $(0, 1)$ does make sense. Whenever the absolute value of some or most of the inputs is substantially larger than one, the weighted sum of the inputs, which forms the input to the sigmoid nonlinearity, can be a relatively large number. As a consequence the training algorithm starts when the sigmoid functions are at a position where the derivative is extremely small. As we will see in the next chapter, learning speed depends on the size of this derivative and if this derivative is very small hardly any training of the neural net will occur. The derivative of the sigmoid is largest when the input is near zero. In order to train the net effectively, the input to a sigmoid must not start far off from zero (Elman

& Zipser, 1988; LeCun, Bottou, Orr & Müller, 1998). With the sigmoid non-linearity we scale the inputs to the interval [0, 1].

4.7.3 Coding the outputs

Two possible output units exist: analog and binary output units. In the case of an identity mapping with analog inputs one uses analog output units too, because the input and output signals are conditioned to be the same. This mapping of a signal onto itself is obtained through one or more hidden layers. If the net is used for classification purposes, the outputs make binary decisions: output unit i is trained to give a value near 1 whenever the input belongs to decision region M_i. This presupposes M output units when M different decision regions need to be classified. More 'economical' coding of the outputs (e.g., $\log_2 M$ output units could code M output classes) are not very efficient because part (or all) of the hidden nodes in the layer just before the outputs are simply used as a binary encoder network instead of being used for the task at hand. One further remark must be made concerning output coding: whenever the outputs need to distinguish between elements of two different input sets at the same time, sets with K_1 and K_2 mutually orthogonal elements respectively, two equivalent possibilities exist for the number of outputs. First we could have $K_1 + K_2$ output units of which two units at the same time are active, one from K_1 and one from K_2. In the second case we have $K_1 \times K_2$ outputs and only one of them is active at the same time. Both codings are equivalent since both have $K_1 \times K_2$ different outputs. Let us illustrate this with an example: classification of twelve vowels while at the same time membership of one of three speaker categories (Male, Female or Child) should be given. Possible codings: $12 + 3 = 15$ outputs, with always two active, one of twelve and one of three, or, $12 \times 3 = 36$ outputs with only one active at the same time. The number of outputs, M, gives us an indication for the minimum number of units needed in the preceding hidden layer: $\log_2 M$. When the number of units in the hidden layer is below this minimum, some units are forced to level coding. This may be what we want for some kinds of problems where we are interested in the coding in itself, like the multiplexer problem. Whenever we want to detect 'features' we have to stay above this limit.

4.7.4 Why neural nets?

Using neural nets for classification may have a number of advantages as compared to 'classical' statistical methods. Probably the greatest advantage is that there is no need for assumptions on the underlying distribution of the inputs. In the derivation of the classification possibilities of neural nets, we never had to make any assumption concerning the distribution of the input data. When input data cannot be described well by a distribution function, neural net classifiers outperform statistical classifiers (Niles, Silverman, Tajchman & Bush, 1989). Another pro is that neural nets can be made adaptive. The net can be in the learning mode permanently and adapt itself to 'new' inputs. Since in a neural net the 'knowledge' is distributed over many connections, the classifier can be made very robust. Its performance will not stop but rather de-

grade only gradually with an increasing number of connections that become disabled. The net also shows its robustness when the input data are not complete (or noisy); in many cases, it is able to give the correct output nevertheless. Neural nets can be implemented on any computer and very good learning algorithms exist. The topology of the net is only limited by the computing power of the computer. Nowadays, special parallel hardware exists for the implementation of neural nets.

Unfortunately, neural nets also have some serious drawbacks. It is not yet clear how we can add explicit 'knowledge' to a neural net. The only influence we can exercise is on the topology of the net. We do not know how to modify weights and biases when only 'knowledge' and no algorithm is present, except for some trivial problems such as the construction of a gaussian classifier with a neural net when means and variances of the inputs are known (Lippmann, 1987). Likewise, it can be a tedious task to extract 'rules' or 'knowledge' from a neural net (however, see Carpenter & Tan (1995)). The interpretation of the weights and biases can be difficult, especially when we are confronted with a large net. In this case it will be expedient to design a modularized net, with each module having a specific subtask and the output of each module being integrated afterwards. Although excellent training algorithms exist, they require a great deal of training data and training time for reasonable accuracy. This can be a serious problem when not many data are available. Special net topologies and nonlinearity functions have been developed in order to speed up training time to reduce the amount of data needed (Jones et al., 1990). However, once a neural net has been trained, its response is very fast. Neural nets are indeed very promising but they are not a panacea for just any problem. In general, when an algorithmic solution exists, this algorithm constitutes the preferred solution. Some problems that are very easy to solve with an algorithm, are notably difficult to solve with a neural net. The outstanding example is the parity problem: the net has to decide whether (a possible transformation of) the input has even or odd parity. This is a difficult problem for a neural net because the larger the input numbers, the smaller the relative differences between succeeding numbers. In order to solve this problem, weights can become prohibitively large. Many other problems, e.g., the validation whether a decision region is connected or not, can be reduced to the parity problem (Minsky & Papert, 1969).

4.8 Conclusions

A number of aspects of supervised neural nets have been discussed. Given their simple topology, the powerful combinatorial possibilities in the decision regions are impressive. In the field of automatic speech recognition (ASR) they potentially rival the successful Hidden Markov Models (HMM). Bourlard & Wellekens (1989) showed that compared to HMM, neural nets have advantages: a HMM needs more decisions beforehand, like the number of states, the distribution functions, the permitted transitions and the transition rules. Another disadvantage of HMM is their weak discriminating power. During training the probability of the optimum hypothesis is maximized without minimizing alternative hypotheses. We have seen that during training of a neural net, in conjunction with optimizing the wanted hypothesis, the incorrect ones are be-

ing minimized. In the next chapter we will continue our investigation of neural nets and discuss some alternatives that exist for the cost functions that are used during the training phase.

Chapter 5

Vowel classification with neural nets: a comparison of cost functions[‡]

Abstract

In this chapter the merits of some special cost functions for feedforward neural nets are discussed with respect to their classification performance. Both Juang & Katagiri (1992a) and Hrycej (1992) claim their cost functions, based on minimum classification error (MCE), to be superior to the standard cost function, based on a minimum squared error (MSE) criterion. To falsify their claims we used two test sets, the four-class iris data set of Anderson (1935) with 150 samples of measurements on the iris flower and the twelve-class data set of Van Nierop et al. (1973) with 300 samples of formant frequency measurements on Dutch vowels from 25 female speakers. With these data sets we could not find evidence for superior performance of the MCE cost functions. On the contrary, in the special condition that a blocked updating scheme for the weights and biases was used in the training procedure, the MCE-based cost functions proved to be inferior to MSE.

[‡]This chapter is a modified version of Weenink (1993).

5.1 Introduction

In the previous chapter we have studied some general classification aspects of feedforward neural nets; in this chapter we focus on one important aspect of these neural nets, their cost function. The type of cost function that is used during the training phase of a neural net determines to a large extent its classification performance and, therefore, is an essential part of the neural net. Choosing a wrong cost function can have degrading effects on its performance. In this chapter we want to discuss the merits of cost functions that are based on a criterion called minimum classification error (MCE) in relation to a cost function based on minimum squared error (MSE). In the MSE cost function the classification error is the sum of squares of the differences between the actual outputs and the desired outputs of the neural net.

Two proposed MCE-based cost functions were developed by Hrycej (1992) and Juang & Katagiri (1992a). Especially the latter received considerable attention in the literature (Juang & Katagiri, 1992b; Komori & Katagiri, 1992; Kurinami & Sujiyama, 1992). The discussion here will be focused on these cost functions in relation with supervised feedforward neural nets.

The search for cost functions is motivated by the following list of shortcomings of the MSE cost function[1] with respect to classification (Hrycej, 1992; Hampshire & Waibel, 1990):

1. The winning class is not identified during learning and is not used in the learning rule either. This means that for classification MSE is not necessarily adequate.

2. The inability to consider an arbitrary cost matrix. In MSE classification it is not possible to consider an individually specified cost for each misclassification type, i.e. the cost for classifying a member of the i-th class as a member of the j-th class.

3. MSE, in combination with linear output nodes, i.e. nodes that do not contain the nonlinear function, slows down the convergence of learning by overconstraining the problem. Generally the desired output of the correct class is given the value 1 and the other desired outputs the value 0. It may be clear that a value greater than 1 for the correct output class and a value smaller than 0 for the incorrect classes would be harmless. However, MSE penalizes such cases and it can be expected that these unnecessary constraints slow down the learning process. (For classification in general it is not necessary that the output nodes of a neural net contain a nonlinearity. The nonlinear functions that are used for neural nets, for example the sigmoid function, are always monotonic functions. Monotonic functions preserve rank ordering of their inputs. The output node with the largest input also produces the largest output. Consequently, no nonlinearity in the output nodes need be present.)

4. The MSE cost function is not monotonic with respect to classification when the number of classes, M, exceeds one. In other words, patterns with a 'low' MSE

[1] For the definition of the MSE see equation (5.10).

may be classified wrongly while patterns with a 'high' MSE may be classified correctly. E.g. for a correctly classified pattern it is sufficient that the correct output unit has the value 1 and all other output units have a value only slightly less than 1.[2] This results in a large MSE of approximately $(M-1)/2$. On the other hand, a wrongly classified pattern could have a MSE value as low as 1/2 if the correct output has a value slightly larger than 0 while the other outputs have zero values.

Before we discuss the merits of the alternatives we will have a look at how the cost function influences network parameters.

5.2 The relation between cost function and weights

Before a neural net can be used as a classifier it has to be trained. The purpose of training is to obtain a set of weights and biases that minimizes a certain cost function E over the training set. Training a one-layer net is very simple since the desired output is known and the only weights are those between the outputs and the inputs. This means that weights can be gradually updated until the error is sufficiently small. However, this procedure is not directly applicable to a net with hidden nodes because most of the time one does not know what the hidden nodes should represent: there is no desired output for the hidden nodes. Despite this problem, successful training procedures for nets with hidden layers have been developed. The most common learning algorithm uses a gradient search technique to find the network weights and biases $\mathbf{w}$ that minimize the cost function $E(\mathbf{w})$. It is called the back propagation algorithm (Rumelhart et al., 1986).

The weights and biases of the network are updated iteratively according to:

$$w_{l,ij}(k+1) = w_{l,ij}(k) - \mu \frac{\partial E(\mathbf{w})}{\partial w_{l,ij}}, \tag{5.1}$$

where $w_{l,ij}$ is the weight that connects the input of node j on layer l with the output of node i on layer $l-1$, k is a discrete time index, and μ is a positive constant, called the *learning rate*. Equation (5.1) shows that a new value for a weight can be obtained by subtracting from its current value an amount that depends on the derivative of the cost function $E(\mathbf{w})$ and the learning rate. At the minimum of the cost function its derivative equals zero and the weight will not change anymore. To find this minimum of the cost function $E(\mathbf{w})$, we derive an expression for the partial derivative of the error function with respect to each individual weight in the network. Before we can do so we have to define some terms that will be used in the derivation. For a node j in layer l, the outputs in relation to their inputs are:

$$O_{l,j} = f_l(I_{l,j}) \tag{5.2}$$

[2] This can be seen from the MSE equation (5.10) if we substitute for all the outputs O_j the value 1 and for all except one desired outputs D_j the value 0. One desired output will have the value 1.

$$I_{l,j} = \sum_{k=1}^{N_{l-1}} w_{l,kj} O_{l-1,k}. \tag{5.3}$$

Here $O_{l,j}$ is the output of the node j in the layer l, $w_{l,kj}$ is the weight that connects node k in layer $l-1$ with the j-th node in layer l, $I_{l,j}$ is the input of the nonlinearity f_l of the j-th node in layer l, and N_{l1} the number of nodes at layer $l1$. We define the error at node j of layer l as follows:

$$\delta_{l,j} \equiv -\frac{\partial E}{\partial I_{l,j}}. \tag{5.4}$$

When we use the chain rule on equation (5.4) we get

$$\delta_{l,j} = -\frac{\partial E}{\partial I_{l,j}} = -\sum_{k=1}^{N_{l+1}} \frac{\partial E}{\partial I_{l+1,k}} \cdot \frac{\partial I_{l+k,k}}{\partial I_{l,j}} = \sum_{k=1}^{N_{l+1}} \delta_{l+1,k} \frac{\partial I_{l+1,k}}{\partial I_{l,j}}. \tag{5.5}$$

The last term in the summation can be simplified as

$$\frac{\partial I_{l+1,k}}{\partial I_{l,j}} = \frac{\partial}{\partial I_{l,j}} \sum_{p=1}^{N_{l+1}} w_{l+1,pk} O_{l,k} = \frac{\partial}{\partial I_{l,j}} \sum_{p=1}^{N_{l+1}} w_{l+1,pk} f_l(I_{l,p}) = w_{l+1,jk} f_l'(I_{l,j}). \tag{5.6}$$

The two equations (5.5) and (5.6) combine to:

$$\delta_{l,j} = f_l'(I_{l,j}) \sum_{k=1}^{N_{l+1}} w_{l+1,kj} \delta_{l+1,k}. \tag{5.7}$$

Equation (5.7) expresses the *back propagation* of errors. The errors δ_l at the lower layer l can be calculated from the errors at the next higher layer $l+1$. The derivative of the cost function with respect to the weights can now simply be written as

$$\frac{\partial E}{\partial w_{l,ij}} = \frac{\partial E}{\partial I_{l,j}} \frac{\partial I_{l,j}}{\partial w_{l,ij}} = -\delta_{l,j} O_{l-1,i}. \tag{5.8}$$

The attractiveness of this formulation of the derivative lies in the fact that in equation (5.8) no explicit notion of the cost function figures any more. Derivative information at a layer l is expressed in terms of δ_l and O_l. The specifics of the cost function only enter at the top layer.

If we substitute (5.8) in equation (5.1) we obtain:

$$\Delta w_{l,ij} = w_{l,ij}(k+1) - w_{l,ij}(k) = -\mu \delta_{l,j} O_{l-1,i}. \tag{5.9}$$

This equation says that the amount of weight change in a node ($Deltaw$) linearly depends on the activity at the node's input (O_{l-1}.

When we minimize the errors between the desired outputs and the actual outputs of the net in a quadratic sense, it is called the Minimum Squared Error (MSE) criterion function:

$$E(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^{M} (O_j(\mathbf{p}) - D_j(\mathbf{p}))^2. \tag{5.10}$$

Here $O_j(\mathbf{p})$ denotes the output of the j-th output node of the neural net for pattern $\mathbf{p}$ and $D_j(\mathbf{p})$ the desired output for this pattern on this node. M denotes the number of outputs. This cost function is at a minimum if for all patterns for all output nodes the output of the net is equal to the desired output. For this cost function, the errors at the top level, which propagate back, can simply be calculated according to equation (5.4) as

$$\delta_{L,j} = f_L^{'}(I_{L,j}(\mathbf{p}))(O_j(\mathbf{p}) - D_j(\mathbf{p})). \tag{5.11}$$

Two schemes for updating the weights and biases exist, *incremental* and *blocked* updating. In the incremental updating scheme weights and biases are changed after each training pattern. This scheme is typically used in adaptive sessions when the total training set is not available or changes continuously. When the total training set is fixed a *blocked* updating of the weights is more favourable. In this case the weights and biases are updated only after each *epoch* (an epoch being one full presentation of the entire training set). Usually this is faster than updating after each training sample. Moreover, blocked updating has better convergence properties because the cumulative gradient, which is a mean gradient over the training set, converges to zero for the optimal parameter values.

In the minimization of the cost function we can use gradient information in a special way. The standard minimization method, as described by equation (5.1), is the *steepest descent method*. In this method weight changes are always in the direction of the gradient. This method leads to a not very good algorithm of minimization. The problem with the steepest descent method is that it will perform many small steps in going down a long, narrow valley, even if the valley has a perfect quadratic form. Because the new gradient at the minimum point of any line minimization is perpendicular to the direction just traversed, one must always make a right angle turn, which does not, in general, take one to the minimum. Instead we want a way of proceeding not down the new gradient, but rather in a direction that is somehow constructed to be conjugate to the old gradient and previous directions. *Conjugate gradient* methods accomplish this and therefore are, under many circumstances, superior to steepest descent methods. A further introduction to conjugate gradient and other methods of minimization can be found in Nocedal & Wright (1999). In our feed forward neural net simulation in the PRAAT program we have implemented two minimization algorithms. The simplest minimization method implemented is steepest descent with an (optional) momentum term. The second, more powerful, method is Powell's conjugate gradient method.

A sensible strategy to guarantee that learning will occur is to choose the initial random weights in such a way that the magnitude of the typical input to a unit is somewhat less than unity. This can be achieved by initializing the weights in a layer to a random number in the interval $(-1/\sqrt{n}, 1/\sqrt{n})$, where n is the number of units which feed forward to this layer. As was discussed in section 4.7.2 we then have to scale the inputs to the interval $(0, 1)$.

5.3 The cost function by Juang & Katagiri

The first alternative to MSE that we consider was formulated by Juang & Katagiri (1992a). It contains an explicit notion of the winning class and therefore addresses the first point of the list of "shortcomings of MSE" in section 5.1. They define the cost function for a pattern $\mathbf{p}$ to be a sigmoid function of a continuous misclassification measure d_k:

$$E(\mathbf{p}) = \sigma(d_k(\mathbf{p})) = \frac{1}{1 + e^{-d_k(\mathbf{p})}}, \tag{5.12}$$

where d_k is defined as:

$$d_k(\mathbf{p}) = -O_k(\mathbf{p}) + \left[\frac{1}{M-1} \sum_{j \neq k} O_j^{\eta}(\mathbf{p}) \right]^{\frac{1}{\eta}}. \tag{5.13}$$

Here pattern $\mathbf{p}$ is supposed to belong to class k (the correct output class), $O_k(\mathbf{p})$ is the output of the correct node of class k resulting from input pattern $\mathbf{p}$, η is a positive number and there are M classes. In this formula, the correct class appears explicitly, via $O_k(\mathbf{p})$, and the incorrect classes enter in a weighted sum (the term enclosed by the square brackets). When η approaches ∞, the misclassification measure becomes

$$d_k(\mathbf{p}) = -O_k(\mathbf{p}) + \max_{i \neq k} O_i(\mathbf{p}), \tag{5.14}$$

where i is the index of the class with the largest output value other than the correct class k. It is clear that in this case $d_k > 0$ implies misclassification and $d_k < 0$ means correct decision. The errors at the *linear* output nodes, according to equation (5.4), then become

$$\delta_{L,j} = -\frac{\partial E}{\partial I_{L,j}} = -\frac{\partial E}{\partial O_{L,j}} = \frac{\partial \sigma(d_k)}{\partial d_k} \sigma(d_k) \frac{\partial d_k}{\partial O_{L,j}} = -\sigma(d_k)(1 - \sigma(d_k)) \frac{\partial d_k}{\partial O_{l,j}}, \tag{5.15}$$

where the argument pattern vector $\mathbf{p}$ is implied, L is the index of the output layer, and,

$$\frac{\partial d_k}{\partial O_{L,j}} = \begin{cases} -1 & j = k \\ \frac{O_{L,j}^{\eta-1}}{M-1} \left\{ \frac{1}{M-1} \sum_{i,i \neq k} O_{L,i}^{\eta} \right\}^{-1+1/\eta} & j \neq k \end{cases} \tag{5.16}$$

which, for $\eta \to \infty$, becomes:

$$\frac{\partial d_k}{\partial O_{L,j}} = \begin{cases} -1 & j = k \\ 1 & j = \arg\max_i O_i \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

We emphasise again that in equations (5.15) – (5.17) it is understood that the output units are linear, i.e. no sigmoid function is active in these units.

5.3.1 Tests with actual data

In all subsequent comparisons of classification performances between the two cost functions the training data set and the test data set will be the same. Two data sets will be used for comparing classification performances: Fisher's iris data set and Van Nierop et al.'s female vowel formant data.[3]

The iris data consist of four measurements made by Anderson (1935) on 150 samples of three species of iris flowers: iris setosa, iris versicolor and iris virginica. The four measurements are sepal length, sepal width, petal length and petal width. Fifty tokens are available for each of the three classes. The iris data set was one of the test sets used by the authors of this MCE-based cost function and is used extensively in the literature as a reference set (Fisher, 1936). However, we must note that it is not a very interesting data set for classification since standard linear discriminant analysis with the PRAAT program already gives a classification rate of 98.0% correct with only 3 out of 150 items being misclassified. This means that the misclassification of 2.2% that Juang & Katagiri obtain by their equation (5.12) is not impressive (see their table III). In fact, with our neural net simulation in the PRAAT program we easily reach 0.7% misclassification with a one-layer neural net of topology $(4, 3)$[4], i.e. four inputs, three outputs, no hidden units, a sigmoid nonlinearity, and the MSE cost function. This contrasts heavily with their 12.3% misclassification, which was simulated with a net of topology $(4, 15, 3)$ and MCE. With MSE, 15 hidden units, and a nonlinear output layer we also obtained 99.3% correct classification. The iris data set does not show the superiority of the MCE cost function as Juang & Katagiri argue. On the contrary, there is a slight superiority for the MSE-based cost function.

The principal weakness of this MCE-based cost function will reveal itself with a data set that needs considerably more output classes than the iris data set, such as, for example, the formant frequency measurements of 25 female speakers by Van Nierop et al. (1973), which needs twelve output classes. This set consists of the first three formant frequencies in Hertz of the twelve Dutch vowels /u, ɔ, o, ɑ, a, ʏ, ø, y, i, ɪ, e, ɛ/. We have scaled these formant frequency values to the interval $(0, 1)$ according to the following formula:

$$F'_i = \frac{f(F_i) - f(F_{i,\min})}{f(F_{i,\max}) - f(F_{i,\min})} \tag{5.18}$$

Since all vowel classifications in the Van Nierop et al. paper were performed on $\log(F)$ values, the function f was chosen to perform a logarithmic formant frequency transformation by taking $f(x) = \ln(x)$. This transformation is then followed by a linear scaling. The following values were used for the parameters of this linear scaling: the minimum formant frequency values, $F_{i,\min}$, for the first three formants were chosen to be 200, 500, and 1500 Hz, respectively. The maximum formant frequency values, $F_{i,\max}$, were chosen as 1500, 3500, and 4500 Hz, respectively. This linear

[3] Both data sets have been incorporated in the PRAAT program. The command `Create iris data set` in the `New` menu results in a 150 × 4 `TableOfReal` object with row labels '1', '2' or '3' that indicate class membership and column labels 'sl', 'sw', 'pl' and 'pw'. The command `Create TableOfReal (Van Nierop 1973)... 0` in the `New` menu results in a 300 × 3 `TableOfReal` object with the twelve vowel classes as row labels and F1, F2 and F3 as column labels.

[4] All nodes are analog nodes.

scaling makes logarithmic scaling independent of the base of the logarithm and has the additional advantage that all transformed frequencies are in the range $(0, 1)$ which guarantees better training.

With MSE we get excellent classification on the Van Nierop et al. data set as is shown in the last column of table 5.1. When the MCE cost function (5.12) in combination with equation (5.13) was used for *finite* η, classification results were generally poorer. For a net of topology $(3, 10, 12)$, the percentage correct classification, on the average, was less than 70% with the following settings of the simulation program: weights were initialized at a random value between 0.1 and -0.1, a blocked update scheme was used with conjugate gradient minimization, $\eta = 4$ was chosen in equation (5.13), linear output nodes were used, the number of iterations was chosen sufficiently large (>10000) to guarantee good minimization. The performance of less than 70% correct classification is substantially below the more than 86.6% correct classification obtained with the combination of MSE and nonlinear output nodes. Apart from the worse classification performance, we notice that the blocked minimization with MCE more often got stuck in a local minimum than MSE minimization. Using a pattern-by-pattern update and choosing appropriate values for μ and α did not help. A careful look at the patterns that were not correctly classified revealed a flaw in this MCE-based cost function. For finite values of η, the value of d_k can become very negative, meaning very low cost, even when the output value of the correct class is very much smaller than that of one (or more) incorrect class(es). The derivative of the cost function, equation (5.16), is very small as well in this situation, meaning that virtually no correction on this unfavourable situation is taking place. As long as the average of the $M - 1$ values of O_j^η stays much below the value O_k^η, the misclassification does not add much to the cost function. Indeed, the total cost function can reach any small positive value ϵ without 100% correct classification, a very undesirable property. For example, in one session with MCE simulation as above, the total cost was minimized from an initial value of 250.0 to a value of 0.0013 with only 67.3% correct identification. In another session the total cost was reduced from an initial 150.0 to a final 0.00038 and, despite a reduction of the cost with a factor of 10^5, only 58.3% correct classification resulted. When the number of classes (M) increases, the probability that this phenomenon occurs is likely to increase.

Since finite η does not do the job, the only formulation of this MCE cost function that needs checking is the limiting case $\eta \to \infty$: only the difference of the output for the correct class and the highest output of the resulting output units appears in the cost function. The measure in equation (5.14) for d_k now clearly is better coupled to classification performance than before: a negative value represents correct classification. In table 5.1 we have accumulated some results of testing this MCE-based cost function. We have tested MCE by a pattern-by-pattern weight update; for each iteration the patterns were randomized. The values chosen for the gradient descent for the pattern-by-pattern update were: $\mu = 0.003$ and $\alpha = 0.9$. The number of iterations was chosen to be sufficiently high (50000). We had to use this pattern-by-pattern updating scheme because the blocked updating scheme did not perform reliably with this cost function; it got stuck many times in a local minimum. It seems that with MCE in the blocked update case some sort of cancellation of weight changes often takes

Table 5.1. Comparison of classification performance between the MSE cost function and the MCE cost function by *Juang & Katagiri* with $\eta \to \infty$. The topology of the neural net was (3, N, 12). The training data set of van Nierop et al. was used (see text). MCE was tested with pattern-by-pattern update with randomization ($\mu = 0.003$ and $\alpha = 0.9$). The columns, from left to right, denote the number of hidden units, the MCE-cost after training, and the percentages correct for MCE- and MSE-based training, respectively. The percentage correct derived in the Van Nierop et al. study via maximum likelihood classification was 79.0%.

	#Hidden	Cost	MCE (%)	MSE (%)
	2	72.0	77.0	75.6
	3	65.0	78.0	79.3
	4	58.2	81.3	80.3
	5	55.9	82.0	83.0
	6	53.7	83.0	82.6
	7	52.5	83.0	86.6
finite η	10		<70.0	> 86.6

place, in such a way that no effective updating is possible any longer. The results in table 5.1 show that with the van Nierop et al. data set, the results for MCE, especially for topologies with a small number of hidden units, are satisfying. But these results come at a great cost: many times the blocked updating of weights with fast conjugate gradient minimization cannot be used, and pattern-by-pattern updating with steepest descent has to be used instead. Furthermore, the learning parameters (μ, α) have to be optimally adjusted to guarantee proper minimization. Pattern-by-pattern updating takes considerably more computer time than a blocked update. Moreover, our powerful minimization algorithms cannot be used. Careful testing with more difficult artificially generated data sets with strongly overlapping classes showed clear superiority of the MSE cost function in combination with nonlinear output nodes over MCE with linear output nodes.

In summary, we could not find convincing evidence for the superiority of this MCE-based cost function over the standard MSE-based cost function.

5.4 The cost function by Hrycej

The cost function by Hrycej (1992) is the simplest form of a function that imposes no weight changes if classification is correct. It has a non-zero gradient only in the region were the cost is positive. The cost function is:

$$E(\mathbf{p}) = C_{ki}\,\mathrm{pos}(O_i(\mathbf{p}) - O_k(\mathbf{p})) \tag{5.19}$$

in which k is the index of the correct class of pattern $\mathbf{p}$, i the index of the largest output, C_{ki} is the element of the cost matrix that denotes the cost of misclassifying a pattern belonging to the correct class k as belonging to the incorrect class i, and the function $\text{pos}(u)$ is defined as:

$$\text{pos}(u) = \begin{cases} u & u > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.20}$$

The cost matrix C_{ki} need not be symmetric and can be any general matrix. The errors at the output level can be expressed as

$$\delta_{L,j} = \begin{cases} -C_{ki}\,\theta(O_i - O_k), & j = k \\ C_{ki}\,\theta(O_i - O_k), & j = i \\ 0, & \text{otherwise} \end{cases} \tag{5.21}$$

where $\theta(u)$ denotes the step function, defined by

$$\theta(u) = \begin{cases} 1 & u > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.22}$$

This cost function, in combination with blocked updating, has the admirable property that it is a convex function with regard to the classifier parameters, i.e. the weights (Hrycej, 1992). This means that the global minimum of the cumulative cost function can be found by gradient descent.

5.4.1 Tests with actual data

In table 5.2 we have accumulated some results of the comparison of MCE versus MSE. The data set used for the comparison was again Van Nierop et al.'s vowel formant frequency data set of 25 Dutch female speakers. We mention that, as was the case with the MCE-based cost function of the previous section, a blocked updating scheme of the weights was not very successful. Again, we had to use incremental updating. We chose $\mu = 0.003$ and $\alpha = 0.9$. The classification results for this MCE-based cost function were again not very impressive. We did several other tests with data sets with strongly overlapping classes and this MCE cost function did not perform well. Many times it got stuck in a local minimum without any substantial classification performance. Careful analysis of the resulting states lead us to detect a defect in this cost function: The main weakness of the Hrycej cost function is that it is too sensitive to scale: a trivial reduction of all output weights and biases with a factor α $(0 < \alpha < 1)$ reduces the outputs with the same factor because the output nodes are linear. The net effect of this reduction is that the total cost is reduced with the same factor, but without any implication on the classification performance whatsoever. The global cost can be reduced to any small positive number ϵ, without affecting the classification at all. A very undesirable property for a cost function.

Table 5.2. Comparison of classification performance between the MSE cost function and the MCE cost function by Hrycej. Incremental updating scheme with $\mu = 0.003$ and $\alpha = 0.9$. For further details see table 5.1.

# Hidden	Cost	MCE (%)	MSE (%)
2	0.43	72.0	75.6
3	0.42	74.3	79.3
4	0.34	73.0	80.3
5	0.24	73.3	83.0
6	0.44	71.3	82.6
7	0.86	72.0	86.6

5.5 Discussion on cost functions

Most of the criticism formulated in section 5.1 on MSE-based learning is only appropriate for the combination of MSE and *linear* output units. The only serious objection that MSE has no remedy for, is the first one of the list: the MSE cost function is not necessarily optimal for classification. The rest of the objections can be dealt with easily, as we will demonstrate. A class-specific cost can be incorporated in the MSE cost function of equation (5.10) in an analogous way as was done in the previous section with the Hrycej cost function:

$$E(\mathbf{p}) = \frac{1}{2} \sum_{j=1}^{M} C_{kj}(O_j - d_j)^2. \tag{5.23}$$

In this cost function, pattern $\mathbf{p}$ belongs to class k and C_{kj} is the cost when class k is misclassified as class j. This reduces to the standard MSE formulation when all C_{kj} are equal to 1. The errors at the output level can be calculated in a simple way:

$$\delta'_{L,j} = f'_L(I_{L,j}) \cdot C_{kj}(O_j - d_j), \tag{5.24}$$

where f is the function present at the output nodes.

As was explained in section 5.1, when used in combination with linear units, the MSE cost function slows down learning. However, this need not worry us since MSE and linear output nodes were not meant for each other. The solution is to change the output function to a sigmoid. This immediately creates the necessary freedom and removes the unnecessary constraints because the domain of the sigmoid is $(-\infty, +\infty)$. The price we have to pay for changing the output function to a sigmoid is an increase in the learning time. All MSE classification tests in this paper were performed with sigmoid nonlinearities in *all* nodes. A cost function that is monotonic with respect to classification cannot be a function of all outputs at the same time. When the number of classes is substantial and the cost function is non-monotonic, it is always possible that misclassified patterns exist with low cost. It may be that for these misclassified

patterns the way to go in weight space in order to reach perfect classification is either partly uphill or very slowly downhill. In the averaging performed by a cumulative update this can usually be remedied. The only objection against MSE that remains valid is the first argument in the list of section 5.1: MSE is not necessarily optimal for classification. From the discussion above we must conclude, however, that although MSE as a cost function is probably suboptimal, it is certainly hard to beat.

In the next chapter we will use a neural net with an MSE cost function to model speaker adaptive vowel normalization.

Chapter 6

Modelling speaker normalization by adapting the bias in a neural net[‡]

Abstract

In this chapter we present our first attempt at modeling speaker normalization in terms of bias adaptation. The model is tested on vowel formant data from a variety of speakers. In this model vowel identification is considered to be based on the integration of a number of decisions. Each decision is of the same simple form: decide whenever the accumulated evidence exceeds a certain threshold (bias). Changing the bias can influence a decision and consequently the identification. This process could equally well describe human vowel perception as neural net classification.

This model will be tested with formant frequency data from Pols et al. (1973), Van Nierop et al. (1973) and Weenink (1986a). These tests show that although the model seems to work allright an unrealistic amount of test data is needed for the adaptation to perform well.

[‡]This chapter is a modified version of Weenink & Pols (1999).

6.1 Introduction

In this chapter we will try to explore the knowledge that was gained in the previous chapters about neural nets and incorporate it in a general model about speaker normalization. The listening experiments with manipulated vowel segments in blocked and mixed speaker conditions were described in chapter 2. In these experiments there was a significant difference in identification errors between the blocked and the mixed speaker condition. As subjects listened to more stimuli from the same speaker (blocked condition), they made fewer identification errors. This result has also been noted by many other researchers (Strange et al., 1976; Verbrugge et al., 1976; Macchi, 1980; Assmann et al., 1982). Along another line of experiments (Ladefoged & Broadbent, 1957; Dechovitz, 1977; Van Bergem, Pols & van Beinum, 1988) it was shown that by manipulating the context, exactly the same stimulus could evoke different responses (this phenomenon, by the way, occurs in all aspects of human perception). The generally accepted explanation for this type of listening experiments is that context enables the user to construct a frame of reference for a particular speaker and this frame of reference makes it easier to identify the stimulus. The outcome of these experiments described above can be generalized. In all these experiments we are confronted with the fact that a different context can evoke a different response to the same stimulus. We must therefore conclude that in many cases the identity of the stimulus can only be defined in relation to its context. The context provides a means for the listener to adapt to a speaker. Instead of normalization we prefer the term *adaptation*. In order to describe and model this adaptation process we have to define how we view the identification process of a vowel.

6.2 The model

In traditional normalization procedures the general idea is that, for an unknown vowel with formants (F_1, F_2), the normalizing transformation generates a new (F_1', F_2'). The identity of the unknown vowel with normalized formants (F_1', F_2') is then determined by searching for the minimum distance between the normalized vowel and normalized prototypes according to some distance metric in the normalized space (see Adank (2003, chapter 2) for an overview). Besides the Euclidean distance metric, many other distance criteria have been developed (Shikano & Itakura, 1992; Gray & Markel, 1976). We propose a different normalization model which can easily be simulated with a neural net. In the present model the extrinsic and intrinsic aspects of normalization are interwoven.[1] It is based on the following hypotheses concerning the identification/normalization process:

1. Identification is the integration of a number of decisions. Each individual deci-

[1] The transformations are sometimes divided into the classes *extrinsic* and *intrinsic*. This division depends on whether the normalization procedure uses more information than available in the vowel and its immediate context or not. For example an extrinsic procedure might use the average vowel position of a speaker in the calculation. An intrinsic procedure only uses information present within the vowel and its immediate context.

sion has the same basic structure: a decision is made if accumulated evidence exceeds a certain threshold (bias).

2. The way to accumulate evidence is fixed, the threshold (bias) is variable. The general strategy to make a decision is fixed, but the amount of evidence necessary to force a decision may be variable. A change in the bias shifts an individual decision and consequently can alter the outcome of the identification process.[2]

3. Adaptation can be modelled as a variation of bias. The listener may use context to adjust the bias(es) if necessary.

This model of adaptation can be implementation with feedforward neural nets. The decisions made by a neural net can be interpreted in terms of the first hypothesis of the model. The summing of the multiplications of the inputs to a node with the appropriate weights (e.g., in formula 4.2) can be viewed as the gathering of evidence. With a nonlinearity of the Heaviside form a decision for a 1 is made whenever the evidence (the sum of the products) exceeds the bias. When the sum is less than the bias a decision 0 is made. With a continuous function as the nonlinearity, e.g., the sigmoid function, the decisions need not be binary values but can attain any value between 0 and 1. A feedforward neural net can be viewed as the integration of all these decisions.

6.3 Geometrical interpretation

The second hypothesis of the model has a nice geometrical interpretation. As was explained in section 4.2, the linear combination of the n inputs of a node, when equated to a constant, forms the equation of a hyperplane in n-dimensional space. This hyperplane separates the input space into two regions, one on either side of the hyperplane. Through the nonlinearity, the output attains two different values on either side of the hyperplane. The hyperplane thus forms a decision boundary. When the bias changes, the orientation of this hyperplane remains fixed. However, bias changes correspond to parallel translations of this plane. This can most easily be visualised in two dimensions. The decision boundary in this case is a line. In fig. 6.1 the decision boundaries of a node with two inputs are shown for three values of the bias θ for fixed weights w_1 and w_2. A line through the origin would correspond to $\theta = 0$. As can be verified, by only varying the bias, the position of the line changes parallel to the line through the origin. This results in a change of the class membership for those inputs that lie between the dashed lines. Varying the bias therefore is a means to change the decision boundaries between classes.

Of course, there are more ways to change the decision boundaries between the classes. The most drastic one is to change all weights instead of only the biases. In this way not only parallel movements of the boundaries are possible, but rotations as well. However, we have to keep in mind that here we want to describe adaptation

[2] See chapter 10 for a different view.

and not the whole process of learning. In the learning process all weights and biases are changed to values that minimize a particular cost function. Adaptation is not the process of learning anew. The problem is how to learn 'new' things without forgetting knowledge acquired previously. With adaptation we mean here that we can make certain small changes to our perceptual strategy to be able to cope better with some stimuli that lie near decision boundaries given a certain context.

The term adaptation has a slightly different meaning in the field of adaptive pattern recognition. In the latter field, adaptive stands for incrementally updating the weights after each input pattern. These networks are based on self-organization and are practically always in the training phase. Adaptation is then described as a balanced interaction between assimilation and accommodation. Assimilation corresponds to assigning patterns to categories and improving the definition of the corresponding category. New patterns are integrated into existing structures, which leads to the strengthening of these structures. Accommodation can be viewed as changing the category system of the classifier upon arrival of completely novel patterns. Existing structures are modified by new patterns. Overemphasized assimilation leads to egocentric, nonadaptive behaviour, while too much accommodation leads to instability and poor generalization. The problem of keeping a balance between assimilation and accommodation is also called the *stability-plasticity dilemma*. In the field of adaptive pattern recognition, Grossberg's Adaptive Resonance Theory (ART) deals with this problem (Grossberg, 1976; Carpenter & Grossberg, 1987a). This will be discussed in more detail in chapter 10 where we will develop another adaptation model that is based on ART.

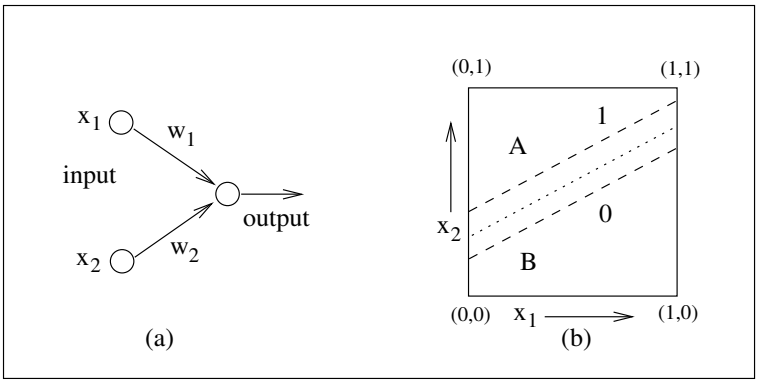

Figure 6.1. An example of the effect of bias change on the decision boundary. In the left figure the topology of the net is shown. The right figure shows three parallel decision boundaries $w_1x_1 + w_2x_2 + \theta = 0$ for three different values of the bias θ and fixed weights w_1 and w_2.

6.4 The test data sets

As training sets we will use the formant frequency data sets of Pols et al. (1973) and Van Nierop et al. (1973). For easy reference some of their classification results, those based on three formant frequencies per vowel, have been accumulated in table 6.1.All entries in the table are based on the logarithmically transformed frequencies of the first three formants. Apart from the twelve standard vowel categories, they also considered classification on a grouping of the vowels into nine categories. These nine categories consisted of six single vowel categories with /u/, /ɑ/, /a/, /y/, /i/, /ɛ/, and three categories with the long-short vowel pairs /o, ɔ/, /ø, ʏ/, and, /e, ɪ/. The classification results on these grouped categories are also accumulated in the table.

Table 6.1. Percent correct classification results based on the three logarithmically transformed formant frequencies per vowel of 50 male speakers, labelled M50, and 25 female speakers, labelled W25. The centred data have been normalized by subtracting the difference between the speaker centroid and the group centroid. These results are a compilation from Pols et al. (1973) and Van Nierop et al. (1973). For more details see text.

Speakers	Non-Centred		Centred	
	12 Categories	9 Categories	12 Categories	9 Categories
M50	75.3	89.3	80.5	95.5
W25	79.0	93.0	84.0	97.3

In order to compare the classification results of the neural nets with these data, we also performed training and testing both with and without grouping of vowel categories. From both training sets we will only use the formant frequencies since the Weenink (1985) test sets that were described in chapter 2 do not contain formant level information. The training set with 50 male speakers contains 600 (= 50 × 12) patterns, the set with the 25 female speakers contains 300 (= 25 × 12) patterns. The testing of the model consists of the following phases.

- Pre-processing of the formant frequency values. The procedure used for the i-th formant was the same as that we have used in the previous chapter (see equation (5.18)). The chosen formant extrema were also the same.

- Training. Adjust all weights and biases of the neural network according to the MSE cost function with formula (5.1). This cost function was shown to be very efficient in chapter 5. The total number of weights and biases of a two layer network with n inputs, h hidden units and m outputs is the sum of the number of weights and biases from the inputs to the hidden layer, $h(n+1)$, and the number of weights and biases from the hidden layer to the output layer, $m(h+1)$, which amounts to $h(n+m+1)+m$. This last number is equal to the dimensionality of the search space. For a two-layer neural net we can see

that the number of parameters that have to be modelled increases *linearly* both with the dimensionality of the input space (n) as well as with the number of hidden units (h). This implies that the size of the training set needed only has to increase linearly when the input dimensionality increases. This still remains true for neural nets with more than one hidden layer.

- Adaptation to the test set. Here we perform a subminimization: all weights and biases of the trained network keep their value and only the m biases of the output layer or the n biases in the hidden layer are varied in the minimization process. The search space thus is reduced to either m-dimensional or n-dimensional. As test sets we use the Pols et al. (1973), the Van Nierop et al. (1973) and the Weenink (1985) formant frequency data. These sets consist of the first three formant frequencies of twelve vowels produced by male, female, and children speakers.

- Classification of the test sets. The percentage correct classifications of the test sets are collected. Correct classification occurs when the output of the 'correct' output unit is bigger than the output of any other output unit, whatever the maximal output level is (there is no minimum level that the output should attain).

6.5 The number of parameters

In the simulations that follow, we vary the topology of the net, i.e. the number of hidden units, between two and seven. When the number of hidden nodes is seven, the number of weights and biases in the neural net is approximately equal to the number of parameters in the Van Nierop et al. (1973) classification model for the grouped vowels. This is based on the following assumptions: in the Van Nierop et al. paper the classification was based on ellipses in three dimensions that did not have their main axes in parallel. These ellipses that determine how distance is measured were calculated from the individual covariance matrices. Van Nierop et al.'s classification method is very similar to a Quadratic Discriminant Analysis as implemented in the PRAAT program (see section 3.5.6 for more information).

The number of parameters for a general ellipse in n dimensions can easily be calculated. We start with the number of parameters that specify a two-dimensional ellipse. The position of the "centre" of the ellipse is determined by two parameters. Three additional parameters specify the ellipse: two parameters specify the major axis, i.e. axis length and the angle with respect to the x axis. To specify the minor axis we need only one length parameter since its orientation, orthogonal to the major axis, is fixed. A three-dimensional ellipse needs three parameters to define its centre and six parameters to define its form: three parameters for the principal axis, i.e. two for its orientation and one for its length, and another three that define the resulting two-dimensional ellipse. For a four-dimensional ellipse we need four parameters for the principal axis plus six for the resulting three-dimensional ellipse. For specification of its centre we need an additional four. To specify the form in n dimensions we need n parameters for the principal axis plus the number of parameters to specify the resulting

$(n-1)$-dimensional ellipse. This expands to $(n)+(n-1)+(n-2)+\cdots 4+3+2+1$, which is equal to $n(n+1)/2$. When we add the n parameters for the centre this sums to $n(n+3)/2$.[3] For m output classes the total number of parameters necessary will be $mn(n+3)/2$. We note that the number of parameters increases *linearly* with the number of categories and *quadratically* with the dimensionality of the input space. This makes this type of classification unfavourable for large input dimension n as compared to the neural net where the size of the net only increases linearly with the input dimensionality. This also implies that the size of the data set needed to train a classifier based on the Mahalanobis distance, which was defined in equation 3.25 on page 39, has to increase quadratically with the input dimensionality because this classification is based on multi-dimensional ellipses (the dimensionality of the inverse covariance matrix is $n(n+1)/2$). The need for input-dimension-reduction techniques for classification based on Mahalanobis distance is therefore very urgent. Fixing all directions of all the ellipses in parallel reduces the number of parameters with a factor of m. A simple classification based on means and standard deviations along the "standard" directions i.e. the x and y axes and so on, involves $2nm$ parameters.

For a two-layer neural net with n inputs and m outputs, the number h of hidden nodes can be varied. We want to find the number h for which the number of parameters in the net attains the number of parameters used for the classification based on the Mahalanobis distance. This number functions as the upper limit for the number of hidden nodes in our tests. The value for h can be calculated by equating the number of parameters in both models:

$$h(n+m+1)+m = \frac{mn(n+3)}{2}, \tag{6.1}$$

which translates to

$$h = \frac{m(n(n+3)-2)}{2(n+m+1)}. \tag{6.2}$$

For twelve output classes the equation above gives $h = 6.0$, for nine output classes $h = 5.5$. To make a fair comparison between the neural net method and the Mahalanobis distance the number of hidden nodes should be six or lower. We decided to use seven as the maximum number of hidden nodes and two as the minimum number in the training.

6.6 Frequency scales

Before extensively testing our adaptation model, we first want to check whether a logarithmic frequency scale is appropriate.

In table 6.2 on page 89 the classification results are displayed for the Van Nierop et al. data set with formant frequencies transformed to three different frequency scales: linear, logarithmic and Bark scale, respectively. This scaling is performed according to equation (5.18) with appropriate values for the function f. These values are $f(x) = x$

[3] An alternative calculation is based on QDA: since the form of each ellipse is based on a covariance matrix which is a square symmetric real matrix, the number of independent parameters follows directly.

and $f(x) = \ln(x)$ for linear and logarithmic scaling, respectively. For the Bark scale the following formula was used (Schroeder, 1977):

$$f(x) = 7 \ln\left(\frac{x}{650} + \sqrt{1 + \left(\frac{x}{650} \right)^2} \right), \tag{6.3}$$

where the result $f(x)$ is in Barks and the frequency x is in Hz.

In general, one can never guarantee that the minimization of a function in multi-dimensional space succeeds in finding the global minimum. Therefore, a number of trials were performed with different starting positions in weight space for each topology. The numbers in the table are the maximum classification results of these trials. From table 6.2 we note that, when the number of hidden nodes increases, the classification results get better. It is also clear that no significant differences exist between the different frequency representations. This notion is in line with Watrous (1993), who could not find classification differences between logarithmically and linearly scaled formant frequencies either. The fact that no differences in classification performance is noticeable between the different frequency scales is at first sight surprising. One could argue that since Bark- and log-frequencies give more compact representations and are more perceptually motivated, this would result in better classification results. We have to notice that a 'better' compactness under these circumstances always meant implicitly that the resulting distribution better matched a normal distribution. Most classifiers are implicitly based on normal distributions. But neural nets do not base their internal representations on a normal distribution of the inputs. They are powerful enough to make their own internal representations of the input data, irrespective of the distribution. The generalizing capabilities do of course depend on how well the input space is covered by the input data. Since the choice of frequency seems to be irrelevant for our data and both the Van Nierop et al. and the Pols et al. data sets contain logarithmically transformed formant frequencies, in what follows we will continue to use logarithmically transformed formant frequencies.

In the last two columns of table 6.2 we show the percentages correct classification for the log transformed vowels with twelve and nine categories, respectively. When we compare, for different numbers of hidden units, the numbers in this table with the numbers in table 6.1, we see that our classification results compare favourably with the results of Van Nierop et al. (79.0% correct for twelve vowel categories and 93.0% for nine vowel categories). A net with two hidden nodes already gives excellent classification results. This is remarkable because according to table 4.1 two hidden nodes in a three-dimensional input space can form four cells and we have twelve vowel categories. The table gives a correct representation when the nonlinearity is of the Heaviside form. We, however, use the sigmoid nonlinearity and we have seen that a very powerful aspect of neural nets, equipped with this type of nonlinearity, is level coding. Considering this aspect of neural nets it may be less surprising to find already good classification performances for a neural net with just two hidden nodes.

It is even more surprising that a net with six hidden nodes gives almost the same classification results as the Van Nierop et al. (1973) classification on centred data. For centred data the difference between the speaker centroid and the group centroid has

Table 6.2. Percentages correct classification of neural nets trained and tested with the female formant frequency data of Van Nierop et al. (1973). Input data were presented on a linear, a Bark and a logarithmic frequency scale, respectively. The neural nets had one hidden layer and their topologies were $(3, h, m)$. The number of hidden units, h, is indicated in the first column of the table. The number of outputs, m, is 12, except for the last column, the grouped data, for which m is 9. Training set and test set were identical.

# Hidden	Lin (%)	Barks (%)	Log (%)	Log Grouped (%)
2	76.3	77.3	75.6	91.0
3	79.3	78.0	79.3	95.0
4	81.0	81.6	80.3	95.3
5	82.3	82.0	83.0	96.0
6	84.6	83.6	82.6	97.0
7	86.0	88.3	86.6	98.0

been subtracted. The speaker centroid is the average of all the vowels from a speaker. The group centroid is the average of all the speaker centroids. By subtracting this difference we make all speaker centroids for the transformed vowels equal. Centred data is a form of extrinsic normalization. This shows that a neural net without speaker normalization performs almost as well as Van Nierop et al. with speaker normalization.

6.7 Test of the adaptation model

In table 6.3 we have collected the results on the test of our adaptation model with neural nets that were trained with the logarithmically transformed formant frequency values of the 50 male speakers of Pols et al. (1973). The grouped data with nine output categories were used, so the distinction between similar long and short vowels was ignored. The topology of the neural nets used was $(3, h, 9)$, with $h = 3$ or $h = 7$. The number of hidden nodes (h) is indicated in the first column of the table. The second column shows the test sets used: MG10, WG10 and CG10 are sets with the formant frequency values of ten men, ten women and ten children , respectively, where the twelve Dutch vowels are grouped into nine categories. MG50 and WG25 are the data sets of Pols et al. and Van Nierop et al. with vowels grouped into nine categories. In the third column the percentages correct classification of these test sets are shown (for MG50 the test and training set were identical). The general tendency for these test sets is that classification performance gets worse when the speaker category goes from men to women and then from women to children, irrespective of the number of hidden nodes. When the number of hidden nodes increases we expect better classification performance when training and test sets are identical. This is indeed the case for test set MG50: 89.5% and 93.1% correct classification with three and seven

Table 6.3. Classification and adaptation performance of a neural net with topology $(3, h, 9)$, three inputs, h hidden units and nine outputs, trained with the grouped male formant frequency data set of Pols et al. (1973). In the first column the number of hidden nodes is given. The column labelled 'Test set' shows the data set used for testing the net: MG10, WG10 and CG10 denote the set of ten Men, ten Women and ten Children, respectively, MG50 the Pols et al. set of 50 Men and WG25 the Van Nierop et al. set of 25 Women. The column labelled 'Test' denotes the percent correct classification for these test sets. Columns 'H_{all}' and 'O_{all}' contain the percentages correct of the adaptation of the biases of the hidden units and the output units, respectively, for all speakers in the test set together. In the columns 'H_{ind}' and 'O_{ind}' the adaptation was performed for each speaker in the test set individually. The percentage correct classification on data set MG50 in the paper of Pols et al. was 89.3%. For centred data, an extrinsic form of normalization, it improved to 95.5%.

H	Test set	Test (%)	H_{all} (%)	H_{ind} (%)	O_{all} (%)	O_{ind} (%)
3	MG10	91.7	92.5	100.0	95.8	98.3
3	WG10	74.2	91.6	96.7	73.3	76.7
3	CG10	26.7	90.8	98.3	39.1	36.7
	Mean	64.2	91.6	98.3	69.4	70.5
3	MG50	89.3	89.3	96.7	89.3	94.7
3	WG25	78.3	88.0	97.3	83.3	86.7
7	MG10	93.3	94.1	99.2	95.0	96.7
7	WG10	71.7	82.5	85.8	77.5	81.7
7	CG10	51.7	63.3	70.8	58.3	62.5
	Mean	72.2	80.0	85.3	76.9	80.3
7	MG50	93.2	93.2	98.0	93.2	94.5
7	WG25	75.6	84.6	92.0	85.0	83.7

hidden nodes, respectively. Furthermore, the classification performance for test set MG10 also increases from 91.7% to 93.3% correct when the number of hidden nodes increases. This is also the case for test set CG10 where the increase in classification performance almost doubles from 26.7 to 51.7% correct. The test sets of the women speakers, WG10 and WG25, show the opposite effect. Here the classification performance decreases somewhat when the number of hidden nodes increases, which is a bit unexpected (explanation will follow in this section).

In the next column labelled 'H_{all}' we show classification performance after bias adaptation on the hidden layer. The bias adaptation procedure went as follows: we started with a neural network that has been trained on the male data set MG50. In the following adaptation step we trained a subset of the weights of this network again with the test set, i.e. only the bias weights on the hidden layer were allowed to change during the training while all other weights were kept fixed. Training automatically

stopped if the difference between the Minimum Squared Error's on two successive iterations on the test set was smaller than 10^{-7}. For all the test sets this 'training' finished within 200 iterations. In the last step of the adaptation process the neural net was used as a classifier and its performance on the test set was measured. We note a significant increase in percentage correct classification. Especially the results for the topology with three hidden nodes are noticeable: with a change in only three parameters the percentage correct of the women speakers WG10 goes up with almost 20% from 74.2% to 91.6% and WG25 goes up by 9%. For the children CG10 we note an extreme increase from 26.7% to 90.8% correct.

These results get even better when we go to the next column, labelled 'H_{ind}'. These results were obtained by adapting the biases to each individual speaker. The results obtained for the individual speakers were then averaged. The recognition scores for this case came rather close to 100% for all speaker categories, which is quite extraordinary if we realize that the training was performed with data from male speakers.

The last two columns in the table show the performance when the nine biases of the output layer, rather than from the hidden layer, were allowed to adapt. As before, the adaptation was either on the whole test set (O_{all}) or on the individual speakers of the test set (O_{ind}). Both columns show some increase in recognition performance as compared to the third column labelled 'Test'. The increase is substantial if the speaker category of the test set equals the speaker category of the training set (men). The effect is only small when the speaker categories differ. When we compare for all speaker categories the adaptation effect for the biases of the hidden and the output nodes, we note from the table that the best results are obtained with the biases of the hidden nodes. This means that indeed, on the input side, a simple translation of the hyperplanes is sufficient to guarantee proper adaptation and at the same time is powerful enough even for the adaptation of the vowels spoken by children to the vowels as spoken by men. On the output side a translation helps, but is not powerful enough to adapt the children's vowels to the male vowels. Probably, besides a translation we need rotations, which involves many more parameters and therefore is less attractive.

When we investigate the effect of the number of hidden nodes on the classification performance we can note the following: As can be expected, the classification performance, for the case when training and test set are identical, increases when the number of hidden nodes increases. The net has more freedom to model the training set when it has more hidden nodes at its disposal, hence a better classification. This does not necessarily mean that the generalization properties increase with the number of hidden nodes. Generalization depends more on the relation between the number of training patterns and the number of parameters that we want to train. In general one can say that if more training patterns are present in relation to the number of parameters, the generalization properties tend to increase. From table 6.3 we note that the generalization properties are somewhat better for the network with seven hidden nodes. However, the adaptation performance of the net with the smaller number of hidden nodes are better. A possible explanation could be that the net with three hidden nodes necessarily is more general and therefore more easily adaptable. Another explanation could be that although the fraction of the total number of parameters in the net that is to be trained anew is the same in the nets with three and seven hidden nodes,

the number of parameters that is not changed grows linearly and equals $h(n + m)$.

Table 6.4. Classification and adaptation performance of a neural net with topology $(3, h, 9)$ trained with the grouped female formant frequency data set of Van Nierop et al. (1973). For further details see table 6.3 and the text.

H	Test set	Test (%)	H_{all} (%)	H_{ind} (%)	O_{all} (%)	O_{ind} (%)
3	MG10	73.3	90.8	95.8	90.8	85.8
3	WG10	88.3	88.3	93.3	88.3	90.8
3	CG10	55.8	90.8	95.0	75.0	70.0
	Mean	72.5	90.0	94.7	84.7	82.2
3	MG50	68.8	79.0	89.2	78.6	80.8
3	WG25	95.0	95.0	96.3	95.0	96.0
7	MG10	75.8	86.6	90.0	77.5	78.3
7	WG10	84.2	90.8	94.2	85.0	87.5
7	CG10	40.8	55.0	75.8	46.6	41.7
	Mean	66.9	77.5	86.7	69.7	69.2
7	MG50	62.5	76.5	82.3	65.6	70.3
7	WG25	98.0	98.0	98.3	98.0	98.0

Table 6.4 shows the adaptation performance of neural nets that were trained with the logarithmically transformed formant frequency values of the 25 female speakers of Van Nierop et al. (1973), using again the grouped data with nine output categories. The topology of the neural nets used was $(3, h, 9)$, with $h = 3$ and $h = 7$, the same as in the previous table.

The general results are in line with the data in table 6.3: very good adaptation properties, better adaptation when the biases of the hidden layer are changed as compared to a change in the biases of the output layer and better adaption performance for the net with the lower number of hidden nodes. Some small differences, however, do exist. The women data set seems to be more homogeneous, which leads to higher classification results when training and test set are identical (WG25). This results in a somewhat less succesful generalization performance: the results for WG10 in column 'Test' show 88.3% and 84.2% correct for a net with three and seven hidden nodes. This is probably caused by the fact that we do not have enough training data. We may conclude from both tables 6.3 and 6.4 that the adaptation capabilities by only varying the bias of the hidden units of a trained neural net are very powerful. However, in order to get good generalizing capabilities the number of training patterns has to increase.

6.8 Discussion

We discussed a simple model of normalization in terms of adaptation. First we trained the neural net with data that reasonably covered the input space. Next we adapted the net to individual speakers by letting the net only vary a small subset of the weights, namely the biases of either the hidden layer or the output layer. The model proved reasonably successful within the limited task of the classification of static vowel targets. It showed that by selectively training only a subset of all the weights in a neural net, the adaptation of the net towards speakers could be greatly improved. In this field, Watrous (1993) has an interesting approach to normalization that he tests on the Peterson & Barney (1952) data. He uses neural nets, in a two-staged process. First he tries to find the speaker-dependent transformations that are necessary to transform formants to normalized formants. These normalized formants are then input to the second stage, a classifying neural net. His model is based on a clear distinction between extrinsic and intrinsic normalization: the transforming and the classifying net perform these tasks, respectively. Adaptation is performed by an independent optimization of the transforming net. In our approach the same neural net performs the normalization task. Before we can investigate how well this adaptation can be used on bandfilter data (in chapter 9), we first have to discuss some technical issues in the next two chapters.

Chapter 7

Canonical correlation analysis[‡]

Abstract

In this chapter we discuss algorithms for performing canonical correlation analysis to find correlations between two data sets. The canonical correlation coefficients can be calculated directly from the two data sets or from (reduced) representations such as the covariance matrices. The algorithms for both representations are based on singular value decomposition. The methods described here have been implemented in the speech analysis program PRAAT (Boersma & Weenink, 2006). We show how to calculate the correlations between formant frequency values and formant levels, and how to use these correlations to predict the formant frequency values when only the values for the levels are known. Another example will show that an auto-associative neural network actually performs a principal component analysis.

[‡]This chapter is a modified version of Weenink (2003).

7.1 Introduction

Let $\mathbf{X}$ be a data matrix of dimensionality $m \times n$ which contains m replications of an n-dimensional random variable $\mathbf{x}$. The correlation coefficient ρ_{ij} that shows the correlation between the variables x_i and x_j is defined as

$$\rho_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}, \tag{7.1}$$

where the number Σ_{ij} denotes the covariance between x_i and x_j which is defined as

$$\Sigma_{ij} = \frac{1}{m-1}\sum_{k=1}^{m}(X_{ki} - \mu_i)(X_{kj} - \mu_j), \tag{7.2}$$

where μ_i is the average value of x_i. The matrix Σ is called the covariance matrix. From $\mathbf{X}$ we construct the data matrix $\mathbf{A}_x$ by centring the columns of $\mathbf{X}$, i.e. the elements of $\mathbf{A}_x$ are $a_{ij} = X_{ij} - \mu_j$. We can now rewrite the covariance matrix as

$$\Sigma = \frac{1}{m-1}\mathbf{A}'_x\mathbf{A}_x, \tag{7.3}$$

where $\mathbf{A}'_x$ denotes the transpose of $\mathbf{A}_x$.

Note that the correlation coefficient only provides a measure of the *linear* association between the two variables: when the two variables are uncorrelated, i.e. when their correlation coefficient is zero, this only means that no linear function describes their relationship. A quadratic relationship or some other non-linear relationship is certainly not ruled out.

Equation (7.1) shows us the recipe to determine the correlation matrix from the covariance matrix. However, the correlations in the correlation matrix depend very much on the coordinate system that we happen to use. We could rotate the coordinate system in such a way that the projections in the new coordinate system are maximally uncorrelated and this is exactly what a principal component analysis (see chapter 3) achieves: the correlation matrix obtained from the principal components would be the identity matrix, showing only zeros with ones on the diagonal. While each element in the correlation matrix captures the correlation between two variables, the object of canonical correlation analysis is to capture the correlations between two *sets* of variables. Canonical correlation analysis tries to find basis vectors for two sets of multidimensional variables such that the linear correlations between the projections onto these basis vectors are mutually maximized. In the limit when the dimensionality of each set is 1, the canonical correlation coefficient reduces to the correlation coefficient.

We will need this type of analysis when we want to find relations between different representations of the same objects. Here we will demonstrate its usefulness by showing the correlations between principal components and auto-associative neural nets for vowel data.

7.2 Mathematical background

Canonical correlation analysis originates in Hotelling (1936) and the two equations that govern the analysis, as we will show below, are the following:

$$(\Sigma'_{xy}\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2\Sigma_{yy})\mathbf{y} = \mathbf{0} \tag{7.4}$$
$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma'_{xy} - \rho^2\Sigma_{xx})\mathbf{x} = \mathbf{0}, \tag{7.5}$$

where Σ'_{xy} denotes the transpose of $\Sigma_{xy}$and $\Sigma'_{xy} = \Sigma_{xy}$. Both equations look similar and have, in fact, the same eigenvalues. And, given the eigenvectors for one of these equations, we can deduce the eigenvectors for the other, as will be shown in the next section.

7.2.1 Derivation of the canonical correlation analysis equations

In canonical correlation analysis we want to maximize correlations between objects that are represented with two data sets. Let these data sets be $\mathbf{A}_x$ and $\mathbf{A}_y$, of dimensions $m \times n$ and $m \times p$, respectively. Sometimes the data in $\mathbf{A}_y$ and $\mathbf{A}_x$ are called the *dependent* and the *independent* data, respectively. The maximum number of correlations that we can find is then equal to the minimum of the column dimensions n and p. Let the directions of optimal correlations for the $\mathbf{A}_x$ and $\mathbf{A}_y$ data sets be given by the vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. When we project our data on these direction vectors, we obtain two new vectors $\mathbf{z}_x$ and $\mathbf{z}_y$, defined as follows:

$$\mathbf{z}_x = \mathbf{A}_x\mathbf{x} \tag{7.6}$$
$$\mathbf{z}_y = \mathbf{A}_y\mathbf{y}. \tag{7.7}$$

The variables $\mathbf{z}_y$ and $\mathbf{z}_x$ are called the *scores* or the *canonical variates*. The correlation between the scores $\mathbf{z}_y$ and $\mathbf{z}_x$ is then given by:

$$\rho = \frac{\mathbf{z}'_y \cdot \mathbf{z}_x}{\sqrt{\mathbf{z}'_y \cdot \mathbf{z}_y}\sqrt{\mathbf{z}'_x \cdot \mathbf{z}_x}}. \tag{7.8}$$

Our problem is now to find the directions $\mathbf{y}$ and $\mathbf{x}$ that maximize equation (7.8). We first note that ρ is not affected by a rescaling of $\mathbf{z}_y$ or $\mathbf{z}_x$, i.e., a multiplication of $\mathbf{z}_y$ by the scalar α does not change the value of ρ in (7.8). Since the choice of rescaling is arbitrary, we therefore maximize equation (7.8) subject to the constraints

$$\mathbf{z}'_x \cdot \mathbf{z}_x = \mathbf{x}'\mathbf{A}'_x\mathbf{A}_x\mathbf{x} = \mathbf{x}'\Sigma_{xx}\mathbf{x} = 1 \tag{7.9}$$
$$\mathbf{z}'_y \cdot \mathbf{z}_y = \mathbf{y}'\mathbf{A}'_y\mathbf{A}_y\mathbf{y} = \mathbf{y}'\Sigma_{yy}\mathbf{y} = 1. \tag{7.10}$$

We have made the substitutions $\Sigma_{yy} = \mathbf{A}'_y\mathbf{A}_y$ and $\Sigma_{xx} = \mathbf{A}'_x\mathbf{A}_x$, where the Σ's are covariance matrices (the scaling factor to get the covariance matrix, $1/(m-1)$, can be left out without having any influence on the result). When we also substitute $\Sigma_{yx} = \mathbf{A}'_y\mathbf{A}_x$

we use the two constraints above and write the maximization problem in Lagrangian form:

$$L(\rho_x, \rho_y, \mathbf{x}, \mathbf{y}) = \mathbf{y}'\Sigma_{yx}\mathbf{x} - \frac{\rho_x}{2}\left(\mathbf{x}'\Sigma_{xx}\mathbf{x} - 1\right) - \frac{\rho_y}{2}\left(\mathbf{y}'\Sigma_{yy}\mathbf{y} - 1\right), \tag{7.11}$$

We can solve equation (7.11) by first taking derivatives with respect to $\mathbf{y}$ and $\mathbf{x}$:

$$\frac{\partial L}{\partial \mathbf{x}} = \Sigma_{xy}\mathbf{y} - \rho_x\Sigma_{xx}\mathbf{x} = \mathbf{0} \tag{7.12}$$

$$\frac{\partial L}{\partial \mathbf{y}} = \Sigma_{yx}\mathbf{x} - \rho_y\Sigma_{yy}\mathbf{y} = \mathbf{0}. \tag{7.13}$$

Subtractig $\mathbf{x}'$ times the first equation from $\mathbf{y}'$ times the second yields

$$\begin{aligned} 0 &= \mathbf{y}'\Sigma_{yx}\mathbf{x} - \rho_y\mathbf{y}'\Sigma_{yy}\mathbf{y} - \mathbf{x}'\Sigma_{xy}\mathbf{y} + \rho_x\mathbf{x}'\Sigma_{xx}\mathbf{x} \\ &= \rho_x\mathbf{x}'\Sigma_{xx}\mathbf{x} - \rho_y\mathbf{y}'\Sigma_{yy}\mathbf{y}. \end{aligned}$$

Together with the constraints of equations (7.9) and (7.10) we must conclude that $\rho_x = \rho_y \equiv \rho$. When Σ_{xx} is invertible we get from (7.12)

$$\mathbf{x} = \frac{\Sigma_{xx}^{-1}\Sigma_{xy}\mathbf{y}}{\rho}. \tag{7.14}$$

Substitution in (7.13) gives essentially equation (7.4):

$$(\Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2\Sigma_{yy})\mathbf{y} = \mathbf{0}. \tag{7.15}$$

In an analogous way we can get the equation for the vectors $\mathbf{x}$ as:

$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} - \rho^2\Sigma_{xx})\mathbf{x} = \mathbf{0}. \tag{7.16}$$

Because the matrices Σ_{xy} and Σ_{yx} are each other's transpose we write the canonical correlation analysis equations as follows:

$$(\Sigma'_{xy}\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2\Sigma_{yy})\mathbf{y} = \mathbf{0} \tag{7.17}$$

$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma'_{xy} - \rho^2\Sigma_{xx})\mathbf{x} = \mathbf{0}. \tag{7.18}$$

We can now easily see that in the one-dimensional case both equations reduce to a squared form of equation (7.1). The equations (7.17) and (7.18) are so called generalized eigenvalue problems. Special software is needed to solve these equations in a numerically stable and robust manner. In the next section we will discuss two methods to solve these equations. Both methods have been implementend in the PRAAT program.

7.2.2 Solution of the canonical correlation analysis equations

We have to consider two cases here: the simple case when we only have the covariance matrices, or, the somewhat more involved case, when we have the original data matrices at our disposal.

7.2.2.1 Solution from covariance matrices

We will start with the simple case and solve equations (7.17) and (7.18) when we have the covariance matrices Σ_{xx}, Σ_{xy} and Σ_{yy} at our disposal. We will solve one equation and show that the solution for the second equation can be calculated from it. Provided Σ_{yy} is not singular, a simpler looking equation can be obtained by multiplying equation (7.17) from the left by Σ_{yy}^{-1}:

$$(\Sigma_{yy}^{-1}\Sigma_{xy}'\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2)\mathbf{y} = \mathbf{0}. \tag{7.19}$$

This equation can be solved in two steps. First we perform the two matrix inversions and the three matrix multiplications. In the second step we solve for the eigenvalues and eigenvectors of the resulting general square matrix. From the standpoint of numerical precision, actually performing the matrix inversions and multiplications would be a very unwise thing to do because with every matrix multiplication we lose numerical precision. Instead of solving equation (7.17) with the method described above, we will rewrite this generalized *eigenvalue* problem as a generalized *singular value* problem. To accomplish this we will need the Cholesky factorization of the two symmetric matrices Σ_{xx} and Σ_{yy}.

The Cholesky factorization can be performed on symmetric positive definite matrices, like covariance matrices, and is numerically very stable (Golub & Van Loan, 1996). Here we factor the covariance matrices as follows:

$$\begin{aligned}\Sigma_{yy} &= \mathbf{U}_y'\mathbf{U}_y \\ \Sigma_{xx} &= \mathbf{U}_x'\mathbf{U}_x,\end{aligned}$$

where $\mathbf{U}_y$ and $\mathbf{U}_x$ are upper triangular matrices with positive diagonal entries. Let $\mathbf{K}$ be the inverse of $\mathbf{U}_x$; then we can write

$$\Sigma_{xx}^{-1} = \mathbf{K}\mathbf{K}'. \tag{7.20}$$

We substitute this in equation (7.17) and rewrite as

$$((\mathbf{K}'\Sigma_{xy})'(\mathbf{K}'\Sigma_{xy}) - \rho^2\mathbf{U}_y'\mathbf{U}_y)\mathbf{y} = \mathbf{0}. \tag{7.21}$$

This equation is of the form $(\mathbf{A}'\mathbf{A} - \rho\mathbf{B}'\mathbf{B})\mathbf{x} = \mathbf{0}$ which can be solved by a numerically very stable generalized singular value decomposition of $\mathbf{A}$ and $\mathbf{B}$, without actually performing the matrix multiplications $\mathbf{A}'\mathbf{A}$ and $\mathbf{B}'\mathbf{B}$ (Golub & Van Loan (1996); see also chapter 3). We have obtained this equation by only one matrix multiplication, two Cholesky decompositions and one matrix inversion. This allows for a better estimation of the eigenvalues than estimating them from equation (7.19). The square roots of the eigenvalues of equation (7.21) are the canonical correlation coefficients ρ. The eigenvectors $\mathbf{y}$ tell us how to combine the columns of $\mathbf{A}_y$ to get this optimum canonical correlation.

We will now show that the eigenvalues of equations (7.17) and (7.18) are equal and that the eigenvectors for the latter can be obtained from the eigenvectors of the former. We first multiply (7.17) from the left by $\Sigma_{xy}\Sigma_{yy}^{-1}$ and obtain

$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}'\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2\Sigma_{xy})\mathbf{y} = \mathbf{0},$$

which can be rewritten by inserting the identity matrix $\Sigma_{xx}\Sigma_{xx}^{-1}$ as

$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}'\Sigma_{xx}^{-1}\Sigma_{xy} - \rho^2\Sigma_{xx}\Sigma_{xx}^{-1}\Sigma_{xy})\mathbf{y} = \mathbf{0}.$$

Finally we split off the common $\Sigma_{xx}^{-1}\Sigma_{xy}$ part on the right and obtain

$$(\Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{xy}' - \rho^2\Sigma_{xx})\Sigma_{xx}^{-1}\Sigma_{xy}\mathbf{y} = \mathbf{0}. \tag{7.22}$$

We now use (7.14) and obtain equation (7.18). This shows that the eigenvalues of equations (7.17) and (7.18) are equal and that the eigenvectors $\mathbf{x}$ for equation (7.18) can be obtained from the eigenvectors $\mathbf{y}$ of equation (7.17) as $\mathbf{x} = \Sigma_{xx}^{-1}\Sigma_{xy}\mathbf{y}$.

7.2.2.2 Solution from data matrices

When we have the data matrices $\mathbf{A}_x$ and $\mathbf{A}_y$ at our disposal we do not need to compute the covariance matrices $\Sigma_{xx} = \mathbf{A}'\mathbf{A}_x$, $\Sigma_{yy} = \mathbf{A}_y'\mathbf{A}_y$ and $\Sigma_{xy} = \mathbf{A}_x'\mathbf{A}_y$ from them. Numerically spoken, there are better ways to solve equations (7.4) and (7.5). We will start with the singular value decompositions

$$\mathbf{A}_x = \mathbf{U}_x\mathbf{D}_x\mathbf{V}_x' \tag{7.23}$$
$$\mathbf{A}_y = \mathbf{U}_y\mathbf{D}_y\mathbf{V}_y' \tag{7.24}$$

and use them to obtain the following covariance matrices

$$\begin{aligned}\Sigma_{xx} &= \mathbf{A}_x'\mathbf{A}_x = \mathbf{V}_x\mathbf{D}_x^2\mathbf{V}_x'\\ \Sigma_{yy} &= \mathbf{A}_y'\mathbf{A}_y = \mathbf{V}_y\mathbf{D}_y^2\mathbf{V}_y'\\ \Sigma_{xy} &= \mathbf{A}_x'\mathbf{A}_y = \mathbf{V}_x\mathbf{D}_x\mathbf{U}_x'\mathbf{U}_y\mathbf{D}_y\mathbf{V}_y',\end{aligned} \tag{7.25}$$

where we used the orthonormalities $\mathbf{U}_x'\mathbf{U}_x = \mathbf{I}$ and $\mathbf{U}_y'\mathbf{U}_y = \mathbf{I}$. We use these decompositions together with $\Sigma_{xx}^{-1} = \mathbf{V}_x\mathbf{D}_x^{-2}\mathbf{V}_x'$ to rewrite equation (7.4) as

$$(\mathbf{V}_y\mathbf{D}_y\mathbf{U}_y'\mathbf{U}_x\mathbf{U}_x'\mathbf{U}_y\mathbf{D}_y\mathbf{V}_y' - \rho^2\mathbf{V}_y\mathbf{D}_y^2\mathbf{V}_y')\mathbf{y} = \mathbf{0}, \tag{7.26}$$

where we used the orthonormalities $\mathbf{V}_x'\mathbf{V}_x = \mathbf{I}$ and $\mathbf{V}_y'\mathbf{V}_y = \mathbf{I}$. Next we multiply from the left with $\mathbf{D}_y^{-1}\mathbf{V}_y'$ and obtain

$$(\mathbf{U}_y'\mathbf{U}_x\mathbf{U}_x'\mathbf{U}_y\mathbf{D}_y\mathbf{V}_y' - \rho^2\mathbf{D}_y\mathbf{V}_y')\mathbf{y} = \mathbf{0}, \tag{7.27}$$

which can be rewritten as

$$((\mathbf{U}_x'\mathbf{U}_y)'(\mathbf{U}_x'\mathbf{U}_y) - \rho^2\mathbf{I})\mathbf{D}_y\mathbf{V}_y'\mathbf{y} = \mathbf{0}. \tag{7.28}$$

This equation is of the form $(\mathbf{A}'\mathbf{A} - \rho\mathbf{I})\mathbf{x} = \mathbf{0}$ which can be easily solved by the substitution of the singular value decomposition (svd) of $\mathbf{A}$. The svd of $\mathbf{U}_x'\mathbf{U}_y = \mathbf{U}\mathbf{D}\mathbf{V}'$ substituted in equation (7.28) leaves us after some rearrangement with

$$(\mathbf{D}^2 - \rho^2\mathbf{I})\mathbf{V}'\mathbf{D}_y\mathbf{V}_y'\mathbf{y} = \mathbf{0}, \tag{7.29}$$

where we used the orthonormalities $\mathbf{U}'\mathbf{U} = \mathbf{V}'\mathbf{V} = \mathbf{I}$. This equation has eigenvalues $\mathbf{D}^2$, and the eigenvectors can be obtained from the columns of $\mathbf{V}_y\mathbf{D}_y^{-1}\mathbf{V}$. In an analogous way we can reduce equation (7.5) to

$$(\mathbf{D}^2 - \rho^2\mathbf{I})\mathbf{U}'\mathbf{D}_x\mathbf{V}'_x\mathbf{x} = \mathbf{0}, \tag{7.30}$$

with the same eigenvalues $\mathbf{D}^2$. Analogously, the eigenvectors are obtained from the columns of $\mathbf{V}_x\mathbf{D}_x^{-1}\mathbf{U}$.

We have now shown that the algorithms above significantly reduce the number of matrix multiplications that are necessary to obtain the eigenvalues. Most importantly, we do not actually need to perform the matrix multiplications to obtain the covariance matrices in equations (7.25). We only need two singular value decompositions and one matrix multiplication $\mathbf{U}'_x\mathbf{U}_y$. The latter multiplication is numerically very stable because both matrices are column orthogonal.

7.2.2.3 Solution summary

We have shown two numerically stable procedures to solve the canonical correlation equations (7.4) and (7.5). In both procedures the data matrices $\mathbf{A}_x$ and $\mathbf{A}_y$ were considered as two separate matrices. The same description can be given if we use the *combined* $m \times (p + n)$ data matrix $\mathbf{A}_{y+x}$. In this matrix the first p columns equal $\mathbf{A}_y$ and the next n columns equal $\mathbf{A}_x$. Its covariance matrix can be decomposed as:

$$\Sigma_{y+x} = \mathbf{A}'_{y+x}\mathbf{A}_{y+x} = \begin{bmatrix} \Sigma_{yy} & \Sigma_{yx} \\ \Sigma_{xy} & \Sigma_{xx} \end{bmatrix}.$$

The problem has now been reformulated as obtaining correlations between two groups of variables within the *same* data set. This formulation has been adopted in the PRAAT program.

7.3 A canonical correlation analysis example

As an example we will use the data set of Pols et al. (1973). This data set is available as a `TableOfReal` object in the PRAAT program: the first three columns in the table contain the frequencies of the first three formants in Hertz and the next three columns contain the levels of the formants in decibel below the overall sound pressure level (SPL) of the measured vowel segment. There are $600 = 50 \times 12$ rows in this table. Because the levels are all given as positive numbers, a small number means a relatively high value for the bandfilter level while a large number means a relatively low value. To get an impression of this data set we have plotted in figure 7.1 the standardized logarithmically transformed first and second formant against each other. This figure is the standardized version of figure 3.3. In the next subsection more details about the transformation will be given.

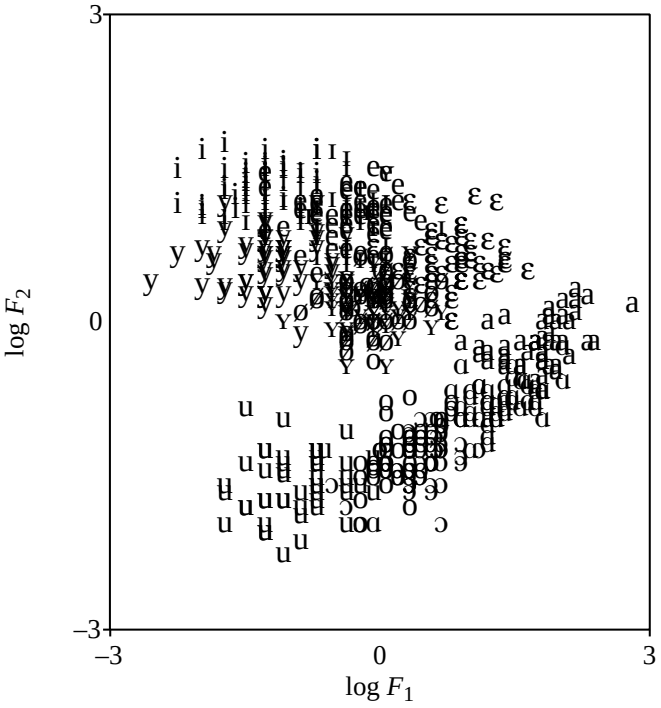

Figure 7.1. The standardized logarithmically transformed first and second formant frequencies of the Pols et al. (1973) data set.

7.3.1 Finding correlations between formant frequencies and levels

As an example of applying a canonical correlation analysis, we will try to find the canonical correlation between the three formant frequency values and the three levels. Instead of the frequency values in Hertz we will use logarithmic values and standardize all columns[1] (for each column separately: subtract the column average and divide by the standard deviation). Before we start the canonical correlation analysis we will first have a look at the *Pearson* correlations within this data set. These correlations are displayed in table 7.1. Actually we show two correlation matrices in the same table, making use of the fact that a correlation matrix is a symmetric matrix whose diagonal elements are all equal to 1. The upper triangular part shows the correlations for the formant frequencies in Hertz while the lower triangular part shows the correlations for the logarithmically transformed formant frequencies. The column labels belong to the upper triangular part while the row labels belong to the lower triangular part.

[1] The standardization is, strictly speaking, not necessary because correlation coefficients are invariant under standardization.

```
Create TableOfReal (Pols 1973)...  yes                ▷ F1,2,3 (Hz) and levels L1,2,3.
Formula...  if col < 4 then log10(self) else self endif   ▷ To log(F1,2,3).
Standardize columns
To Correlation
Confidence intervals...  0.95 0 Ruben                 ▷ Bonferroni correction.
```

Script 7.1. Calculating correlations and confidence intervals.

For example, the number −0.338 in the row labelled log F_1 and the column labelled F_2 represents the correlation between F_1 and F_2, while the number −0.302 in the row labelled log F_2 and the column labelled F_1 represents the correlation between log F_1 and log F_2. We clearly see in the table that the correlation pattern in the upper triangular part follows the pattern in the lower triangular part for the logarithmically transformed frequencies. To get an impression of the variability of these correlations, we have displayed in table 7.2 the confidence intervals at a confidence level of 0.95. We used Ruben's approximation for the calculation of the confidence intervals and applied a Bonferroni correction for the significance level (Johnson, 1998, page 39). Script 7.1 summarizes.

Table 7.1. Correlation coefficients for the Pols et al. (1973) data set. The entries in the lower triangular part are the correlations for the logarithmically transformed frequency values while the entries in the upper part are the correlations for frequency values in Hertz. The lower part reproduces the values found in table 3.1.

	F_1	F_2	F_3	L_1	L_2	L_3
log F_1		−0.338	0.191	0.384	−0.507	−0.014
log F_2	−0.302		0.190	−0.106	0.530	−0.568
log F_3	0.195	0.120		0.113	−0.036	0.019
L_1	0.370	−0.090	0.116		−0.042	0.085
L_2	−0.533	0.512	−0.044	−0.042		0.127
L_3	−0.021	−0.605	0.017	0.085	0.127	

The lower triangular part of table 7.1 in which the correlations of the logarithmically transformed formant frequency values are displayed, is reproduced from table 3.1. The correlation matrix shows that high correlations exist between some formant frequencies and some levels, as was already discussed in section 3.5.3.

To obtain the canonical correlations between the formant frequencies and formant levels we first let the PRAAT program construct a `CCA` object from the `TableOfReal` object. This object will next be queried for the canonical correlations. In the construction of the `CCA` object, the first three columns in the `TableOfReal` object, namely those that contain the formant frequencies, are associated with the matrix $\mathbf{A}_y$, and the last three columns, which contain the formant levels, are associated with the matrix $\mathbf{A}_x$. Then, the calculations as outlined in section 7.2.2.2 are used to determine the canonical correlations. Script 7.2 on the following page summarizes.

Table 7.2. Confidence intervals at a 0.95 confidence level of the correlation coefficients in the lower triangular part of table 7.1. Confidence intervals were determined by applying Ruben's approximation and a Bonferroni correction was applied to the confidence level. The upper and lower triangular part display the upper and lower value of the confidence interval, respectively. For example, the confidence interval for the −0.533 correlation between L_2 and $\log F_1$ is (−0.614, −0.442).

	$\log F_1$	$\log F_2$	$\log F_3$	L_1	L_2	L_3
$\log F_1$		−0.189	0.307	0.469	−0.442	0.099
$\log F_2$	−0.407		0.236	0.030	0.595	−0.522
$\log F_3$	0.077	0.001		0.232	0.076	0.136
L_1	0.262	−0.207	−0.004		0.078	0.203
L_2	−0.614	0.417	−0.162	−0.161		0.243
L_3	−0.140	−0.675	−0.103	−0.035	0.007	

```
select TableOfReal pols_50males                ▷ The log(F) values.
To CCA... 3                                    ▷ We have 3 dependent variables.
Get correlation... 1
Get correlation... 2
Get correlation... 3
```

Script 7.2. Canonical correlation analysis.

In table 7.3 we show the canonical correlations together with the eigenvector loadings on the variables. The eigenvectors belonging to the first and the second canonical correlation have also been drawn in figure 7.2 with a solid line and a dotted line, respectively. In this figure the plot on the left shows the weighting of the frequencies.

Table 7.3. The canonical correlations between formant frequencies and formant levels and their corresponding eigenvectors.

	ρ	$\log F_1$	$\log F_2$	$\log F_3$	L_1	L_2	L_3
1	0.867	−0.187	0.971	−0.148	−0.092	0.714	−0.694
2	0.545	0.891	0.443	−0.099	0.646	−0.428	−0.632
3	0.072	0.166	0.017	−0.986	−0.788	−0.530	−0.313

We see that for the first eigenvector most of the weight is put on $\log F_2$, and that the other two frequencies are barely weighted. On the other hand, for the weighting of the levels, the first eigenvector shows approximately equal weighting of the second and third level (in an absolute sense). This is confirmed by the data in table 7.1, which show a high correlation, 0.512, between $\log F_2$ and L_2 and the highest correlation,

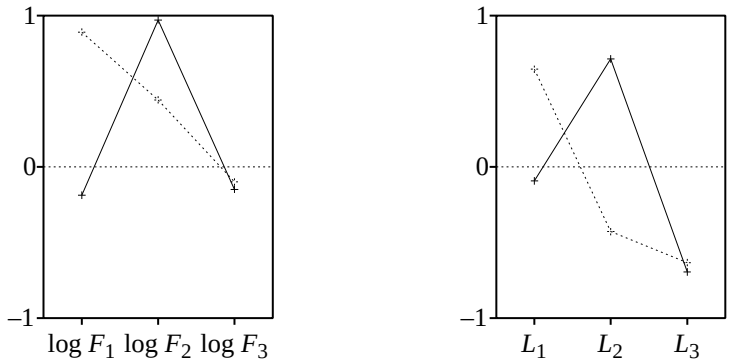

Figure 7.2. The eigenvectors corresponding to the first (solid line) and the second canonical correlation (dotted line).

-0.605, between $\log F_2$ and L_3. Table 7.3 indicates that the weightings of L_2 and L_3 in the first eigenvector are even larger than in table 7.1.

7.3.2 Using the correlations for prediction

The outcome of the canonical correlation analysis on the Pols et al. data set was three canonical correlations, ρ_i, with their associated eigenvectors $\mathbf{x}_i$ (levels) and $\mathbf{y}_i$ (frequencies). These eigenvectors can be used to construct the scores (canonical variates) $\mathbf{z}_y$ and $\mathbf{z}_x$ by projecting the data matrices on the eigenvectors, as was shown in equations (7.7) and (7.6), respectively. In figure 7.3 we have drawn a scatter plot of the first canonical variates. The straight line shows the empirical relation $y_1 = 0.867x_1$ for the first canonical correlation. We note two separate clusters, one for the back vowels and another for the front vowels. The main ordering principle in the figure is from front to back, as can also be seen from the first eigenvector for the formants in figure 7.2, which is dominated by the second formant frequency. The linear part of the relation between these canonical variables can be exploited by predicting one from the other. In the following we will try to predict formant frequency values from formant levels. We start with the equations for the canonical variates and write

$$\mathbf{z}_{y,i} = \rho_i \mathbf{z}_{x,i}, \text{ for } i = 1, 2 \text{ and } 3. \tag{7.31}$$

These three equations show the optimal linear relation between a linear combination of formant frequencies and formant levels, the $\mathbf{z}_y$ and the $\mathbf{z}_x$, respectively. Equation (7.31) could also be interpreted as a prescription to determine the $\mathbf{z}_y$when only the $\mathbf{z}_x$are given. In the equation above the vectors $\mathbf{z}$ are three-dimensional. For every element j of the vectors $\mathbf{z}$, we can substitute back the original variables and obtain the

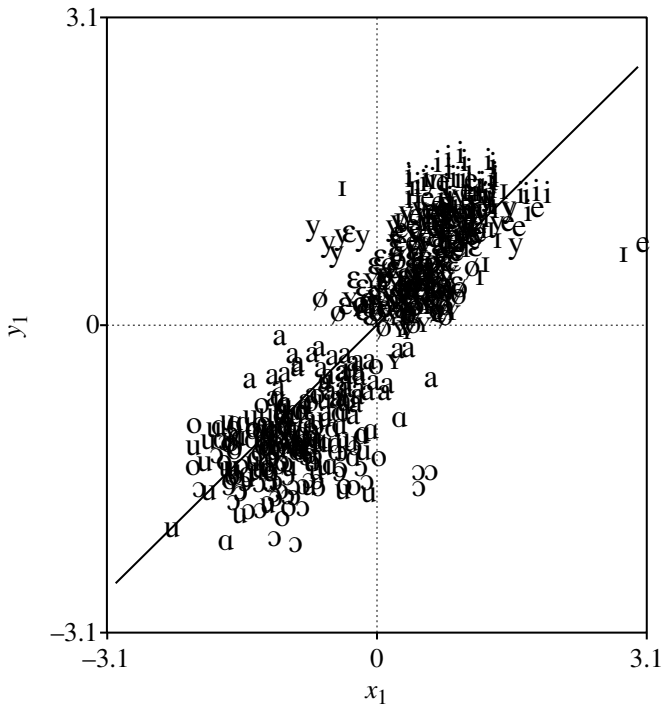

Figure 7.3. A scatter plot of the first canonical variates. The straight line shows the canonical correlation relation $y_1 = \rho_1 x_1$, where $\rho_1 = 0.867$.

following equation

$$\mathbf{Yf} = \mathbf{D}(\rho_i)\mathbf{Xl}, \tag{7.32}$$

where $\mathbf{f}$ and $\mathbf{l}$ are the vectors with the three formant frequencies and levels, respectively, and $\mathbf{D}(\rho_i)$ is a diagonal matrix. The $\mathbf{Y}$ and $\mathbf{X}$ are the eigenvectors. Now, because the $\mathbf{Y}$ are orthonormal, we can write the solution as

$$\mathbf{f} = \mathbf{Y}'\mathbf{D}(\rho_i)\mathbf{Xl}. \tag{7.33}$$

When we train a discriminant classifier with the standardized formant frequency values and use the same set we used for the training as input for the classifier, we obtain 73.9% correct classification with the 12-label set (discriminant analysis with the PRAAT program has been covered in chapter 3). When we use the formant *levels* to predict the formant *frequencies* and subsequently use these predicted formant frequencies as input to the classifier, we get 26% correct classification. Using only formant levels for discriminant classification gives 37.2% correct. Both classifications are above chance (8.5%). The following script summarizes.

```
select TableOfReal pols_50males
plus CCA pols_50males
Predict...  4                                          ▷ Start column is 4.
Select columns where row...  "1 2 3" 1                 ▷ Select only F1, F2, F3.
Rename...  f123
To Discriminant                                        ▷ Train the classifier.
plus TableOfReal f123
To ClassificationTable...  y y                         ▷ Use linear discriminant.
To Confusion                                           ▷ Get the confusion matrix.
fc = Get fraction correct
```

Script 7.3. Prediction from canonical correlations.

7.4 Principal components and auto-associative neural nets

7.4.1 Introduction

In this section we try to use canonical correlation analysis to demonstrate that appropriately chosen neural nets can also perform principal component analysis. We will do so by comparing the output from an auto-associative neural net with the output of a principal component analysis by means of canonical correlation analysis. As test data set we will use only the three formant frequency values from the Pols et al. data set. In order to make the demonstration not completely trivial we compare two-dimensional representations. This means that in both cases some data reduction must take place.

7.4.2 The auto-associative neural net

An auto-associative neural net is a supervised neural net where each input is mapped onto itself. We will use here the supervised feedforward neural net as is implemented in the PRAAT program. Auto-associativity in these nets can best be accomplished by making the output units linear[2] and the number of dimensions of the input and output layer must be equal. The trivial auto-associative net has no hidden layers and maps its input straight onto its output. Interesting things happen when we compress the input data by forcing them through a hidden layer with fewer units than the input layer. In this way the neural net has to learn some form of data reduction. This reduction must probably be some way of principal component analysis in order to maintain as much variation as possible in the transformation from input layer to output layer.

Since our input data is three-dimensional, the number of input and output nodes for the neural network is already fixed and the only freedom in the topology that is left is the number of hidden layers and the number of nodes in each hidden layer. To keep the comparison as simple as possible, we will use only one hidden layer in this task, with two nodes in this layer. The resulting topology for the supervised feedforward

[2]This linearity is only for the output nodes, the hidden nodes still have the sigmoid non-linearity.

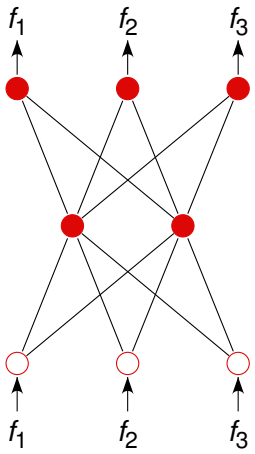

Figure 7.4. Topology of the supervised auto-associative feedforward neural net used for learning the associations between logarithmically scaled formant frequency values.

neural net is a (3,2,3) topology, i.e. three input nodes, two hidden nodes and three output nodes. A network with this topology has only 17 adaptable weights: nine weights for the output layer and eight weights for the hidden layer. The topology of this network is displayed in figure 7.4.

In the training phase we try to adjust the weights of the network in such a way that when we propagate an input through the neural net, the output activation of the neural net will equal the input. Of course, this is not always possible for all inputs and therefore we try to make them as close as possible on average. Closeness is then mathematically defined as a minimum squared error criterion.

7.4.3 Data preprocessing

In order to guarantee proper training we have to arrange for all inputs to be in the interval (0, 1). We have scaled all formant frequency values as

$$f_i = \log \frac{F_i}{(2i-1)500} + 0.5, \text{ for } i = 1, 2 \text{ and } 3. \tag{7.34}$$

In this formula formant frequencies F_i in Hertz are first scaled with respect to the resonance frequencies of a straight tube which are at frequencies of $(2i-1)500$ Hz.

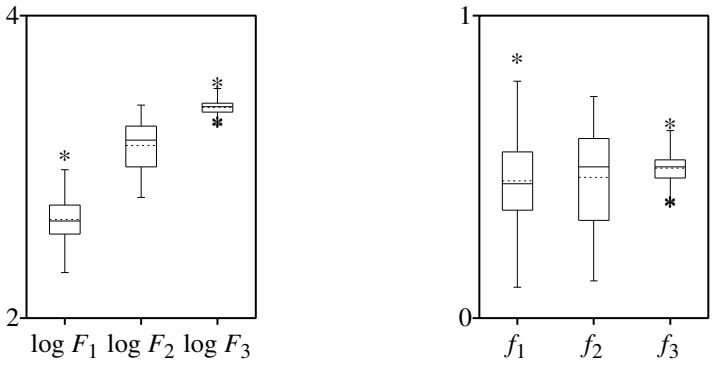

Figure 7.5. Box plots before (left) and after (right) scaling the logarithmically transformed frequency values. The f_i are scaled to the interval (0, 1) according to equation (7.34). The dotted lines in the box plots indicate the mean values.

Next the logarithm of this fraction is taken.[3] Since the logarithm of this fraction can take on negative values we add the factor 0.5 to make the number positive. This formula is very similar to formula (5.18), we find the former more "attractive" because the scalings involved, i.e. $(2i-1)500$, are the resonance frequencies of a straight tube, whereas in equation (5.18) the minimum and maximum values where chosen ad hoc.

To show the effect of this scaling we have drawn in figure 7.5 box plots of the data before and after the scaling. A "box plot", or more descriptively a "box-and-whiskers plot", provides a graphical summary of data. The box is marked by three solid horizontal lines which, from bottom to top, indicate the position of the first, second and third quartile. The box height therefore covers 50% of the data (the line of the second quartile is also called the *median*). In the PRAAT version of the box plot, the box has been extended with a dotted line that marks the position of the mean. The lengths of the vertical lines, the "whiskers", show the largest/smallest observation that falls within 1.5 times the box height from the nearest horizontal line of the box. If any observations fall farther away, the additional points are considered outliers and are shown separately.

Besides scaling the values to the (0, 1) interval we also note that the locations of the scaled formant frequency values have become more equalized. The following script summarizes the scaling.

[3] It is not strictly necessary to take the logarithm. The scaling with the corresponding odd multiple of 500 Hz for each formant is already sufficient to move all values into the interval (0.4,2.2]. Subsequently dividing by a factor somewhat greater than 2.2 would yield numbers in the (0,1) interval. Taking an extra logarithm, however, achieves a somewhat better clustering. A discriminant classification with identical training set and test set shows 73.9% correct for the logarithmic scaling, as was already shown in section 7.3.2, versus 72.8% for the alternative scaling discussed in this footnote.

```
Create TableOfReal (Pols 1973)...  no                    ▷ Only frequencies, no levels.
Formula...  log10 (self / ((2*col-1)*500)) + 0.5         ▷ Equation (7.34).
```

Script 7.4. Scaling of the formant frequencies to the (0, 1) interval.

7.4.4 Training the neural net

After preprocessing the data we have a table in which all elements are within the (0, 1) interval. We duplicate this table and cast the two resulting objects to a `Pattern` object and an `Activation` object, respectively. These two objects function as the input and output for the auto-associative feedforward neural net. The next step is then to create a neural net of the right topology, select the input and the output objects and start learning. Preliminary testing showed that 500 learning epochs were sufficient for learning these input-output relations.

Because the learning process uses a minimization algorithm that starts the minimization with random weights, there always is the possibility that learning gets stuck in a local minimum. We cannot avoid these local minima. However, by repeating the minimization process a large number of times, each time with different random initial weights, we hope to find acceptable learning in some of these trials. We therefore repeated the learning process 1000 times and each time used different random initial weights. The repeated learning only took 27 minutes of cpu-time on a computer with a 500-MHz processor. It turned out that after these 1000 learning sessions all the obtained minima were very close to each other. The distribution of the minima in this collection of 1000 was such that the absolute minimum was 0.5572, the 50% point (median) was at 0.5575 and the 90% point at 0.5580. If we consider that the training data matrix had 600 rows (replications) and each row is a three-dimensional vector with values in the interval (0, 1) and this minimum is the sum of all the squared errors on the test set, then excellent learning has taken place. Script 7.5 summarizes the learning process.

7.4.5 The comparison

Now that the best association between the three-dimensional outputs and inputs by means of two hidden nodes has been learned by the neural net, we want to compare this mapping with the results of a two-dimensional principal component analysis. We want to obtain the representation of all the inputs at the two nodes of the hidden layer. This can be done by presenting an input to the trained neural net, let the input propagate to the first hidden layer and then record the activation of the nodes in this layer. The input to the neural net will therefore be a 600×3 table and the output will be the activation at the hidden layer, a table of dimension 600×2. Script 7.6 summarizes.

The mapping to the principal component plane of the scaled data is simple to obtain. See chapter 3 for more information on principal component analysis. The first two principal components explain 95.8% of the variance. Script 7.7 summarizes.

To get more insight in the results of the two different analyses we have plotted in figure 7.6 the neural net and principal component representations of the formant data

```
min_global= 1e30                                          ▷ Initialize to some large value.
Create Feedforward Net...  3_2_3 3 3 2 0 y                ▷ Topology (3, 2, 3).
for i to 1000
   select FFNet 3_2_3
   Reset...  0.1                                          ▷ All weights random uniform in [-0.1, 0.1].
   plus Activation pols_50males
   plus Pattern pols_50males
   Learn (SM)...  500 1e-10 minimum squared error         ▷ 500 epochs.
   select FFNet 3_2_3
   min = Get minimum
   if min < min_global
      min_global = min
      Write to short text file...  3_2_3                  ▷ Save FFNet object to disk.
   endif
endfor
```

Script 7.5. Training the neural net.

```
select FFNet FFNetmin                                     ▷ Select the trained neural net
plus Pattern pols_50males                                 ▷ + the input.
To Activation...  1                                       ▷ Layer 1 is the hidden layer.
```

Script 7.6. Get activation at hidden layer.

```
Create TableOfReal (Pols 1973)...  no                     ▷ No levels.
Formula...  log10 (self / ((2*col-1)*500)) + 0.5
To PCA                                                    ▷ Principal Component Analysis.
vaf = Get fraction variance accounted for...  1 2
plus TableOfReal pols_50males
To Configuration...  2                                    ▷ The 2-dimensional mapping.
```

Script 7.7. Mapping to the principal component plane.

preprocessed according to equation (7.34). The figure on the left shows the representation in the hidden layer, the figure on the right displays the data in the principal component plane. Closer inspection shows that after reflecting the second plot around the pc1-axis both representations will look very similar. When we compare them to figure 7.1, we notice a great resemblance, which shows that predominantly only the first two formant frequencies contribute to the representations in figure 7.6.

We can now combine the two representations in one 600×4 data matrix and calculate the correlations between the columns of this matrix. The correlation coefficients are shown in the upper diagonal part in table 7.4. Script 7.8 summarizes.

As for the principal components, the table confirms that the correlation coefficient between the first and the second principal component is zero, as it must be, since the whole purpose of principal component analysis is to remove correlations between dimensions. The representations at the two hidden nodes are not independent from

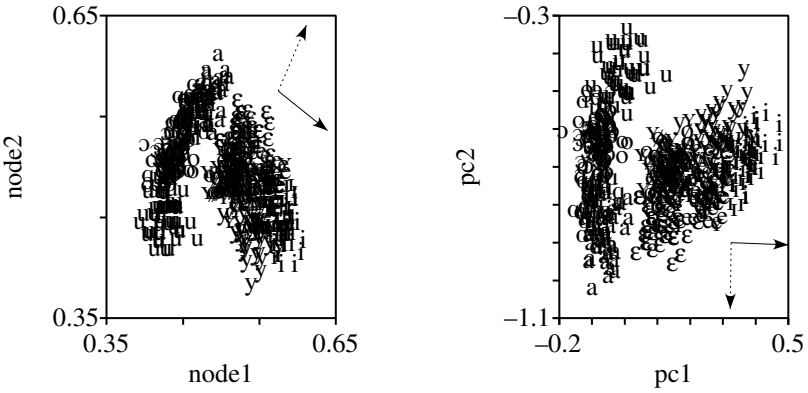

Figure 7.6. Two different representations of the formant frequency data scaled according to equation (7.34). Left: the representation at the hidden layer of the neural net of figure 7.1 with topology (3, 2, 3). Right: the plane with the first two principal components. The solid and dotted arrows are data taken from table 7.5 and indicate the directions of the eigenvectors for the first and second canonical correlation, respectively.

```
select TableOfReal hidden
plus TableOfReal pca
Append columns                    ▷ Now 2 times 2 columns → 4 columns.
Rename... hidden_pca
To Correlation
```

Script 7.8. Correlations between the hidden layer and the principal component representations.

each other, as the (negative) correlation coefficient between node 1 and node 2 shows. Substantial correlations exist between the two neural dimensions and the principal component dimensions. However, the two plots in figure 7.6 suggest that there is more correlation than is shown in the table. This is where a canonical correlation analysis can be useful. The results of the canonical correlation analysis between the two-dimensional representation at the hidden nodes and the two-dimensional principal component representation are displayed in table 7.5. Besides canonical correlation coefficients, the table also shows the eigenvectors. Additionally, the eigenvectors are graphically displayed in figure 7.6 with arrows. The two arrows in the left and the right plot, drawn with a solid line, are the directions of maximum correlation between the two representations: when we project the 600 two-dimensional data points on these directions, the resulting two 600-dimensional data vectors have the maximum obtainable canonical correlation coefficient of 1.000. The second coefficient also equals 1, rounded to three digits of precision. The corresponding eigenvectors are drawn as the

Table 7.4. The correlation coefficients for the combined representations of formant frequencies at the hidden nodes of a neural network and principal components. The lower diagonal part contains the correlations after a Procrustes similarity transform on the hidden nodes representation (see section 7.4.6). For clarity, the diagonal has been left out.

	node1	node2	pc1	pc2
node1′		−0.363	0.927	−0.376
node2′	−0.055		−0.686	−0.727
pc1	1.000	−0.029		0.000
pc2	−0.025	1.000	0.000	

Table 7.5. Characteristics of the canonical correlation analysis between the two-dimensional representation of formant frequencies at the hidden nodes of a neural network and the two principal components. Canonical correlation coefficients and corresponding pairs of eigenvectors are shown.

	ρ	node1	node2	pc1	pc2
1	1.000	0.854	−0.520	0.999	−0.033
2	1.000	0.488	0.873	−0.017	−1.000

```
select TableOfReal hidden_pca                    ▷ 4 columns.
To CCA... 2                                      ▷ 2 dependent variables.
plus TableOfReal hidden_pca
To TableOfReal (scores)... 2
```

Script 7.9. Get canonical variates (scores).

arrows with a dotted line. This shows that the neural net performs a PCA analysis. In figure 7.7 we have plotted the canonical variates (scores) for this analysis. Script 7.9 summarizes.

We see from the plots in figure 7.7 a nice agreement between the scatter plots of the neural net scores on the left and the principal component scores on the right. However, we note from figure 7.6 that the two eigenvectors **y** in the plot on the left are not mutually orthogonal because the angle between the two eigenvectors is not 90°. The same applies to the two eigenvectors **x**, they are not orthogonal either (although harder to see in the figure, the numbers in table 7.5 are convincing: the inproduct of the two eigenvectors **x** is $0.999 \cdot (-0.017) + (-0.033) \cdot (-1.000)$ which equals 0.016017.). This is a characteristic of equations like (7.4) and (7.5): in general these equations do not have eigenvectors that are orthogonal. Because the scores (canonical variates) are obtained by a projection of the original data set on the eigenvectors of the canonical

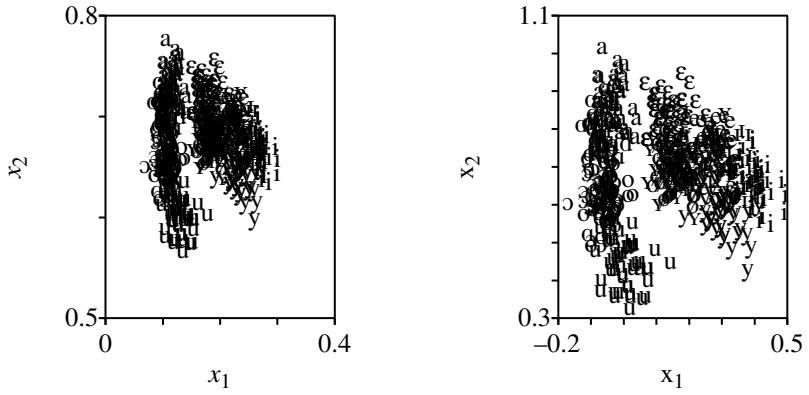

Figure 7.7. Scatter plots of canonical variates for the dependent (left) and the independent data set (right). The dependent and independent data sets are the neural net data and the principal component data set, respectively.

correlation analysis, the resulting scatter plots will show a somewhat distorted map of the original data. This is in contrast with principal component analysis, where the eigenvectors are orthogonal and therefore the new principal dimensions are a mere rotation of the original dimensions. This means that a principal component analysis does not change the structure of the data set and relative distances between the points in the data set are preserved. In the mapping to the canonical variate space, the structure of the data set is not preserved and the relative distances have changed.

7.4.6 Procrustes transform

It is possible, however, to transform one data set to match another data set, as closely as possible, such that the structure of the transformed data set is preserved. A transformation that preserves the structure of a data set leaves all relative distances between the data points intact. The only admissible operations that preserve structure are dilation, translation, rotation and reflection. This *similarity* transformation which is also implemented in the PRAAT program, is called a *Procrustes transform* and we can write the equation that governs the transformation of data set $\mathbf{X}$ into $\mathbf{Y}$ as follows:

$$\mathbf{Y} = s\mathbf{X}\mathbf{T} + \mathbf{1}\mathbf{t}'. \tag{7.35}$$

In this equation s is the dilation or scale factor, $\mathbf{T}$ is an orthonormal matrix that incorporates both rotation and reflection, $\mathbf{t}'$ is the translation vector, and $\mathbf{1}$ is a vector of ones. Given data sets $\mathbf{X}$ and $\mathbf{Y}$, a Procrustes analysis yields the parameters for s, $\mathbf{t}$ and $\mathbf{T}$. The total number of parameters for a Procrustes transform in n-dimensional space is: $n(n-1)/2$ for the rotation and reflection matrix, n for the translation vector and 1 for the scale factor.

The equation above transforms **X** into **Y**. The inverse, the one that transforms **Y** into **X**, can easily be deduced from equation (7.35) and is:

$$\mathbf{X} = \frac{1}{s}(\mathbf{Y} - \mathbf{1t}')\mathbf{T}'. \tag{7.36}$$

More details of the Procrustes transform and the analysis can be found in Borg & Groenen (1997). In figure 7.8 we show the result of a Procrustes analysis on the neural net and the principal component data sets. The plot on the left is the Procrustes

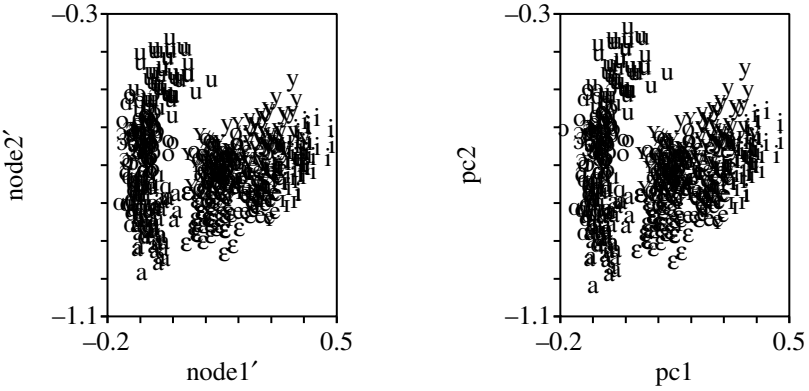

Figure 7.8. Scatter plots of the Procrustes-transformed neural net representation (left) and the principal component representation (right). The plot on the left is obtained from the left plot in figure 7.6 by a clockwise rotation of 31°, followed by a reflection around the horizontal axis, a scaling by a factor 2.98 and a translation with the vector (-0.42, 1.35). The plot on the right is only for comparison and shows the same data as the plot on the right in figure 7.6.

transform of the neural net data set and was obtained from the plot in figure 7.6 by a clockwise rotation with an angle of approximately 31°, followed by a reflection around the horizontal axis, a scaling by a factor 2.98 and a translation with the vector (-0.42, 1.35). The parameters for this transform were obtained from matching the two-dimensional neural net data set with the two-dimensional principal component data set. The two plots now look very similar. In table 7.4 we show in the lower diagonal the correlation coefficients between the Procrustes-transformed neural net data set and the principal component data set. These correlations were also obtained, in a manner analogous to the data in the upper diagonal part, by appending columns into a combined data set. The table shows that the correlation between node1′ and pc1 is 1.000 and that the correlation between node2′ and pc2 also equals 1.000. Script 7.10 summarizes.

Table 7.4 shows the number −0.363 for the correlation coefficient of node1 and node2, and the number −0.055 for the correlation coefficient of node1′ and node2′.

```
select Configuration pca
plus Configuration hidden
To Procrustes
plus Configuration hidden
To Configuration                                    ▷ Apply Procrustes.
Rename... hiddenp
To TableOfReal
plus TableOfReal pca
Append columns                                      ▷ Combine the two tables.
To Correlation
```

Script 7.10. Correlation of Procrustes-transformed data with principal components.

This shows that node1′ and node2′ have become less correlated as compared to node1 and node2, making these new dimensions more independent from each other. The Procrustus transform did not touch pc1 and pc2 and therefore they stay uncorrelated as the two numbers 0.000 in the row and the column labelled pc2 show. And, finally, the correlations between node1′ and pc1 and, especially, between node2′ and pc2 have increased and are almost perfect now.

7.4.7 Summary

All the data presentations in the preceding sections have shown that there is a great amount of similarity between the internal representation of an auto-associative neural net and a principal component analysis for the Pols et al. formant frequency data set. Although the presentation in these sections constitutes no formal proof and was only used as a demonstration of some of the methods available in the PRAAT program, we hope that it has been made plausible that auto-associative neural nets and principal components bear a lot in common.

7.5 Discussion

We have shown that the canonical correlation analysis can be a useful tool for investigating relationships between two representations of the same objects. Although the mathematical description of the analysis that has been given in this chapter can be considered as a *classical* analysis, the results can also be used with modern robust statistics and data reduction techniques. These modern techniques are more robust against outliers. Essential to these modern techniques is a robust determination of the covariance matrix and the associated mean values (Dehon, Filzmoser & Croux, 2000). The description we have given in section 7.2.2.1 does not prescribe how a covariance matrix is obtained and could therefore be used with these modern techniques. Canonical correlation analysis will be used in section 9.3.2, where we will investigate the relation between fundamental frequency and bandfilter values.

Chapter 8

Accessing the TIMIT acoustic phonetic speech corpus[‡]

Abstract

In this chapter we introduce the TIMIT acoustic-phonetic American-English speech corpus and we will describe how we have made the corpus accessible for analysis for the computer program PRAAT. We will further present some basic statistics about this corpus. The TIMIT corpus gives us the possibility to test various speaker-adaptive vowel normalization procedures, because it contains labelled speech material from 630 different speakers, of which 438 are male and 192 are female.

[‡]This chapter is a modified version of Weenink (1996).

8.1 Introduction

The TIMIT acoustic-phonetic continuous speech corpus (Lamel et al., 1986) resulted from the joint efforts of several American speech research sites. The text corpus design was done by the Massachusetts Institute of Technology (MIT), Stanford Research Institute and Texas Instruments (TI). The speech was recorded at TI, transcribed at MIT, and has been maintained, verified and prepared for CDROM production by the American National Institute of Standards and Technology (NIST), and was made available via the Linguistic Data Consortium.

The TIMIT corpus contains a total of 6300 sentences. Each of 630 speakers from eight major dialect regions of the United States of America spoke ten sentences. Approximately 70% of the speakers were male and 30% were female. The speaker's dialect region was defined as that geographical area were s/he had lived during childhood years. Dialect number 8 (Army Brat) was assigned to people who had moved around a lot during their childhood and to whom no particular dialect could be attributed. Table 8.1 shows the dialect distribution of the male and female speakers for the total and the training part of the database (see below).

The ten sentences produced by each speaker consisted of two so-called SA-type, five SX-type and three SI-type sentences.

The *SA sentences* are dialect sentences and were meant to expose the dialectal variants of the speakers. Only two different SA-type sentences were designed, `sa1` and `sa2`, and they were spoken by all 630 speakers. The `sa1` sentence is "She had your dark

Table 8.1. TIMIT dialect distribution of speakers. The first column contains the dialect number, followed by the geographical region. The third, fourth and fifth columns contain the number of male speakers, female speakers and the total number of recorded speakers in the corresponding dialect region, respectively. The last two columns contain the number of male and female speakers in the `train` part of the database (see text for an explanation). The last row contains the column sums.

		Training + Test			Training	
Dialect	Region	Male	Female	Total	Male	Female
1	New England	31	18	49	24	14
2	Northern	71	31	102	53	23
3	North Midland	79	23	102	56	20
4	South Midland	69	31	100	53	15
5	Southern	62	36	98	45	25
6	New York City	30	16	46	22	13
7	Western	74	26	100	59	18
8	Army Brat	22	11	33	14	8
		438	192	630	326	136

suit in greasy wash water all year" and the `sa2` sentence is "Don't ask me to carry an oily rag like that".

The *SX sentences* are phonetically-compact sentences and were designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest. There were 450 different phonetically compact SX-type sentences, and, given that each speaker produced five of these sentences, each SX sentences was reproduced by seven speakers ($= 630 \times 5/450$). An example of an SX-type sentence is `sx217`: "How permanent are their records?".

The phonetically-diverse *SI sentences* were selected so as to add diversity in sentence types and phonetic contexts. The selection criteria maximized the variety of allophonic contexts found in the texts. Each speaker reproduced three unique utterances out of the 1890 different phonetically diverse SI-type sentences. An example of such an SI-sentence is `si1027`: "Even then, if she took one step forward he could catch her". Table 8.2 gives a resumé.

Table 8.2. Overview of type of sentences in relation to the speakers, train and test part of the database taken together.

Type	#Sentences	#Speakers/Sentence	Total	#Sentences/Speaker
sa (dialect)	2	630	1260	2
sx (compact)	450	7	3150	5
si (diverse)	1890	1	1890	3
	2342		6300	10

The speech material was divided into a `train` set and a `test` set that contain 462 and 168 speakers, respectively, according to the following criteria:

- Roughly 20 to 30% of the corpus should be used for testing purposes, leaving the remaining 70 to 80% for training.
- No speaker should appear in both the training and testing portions.
- All the dialect regions should be represented in both subsets, with at least one male and one female speaker from each dialect.
- The amount of overlap of text material in the two subsets should be minimized; if possible no texts should be identical.
- All the phonemes should be covered in the test material; preferably each phoneme should occur multiple times in different contexts.

Besides the speech sound recordings, the TIMIT disk contains documentation, a pronouncing dictionary with 6229 entries and *transcriptions and segmentations of all 6300 recorded sentences* at the sentence level, the word level and the phoneme level, all done by hand. This results in the following impressive numbers: the TIMIT disk

contains 6300 sentences, these sentences contain 54,387 words and all these words contain 241,225 labelled segments.[1]

The pronouncing dictionary lists all 6229 different words used in the speech corpus with their 'standard' phonetic transcription and word stress markers. The TIMIT phonetic transcription alphabet was inspired by ARPABET but is not quite the same. TIMIT uses the two-character ARPABET code for vowels. However, different labels were used for the closure and release part of plosive sounds (Zue & Seneff, 1988). In table 8.3 we show the translation between TIMIT symbols and International Phonetic Association (IPA) phonetic symbols for the vowels in the database.

Table 8.3. Translation table of TIMIT symbols to IPA symbols for vowels only. The first column shows the TIMIT symbol, the second column shows the corresponding IPA-symbol. The vowel *sounds* like the vowel in the word given in the third column. The fourth column shows the TIMIT labeling of the word in the preceding column. The last column indicates the average duration of the vowel in seconds.

TIMIT	IPA	Example	TIMIT labeling of word	Duration (s)
iy	i	b**ee**t	bcl b iy tcl t	0.090
ih	ɪ	b**i**t	bcl b ih tcl t	0.079
eh	ɛ	b**e**t	bcl b eh tcl t	0.091
ey	e	b**ai**t	bcl b ey tcl t	0.128
ae	æ	b**a**t	bcl b ae tcl t	0.149
aa	ɑ	b**o**tt	bcl b aa tcl t	0.123
aw	aʊ	b**ou**t	bcl b aw tcl t	0.163
ay	aɪ	b**i**te	bcl b ay tcl t	0.145
ah	ʌ	b**u**t	bcl b ah tcl t	0.089
ao	ɔ	b**ou**ght	bcl b ao tcl t	0.124
oy	ɔɪ	b**oy**	bcl b oy	0.162
ow	o	b**oa**t	bcl b ow tcl t	0.127
uh	ʊ	b**oo**k	bcl b uh kcl k	0.076
uw	u	b**oo**t	bcl b uw tcl t	0.076
ux	ü	t**oo**t	tcl t ux tcl t	0.108
er	ɝ	b**ir**d	bcl b er dcl d	0.118
ax	ə	**a**bout	ax bcl b aw tcl t	0.049
ix	ɪ	deb**i**t	dcl d eh bcl b ix tcl t	0.052
axr	ɚ	butt**er**	bcl b ah dx axr	0.082
ax-h	ə̥	s**u**spect	s ax-h s pcl p eh kcl k tcl t	0.034

[1] When we subtract the 12,600 (= 6300 × 2) begin and end markers 'h#' we are left with 228,625 labelled phonemes.

8.2 File formats

The speech and associated data is organized on the CDROM according to the following file path hierarchy:

`timit/<USAGE>/<DIALECT>/<SEX><SPEAKER>/<SENTENCE>.<FILETYPE>`

The top level directory is named `timit` and the other symbols in the path may obtain the following values:

`<USAGE> := (train|test)`
`<DIALECT> := (dr1|dr2|dr3|dr4|dr5|dr6|dr7|dr8)`
`<SEX> := (m|f)` male or female
`<SPEAKER> := xyzd` (three characters followed by a digit)
`<SENTENCE> := (sa|si|sx)n` (sentence number `n`)
`<FILETYPE> := (wav|txt|wrd|phn)`

For example, file `timit/train/dr3/fpaz0/si2223.phn` refers to the transcription at the phoneme level for sentence `si2223` from the *female* speaker with initials `paz0` from the *North Midland* dialect `dr3` in the `train` part of the database.

8.2.1 Audio files

All 6300 recorded sentences are in separate binary files with extension `.wav`. These binary files with audio data have a special format by which they can be recognized and subsequently read by a specialized computer program that does not need external knowledge. In the binary files, audio data is preceded by information about the data according to a prescribed format, the so called SPHERE header which is 1024 bytes long. We will henceforth call these files *NIST sound files*.[2] Each NIST sound file always starts with a standard sequence of 16 characters by which they can be identified: `NIST_1A\n\s\s\s1024\n` where `\n` and `\s` stand for the ASCII newline and space character, respectively. The rest of the header contains, among other things, information about the number of audio channels, the number of samples per channel, the number of bytes per sample, the sampling rate and the type of data compression. This information is sufficient to read the speech data that follow the header.

8.2.2 Label files

Unfortunately the accompanying description label files do not contain enough information to be self-contained and cannot be read without additional knowledge. Their only identification is that they carry the same filename as the sound file they were derived from and have different extensions like `.txt`, `.wrd` or `.phn`. They all are ASCII text files and describe the recording at an ever increasing level of detail. Each file contains one or more lines, each line has three items that are separated by spaces. Each line starts with two sample numbers followed by a string. These two numbers, with the second number always larger than the first, mark a segment. Information about the

[2] NIST or the National Institute of Standards and Technology is the American non-regulatory federal agency that also coordinates the yearly DARPA Automatic Speech Recognition evaluations.

duration of the segment can only be calculated with information about the sampling frequency. Sampling frequency information is not given in the label files. Files with extension `.txt` contain only one line with the text of the sentence in the string, files with the extension `.wrd` contain a line for each word in the sentence and files with the extension `.phn` contain one line for each phonetic symbol that occurs. A `.phn` file always starts and ends with the symbols `h#`.

Although a robust TIMIT label file recognizer cannot be build, we have nevertheless exploited the structure of the TIMIT label files to program a *heuristic* "TIMIT label file recognizer" into the PRAAT program. Algorithm 8.1 incorporates the heuristics that we have used.

if ((first two lines each show two numbers followed by a string) **and**
(number$_1 \geq 0$) **and** (number$_2$ > number$_1$) **and**
(number$_3 \geq$ number$_2$) **and** (number$_4$ > number$_3$) **and**
(((string$_1$ == 'h#') **and** is_phonetic_label (string$_2$)) **or**
(is_lowercase(string$_1$) **and** is_lowercase(string$_2$))))
then
file is TIMIT label file
end if

Algorithm 8.1. TIMIT label file recognizer.

8.3 Accessibility of the material

In order to be able to visualize the speech corpus and (part of) the associated data we decided to extend the general speech analysis computer program PRAAT (Boersma & Weenink, 2006) with the possibility to recognize and read the audio files and phonetic label files and make them accessible in the program.

As was described above, the recognition of a NIST audio file is simple, since it always starts with the same identification string. Reading the data is also straightforward since all information for the interpretation of the data is contained within the header in a fixed format.

Although the NIST audio files in the TIMIT database are not in a compressed format, we nevertheless incorporated A-law and μ-law and embedded-shorten decompression algorithms in our PRAAT program.[3] In this way the PRAAT program can also read other databases that use audio files in NIST format such as the Dutch Polyphone Database (Den Os, Boogaart, Boves & Klabbers, 1995), the Groningen corpus (Sulter & Schutte, 1994) and the Translanguage English Database (Lamel et al., 1994).

All the audio files on the TIMIT CDROM have a sampling frequeny of 16 kHz and are quantized with 16 bits per sample, although the actual resolution of many files is not better than approximately 12 bits per sample.

[3] The shorten audio compressor has been removed recently because we obtained no permission to distribute its modified source code.

The TIMIT label files with extensions `.wrd` and `.phn` can be recognized by PRAAT's label file recognizer and then read from file into a `TextGrid` object. In this process sample numbers were converted to seconds by dividing by the sampling frequency.

8.3.1 A phoneme database

To be able to select specific phonemes in the TIMIT speech corpus, such as, for example, vowels in stressed or unstressed syllables in penultimate word position, we decided to build a database with information about every phoneme in the corpus.[4] We have 241,225 records in the database (in table 8.4 the phoneme inventory is shown with more details). Each record contains the following fields:

ph_id The phoneme label. The label symbols can be found in table 8.4.

ph_tmin Start time of the phoneme with respect to the start of the sentence in seconds. All sentences start at time 0 seconds.

ph_tmax End time of the phoneme with respect to the start of the sentence in seconds.

ph_dur Duration of the phoneme, defined as the difference between end time and start time. Added for convenience only.

ph_type Phoneme type. One uppercase letter from the set `S`, `A`, `F`, `N`, `G`, `V`, `O`, indicating Stop, Affricate, Fricative, Nasal, Glide&Semivowel, Vowel and Other, respectively. For the assignment to the groups see table 8.4.

ph_stress Stress. A number `0`, `1`, or `2`, indicating no stress, primary stress and secondary stress, respectively. In section 8.3.2 we will explain how we obtain this value.

ph_in_wd Position of the phoneme in the word, i.e. a number from 1 to n, where n is the number of phonemes in the word.

ph_id_l Label of the phoneme on the left.

ph_type_l Type of the phoneme on the left.

ph_id_r Label of the phoneme on the right.

ph_type_r Type of the phoneme on the right.

wd_id Orthographic representation of the word that contains this phoneme.

wd_tmin Start time of the word.

wd_tmax End time of the word.

wd_dur Duration of the word. Again, added for convenience only.

[4] We used the PostgreSQL relational database that is freely available for the Linux operating system. More information about this database can be found at the following URL `http://www.postgresql.org/`

wd_ph Transcribed phoneme string for the word.

wd_nph Number of phonemes in the word.

wd_nsyl Number of syllables in the word. The simple heuristic to determine the number of syllables is: the number of syllables in the word equals the number of vowels.

wd_in_st Position of word in the sentence, implying a number from 1 to n, where n is the number of words in the sentence.

st_id The identification of the sentence in which the phoneme occurs. This can be one of `sa1`, `sa2`, `sx1`, `...`, `sx450`, or, `si1`, `...`, `si1890`.

st_tmin Start time of the sentence.

st_tmax End time of the sentence.

st_dur Duration of the sentence. Again, added for convenience only.

sp_id Speaker identification. A four character string. This string together with the data for the next three descriptors were obtained directly from the directory specifiers

sp_dr Dialect region of the speaker. One of the numbers `1` to `8`.

sp_sex Sex of the speaker. Either `m` or `f`.

usage Either `train` or `test`.

8.3.2 Obtaining stress information

The start time and the end time were obtained by dividing the sample numbers that mark the phoneme by 16,000, the sampling frequency of the audio files. The fields *st_id* (sentence identification), *sp_id*, *sp_dr*, *sp_sex* and *usage* can all be obtained directly from the file path.

Because the `.phn` label files do not contain any information about actually realized word accent (stress), we could only rely on the stress information in the pronouncing dictionary available on the CD-ROM. This dictionary is a simple text file and it is formatted so that each line contains a word in lower case followed by two spaces and then the quasi-phonemic transcription given between slashes. There are 6229 entries in the dictionary. Stress markers are given as `1` for primary stress and `2` for secondary stress, tagged on to the end of the vowel symbol. One pronunciation is provided per entry except in the case where the same orthography corresponds to different parts of speech with different pronunciations, and both forms exist in the TIMIT label files. To differentiate these words, multiple entries are given, with the syntactic class following the symbol `~`. The classes found in the lexicon are: `~n` for a noun, `~v` for a verb, `~adj` for an adjective, `~pres` for present tense and `~past` for past tense. An example is the word “live”, with the entries:

```
live~v /l ih1 v/
live~adj /l ay1 v/
```

A two-syllable entry, e.g. the word “accent”, will look like this:

```
accent /ae1 k s eh2 n t/.
```

The realized pronunciation, however, often differs from the ideal pronunciation in the dictionary, both in terms of actually spoken phonemes and word stress. This means that to find the position in the realized phoneme string where the stress is supposed to occur, we have to find a match between this realized phoneme string and the ideal phoneme string. We can do this with a dynamic programming matching algorithm that goes as follows:

for i = 1..6300
  Get `.phn` phonetic label file
  Read realized phonene string
  Get `.wrd` word label file
  Construct ideal phoneme string from word label file
  Construct associated cost matrix
  Find optimal path through matrix
  Assign the realized stress
endfor

Algorithm 8.2. TIMIT stress info.

The ideal phoneme string for a sentence can be constructed from the word label file by concatenating all the phonemes from the word entries in the pronouncing dictionary. Because the plosives in the phonetic label files have closure and burst separately labelled, we choose to adapt the dictionary by inserting before every plosive (`b`, `d`, `g`, `p`, `t`, `k`, `jh` and `ch`) the corresponding closure string. The realized phoneme string is simply the concatenation of all the labels in the phonetic label file.

With these two strings we can construct a cost matrix. A matrix element at position (i, j) measures the cost of the confusion of realized phoneme i with ideal phoneme j. By varying the costs associated with a particular confusion, we can differentiate between vowel-consonant, within-category and between-categories confusions. We made confusions between a vowel and a consonant, and vice versa, most costly. Next most costly were the remaining between-categories confusions, for example, confusing a nasal with a glide or a fricative with a nasal. We made within-category confusions least costly.[5]

A Viterbi dynamic programming algorithm was used to find the path of lowest cost through the cost matrix. The path was constrained to start at the lower left corner of the matrix and to end in the upper right corner. When the realized string equals the ideal string the optimal path follows the diagonal of this matrix. Both for square and rectangular matrices, the path did not deviate much from the ‘diagonal’. We used this optimal path to map the accent from the phonemes of the ideal string on the vowels

[5] The actual confusion costs used in the algorithm were: `vowelConsonant = 10`, `withinCategory = 3`, `betweenRestCategories = 5`.

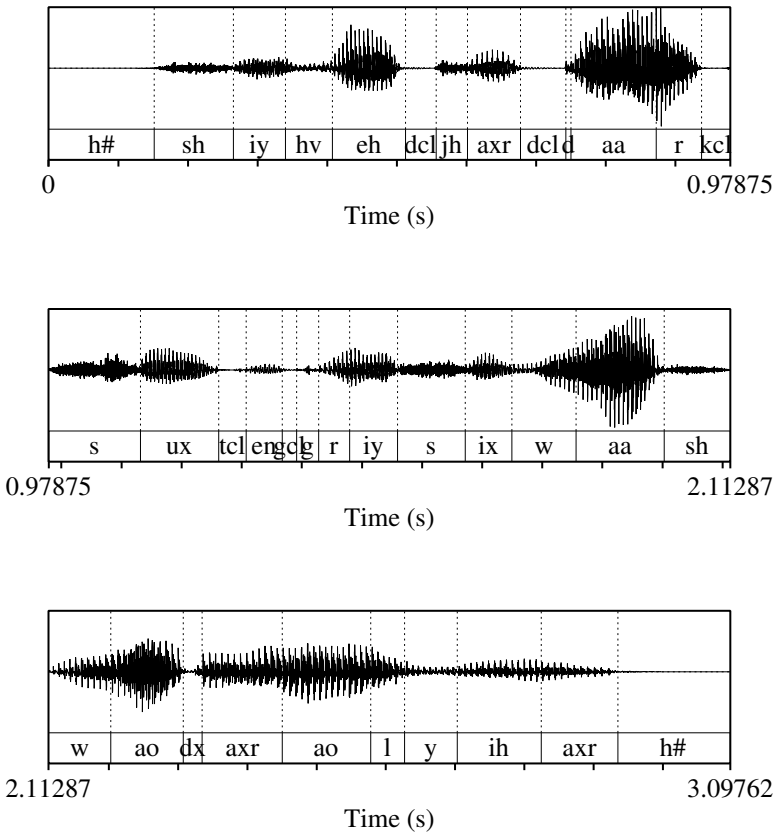

Figure 8.1. The sound waveform and the labels for sentence `sa1`, "She had your dark suit in greasy wash water all year", from speaker `mjsw0` from dialect `dr1` in the `test` part of the database.

from the realized string. In case of insertions, i.e. an ideal phoneme matches two or more realized phonemes, the stress was placed on the last vowel. As an example we will show for sentence `sa1` from speaker `mjsw0` from dialect `dr1` in the `test` part of TIMIT some details of this stress assignment processing.[6] In figure 8.1 we show the sound file with its labels. To show all labels reasonably well the sound has been drawn in three sequential parts that start and end at exact label interval boundaries. For maximum visibility, the amplitude of the sound has been scaled too. The minimum path in a stylized cost matrix is displayed in figure 8.2. The realized phoneme labels are displayed below the cost matrix and the ideal label string is displayed on the left

[6] The relative path names for the sound file and the label file will be `../test/dr1/mjsw0/sa1.wav` and `../test/dr1/mjsw0/sa1.phn`, respectively

of the matrix in the second column. There were 42 labels in the ideal string and 36 labels in the realized string. The first column displays the row numbers in units of five and the stress marker `1` is indicated in the third column. When the realized phonemes all equal the ideal phonemes the path is plain diagonal and would be filled with '`o`' symbols. We see that this is not the case in the table; this has several reasons.

substitution The speaker realized another phoneme instead of the canonical phoneme. The path still runs diagonal but shows a '±' symbol instead of a '`o`'. The table shows nine substitutions. For example, in row four, the voiceless `hh` in the canonical form is substituted by the voiced variant `hv` and in row five the `ae` is substituted by `eh`.

deletion The speaker did not realize the phoneme at all. This effect shows itself in the table when the path is directed vertically. Of the seven deletions in the table, four are from plosives. In row 35 the plosive is released as a tap `dx`, in row 16 and 20 the plosives are unreleased and in row 7 the plosive is released as an affricate.

insertion The speaker realized an extra phoneme. The path is horizontal then. This happens only once, at row 40 where an `axr` is inserted.

8.4 Phoneme statistics

We will start with a summary of how the 241,225 items in the database are divided over phonemes and dialect regions. This data is presented in tabel 8.4 which shows how the number of occurrences of the 61 different labels in the TIMIT database are distributed over the dialect regions. The last column shows the total number of occurrences of a particular label and we notice large differences in numbers between the labels: from 43 occurrences for the label `eng` to 12600 for the label `h#`. Because every sentence starts and ends with `h#`, it occurs $630 \times 10 \times 2 = 12600$ times. Therefore, the cells in the `h#`-row display the number of sentences in each dialect multiplied by two. If we divide this number by 20 we simply obtain the number of speakers.

Table 8.4. Phoneme inventory of the TIMIT acoustic phonetic database by dialect region. The first column shows the TIMIT phoneme label and the second column a typical word in which this phoneme label occurs. The next eight columns, labelled dr1, ..., dr8, show the number of phonemes in each dialect group. The last column shows the row sum, i.e. the total number of occurrences of this phoneme. The last row in the table shows the column totals, i.e. the total number of phonemes per dialect. The rows are grouped according to phoneme type.

Label	Word	dr1	dr2	dr3	dr4	dr5	dr6	dr7	dr8	Total

Table 8.4. Continued.

Label	Word	dr1	dr2	dr3	dr4	dr5	dr6	dr7	dr8	Total
					Vowels					
aa	b**o**tt	333	726	623	656	619	303	743	194	4197
ae	b**a**t	425	880	882	822	851	425	847	272	5404
ah	b**u**t	255	493	511	453	527	261	532	153	3185
ao	b**ough**t	296	623	633	714	688	277	637	228	4096
aw	b**ou**t	70	166	146	161	136	86	142	38	945
ax	**a**bout	400	774	833	867	825	302	706	249	4956
ax-h	s**u**spect	49	86	69	86	79	29	76	19	493
axr	butt**er**	319	816	811	806	732	312	743	251	4790
ay	b**i**te	256	528	534	492	493	234	512	193	3242
eh	b**e**t	409	867	870	836	830	382	838	261	5293
er	b**ir**d	178	521	498	447	389	188	443	182	2846
ey	b**ai**t	245	463	489	492	485	234	531	149	3088
ih	b**i**t	479	1097	1034	1145	1098	469	1069	369	6760
ix	deb**i**t	963	1820	1811	1798	1743	934	1916	602	11587
iy	b**ee**t	693	1601	1517	1576	1560	660	1504	552	9663
ow	b**oa**t	240	452	466	447	473	226	465	144	2913
oy	b**oy**	81	167	171	119	116	71	166	56	947
uh	b**oo**k	67	112	118	114	114	72	109	50	756
uw	b**oo**t	83	155	111	86	107	60	93	30	725
ux	t**oo**t	169	358	444	411	406	177	401	122	2488
					Affricates					
ch	**ch**oke	93	151	199	159	183	69	182	45	1081
jh	**j**oke	124	232	268	259	278	100	234	86	1581
					Stop closures					
bcl	**b**ee	211	403	432	405	432	181	468	153	2685
dcl	**d**ay	528	1028	1043	1054	1036	469	1089	338	6585
gcl	**g**ay	233	492	498	467	465	216	515	145	3031
kcl	**k**ey	603	1293	1238	1243	1221	560	1259	406	7823
pcl	**p**ea	262	609	570	579	570	257	575	187	3609
tcl	**t**ea	672	1439	1422	1444	1411	678	1473	439	8978
					Stops					
b	**b**ee	242	467	504	471	459	206	543	175	3067
d	**d**ay	390	800	749	729	721	354	802	248	4793
dx	dir**t**y	273	595	599	581	542	279	576	204	3649
g	**g**ay	211	451	462	421	437	191	466	133	2772
k	**k**ey	493	1080	1028	1033	979	469	1081	325	6488
p	**p**ea	263	599	549	561	541	255	595	182	3545
q	ba**t**	377	840	700	741	839	363	762	212	4834
t	**t**ea	413	982	904	950	940	421	974	315	5899

Table 8.4. Continued.

Label	Word	dr1	dr2	dr3	dr4	dr5	dr6	dr7	dr8	Total
					Fricatives					
dh	**th**en	308	600	649	625	592	290	616	199	3879
f	**f**in	256	503	493	460	481	243	508	184	3128
s	**s**ea	796	1674	1655	1565	1545	758	1597	524	10114
sh	**sh**e	242	487	450	492	472	240	492	159	3034
th	**th**in	72	171	151	179	150	73	171	51	1018
v	**v**an	227	420	435	439	419	168	433	163	2704
z	**z**one	374	772	840	833	808	357	780	282	5046
zh	a**z**ure	15	34	38	40	47	16	26	9	225
					Glides					
el	bott**le**	110	207	184	212	214	82	206	79	1294
hh	**h**ay	103	205	226	184	220	110	205	60	1313
hv	a**h**ead	115	241	272	236	222	110	251	76	1523
l	**l**ay	638	1336	1346	1300	1271	586	1269	411	8157
r	**r**ay	643	1473	1432	1445	1434	650	1510	477	9064
w	**w**ay	329	725	713	693	698	331	673	217	4379
y	**y**acht	191	373	413	354	364	184	363	107	2349
					Nasals					
em	bott**om**	14	27	31	24	25	10	24	16	171
en	butt**on**	78	151	158	144	132	79	168	64	974
eng	wash**ing**ton	3	8	7	9	8	5	2	1	43
m	**m**om	428	880	880	870	843	388	835	305	5429
n	**n**oon	734	1596	1509	1529	1493	691	1531	486	9569
ng	si**ng**	135	287	272	253	297	149	279	72	1744
nx	wi**nn**er	90	215	246	235	185	98	194	68	1331
					Others					
epi	silence	179	308	310	290	305	171	331	106	2000
h#	begin/end	980	2040	2040	2000	1960	920	2000	660	12600
pau	pause	117	213	195	221	252	73	202	70	1343
		18575	39112	38681	38257	37762	17552	38733	12553	241225

8.5 Characteristics of the vowel material

In order to gather some information on the spectral characteristics of the vowels in the database we performed a number of analyses on them. Because males and females have different voice characteristics we first separated the vowels into a male and a female set. We did this for the training part as well as for the test part of TIMIT. Table 8.5 summarizes.

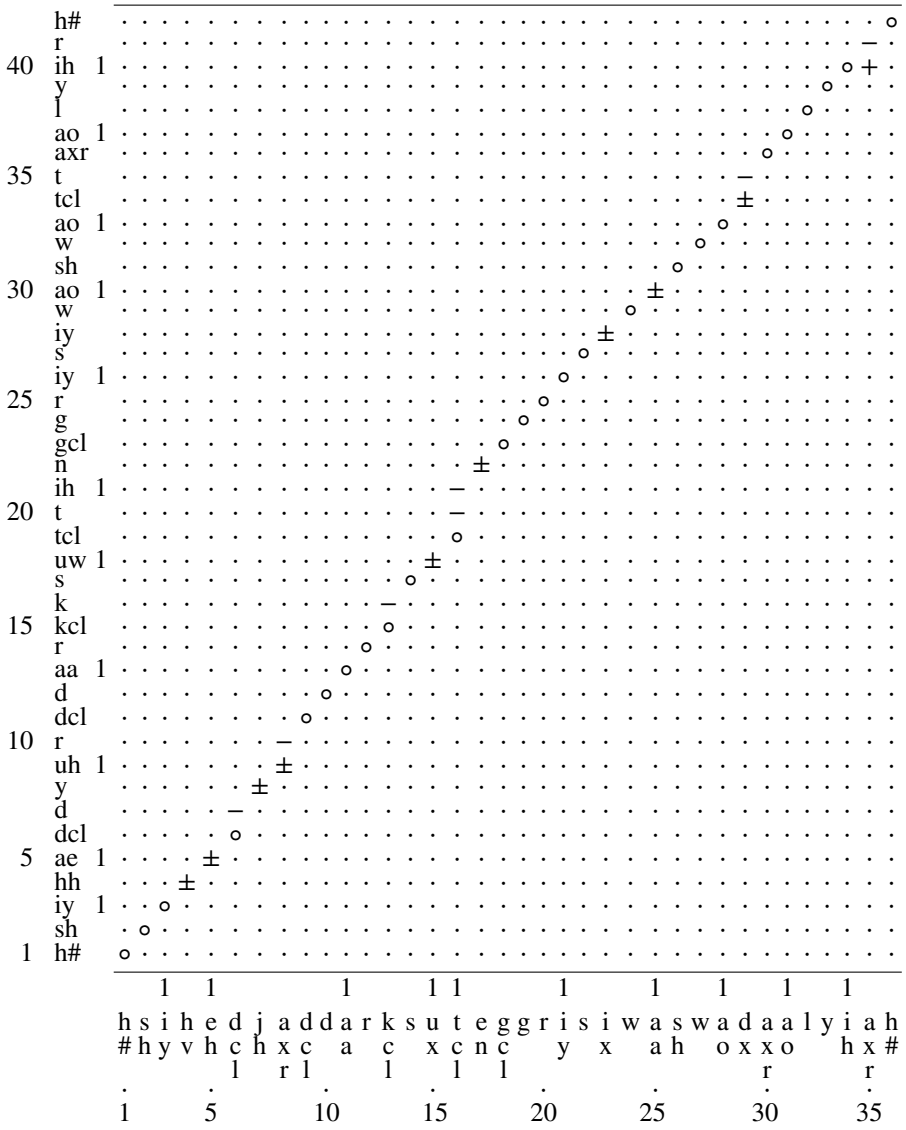

Figure 8.2. The stylized cost matrix with the optimal path for the sentence "She had your dark suit in greasy wash water all year" of speaker `mjsw0`. The first column shows the row number of the matrix in units of 5, each unit being one label. The second column shows the ideal phoneme string as found in the dictionary. The third column indicates dictionary stress with a `1`. At the bottom of the cost matrix we show the transposed stress and the realized phonemes. The symbols that form the path are $\circ, +, -, \pm$ and indicate identity, insertion, deletion, and substitution, respectively.

Table 8.5. The number of vowels in the TIMIT database split up by speaker sex.

	Male	Female	Total
Training	40468	13889	54357
Test	16995	7022	24017
	57463	20911	78374

8.5.1 Analysis of the vowels

Although only the vowel parts of the sentences in the TIMIT database have our current interest, the analyses that will be described were performed on the complete sound files. These analyses involved a fundamental frequency analysis, a filterbank analysis and an intensity analysis. Frames of interest at one or more locations within a vowel can easily be extracted from the results of these analyses. We decided to extract three analysis frames for each vowel: one frame at the midpoint and two frames at 25 ms before and after the midpoint.[7] In rare cases, if the vowel segment is very short, the last two points could lie in the surrounding phonemes (see also figure 8.9 and its explanation in section 8.6). These positions can easily be calculated from the information in our own database since the starting time and the finishing time of each vowel segment were recorded (see section 8.3.1). For an analysis window with a duration of 20 ms this amounts to the measurement of a vowel nucleus of approximately 70 ms duration. The frames of interest were collected in a table and speaker and vowel information were saved too. We will now describe each analysis in somewhat more detail.

8.5.1.1 Fundamental frequency analysis

The fundamental frequencies or pitches[8] were determined by the autocorrelation method as implemented in the PRAAT program. This method consists of two steps: in the first step a number of pitch candidates are found for each frame and in the second step the optimal candidate in each frame is selected. The details of the algorithm are described in Boersma (1993). Both steps in the pitch algorithm depend on a number of parameters.

The first step, finding the candidates depends on two parameters: `timeStep` which determines the time interval between consecutive pitch measurements and `minimumPitch` which determines the minimum pitch that can be measured; at the same time this parameter determines the length of the analysis window. We have chosen a value of 5 ms for time step and a value of 75 Hz for minimum pitch.

[7] As an alternative we could have performed the analyses at times 25% from the start and 25% before the end of the vowel, for example. However, this choice is more dependent on a correct labelling of the start time and end time of the vowel. The centre of a vowel is simpler to detect automatically.

[8] Actually the numbers obtained from PRAAT's pitch measurements are perception oriented and the term 'pitch' would be more accurate than 'fundamental frequency'.

The parameters for the second step, finding the optimal candidates, were set to their standard values.[9]

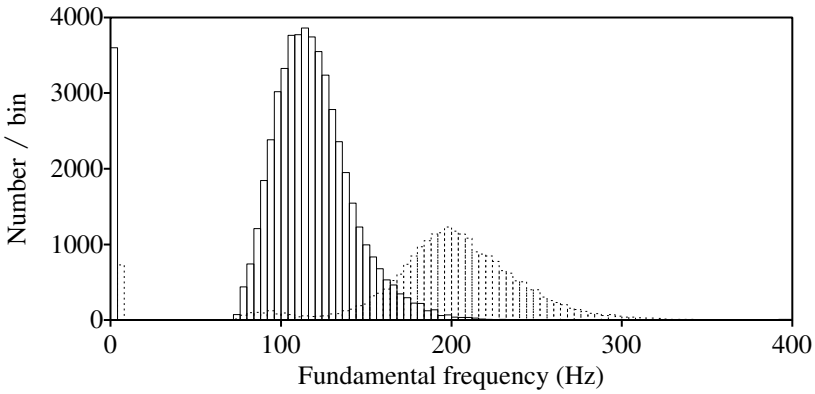

Figure 8.3. The distributions of fundamental frequency, as measured at the midpoint of the vowels, for the 54357 male and 24017 female vowels drawn with solid and dotted lines, respectively. The bin width was chosen as 4 Hz. The larger column near 0 Hz shows the number of unvoiced decisions for the male vowels. The smaller column, which has been offset by one bin, shows the unvoiced decisions for the female vowels.

In figure 8.3 we have plotted the distributions of the fundamental frequencies that were measured at the midpoints of the vowels. The distribution drawn with solid lines shows data from the male speakers. The dotted lines show the data for the female speakers. For some 5.5% of the vowels, 4327 in total, split up into 3598 and 729 for males and females, respectively, no pitch could be determined. These vowels were assigned a fundamental frequency of 0 Hertz. Their presence is indicated in the figure with bars at the first and second bin near 0 Hertz, respectively. There can be various reasons why a stable fundamental frequency could not be measured. In figure 8.4 we show some examples of vowels where no fundamental frequency could be established at the midpoints. All these examples occur in the sentence `sa1` from the training part of TIMIT as pronounced by male speaker `cpm0` from dialect region `dr1`. For optimum display all sound amplitudes in the oscillograms have been scaled. The top display shows the oscillogram of the complete sentence. The next four displays show enlarged versions of the intervals that are marked in the top display by dotted lines and are labelled (a), (b), (c) and (d), respectively. The vowel segments of interest are indicated with a capitalized label. In display (a) we have a voiceless vowel `ax-h`. In (b) we see period halving in the vowel `aa` at approximately 0.73 s. The fundamental frequency

[9] These parameters and their standard values were: `silenceThreshold = 0.03`, `voicingThreshold = 0.45`, `octaveCost = 0.01`, `octaveJumpCost = 0.35`, `voicedUnvoicedCost = 0.14`, `pitchCeiling = 600 Hz`.

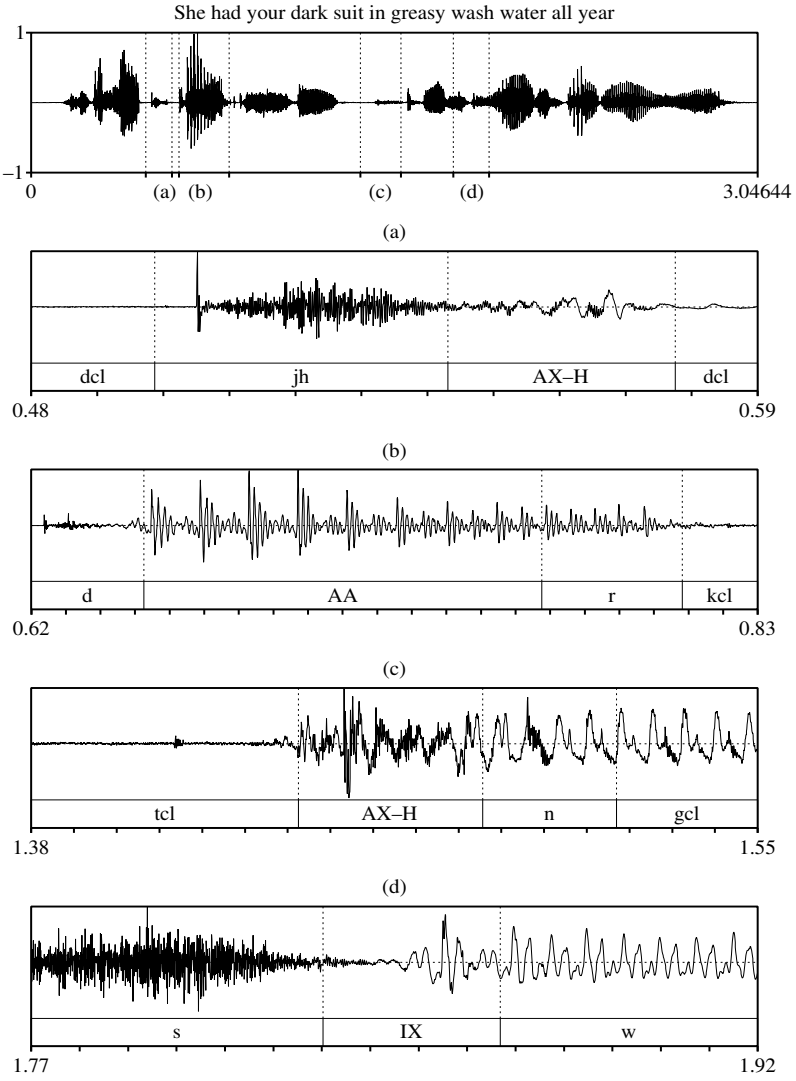

Figure 8.4. Some examples of vowels where no fundamental frequency could be established at the midpoints. The sentence given at the top is `sa1` from the training part of TIMIT and is pronounced by male speaker `cpm0` from dialect region `dr1`. All sound amplitudes per oscillogram have been scaled for optimum display. The top display shows the oscillogram of the complete sentence. The next four displays show enlarged versions of the intervals that are marked in the top display by dotted lines and are labelled (a), (b), (c) and (d), respectively. The vowels of interest are indicated with a capitalized label. (a) Voiceless vowel `ax-h`. (b) Period halving in `aa`. (c) Noisy `ax-h`. (d) Incorrect segmentation for `ix`.

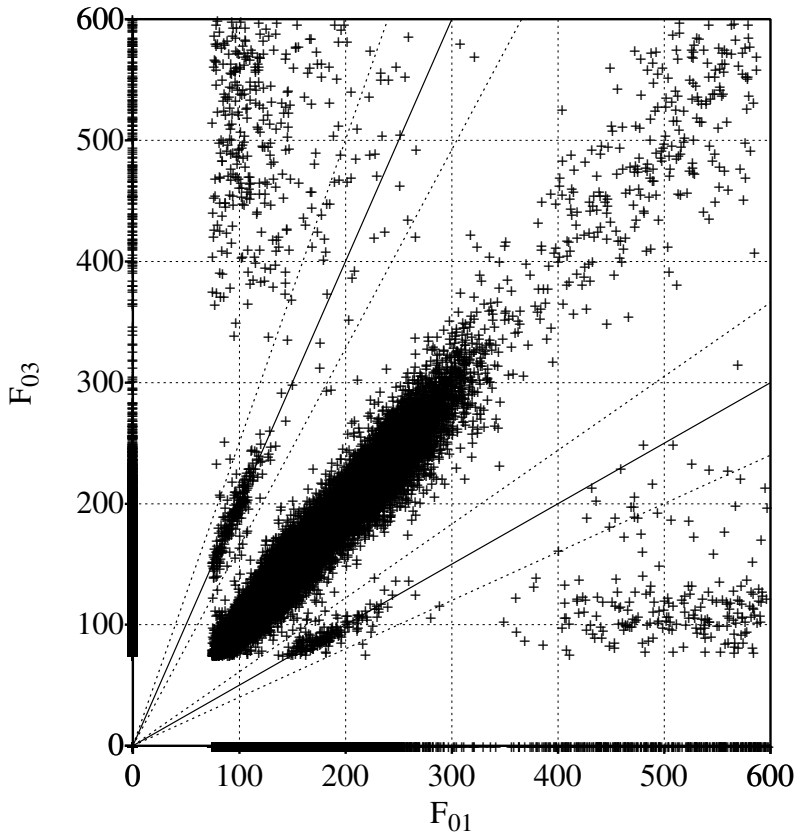

Figure 8.5. Scatter plot of the fundamental frequencies at the first and last position in a vowel. The two solid lines originating at the origin mark where the fundamental frequencies differ by two octaves, i.e. one octave up and one octave down from the cloud in the middle. The points on the axes mark unvoiced frames.

before the doubling is approximately 70 Hz and therefore falls below our chosen lower limit of 75 Hz. In (c) the vowel `ax-h` is very breathy and no fundamental frequency was assigned because the voicing threshold was not reached. In (d) the vowel `ix` has been incorrectly segmented. For a correct segmentation the left border of the vowel should have been moved some 11 ms to the right. With the incorrect segmentation there is a noisy part at the beginning of the vowel and therefore no stable fundamental frequency could be measured.

In figure 8.5 we show the variability over time of the fundamental frequency measurements. For each analysed vowel we have plotted the F_0's from the first and the

third analysis frames against each other in a scatter plot. The corresponding axes are labelled F_{01} and F_{03}, respectively. The midpoints of these analysis frames differ by exactly 50 ms and for the majority of the cases we expect the fundamental frequencies not to differ too much over this time range. In the plot we see a heavy cloud of points in the neighbourhood of the diagonal. These are the points where the two fundamental frequency measurements were approximately equal, equality applying if the point lies exactly on the diagonal. Above and below the diagonal, in the lower left hand part of the figure, we see two small clouds of points. Here the fundamental frequencies differ by two octaves. The cloud above the diagonal shows the data for which $F_{03} = 2F_{01}$ and the cloud below the diagonal shows the data for which $F_{03} = \frac{1}{2}F_{01}$. The two solid lines in the figure trace these octave relations. Two other clouds of points within the plot are visible: one at the lower right and the other at the top left. These are measurements where one of the two frequencies is very much off, probably due to a noisy signal frame. Finally, the points on the horizontal and vertical axes represent segments where one of the two frequencies could not be measured. These segments were assigned a fundamental frequency of 0 Hz.

To get an indication of the number of measurements associated with these regions in the plot we performed a number of selections on the data and counted the conforming number of data. See table 8.6 for an overview. The first criterion selects all data and the numbers in the table are a replication of the totals in table 8.5. The next three criteria select one of the frames where no fundamental frequency could be measured. They show that, compared to the first and the third frame, fewer frames at the midpoint show a fundamental frequency measurement of 0 Hz. The fifth criterion selects the frames where at least one of the three measurements produces a value of 0 Hz. This occurs approximately once in every five vowels. Criterion six selects the measurements where all three frames produce a value of 0 Hz. This selects the vowels that were realized either voiceless or with too large a noise component to be considered voiced. There were 2819 of these realizations, approximately 3.6% of the data. Criterion nine and ten select the vowels where an octave jump occurs between the first and third frame.

Octave jumps to the lower octave were selected by counting all points in a small "octave cone" limited by the lower two dotted lines in figure 8.5. In this cone, the solid line that marks the lower octave bisects the angle between the dotted lines. The tangent of this octave line is 0.5. We have arbitrarily chosen the tangent of the lower line to be equal to 0.4, this gives us a lower limit where $F_{03} = 0.4F_{01}$. With these two values we can calculate the equation of the other dotted line: $F_{03} = 0.61F_{01}$. For the higher octave line whose tangent equals 2, the two dotted lines intersect at the same angle as for the lower octave and their equations are $F_{03} = 1.64F_{01}$ and $F_{03} = 2.50F_{01}$. Criteria seven and eight select the vowels where the first and third measurement differ by more than one octave. Finally, citerion eleven selects the vowels where the measurements behave in a more or less regular fashion, i.e. the points in the plot lie in the neighborhood of the diagonal and the fundamental frequencys in the first and the third frame are approximately the same, although one could probably argue against the status of measurements that have a fundamental frequency larger than, say, 400 Hz. If instead of an upper limit of 600 Hz, we would had chosen 400 Hz, those high frequencies would have been mapped to a value of 0 Hz.

Table 8.6. The number of fundamental frequency measurements that obey the criterion in the last column, split up into male and female part. F_{01}, F_{02}, F_{03} are the fundamental frequencies in first, second and third analysis frame of each vowel, respectively. Logical combinations 'AND' and 'OR' are shown with $\vee$ and $\wedge$ symbols, respectively. The relations on which the last three criteria are based are shown with dotted lines in figure 8.5. For more details see text.

Male	Female	Total	%	Criterion	
57463	20911	78374	100.0	1	1
8796	2827	11623	14.8	2	$F_{01} = 0$
3598	729	4327	5.5	3	$F_{02} = 0$
8473	2334	10807	13.8	4	$F_{03} = 0$
13404	4102	17506	22.3	5	$F_{01} = 0 \vee F_{02} = 0 \vee F_{03} = 0$
2389	430	2819	3.6	6	$F_{01} = 0 \wedge F_{02} = 0 \wedge F_{03} = 0$
169	21	190	0.2	7	$F_{01} > 0 \wedge F_{03} > 0 \wedge F_{03}/F_{01} < 0.40$
257	45	302	0.4	8	$F_{01} > 0 \wedge F_{03} > 0 \wedge F_{03}/F_{01} > 2.50$
22	164	186	0.2	9	$F_{01} > 0 \wedge (0.40 < F_{03}/F_{01} < 0.61)$
35	231	266	0.3	10	$F_{01} > 0 \wedge (1.64 < F_{03}/F_{01} < 2.50)$
40587	19500	60087	76.7	11	$F_{01} > 0 \wedge (0.61 \leq F_{03}/F_{01} \leq 1.64)$

8.5.1.2 Filter bank analysis

The filter bank analysis that we have implemented in the PRAAT program uses filters that are equally spaced on a Bark frequency scale. The associated object type is `BarkFilter`. The number of filters is optional and we have chosen to represent a vowel analysis frame with 18 filter values. With 18 filters, the centre frequency of the 18$^{\text{th}}$ filter being 4227 Hz, we cover the most important frequency region for vowels, which runs approximately to 5000 Hz. All filters have a bandwidth of 1 Bark and are spaced 1 Bark apart. The first filter is positioned at a frequency of 1 Bark. The filtering is simulated in software with a filtering function specified by Sekey & Hanson (1984) as

$$10 \log F(z) = 7 - 7.5(z - 0.215) - 17.5\sqrt{0.196 + (z - 0.215)^2}, \qquad (8.1)$$

where z is the frequency in Bark. The relation between frequencies in Bark and frequencies in Hertz is given by the following equation:

$$z(f) = 7 \ln \left(\frac{f}{650} + \sqrt{1 + \left(\frac{f}{650} \right)^2} \right). \qquad (8.2)$$

The inverse relation is:

$$f(z) = 650 \sinh \frac{z}{7}. \qquad (8.3)$$

Equation (8.2) is visualized in figure 8.6a where the horizontal axis is in Hertz and the vertical axis in Bark. The figure clearly shows approximate logarithmic behaviour above 650 Hz and approximate linear behaviour below this frequency.

The 18 Sekey and Hanson filter functions described by equation (8.1) are displayed in figure 8.6b on a Bark frequency scale. Figure 8.6c displays the same filter functions but now on a linear frequency scale in Hertz. The centre frequency of the 18^{th} filter has been marked with a dotted line at $z = 18$ Bark or $f = 4228$ Hz.

After choosing the "Analysis window length" and "Time step" parameters, the filter bank analysis proceeds as follows:

1. Apply a Gaussian window to the sound frame.
2. Convert the windowed sound frame to a `Spectrum`.
3. Convert the spectral amplitudes to *energy* values by squaring the real and imaginary parts and multiplying by the frequency distance between two successive frequency points in the spectrum.[10]
4. For each of the N filters in the filter bank we determine the inner product of its filter function with the energies as determined in the previous step. The result of each inner product is the energy in the corresponding filter.
5. Convert the energies in each filter to power by dividing by the *window length*.
6. Correct the power, due to the windowing of the frame, by dividing by the integral of the *squared* windowing function.
7. Convert all power values to *dB* according to $10\log(power/4 \cdot 10^{-10})$.
8. Write the power values into the corresponding `BarkFilter` object.

As an indication of how a BarkFilter spectrogram compares with a standard spectrogram we have displayed in figures 8.7a and b both representations of the same sentence `sa1`.[11] We clearly see a more expanded low-frequency region and a more compressed high-frequency region in the BarkFilter representation. The frequency quantization into 18 bins is also clearly visible, as well as a loss of spectral detail.

8.6 Selection of the vowel material

The tasks chosen in this thesis have to do with classification of mainly monophthongal vowels. In order to gather some information on the spectral characteristics of the vowels in the database we performed a number of analyses on them. Because males and females have different voice characteristics we separated the vowels into a male

[10] In the spectrum of a real signal the content at positive and negative frequencies is related. The Spectrum object in PRAAT therefore only needs to store the values for positive frequencies because the values for negative frequencies can calculated from them. Taking negative frequencies into account for the energy calculation boils down to multiplying all values by 2, and weighting the first and the last energy with a factor 1/2.

[11] The commands to perform the analyses were:

```
To BarkFilter...  0.025 0.005 1 1 19
```

and

```
To Spectrogram...  0.025 5000 0.005 20 Gaussian
```

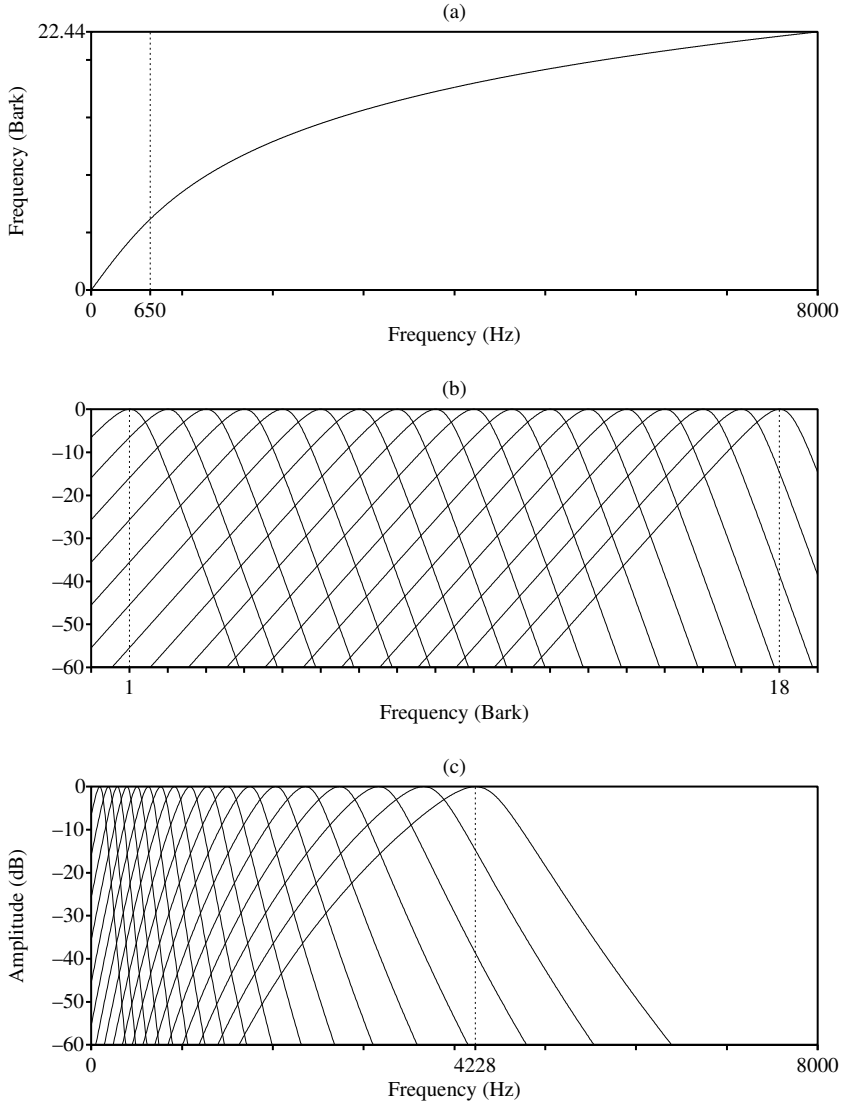

Figure 8.6. (a) The relation between the frequencies in Bark and Hertz. (b) The 18 filter functions on a Bark frequency scale. (c) The 18 filter functions on a frequency scale in Hertz.

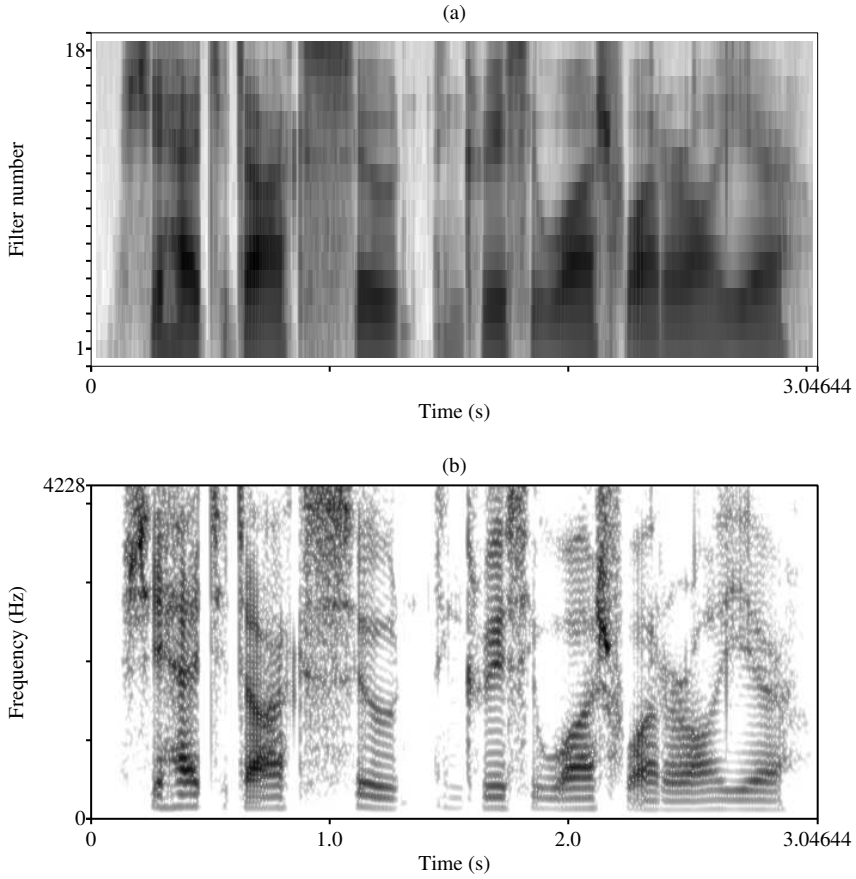

Figure 8.7. Two different spectral representations of the sa1 sentence "She had your dark suit in greasy wash water all year". (a) BarkFilter. (b) Spectrogram.

and a female set. We did this for the training part as well as for the test part of TIMIT. In this section we only display some global characteristics of the male vowels in the training part of the database (40468 vowels). The analyses that we performed will be explained in more detail in the next chapter. Here it suffices to say that the following steps were performed:

- Select all the vowels produced by men in the training part of the database. This results in 40468 vowel segments for all 20 different vowel labels and 326 speakers, representing, on the average, 6.2 realisations per vowel per speaker.

- Perform a bandfilter analysis on three frames in each vowel: a frame at the midpoint and two frames at 25 ms before and after the midpoint. These positions

can be calculated from the information in the database, in the way described in section 8.3.1. In this way we measure a vowel nucleus of approximately 70 ms duration when the analysis window is 20 ms wide. The bandfilter analysis for each frame results in 18 values, as described in section 8.5.1.2.

- Because each speaker has several reproductions of most of the 20 vowels, we calculate the average representation for each vowel for each speaker. This reduces the 40468 vowel representations to 6008 vowel representations. If every speaker had realized all 20 vowels one or more times, we would have had $6520 = 326 \times 20$ representations. The lower number of 6008 thus indicates that not all 326 speakers produced all 20 vowels. We do this averaging per vowel to give all vowels the same weight in some of the subsequent analyses, because we want representations that do not depend too much on differences in the number of occurrences of a vowel in the database.
- Perform principal component analyses, as described in chapter 3.

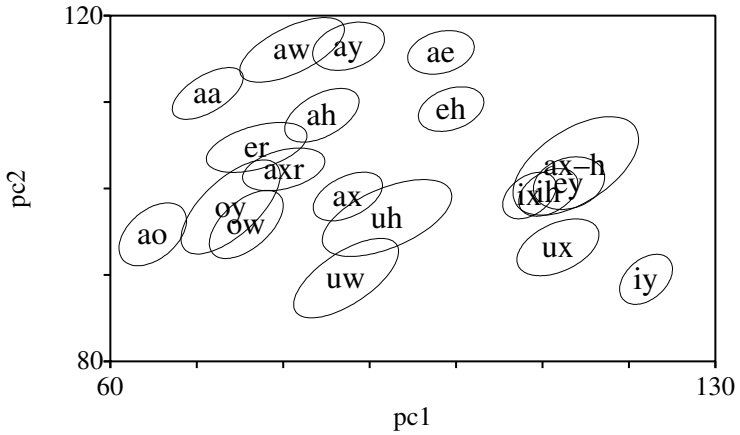

Figure 8.8. The positions and 0.5 σ ellipses of the 20 vowels in the principal component plane as averaged over the 326 male speakers in the training part of the TIMIT database. The horizontal axis was inverted to get the main axes of the ellipses to point from the lower left to the upper right. The first and second principal components explain 52% and 16% of the variance, respectively.

To give some insight in the vowel structure we present in figure 8.8 the average positions of the vowels in the principal component plane. This plane was calculated from the 6008×18 labelled numbers from the central analysis frame in the vowels. The first and second principal components together account for 68% of the variance; 52% is accounted for by the first component, and 16% by the second component. The orientation of the vowels in this plane is reasonably in accordance with the vowels in

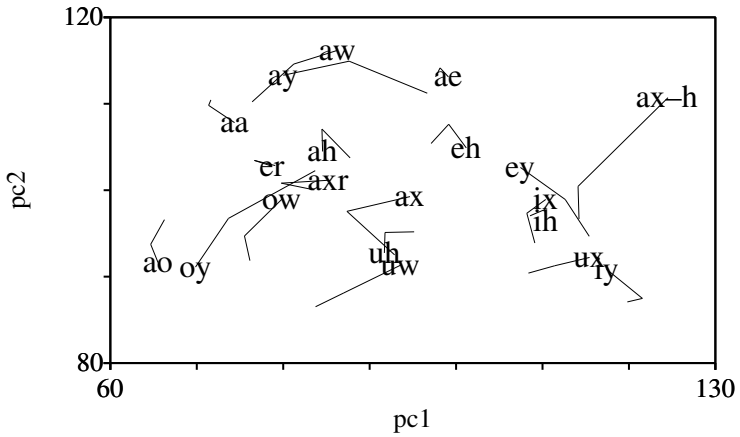

Figure 8.9. The positions in the principal component plane of the begin, middle and end parts of the 20 vowels, averaged over the 326 male speakers in the training part of the database. The positions in the three parts are connected with a line. The start position has been labelled with the corresponding vowel symbol. To conform to figure 8.8, pc1 has been inverted.

the traditonal formant plane, as can be seen, for example, in the paper by Hillenbrand, Getty, Clark & Wheeler (1995). The 0.5σ ellipses in the figure give an indication of the variability for each vowel. The directions of the principal axes of all these ellipses are almost the same, which suggests that these variations are due to the different vocal tract lengths of the speakers. We see that the sizes of the ellipses show some small variability. The `ax-h`, being on average the shortest vowel in the database, shows the largest variability. We will offer an explanation of this strange effect below.

In tables 8.3 on page 120 and 8.4 on page 127 it was indicated that the TIMIT database distinguishes 20 different vowels. Many authors have made vowel groupings to reduce this number of vowels to a more managable number. On the basis of figure 8.8 on the facing page one could argue that some of the vowel groupings used in papers on vowel recognition are problematic. For example, Lee & Hon (1989) and Halberstadt & Glass (1997) make the following vowel groupings {`ih`, `ix`}, {`ah`, `ax`, `ax-h`}, {`aa`, `ao`}, {`er`, `axr`} and {`uw`, `ux`}, resulting in 14 "different" vowel labels. Only the vowels in the groupings {`ih`, `ix`} and {`er`, `axr`} lie acoustically "close"to each other in the figure. The grouping {`uw`, `ux`} shows a complementary distribution: they represent the same phoneme but two symbols have been used because `uw` and `ux` sound different. In the grouping {`ah`, `ax`, `ax-h`} the `ax-h`, being the voiceless pendant of `ah`, shows acoustic characteristics that place it very apart from the other two members in the figure. This is reflected by the large number of confusions between the group with `ax-h` and the group with `ih` in figure 1 in the paper of Halberstadt & Glass (1997), where they give baseline classifier confusions.

Some insight in the *dynamics* of the vowels can be gained by trying to represent spectral change. In figure 8.9 we have represented spectral change in terms of change in projections on principal directions. In this figure the avarage vowels in the start, the middle and the end frames were projected on the plane spanned by the first two principal components that resulted from an analysis of the three frames for these vowels. We therefore performed a principal component analysis on the 18024 × 18 data matrix which simply consists of the three 6008 × 18 data matrices for these three frames appended. In the next step the average positions in the three frames were projected on the principal component plane.[12]

The projections of the three frames are connected with lines. In this way, large spectral change should correspond to long lines. We see in the figure that short vowels have short lines, the only exception being `ax-h`, and that long vowels have long lines, indicating diphthongization. Clearly `oy`, `ey` and `ay` are diphthongs. Contrary to our intuition, the shortest vowel in the database, `ax-h`, shows the largest spectral change and it also showed the largest variability in figure 8.8. This effect, however, can be explained just *because* the `ax-h` is so short. Its average duration according to table 8.3 is only 34 ms, which means that parts of the first and the third analysis frame are located outside the `ax-h` vowel and are biased by the preceding and following phonemes, respectively. Due to its short duration it will be influenced relatively stronger by its neighbouring sounds and therefore shows a relatively large variability. `ix` and `ax`, being monophthongs, with average durations of 52 and 49 ms, respectively, also show a relatively large change between the first and the third frame due to context. Meng & Zue (1991) select 13 what they called monophthong vowels, being {`iy`, `ih`, `eh`, `ey`, `ae`, `aa`, `ah`, `ao`, `ow`, `uh`, `uw`, `ux`, `er`}. One look at figure 8.9 shows that `ey`, with an average duration of 128 ms, cannot be regarded as a monophthong. Also the monophthong status of `ow`, with an average duration of 127 ms, is questionable.

Clearly, the groupings advocated by Lee & Hon (1989) and Halberstadt & Glass (1997) as well as the selections made by Meng & Zue (1991) have their drawbacks. For the time being we make no selection at all and stick with static analyses made at the central part of the vowel keeping all 20 different vowel categories. Grouping can always be done afterwards.

8.7 Conclusion

In this chapter we have given a description of the TIMIT acoustic phonetic speech corpus and how the labelled material was made accessible for the PRAAT program. We have made all information about the 241,225 labelled segments in this corpus accessible via a relational database. As a consequence, this database can be queried with the standard SQL database query language, leading to the analyses already presented in this chapter and to more analyses, which will be presented in the following chapters.

[12] We note that the resulting figure could also be obtained by a projection on the plane from an analysis of the second frame alone, as the eigenvectors in the three frames are almost the same.

Chapter 9

Normalizations on bandfilter data from TIMIT[‡]

Abstract

In the first part of this chapter we explore the spectral differences between vowels produced by male and female speakers. We show a successful extrinsic normalization method based on the Procrustes transform that is able to reduce the differences between vowel bandfilter spectra from male and female speakers.

In chapter 6 we have successfully applied a normalization of the formant frequency data from 50 male speakers and 25 female speakers by adapting the biases in a feed-forward neural network. In the last part of this chapter we will test this model again on bandfilter data from a much bigger data set. The bandfilter data are measurements at the central part of all the vowels in the TIMIT database. The results indicate that bias adaptation is successful also on bandfilter data. The bias adaptation at the output layer showed a slight advantage over the adaptation at the hidden layer.

[‡]This chapter is a modified version of Weenink (1996).

9.1 Introduction

In chapter 6 we described a vowel normalization model on formant frequency data obtained from the data sets of Pols et al. (1973) and Van Nierop et al. (1973). This model was based on adapting only the biases of the hidden layer in a neural network. The results in that chapter indicated that, by only retraining the biases, a neural network of the feedforward type was able to learn the vowel representation for other speakers. For example when trained with male data it reasonably well classifies female data after adaptation of the biases. The results indicated that the model was more successful if the biases of the hidden layer were modified than if the biases of the output layer were modified. See tables 6.3 and 6.4 for these results.

Conceptually, bias adaptation is a simple idea. First we train a neural network with vowel data from a number of speakers. This neural net can then be used as a vowel classifier. Next, for each individual new speaker we retrain only the bias weights in a specified layer. This retrained neural net can then be used as a vowel classifier for that particular speaker. The advantage of using this model is that modifying only bias weights requires less (re)training effort.

The motivation behind this is the following. We know that differences exist between productions of the same vowel by different speakers. The neural network, trained with material from a number of speakers, cannot possibly recognize all vowels from different speakers equally well. To cope with a new, possibly divergent, speaker, the network parameters have to change. Instead of modifying all weights in the network, we proposed to only change the biases in one layer. A modification of the bias weights of, for example, the output layer, shifts the decision boundaries in weight space in a parallel fashion. In other words, it leaves the relative positions of vowel categories intact and only shifts the decision boundaries between them. This is effective because the vowel spaces of different speakers tend to be similar.

In this chapter we want to test whether this can be done with bandfilter data as well. However, no bandfilter data is available for the data sets used in chapter 6. The audio material on which the formant frequency measurements were based is not available any longer, so a new bandfilter analysis is impossible. We therefore decided to use the available TIMIT data set and perform bandfilter analysis on it. A second and more important reason to switch to a bandfilter representation is that bandfilter values can be reliably determined automatically, whereas formant frequency values cannot. A third reason for choosing TIMIT is its widely accepted status as a standard data set of considerable size (Lee & Hon, 1989; Meng & Zue, 1991; Halberstadt & Glass, 1997).

Measuring formant frequency values is a tedious and error-prone job that cannot be fully automated. Given the amount of vowels in the TIMIT database we decided not to use formant frequencies. Bandfilter analysis can be applied automatically and lower-dimensional representations can be obtained that correlate very well with formant frequencies and with perceptual data, as was shown by Pols, Van der Kamp & Plomp (1969). Bandfilter analysis is, of course, not a one-size-fits-all solution. In the low-frequency region we may have an interaction between fundamental frequency and filter bandwidth: for high-pitched sounds some of the bandfilters do not contain energy, because, spectrally speaking, there are no harmonics of the fundamental within

the bandwidth of these filters (Van Alphen, 1992, page 32). As an example, consider a filterbank where the first filter is centered at 100 Hz and has a bandwidth of 100 Hz. A signal with a fundamental of 200 Hz has almost no energy within this filter's bandwidth. Just by being "empty" because of the high pitch of a vowel and not because of its timbre, this filter would introduce a large variance that could ruin a later processing step like principal component analysis based on variances. At the same time this offers opportunities to discriminate higher-pitched vowels from lower-pitched vowels. We could use this to discriminate between the vowels from male and female speakers, since their fundamental frequencies differ a lot, as can be seen from figure 8.3.

The general outline of this chapter is as follows. In the next section we introduce the naming conventions for the subsets of the TIMIT vowel database that we use. In section 9.3 we explore these vowels and present some information on the differences between male and female vowel spectra. We show results on classifying a spectrum as either "male" or "female". In section 9.3.3 we introduce a very powerful method based on Procrustes transformations, to map female spectra to male spectra and vice versa. In section 9.4 we continue the investigation started in chapter 6 whether differences between vowel spectra from different speakers can be equalized by a neural net. The intention is that by allowing only the bias weights of the neural net to change, the net adapts to every speaker. We also describe data reduction to reduce the training times for the neural networks. The chapter ends with a discussion.

9.2 Data set nomenclature

In order to have some shorthand notation for the different spectral data subsets from TIMIT we developed the following naming scheme for a subset: <SEX><PART>-<SUMMARY><DIMENSION>. In this scheme the most important division of the data, <SEX>, is according to whether they originated from a *male* or *female* speaker. This will be indicated by the start sequence M or F. If both speaker sets are involved we start with the MF combination. The next division, <PART>, will be according to whether the data originated from the *test* or the *training* part of TIMIT. This will be indicated by -T for the test part and -L for the training part (-L stands for *learning*). If the dataset has both the test and training part included, we do not use any symbol. In the next part, <SUMMARY>, an -S symbol indicates that we use summary data, i.e. one representation per vowel per speaker, and an -A symbol indicates that we use all the vowels of a speaker. More information on how we obtained the summary data will be presented in the next section. The last part of the naming scheme, <DIMENSION>, indicates the dimensionality of the data set. If the dimensionality is 18, i.e. the full dimensionality of the spectrum, this number will not be shown. The following list, in which the symbol _ represents 'empty', summarizes the naming scheme.

<SEX> := (M|F|MF), male, female, both
<PART> := (-L|-T|_), learning (training) or test part or both
<SUMMARY> := (-S|-A), speaker summary, or all
<DIMENSION> := (_|-<NUMBER>).

To give an indication of the number of items in some of these partitions, table 9.1 gives a summary. The first line in the table shows that data set M-L-A, which contains all

Table 9.1. Summary of the naming scheme, for further explanation see text.

Name	Entries	Name	Entries	Speakers	Description
M-L-S	6008	M-L-A	40468	326	Males, training part
M-T-S	2070	M-T-A	13889	112	Males, test part
M-S	8078	M-A	54357	438	M-L-· + M-T-·
F-L-S	2490	F-L-A	16995	136	Females, training part
F-T-S	1011	F-T-A	7022	56	Females, test part
F-S	3501	F-A	24017	192	F-L + F-T
MF-S	11579	MF-A	78374	630	M-·-· + F-·-·

vowel spectra from all male speakers in the training part of TIMIT, has 40468 entries. These 40468 vowel entries are condensed to 6008 entries in data set M-L-S, in which for each speaker multiple entries for a vowel have been reduced to only one average entry.

9.3 Characteristics of the vowel material

The TIMIT database offers a great opportunity to perform vowel recognition and normalization, since all of its vowels have been segmented and labelled. In the previous chapter it was already outlined how we have made TIMIT generally accessible within the PRAAT program by making the audio files and the label files readable.

We have used all the vowels in TIMIT in the analysis in this chapter. Because the number of occurrences per vowel and per speaker differ very much, as can be seen from table 8.4, we can expect that any learning method that uses the raw vowel data will be biased towards the vowels that occur most frequently. In order to prevent this bias, we decided to compress the vowel material by averaging all the occurrences of the same vowel for each speaker in the same way as was done in section 8.6 for the male vowels in the training part of the database. The compression resulted in a 11579 × 18 summary table with one realization per vowel per speaker, i.e. at most 20 entries per speaker. These summary data are indicated with an -S in table 9.1. The number of entries in MF-S is actually less than the maximum possible, which is 12600 (= 630 × 20), since not all 630 speakers realized all 20 different vowels.

The motivation for splitting the data in a male (M-S) and a female (F-S) part can be found in table 9.2 and in the figures 9.1, 9.2 and 9.3 and, partly, also in figure 8.3. Table 9.2 shows the result of the vowel recognition performance of a discriminant classifier trained with three "different" data sets and tested with the same data sets. These data sets were MF-S, the total summary data set discussed above, containing

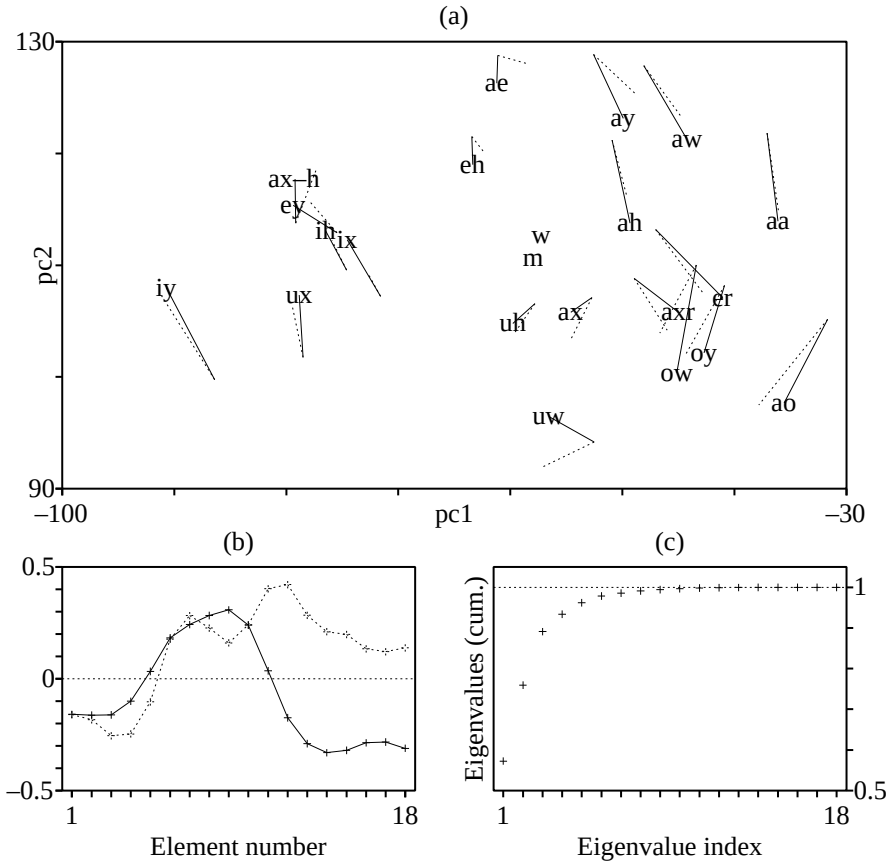

Figure 9.1. Characteristics of male and female vowel centroids in a common PCA eigenspace.
(a) Vowel centroids. The average male vowel centroids are labelled. Each label is the starting point of a solid line that ends at the female vowel centroid. The ends of dotted lines, which start at a female vowel centroid, indicate the positions of the female vowel centroids after a Procrustes transformation to optimally match the male vowel centroids (to be discussed in section 9.3.3). The points labelled 'm' and 'w' are the average male and female spectra, respectively.
(b) The first and second eigenvectors drawn with a solid and a dotted line, respectively.
(c) The cumulative eigenvalues as fractions of the total sum of the eigenvalues.

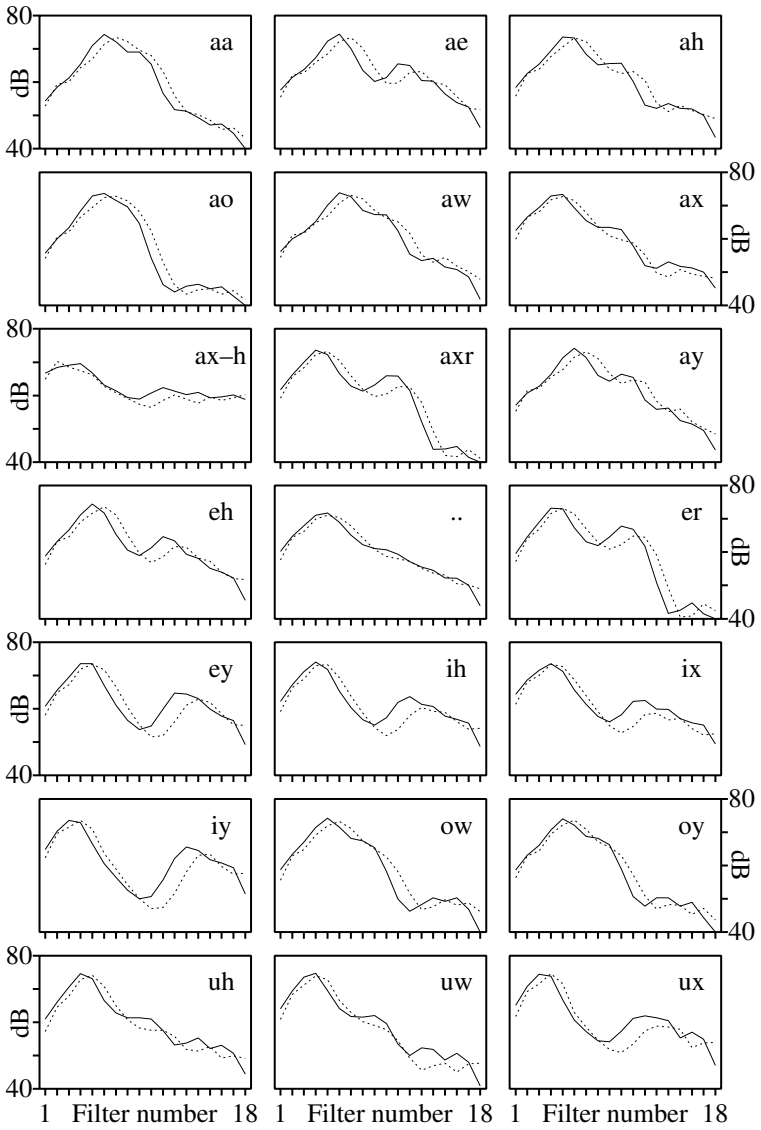

Figure 9.2. The average male and female vowel spectra. The male and female spectra are drawn with solid and dotted lines, respectively. The two spectra at the position in the middle, i.e. fourth row and second column, are the overall average male and average female spectra.

Table 9.2. Fractions correct with a discriminant classifier. The classifier was trained with the data sets displayed in the first column and tested with the data sets shown in the first row. M-S is the summary data with maximally 20 vowel spectra per male speaker. F-S represents the summary data set for the female speakers. MF-S is their combination (see also section 9.2). The numbers in parentheses show the fractions correct after speaker normalization. The normalization for each male/female speaker was done by subtracting the difference between the average of a speaker's vowels and the average of all the speakers' vowels in the male/female group.

	MF-S	M-S	F-S	Entries
MF-S	0.575 (0.645)	0.634 (0.704)	0.440 (0.510)	11579
M-S	0.533 (0.596)	0.663 (0.734)	0.233 (0.280)	8078
F-S	0.370 (0.398)	0.259 (0.267)	0.624 (0.702)	3501

11579 entries, the M-S subset with the 8078 only-male entries and the F-S subset with the 3501 female-only entries. The row marked MF-S in the table reads as follows: a classifier trained on the whole data summary set, male plus female data, shows 0.575 fraction correct when tested on the same data set. It shows 0.634 fraction correct when tested on the male (M-S) data set only and 0.440 when tested on the female (F-S) data set only. The increase in fraction correct from 0.575 for MF-S to 0.634 for M-S and the following decrease to 0.44 for F-S is due to the fact that MF-S training is biased because M-S and F-S contain very unbalanced amounts of data, as was indicated above and can also be seen from table 9.1. Having separate classifiers for M-S and F-S substantially improves the fractions correct to 0.663 and 0.624, respectively. At the same time, these classifiers show smaller fractions correct when the "other" set is tested, as the numbers 0.233 and 0.259 show. These differences between the speaker groups M-S and F-S still remain if we perform speaker normalization, as the numbers in parentheses show. In the straightforward speaker normalization procedure that we have applied here, we have corrected the bandfilter data by the difference of the speaker's average and the group average. We see that the fractions correct increase substantially but the differences between the M-S and F-S sets remain.

Additional evidence that male and female data behave differently is shown in figure 9.1, which displays the common eigenspace obtained by a principal component analysis on the 20 average male and the 20 average female vowels. The 20 male and 20 female average vowels were determined from M-S and F-S, respectively. These 40 representations were collected in one table and a principal component analysis was performed. In part (a) of the figure, which shows the plane spanned by the first two principal components, the average male vowels are projected and shown with the corresponding label. To avoid a clutter of symbols in the figure, we have indicated the positions of the female average vowels as the endpoint of the solid line that starts at the centre of each male average vowel. In part (b), drawn with a solid curve, the first eigenvector whereas the dotted curve shows the second eigenvector. Part (c) shows

the cumulative sum of the eigenvalues, i.e. the cumulative fraction explained variance. The first component explains as much as 57.2% of the variance and the first two components together explain 75.9%.

In the figure we see that the relative positions of the male and female vowels are not the same: some female vowels, like `ao`, lie higher than their male counterpart while others, such as `iy`, lie lower. The main difference in orientation is in the direction of the second component. To understand these differences we have to look at both the weighting of the second eigenvector and the form of the male and female vowel spectra. The average male and female vowel spectra are displayed in figure 9.2. The male spectra are drawn with a solid curve whereas the female spectra are drawn with the dotted curve. The spectra in the middle position, i.e. fourth row second column, are the overall average male and female spectrum. The position of a vowel in the principal component plane is the result of the inner product of the first two eigenvectors with the vowel's spectrum. From figure 9.1b we note that the second eigenvector weighs the first five spectral values with a negative sign and the following with a positive sign. A larger value for the second principal component of a female vowel thus may occur when the higher part of the female spectrum lies above the male spectrum and/or the lower part is beneath this spectrum. We see that for `ao` the first five bandfilter values of the female spectrum lie below the male values and that the contribution of the following six filters is definitely positive. The total contribution of the last six filters is negative but this contribution is only small. The net effect is, as was shown in figure 9.1a, the female `ao` lying above the male `ao`. For the `iy` vowel we note from the spectra that at the interval where the difference between the spectra is largest, i.e. somewhere between filter numbers 10 and 14, we also have the largest weights from the eigenvector. Since the difference in this interval is negative, i.e. the female spectrum lies below the male spectrum, this results in the relative positions shown in the principal component plane, where the female `iy` lies below the male `iy`.

If we consider all the 20 vowels in the principal component plane, then, on average, the female vowels lie somewhat higher than the male ones, as can be seen from the positions of the average male and female spectra. These average spectra are indicated in figure 9.1 with an 'm' and a 'w', respectively. We must conclude from the figure, however, that a common basis for the M-S and F-S data sets would not be very optimal.

Figure 9.1 was purely based on the distribution of variance. In figure 9.3 we have plotted vowel centroids in the eigenspace of a discriminant classifier trained on the M-S data set. A common eigenspace cannot be constructed as easily as for figure 9.1 because we do not have equal numbers of male and female spectra and we cannot use the trick of the vowel centroids. In part (a) we display the male vowel centroids in the Linear Discriminant (LDA) eigenspace. The vowel centroids for the F data set are indicated in the same picture as the endpoints of the solid lines that start at the male vowel centroids. Parts (b) and (c) display the first and the second eigenvectors. This picture also shows different structures for male and female vowel centroids but the positions of the centroids are not as clearly interpretable in terms of the traditional high, low, front and back features as in figure 9.1.

Again, the LDA projections show that a common basis for the male and female data sets would not be very optimal.

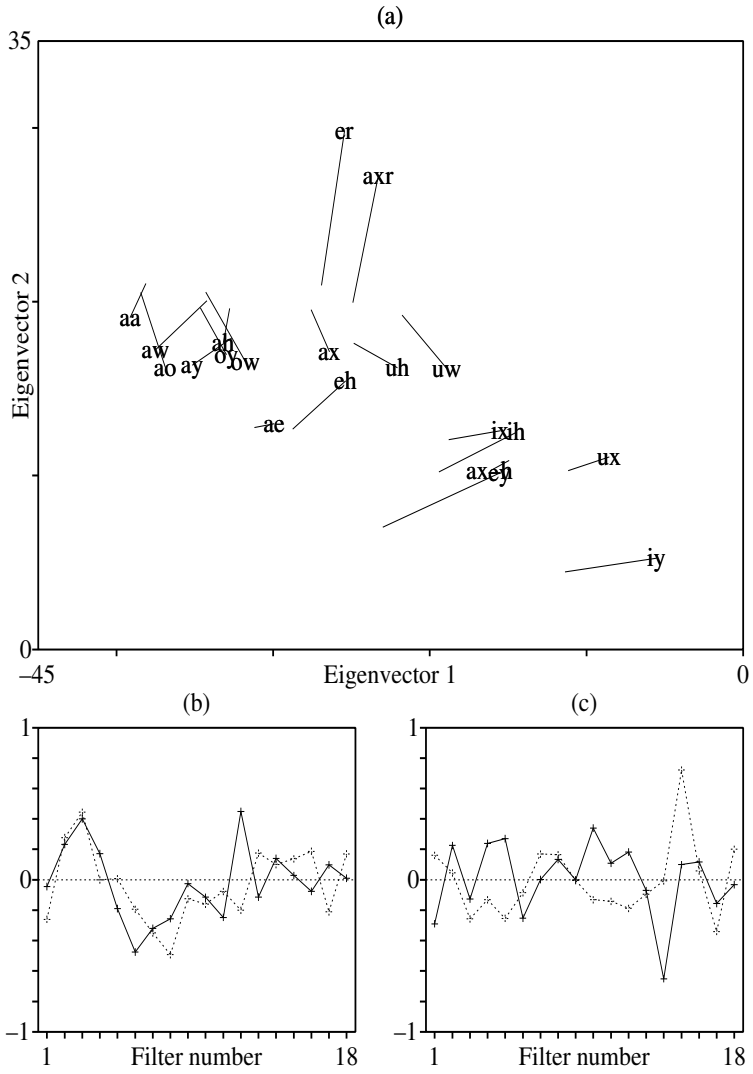

Figure 9.3. Characteristics of the male and female data sets in the male LDA eigenspace. (a) Projection of the vowel centroids in the male eigenspace. The average male vowel centroids are labelled. Each label is the starting point of a solid line that ends at the average female vowel position. (b) First eigenvector. (c) Second eigenvector. The eigenvectors are represented by solid curves for the males and by dotted curves for the females.

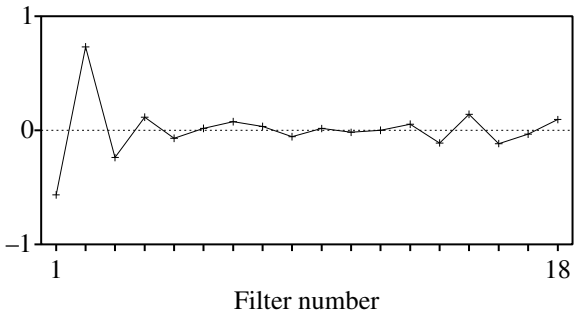

Figure 9.4. The eigenvector from the two-class male/female discriminant classifier.

9.3.1 Classifying a spectrum as male or female

Now, given that the male and female data sets M-S and F-S behave differently, we may ask how well both sets can be discriminated from each other in an automatic fashion. To perform such a test, we relabelled the spectra in the MF-S table. Instead of the vowel labels we substitute either "m" or "f", depending on whether the vowel was from a male or a female speaker. We then have a two-class discrimination problem. A discriminant classifier with identical training and test sets showed 0.937 fraction correct with pooled covariance matrices. If we do not pool the two covariance matrices but use an individual covariance matrix per speaker category, the fraction correct rises to 0.961. If we use a strict separation between training and test set, i.e. for training we use only speakers from the training part of TIMIT and for testing we only use speakers from the test part, we still have 0.913 fraction correct for pooled covariances and 0.948 for individual covariances. According to the naming scheme these training and test sets are MF-L-S and MF-T-S.

Because we have two classes, the discriminant space is one-dimensional and therefore we only have one eigenvector that maps the bandfilter spectra on this space. In figure 9.4 this eigenvector is displayed. We clearly see that it puts most emphasis on the first two filters. Not surprisingly it is here that fundamental frequency differences between males and females are most easily measurable, because (simplifying a bit) a low pitch has both filters filled, a high pitch only the second. One would guess that leaving these two filters out could ruin the discrimination between male and female spectra. In order to investigate this hypothesis we performed discrimination tests in which we successively reduced the number of filters used in the classification tasks. We split up MF-S in independent training and test sets for the discriminant classifier depending on whether the spectrum was from a speaker in the training or in the test part of TIMIT.

In the first series of 18 tests, we used the filter subsets from filter numbers i to

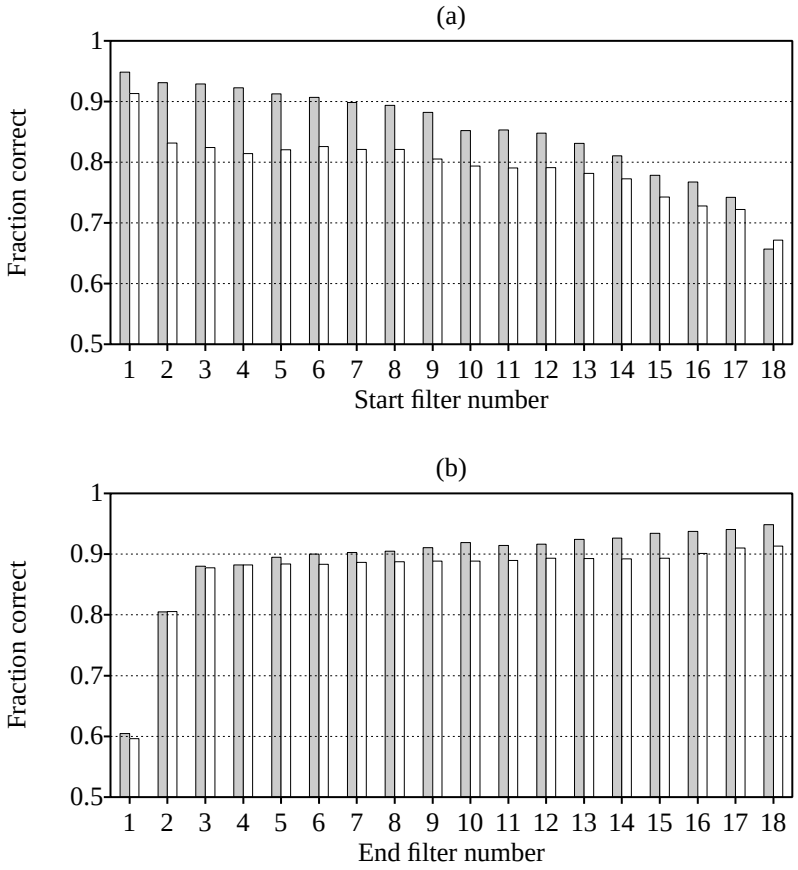

Figure 9.5. Discriminant classifier fraction correct as a function of filter subsets for data set MF-S (split up into independent training and test parts). The light and dark bars show the fractions correct with and without covariance matrix pooling, respectively. (a) Subset with filters from i to 18, where $i = 1..18$. (b) Subset with filters from 1 to j, where $j = 1..18$.

18, where $i = 1..18$, i.e. the number of filters used runs from 18 down to only 1. For example, leaving only the first two filters out corresponds to $i = 3$. In the second series of 18 tests we used the 18 subsets in which the filter numbers varied from 1 to j, where $j = 1..18$. For example, $j = 2$ corresponds to the first two filters only. The results with and without covariance matrix pooling are displayed in figure 9.5. Much to our surprise the fractions correct for subsets turn out to be remarkably high. In bar chart (a) where the fractions correct for the subsets $i..18$ are displayed, we see a gradual decrease from 0.948 for the complete set of 18 filters at bar position 1 to

0.657 for only the 18^{th} filter. Even for this 18^{th} filter the 0.657 fraction correct is well above the 0.5 chance level. In bar chart (b) we see a steep increase in fraction correct from 0.605 for the first filter to 0.88 for the first three filters together. When the number of filters increases to 18 we see a gradual increase in fraction correct, topping of course with 0.948 with all 18 filters. Limiting the filter subset to a range that approximates the telephone speech bandwidth, i.e. filter numbers 3–16, we still obtain 0.895 fraction correct. As the light and dark bars in figure 9.5 show, the effects with and without covariance matrix pooling show the same behaviour. Our conclusion must be that although the filters that show the largest variation due to speaker class, i.e. the first three or four, the resulting gender differences are distributed along the entire spectrum.

9.3.2 Relation between bandfilter values and fundamental frequency

In this section we elaborate on the influence of fundamental frequency on spectral characteristics. In the previous section we noted that although gender differences are distributed along the entire spectrum, low filters show the most prominent effect of fundamental frequency. As figure 8.3 shows, the fundamental frequency distributions of males and females differ significantly. This might suggest that there must be a correlation between fundamental frequency and spectrum. We will investigate this by performing a canonical correlation analysis on the data in the centre part of each vowel. However, as figure 8.3 also shows, for a number of segments no pitch could be determined by the periodicity detector in the PRAAT program, and these segments were assigned a fundamental frequency of 0 Hz. We therefore made a simple cut on the data by selecting only frequencies between the minimum pitch 75 Hz and a more or less artificial upper limit of 350 Hz. In this way we selected 93.4% of the data, i.e. 73165 out of 78374. Since we correlate a one-dimensional vector (fundamental frequency) with an 18-dimensional vector (bandfilter values) this results in only one canonical correlation coefficient whose value is 0.748. Because we have only one dependent variable, the canonical correlation coefficient equals the multiple correlation coefficient. It explains 56% of the variance. In figure 9.6 we show the eigenvector of the CCA analysis. This vector shows the linear combination of bandfilter values that correlates optimally with the fundamental frequency. The eigenvector that results from the canonical correlation analysis is very similar to the eigenvector in figure 9.4. This again indicates the correlation between fundamental frequency and male-female differences.

9.3.3 Procrustes normalization

The previous sections have shown that differences between the male and female vowel spaces exist. The question of this section will be whether these differences can be reduced in some way.

As figure 9.1 shows, where the average male and female spectra are displayed with the symbols ‘m’ and ‘w’, respectively, the first thing that comes to mind is a translation to map these averages on top of one another. In our exploration about other possible

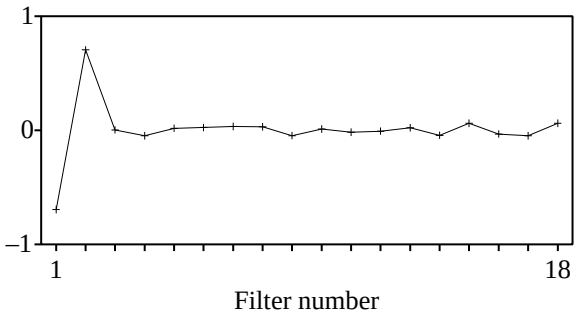

Figure 9.6. The spectral eigenvector that results from a canonical correlation analysis between fundamental frequency and bandfilter values.

manipulations besides translation, we will limit ourselves to shape-preserving transformations. We may translate, scale, rotate or invert the vowel space but we do not want to change the relative distances between the vowels. These relative distances have to be invariant under a transform. The allowable transform that may be a combination of scaling, rotation, reflection and translation is called a Procrustes transform. We have given some background on the Procrustes transform in section 7.4.6. The Procrustes transform boils down to the same feature space transform as was used by Molau (2003, ch. 7). He used a rotation constructed from the product of the eigenvectors of two covariance matrices. The two sets of eigenvectors were first aligned. A Procrustes transform combines the rotation and the alignment of the eigenvectors automatically. The only big decision remaining is whether the transformation is based on the unequal distribution of the vowels or not. We have chosen to give all vowels equal weights.

We use a Procrustes transform to transform the average female vowel system to the male vowel system as closely as possible. In part (a) of figure 9.1, we have indicated the positions of the transformed female vowels as the endpoints of the dotted lines that start at the female average vowel positions. The figure shows that the transformed female vowel positions fit the male positions much better than the untransformed ones. The transformation on the bandfilter spectra seems to work remarkably well. Closer inspection of the 18-dimensional Procrustes transform that actually accomplishes this mapping reveals that the scale factor was only 1.015. The translation vector was different from zero.

To further investigate whether the Procrustes transform could help discrimination, we applied the transform to the F-S data set. As table 9.2 shows, a discriminant classifier trained with M-S has 0.663 fraction correct when tested with the same male data set, but only 0.233 fraction correct when tested with the female vowel summary data set F-S. After applying the Procrustes transform to set F-S, the fraction correct increased to 0.583, which is a large improvement. If we perform the same transforma-

tion procedure on the F-S-9 set, i.e. first obtaining the male and female vowel centroids in the 9-dimensional space, then calculating the Procrustes transform and applying it to F-S-9, the fraction correct increases from 0.254 to 0.573, again a very large improvement. To get a better view on the impact of the Procrustes transformation on the bandfilter spectra we have plotted in figure 9.7 the male and the transformed female vowel centroids. The layout of the figure is identical to figure 9.2. The male spectra are drawn with solid curves while the female spectra are dotted. Again the plot in the middle shows the average spectra. They fall together. The Procrustes transform seems to be very powerful since the differences between the male and female vowel spectra as shown in figure 9.2 are reduced to almost nothing. For most of the vowels a perfect match between the male and the Procrustes-transformed females occurs. Only some vowels, namely uw, ae and ow show minor differences. These differences were also apparent in figure 9.1, where these transformed vowels show the largest difference in position with the male vowels (the transformed female vowels are at the endpoints of the dotted curves).

9.4 Bias adaptation

9.4.1 Introduction

In this section we will continue the investigation that was started in chapter 6 about whether partly retraining a neural net can equalize differences between vowel spectra from different speakers. The starting point is a neural net that has been trained on vowel data. We model speaker adaptation by only retraining a small part of the weights of the neural net for each new speaker. The motivation for doing so is that the positions of the vowels in the vowel spaces of different speakers are comparable. Therefore the expectation is that differences between these vowel spaces can be modelled by changes in only a relatively small part of the weights of the neural network. Theoretically all possible subsets of weights could be chosen to be modified in this adaptation process, however, as was shown in chapter 6, bias changes can be most easily interpreted. The intention is that by allowing only the bias weights of the neural net to be changed, the neural net is able to adapt to each speaker to better discriminate the speaker's vowels. We will use a supervised feedforward neural net to implement these ideas. To reduce some of the training times for these types of networks, we will start with some data reduction techniques.

9.4.2 Data reduction

In the following sections where we want to train neural nets, we preferably want the training times to be short. The best way to achieve this is to reduce the data itself and this can be done, for example, by keeping the dimensionality of the data as low as possible. We therefore try to reduce the 18-dimensional representation of the bandfilter spectrum to a lower-dimensional one. However, we know that reducing the dimensionality can have a degrading effect on quality. In order to find a reasonable balance

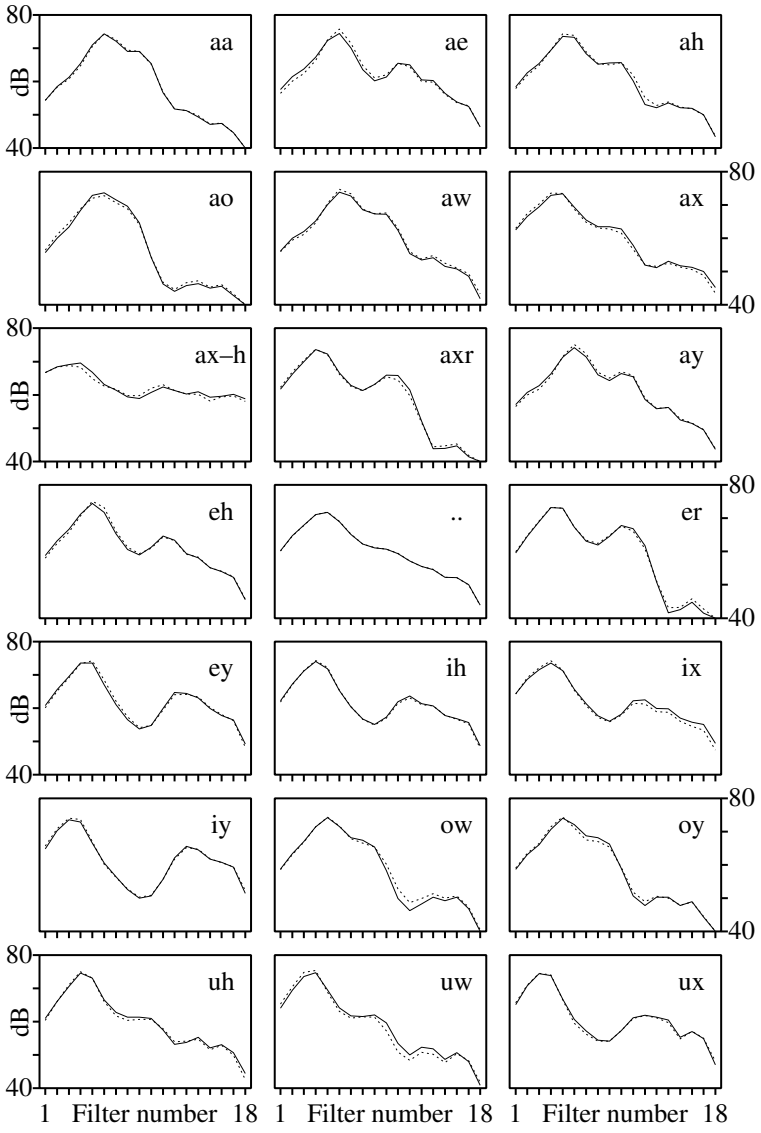

Figure 9.7. The male and female vowel centroids after a Procrustes transform. The male and the transformed female spectra are drawn with solid and dotted curves, respectively. The two spectra at the plot in the middle, i.e. at the fourth row and second column, are the average male and average transformed female spectra; they fall together.

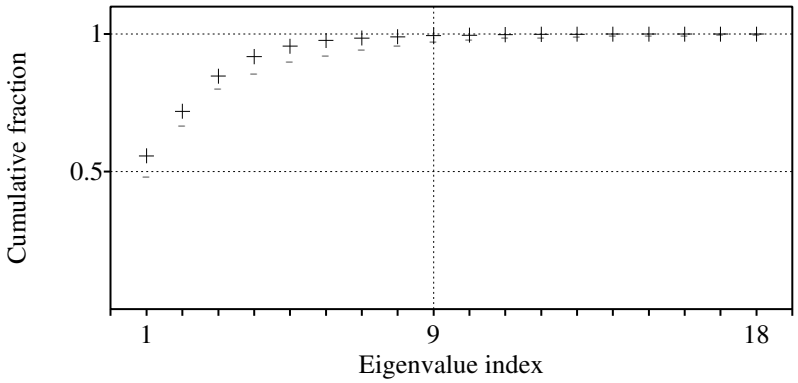

Figure 9.8. The cumulative contribution of the eigenvalues from a principal component analysis (+) and discriminant analysis (-) on data set MF-S.

between data reduction and quality we decided to reduce the dimensions with a factor of two. A reduction to nine dimensions, i.e. principal components, was chosen as a compromise between optimal variance and optimal discrimination quality and is corroborated by figure 9.8. The figure shows that at nine components, the cumulative fractional contribution of the first nine eigenvalues sums almost to 1, or, in complementary terms, the last nine components only contribute a very small fraction, 0.027, of the total variability in the material. According to the naming scheme in section 9.2, the nine-dimensional tables that result by extracting only the first nine principal components from tables M-S, F-S and MF-S will be named M-S-9, F-S-9 and MF-S-9, respectively.

9.4.3 Neural net parameters

The topologies of the feedforward neural nets that we will use are $(9, h, 20)$, where $h = 1, \ldots, 9$. The inputs will be the first nine principal components, the outputs the 20 vowel classes. Since we do not know the optimal number of hidden nodes in advance, we have tested a number of topologies where the number of hidden nodes varies from one to nine. As was shown in section 6.4, the number of parameters in a two-layer net with n inputs, h hidden nodes and m outputs is $h(n + m + 1) + m$. In section 6.5 it was shown that a covariance matrix in n dimensions has $n(n + 1)/2$ independent parameters. Therefore linear discriminant analysis, where all classes share the same covariance matrix, has $mn + n(n + 1)/2$ independent parameters, whereas quadratic discriminant analysis, where we need one covariance matrix for each class, has $mn + mn(n + 1)/2$ independent parameters. In these two expressions the term mn gives the number of independent parameters for m centroids in an n-dimensional space. For fixed inputs and outputs ($n = 9$ and $m = 20$) we find $h = 7$ and $h = 36$ when

we equate the neural net equation to the LDA and QDA equations and solve for the number of hidden nodes h, respectively. An upper limit of nine hidden nodes was chosen to keep the number of parameters in the neural net comparable to the number of parameters for the LDA.

To decrease network training time for finding an optimal solution it sometimes helps to increase the number of hidden nodes, since this introduces extra freedom in the minimization task. However, increasing the number of hidden nodes also increases the risk of overtraining the network. At least two remedies exist against overtraining: reducing the number of hidden nodes and early stopping. We decided to give the net some extra freedom by choosing the maximum number of hidden nodes comparable to the number of parameters for the LDA and have some extra freedom by choosing an upper limit of nine hidden nodes. By selecting nine hidden nodes we hopefully stay out of the overtraining region. By early stopping the learning task, we try to avoid that the neural network zooms in too much on the specific training data and again, hopefully, remains at a stage where the generalization performance of the network is still reasonable. The idea is that after a certain number of epochs, the two criteria *performance on the training set* and *performance on the test set* start to diverge. This moment would be the theoretically optimal time for stopping the minimization. Experiments show that this moment is often very difficult to obtain (LeCun et al., 1998). We did not determine this optimal moment in time yet have fixed the maximum number of iterations in the learning tasks to equal 1000 epochs. Several trials showed that the cost functions levels off at this number of iterations.

9.4.4 Test procedure

We train neural nets of topology $(9, h, 20)$, where $h = 1..9$. These nets have $30h + 20$ parameters. As training sets we use M-L-S and F-L-S. The test sets will allways be independent from the training sets.

We use batch training and the maximum number of epochs was set to 1000. As an error criterion the *Minimum Squared Error* (MSE) measure was taken. This measure has been shown to be very effective (see chapter 5 for a comparison of error measures). Figure 9.9 shows the typical development of the error during the training of a neural network with topology $(9, 9, 20)$.

Now, after the neural net has learned the data set we perform the adaptation process as follows. The data in the test set have been grouped according to speaker identity.

1. We make a copy of the trained neural net and arrange its parameters in such a way that only the biases of the hidden or the output layer are allowed to be modified during adaptation. We note that when the bias weights were then initialized with uniform random numbers drawn from the interval $[-0.1, 0.1]$, the minimization algorithm always got stuck in a local minimum and it was not able to find optimal values for the biases from these starting positions. For the biases of the output nodes it did not matter: good minima could always be found. We therefore did not initialize the biases with random values but started the minimization in the adaptation step with the biases as they resulted from the training phase.

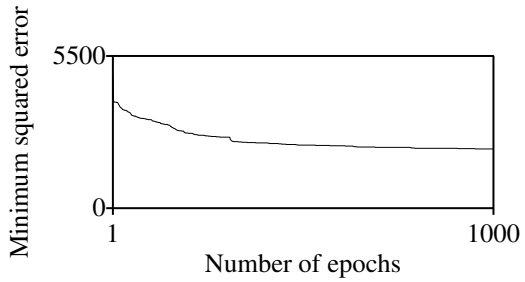

Figure 9.9. Typical development of the cost function during the training of a neural network with topology $(9, 9, 20)$.

2. We select the vowel spectra for a particular speaker and again train the neural net. However, this time only the selected biases are allowed to change. Since the number of data is much smaller than before (maximally 20 spectra) and only a limited number of bias weights have to be modified, we limit the maximum number of allowed epochs in this stage to 100.

3. We then use the neural net as a classifier for the data of the selected speaker and record the fraction correct. Given the amount of data available for the tests with formant frequency values in chapter 6, the test and training sets were identical. Although we have more bandfilter data available, we decided to take identical test and training data sets as well.

4. We start again with step 1, until all test speakers have been processed.

This procedure was repeated for each of the nine different neural net topologies $(9, h, 20)$. Because the learning procedure in a neural net is a very complex minimization in a high-dimensional space, the outcome will not be reproducible. To see whether results were reproducible at all, the whole procedure, i.e. training the neural net and testing the adaptation behaviour for all the test speakers with different neural net topologies, was repeated ten times.

9.4.5 Results

In figure 9.10 we have displayed summaries of the adaptation results. In part (a) the training and test sets were M-L-S-9 and M-T-S-9, respectively, coded as MM. We recall from section 9.2 that these sets are average spectra from male speakers in the training and the test parts of TIMIT, respectively. The same training set was used in part (b), but now the test set was F-S-9, i.e. the complete female summary set coded as MW. Part (c) and (d) also have the same training set, F-L-S-9 but different test sets, F-T-S-9 and M-S-9, respectively. Therefore, in parts (a) and (d) we have a learning task in which the data in the training and test sets belong to the same speaker group,

males and females, respectively. In parts (b) and (c) the training and test data belong to different speaker groups.

In all plots in figure 9.10 the horizontal axes show the number of hidden nodes from 1 to 9, while the vertical axes show the fractions correct from 0 to 1. The "baseline", i.e. no adaptation at all, is indicated with '0' symbols. These symbols represent the fractions correct when testing the trained neural net with an independent test set. The '1' and '2' represent the fractions correct when the biases of the first or the second layer, i.e. the hidden or the output layer of the network, respectively, were adapted. We note that in all four sub-figures adaptation is effective and shows larger fractions correct than the baseline. The adaptation of the output layer '2' has a slight advantage over the adaptation of the hidden layer. We see for the three conditions '0', '1' and '2' that the fractions correct gradually increase and then level off.

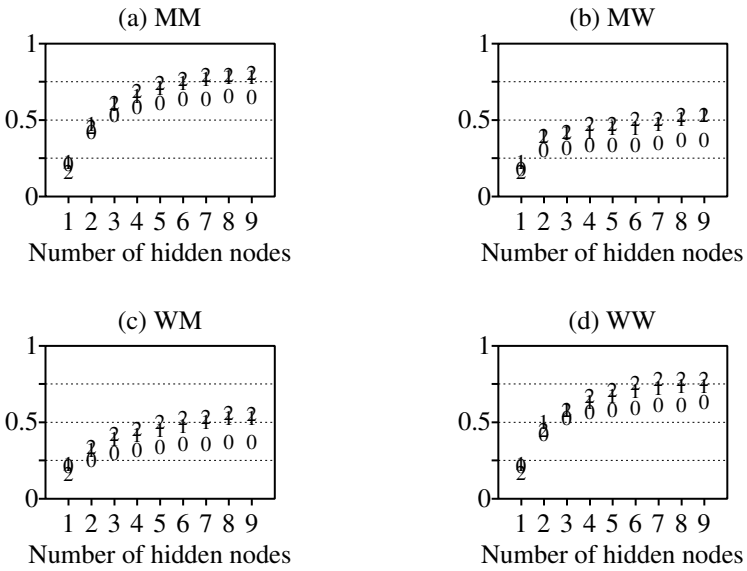

Figure 9.10. Fractions correct classification as a function of the number of hidden nodes in a feedforward neural net with topology $(9, h, 20)$. A '0' indicates the result without speaker adaptation. The symbols '1' and '2' shows results for speaker adaptation of the biases in layer one and two, the hidden and the output layer, respectively. The training and test sets were, respectively: (a) M-L-S-9 and M-T-S-9 (b) M-L-S-9 and F-S-9 (c) F-L-S-9 and M-S-9 (d) F-L-S-9 and F-T-S-9. See section 9.2 for nomenclature.

9.5 Discussion

As we found in chapter 6 for formant frequency values, the bias adaptation model also seems to work well for bandfilter spectra. The classification performance is reasonable, even when compared to human listeners. Cole & Muthusamy (1992) reported that human listeners, presented with vowel sounds from a set of 16 isolated monophthongal and diphthongal vowel classes from TIMIT, agreed about 55% of the time with the labellers. In ten one-hour sessions these listeners where confronted with a random permutation of the set of 2668 vowel sounds, being 16 vowels from 168 male and female speakers from the test part of TIMIT. With more than three hidden nodes figures (a) and (d) in 9.10 show that the neural net without adaptation performs equally well as human listeners when the training and test sets belong to the same speaker category. Bias adaptation, both for the hidden layer and for the output layer, show comparable behaviour, and improve on the results above. However, when training and test sets belonged to different speaker groups, as was the case in figures (b) and (c) of 9.10, the fractions correct were not so impressive. Only for a sufficient number of hidden nodes and adaptation to individual speakers could the performance of human listeners be attained.

As was already indicated by figure 9.1 and confirmed again with these adaptation tests, the differences between male and female vowel spectra cannot be fully annihilated by simply adapting the biases. These positive results for formant frequency data can be explained by the fact that a bias change in the hidden layer has the same effect as a uniform scaling of the formant frequencies. This is like vocal tract length scaling, which is a very succesful formant frequency normalization procedure (Molau, 2003). This is in contrast with the results based on formant frequencies, where we saw excellent performance when test and training sets belonged to different speaker categories. Clearly the type of transformation performed by modifying the biases is not powerful enough to overcome the differences between the male and female bandfilter spectra.

As we saw in section 9.3.3, the differences between male and female bandfilter spectra could be annihilated to a large extent by a Procrustes transform. This type of transform is based on more effective data scalings than are possible by bias weight modifications alone and produces almost identical average spectra, as figure 9.7 showed. The neural net approach and the Procrustes transform could be combined in a two-stage process if we try to classify both male and female data with a reference set obtained from only one speaker category: first Procrustes transform and then neural net adaptation. Being only a minor variation of the same supervised neural net model that has been tested in this chapter, we will not elaborate on this variant any further. In the next chapter we test a potentially more powerful neural net model and try to find a better speaker adaptation model.

Chapter 10

CategoryART: A variation on Adaptive Resonance Theory neural networks[‡]

Abstract

In this chapter we describe our CategoryART, a variation on adaptive resonance theory (ART) neural network models. CategoryART is a predictive ART architecture because it incorporates an ART module and is able to learn to predict a prescribed category given a prescribed n-dimensional input vector. In contrast to ARTMAP, which contains two ART modules, CategoryART contains only one ART module and the map field algorithm has been simplified. The ART module in a CategoryART neural network model can be either a FuzzyART or an ART2-A module. The CategoryART shows excellent performance on a benchmark neural network test, the artificial two-spirals problem, but unfortunately rather poor performance on a real-world vowel recognition test based on TIMIT.

[‡]This chapter is an extended version of Weenink (1997).

10.1 Introduction

In the previous chapter we have used feedforward neural networks to normalize vowel bandfilter spectra. Feedforward neural networks are based on supervised learning and need a lot of training material to adapt to a new speaker. In this chapter we want to explore neural network models that are based on unsupervised learning, in order to test whether these networks show potential in speaker-adaptive vowel identification.

CategoryART is a neural network topology whose dynamics are based on Adaptive Resonance Theory (ART). ART was developed by Grossberg (1976, 1980, 1986) as a theory of human cognitive information processing. ART was the result of an attempt to understand how biological systems are capable of retaining plasticity throughout life, without compromising the stability of previously learned patterns. Somehow, biologically based learning mechanisms must be able to guard stored memories against transient changes, while retaining plasticity to learn novel events in the environment. This tradeoff between continued learning and buffering of old memories Grossberg calls the *stability-plasticity dilemma*. It poses special design problems. For example, in the (supervised) feedforward networks new information gradually washes away old information. Therefore feedforward networks cannot be made stable in a changing environment.

To be able to mimic biological behaviour, the emphasis of ART neural networks lies on *unsupervised learning* and *self-organization*. Self-organization means that the system must be able to build stable recognition categories in real time. Unsupervised learning means that the network learns the significant patterns on the basis of the inputs only; there is no feedback. There is no external teacher that instructs the network to which category a certain input belongs. Other types of learning are *reinforcement learning* and *supervised learning*. In reinforcement learning the net receives only limited feedback, like "on this input you performed well" or "on this input you have made an error". In supervised mode a net receives for each input the correct response. According to Grossberg, unsupervised learning is the substrate on which the other types of learning are based. In biological systems, learning always starts as unsupervised learning: for the newly born hardly any pre-existing categories exist. A system that can learn in unsupervised mode, can always be adjusted to learn in reinforcement mode and in supervised mode. However, a system specifically designed to learn in supervised mode will never be able to perform in unsupervised mode. Needless to say that in unsupervised mode we cannot have a separate training and performance phase, like in supervised mode, because this implies the presence of a homunculus that knows when to alter phases.[1]

These design constraints have led to a series of real-time ART neural network models for unsupervised category learning and pattern recognition. Model families

[1] Boersma, Escudero & Hayes (2003) describe the acquisition of language-specific sound categories as a first and unsupervised stage of a two-stage learning model, in accordance with the proposal of Functional Phonology (Boersma, 1998). In the first *auditory-driven* stage, the statistical distribution of auditory phonetic information leads ultimately to the creation of phonetic categories. These phonetic categories turn into simple abstract phonological categories in the transition to the second stage. During the second *lexically-driven* stage, more abstract representations are developed and multi-dimensional perception will be optimized. The lexicon then acts as a supervisor for achieving more accurate perception.

include ART1, which can stably learn to categorize binary inputs presented in an arbitrary order (Carpenter & Grossberg, 1987b), ART2, which can stably learn to categorize either analog or binary data (Carpenter & Grossberg, 1987a) and ART3, which can carry out parallel search of distributed recognition codes in a multilevel network hierarchy (Carpenter & Grossberg, 1990). The FuzzyART model of Carpenter, Grossberg & Rosen (1991b) is based on fuzzy logic computations and incorporates the ART1 model, since computations from fuzzy set theory reduce to binary computations when the fuzzy variables become binary valued.

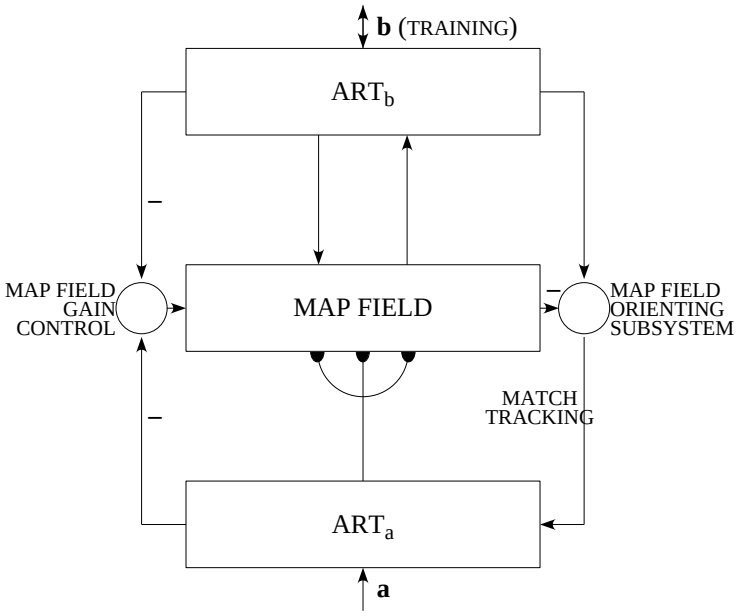

Figure 10.1. Block diagram of a supervised ARTMAP system. Two ART modules are linked by an inter-ART module called the map field. The map field forms predictive associations between categories of the ART modules and realizes a match-tracking rule. If ART$_a$ and ART$_b$ were disconnected each module would self-organize category groupings for their respective input sets.

Besides networks based on these models, *supervised* network architectures have been developed that incorporate one or more of the unsupervised ART modules given above. ARTMAP is one of these architectures (Carpenter, Grossberg & Reynolds, 1991a). Figure 10.1 shows a block diagram of such a system. In supervised mode, mappings are learned between input vectors **a** and output vectors **b**. A familiar exam-

ple of supervised neural networks are feedforward networks with backpropagation of errors, also called backpropagation networks (BP). In chapter 4 an introduction was given to the classifying capabilities of this type of networks and in chapters 6 and 9 they were succesfully applied in adaptive vowel normalization tasks based on formant frequency values and bandfilter values, respectively. Supervision is, however, the only similarity of BP networks with ARTMAP networks. ARTMAP networks are *self-stabilizing* whereas in BP networks new information gradually washes away old information. A consequence of this is that a BP network has separate training and performance phases whereas ARTMAP systems perform and learn at the same time. Besides, ARTMAP networks are designed to work in *real time*, while BP networks are typically designed to work off line, at least during their training phase. Another difference is that while ARTMAP systems can learn both in a fast and in a slow *match* configuration, BP networks can only learn in a slow *mismatch* configuration. This means that an ARTMAP system learns, i.e. adapts its weights, only when the input *matches* an established category, whereas BP networks learn when the input does *not match* an established category. In BP networks there is always the danger of the system getting trapped in a local minimum whereas this is impossible for ART systems. However, in systems based on ART modules, learning may depend upon the order of the input patterns.

CategoryART, which we herewith introduce, is a specialized *fast algorithmic* variant of the ARTMAP class of neural network architectures. It performs incremental supervised learning of recognition categories in response to input vectors presented in arbitrary order. Under supervised learning conditions, CategoryART's internal control mechanisms create stable recognition categories by maximizing predictive generalization while minimizing predictive error, just as the ARTMAP architectures do.

CategoryART differs in several ways from the ARTMAP architecture in figure 10.1: only one ART module is present, and the map field has disappeared. Instead, a simpler algorithm replaces the dynamics of both components. The dynamics of the network, however, are still based on Adaptive Resonance Theory. Originally all learning equations in ART systems are written in the language of real-time systems, i.e. differential equations. In our implementation, as in most algorithmic variants discussed above, steady-state approximations are used that capture the essence of these dynamic equations. Hence we do not have to use integration methods nor will we use differential equations in the formulation of the dynamics of CategoryART.

10.2 Basic features of ART systems

The basic features of Adaptive Resonance Theory and its relation to perception are laid out in a great number of articles by Grossberg and his associates (see for example Grossberg (1986, 1998) for an overview). A block diagram for a typical ART system is displayed in figure 10.2. The main components are the *attentional* subsystem and the *orienting* subsystem. The attentional subsytem consists, among other things, of two fields of neurons, F_1 and F_2, where each field may consist of several layers of neurons. These fields are connected by feedforward and feedback connection weights. The

connection weights form the *long term memory* (LTM) components of the system and multiply the signals along these pathways. The name *short term memory* (STM) will be associated with the pattern of activity that develops on a field as an input pattern is processed. The orienting subsystem is necessary to stabilize the processing of STM and the learning in LTM. As can be seen from the figure, the F_1 field receives input from possibly three sources. These three input sources are the bottom-up input to F_1, the top-down input from F_2, and the gain control signal. To avoid the possibility that mere feedback from F_2 can generate spontaneous activity at level F_1, i.e. to avoid that the system hallucinates, system dynamics are limited in such a way that at least two of the three inputs must be active to generate activity at the F_1 field. This is called the *2/3 rule* in ART. The same rule applies to the three possible input sources for the F_2 level.

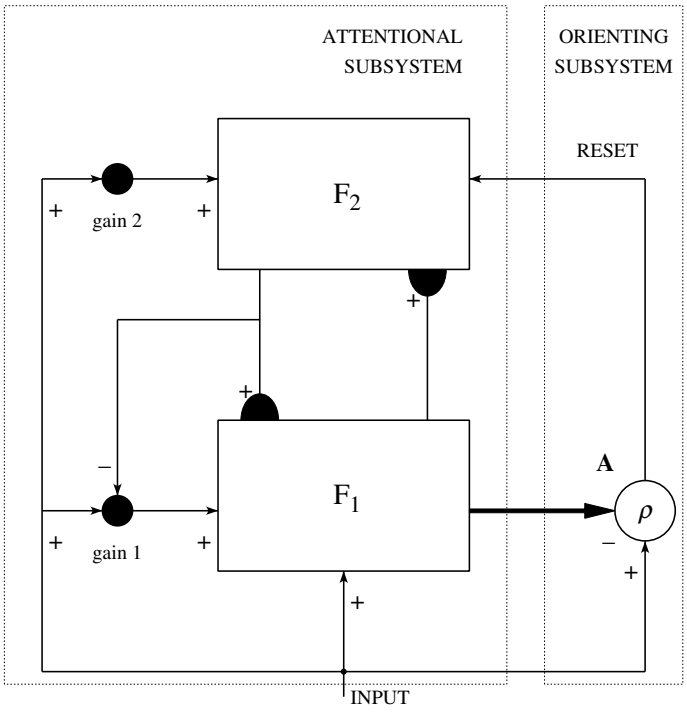

Figure 10.2. Typical ART neural network block diagram. After preprocessing, the input activity pattern is transformed to the first field F_1. Field F_1 is connected to field F_2 with feedforward and feedback connections which are indicated with black half ellipses. These connections form the long term memory components of this system.

All ART systems incorporate basic features, namely pattern matching between

bottum-up input and top-down learned prototype vectors. This matching leads either to a state that focuses attention and triggers stable prototype learning or to a self-regulating parallel memory search. This search ends in either of two ways. First, if an established category is selected in field F_2, then this prototype may be refined to incorporate new information in the input pattern. In this case when an input matches an established category, we speak of *resonance*. This resonant state persists long enough for learning to occur; hence the term *adaptive resonance theory*. Second, if the search ends by selecting a previously untrained node, then learning of a new category takes place. The criterion of an acceptable match is defined by a dimensionless parameter ρ called *vigilance*. Vigilance weighs how close an input must be to the top-down prototype for resonance to occur. Because the vigilance parameter can vary across learning trials, a single ART system is able to encode widely differing degrees of generalization. Low vigilance leads to broad generalization and more abstract prototypes than high vigilance. In the limit of very high vigilance, prototype learning reduces to exemplar learning.

With the help of the diagrams in figure 10.3, we will now follow in detail a typical ART search cycle. Not shown in this figure is the preprocessing field F_0 whose main purpose is a normalization of the input pattern.

(a) After the preprocessing by field F_0, an input pattern I generates a pattern of activity X at field F_1. The 2/3 rule is satisfied here because input I also activates the gain control at the F_1 level. The activation of the gain control is nonspecific because it does not depend on the type of pattern but only on its overall input activity. Pattern X both inhibits A and generates an output signal S from field F_1. Inhibition of A is necessary because otherwise a reset of field F_2 would occur. The signal S is multiplied by the bottom-up connection weights and results in a signal T that inputs to the F_2 level. The signal T produces an output Y from the level F_2. Here also the 2/3 rule is obeyed because the input signal I also nonspecifically activates the gain control for the F_2 level. The signal Y, for example, could result from the activation of the node(s) whose connection weights best matched the signal S.

(b) The pattern Y now generates a top-down signal pattern U which, after being multiplied by the top-down connection weights, results in the prototype pattern V. This prototype pattern V is compared at F_1 with the input pattern I. The result of this comparison is a new pattern of activity X^* at F_1. If V mismatches I at F_1, the resulting activity X^* will have significantly dropped. As a result of this reduction in total activity, less inhibition results at A.

(c) If now the vigilance criterion ρ at A fails to be met, A can release a nonspecific signal to F_2 which inhibits the nodes at F_2 that were most active. As a result the signal Y is reset as well as the feedback signal U and its prototype V.

(d) Pattern X is restored at F_1 and a different STM pattern Y^* becomes active at F_2because the Y nodes are still inhibited. If the top-down prototype due to Y^* also mismatches I at F_1, then the search for an appropriate code continues until

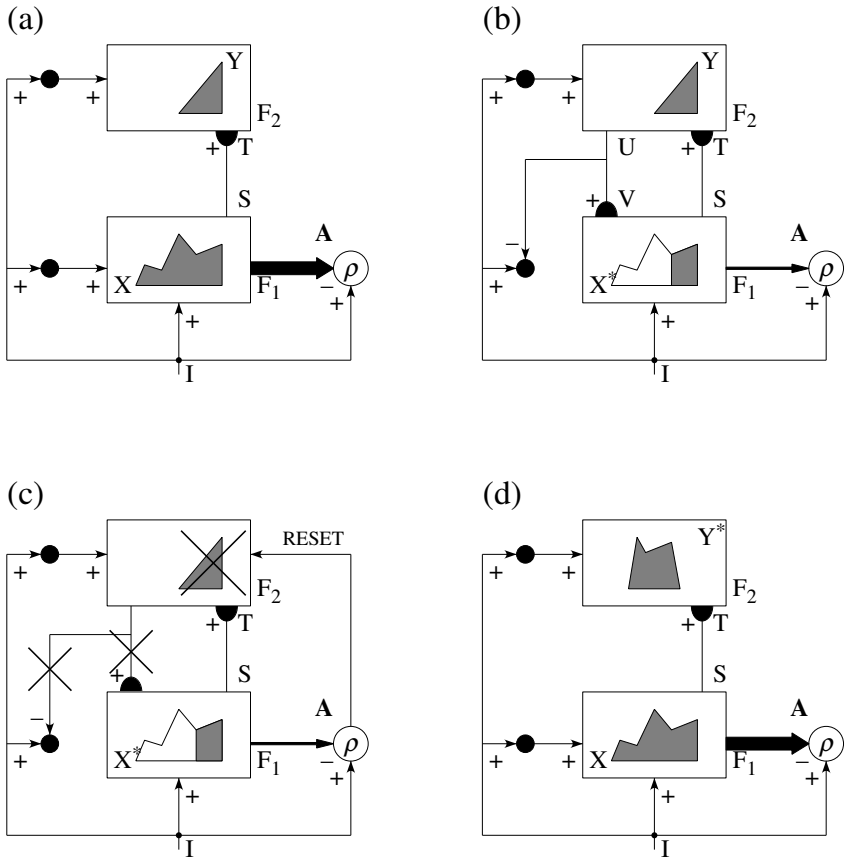

Figure 10.3. ART search for an F_2 code: (a) The input pattern I generates, after being properly normalized, the specific STM activity X at F_1 as it non-specifically activates the orienting subsystem A. Pattern X both inhibits A and generates the output signal pattern S. Signal pattern S is transformed into the input pattern T, which activates the STM pattern Y across F_2; (b) Pattern Y generates the top-down signal pattern U, which is transformed into the prototype pattern V. If V mismatches I at F_1, then a new STM activity pattern X^* is generated at F_1. The reduction in total STM activity which occurs when X is transformed into X^* causes a decrease in the total inhibition from F_1 to A; (c) If the vigilance criterion ρ fails to be met, A releases a nonspecific arousal wave to F_2, which resets the STM pattern Y at F_2; (d) After Y is inhibited, its top-down prototype signal is eliminated, and X can be reinstated at F_1. Once again X generates the input pattern T to F_2 and activates a different STM pattern Y^* at F_2, since Y remains inhibited. If the top-down prototype due to Y^* also mismatches I at F_1, then the search for an appropriate F_2 code continues (adapted from Carpenter & Grossberg (1991).

either a prototype has been found that satisfies the matching criterion at A, or a new category must be established at a previously uncommitted node.

Later we will describe how the ideas of this section can be implemented in the form of an algorithm for our CategoryART. However, before we can explain the supervised CategoryART algorithm, we first have to explain how a basic ART module works. As an example we take the FuzzyART module for unsupervised classification. This FuzzyART module will later be incorporated in the CategoryART model.

10.3 FuzzyART algorithm

The FuzzyART component in CategoryART consists of a preprocessing field of nodes, F_0, which modifies the current input vector $\mathbf{a}$, a field F_1, which receives both bottom-up input from F_0 and top-down input from the field F_2, and a field F_2, which is the output field. We do not need to distinguish between the connection weights of the top-down feedback paths and the bottom-up feedforward paths between the fields F_1 and F_2 in the FuzzyART module: both will be implemented by the same weights. Three parameters determine the dynamics of a FuzzyART network, a *choice* parameter $\alpha > 0$; a *learning rate* parameter $\beta \in [0, 1]$ and a *vigilance* parameter $\rho \in [0, 1]$. The influence of these parameters on the network dynamics will be explained in the following sections.

10.3.1 Preprocessing

When an M-dimensional input vector $\mathbf{a}$, which has all its activities a_i in the interval $[0, 1]$, is presented to the network it is first normalized by the field F_0. This normalization is necessary to guarantee stable category learning. The F_0 output activity vector $\mathbf{I}$ is a simple function of the F_0 input vector $\mathbf{a}$ and its complement vector $\mathbf{a}^c$, namely,

$$\mathbf{I} = (\mathbf{a}, \mathbf{a}^c) = (a_1, \ldots, a_M, 1 - a_1, \ldots, 1 - a_M).$$

The net result of this normalization operation is accordingly a doubling of the length of the input vector $\mathbf{a}$, while at the same time the norm of the new vector will always be equal to M. We use the following definition of the norm of a vector $\mathbf{x}$:

$$|\mathbf{x}| = \sum_{i=1}^{M} |a_i|. \tag{10.1}$$

We then get for the norm of $\mathbf{I}$:

$$|\mathbf{I}| = |(\mathbf{a}, \mathbf{a}^c)| = \sum_{i=1}^{M} a_i + \sum_{i=1}^{M} (1 - a_i) = M. \tag{10.2}$$

10.3.2 Category choice

The input vector $\mathbf{I}$ is now fed forward from the F_1 field to the F_2 field. Both fields are implemented with a single layer of $2M$ and N nodes, respectively. N is the capacity of the F_2 field and at the same time represents the maximum number of categories that this field can accommodate. All nodes in one layer are fully connected with all of the nodes of the other layer, i.e. each of the N category nodes in the F_2 field has $2M$ connections with field F_1. The connection strengths from category node j to the nodes in the F_1 field are represented by the weight vector $\mathbf{w}_j$ ($j = 1, \ldots, M$). Initially, before any learning has occurred all weights in $\mathbf{w}_j$ have the value 1 and each category node is said to be *uncommitted*. A weight vector $\mathbf{w}_j$ is also called a *template*. When a pattern $\mathbf{I}$ is presented at the field F_1, a choice function T_j is defined according to the following formula

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|},$$

where α is the choice parameter and $\wedge$ is the fuzzy AND operator, defined as

$$(\mathbf{x} \wedge \mathbf{y})_i = \min(x_i, y_i).$$

The fuzzy AND operator reduces to the Boolean AND operator in the case of binary vectors. The system is said to make a category *choice* when at most one F_2 node can become active at a given time. The F_2 node with maximum T_j will be chosen to represent the pattern $\mathbf{I}$, and, when the Jth category node is chosen, the output vector $\mathbf{y}$ of the field F_2 is set as $y_J = 1$ and $y_j = 0$ if $j \neq J$. In a choice system, the F_1 activity vector $\mathbf{x}$ obeys the equation

$$\mathbf{x} = \begin{cases} \mathbf{I} & \text{if } \mathrm{F}_2 \text{ is inactive} \\ \mathbf{I} \wedge \mathbf{w}_j & \text{if the } J\text{th } \mathrm{F}_2 \text{ node is chosen.} \end{cases}$$

If the chosen category J meets the vigilance criterion, that is if

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho,$$

then learning can occur. Mismatch reset occurs when the vigilance criterion is not met, and subsequently a new node is chosen. This search process continues until the chosen node satisfies the vigilance criterion. The search order among the nodes in the F_2 layer depends on the choice parameter α. If α is small then the search is more dominated by the pattern with the largest ratio $|\mathbf{I} \wedge \mathbf{w}_j|/|\mathbf{w}_j|$ than by the size of $|\mathbf{I} \wedge \mathbf{w}_j|$ alone. For larger values of α we see that the patterns for which $|\mathbf{I} \wedge \mathbf{w}_j|$ is large dominate the search. We could now make the following hierarchy for the F_2 nodes that will be chosen when an input pattern $\mathbf{I}$ is presented at the F_1 layer (Huang, Georgiopoulos & Heileman, 1995).

(a) If there is a subset node, it will be chosen over an uncommitted node. A subset node has a template $\mathbf{w}_j$ whose components satisfy

$$w_{ji} \leq I_i, \qquad \text{for} \qquad i = 1 \ldots M.$$

This means that for a subset node

$$\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} = 1.$$

(b) Because of the choice parameter $\alpha > 0$, among all the subset nodes, the node with the largest template $\mathbf{w}_j$ will be chosen first.

(c) An uncommitted node will be chosen whenever there are no subset nodes and all committed nodes j satisfy

$$\frac{|\mathbf{I} \wedge \mathbf{w}_j|}{|\mathbf{w}_j|} \leq \frac{1}{2}.$$

In our implementation of the FuzzyART algorithm in the PRAAT program we always maintain a list of committed and uncommitted nodes to speed up the search process.

10.3.3 Learning

The template vector $\mathbf{w}_j$ is updated according to the following equation

$$\mathbf{w}_J^{(\text{new})} = \beta(\mathbf{I} \wedge \mathbf{w}_J^{(\text{old})}) + (1 - \beta)\mathbf{w}_J^{(\text{old})}.$$

When $\beta = 1$ we speak of fast learning. For efficient coding of noisy inputs, we choose the fast learning option when J is an uncommitted node, and then take $\beta < 1$ after the node is committed. Then $\mathbf{w}_J^{(\text{new})} = \mathbf{I}$ the first time category J becomes active. After the commitment, the weight vector update causes the new weight vector to become more aligned with the most recently coded input pattern.

10.4 CategoryART algorithm

Our CategoryART neural network module is a simplification of the ARTMAP module. In the ARTMAP architecture, as shown in figure 10.1, two ART modules, ART_a and ART_b, are linked together via an inter-ART module, F_{ab}, called the map field. The ART_a and ART_b modules could be one of the set ART1, ART2-A or FuzzyART. Two choices are described in the literature: in ARTMAP (Carpenter et al., 1991a) two ART1 modules are combined, and in FuzzyARTMAP (Carpenter, Grossberg, Markuzon, Reynolds & Rosen, 1992) two FuzzyART modules are combined.

Input vectors to ART_a and ART_b are named $\mathbf{a}$ and $\mathbf{b}$, respectively, $\mathbf{x}_a$ and $\mathbf{x}_b$ are the outputs of the corresponding F_1 fields, F_{1a} and F_{1b}, and $\mathbf{y}_a$ and $\mathbf{y}_b$ are the outputs of the corresponding F_2 fields, F_{2a} and F_{2b}. For the map field, let $\mathbf{x}_m$ denote its output vector and $\mathbf{w}_{mj}$ denote the weight vector from the j-th node of F_{2a} to F_{ab}. The map field includes an associative memory and control signals and both are used to form predictive associations between categories of ART_a and ART_b and to realize the match-tracking rule. Match tracking means that a wrong prediction triggers the search

mechanism in the ART$_a$ module anew in order to look for a better match or, if a better match cannot be found, for a new category. Match tracking can reorganize category structure so that predictive errors will not be repeated on subsequent presentations of the same input.

ARTMAP can be used for mapping multidimensional vectors. However, when we want to associate category labels with multi-dimensional vectors, for example, vowel labels with spectral representations, using ARTMAP forces us to represent the category labels as a multi-dimensional input vector to the ART$_b$ network, and initializing ART$_b$'s vigilance to a very high level.

Two possible options for how to choose ART$_b$'s input vector when we have M different categories are: choose vector $\mathbf{b}$ of dimension M and make $b_i = 1$ for category i, or use a binary representation with a p-dimensional vector $\mathbf{b}$, where $2^p \geq M$. This vector $\mathbf{b}$ is processed by the ART$_b$ module, in which different input vectors (resulting from different category labels) should activate different output nodes of field F_{2a}. In effect, ART$_b$ categorizes one to one, each different input is represented by a different output node. This means that for each $\mathbf{b}$, only one output node is active and thus the norm of y_b equals one. As a consequence this makes the map vigilance parameter, ρ_{ab}, in the following match-tracking equation, which is equation (35) in the FuzzyARTMAP implementation of Carpenter et al. (1992), non-effective:

$$|\mathbf{x}_{ab}| < \rho_{ab}|\mathbf{y}_a|,$$

where $\mathbf{x}_{ab}$ is the output of the map field F_{ab}. We note that both in the ARTMAP, as well as in the FuzzyARTMAP implementations of Carpenter et al. (1991a, 1992), the map vigilance parameter is ineffective because the output of the ART$_b$ network, y_b, is always normalized to one.

In our CategoryART algorithm, the second ART system, ART$_b$, whose only function is to form a category representation, and the map field are replaced by an ordered collection of category labels and an array of pointers. There is a pointer to a category label for each node of the F_2 layer of the ART$_a$ module. The condensed CategoryART learning algorithm in pseudo code goes as follows:

Because of the combination of match tracking and fast learning, a single ARTMAP system can learn a prediction for a rare event that is different from that for a cloud of similar frequent events in which it is embedded. This means that eventually noise is also learned since the system cannot know beforehand what constitutes the signal and what the noise.

10.5 Simulation: Learning to tell two spirals apart

To get an impression of the capabilities of a CategoryART network we will describe its performance on a complicated classification task. We reproduce the example from Carpenter et al. (1992) in which they describe the FuzzyARTMAP network: learning to tell two spirals apart in a two-dimensional plane. This benchmark task cannot be learned by a standard feedforward network, which has connections from each layer to the next layer only. According to the authors cited above, Lang & Witbrock (1988)

```
forall (pattern p, categoryLabel c)
    learn (pattern p, categoryLabel c)
        if (categoryLabel c ∉ categoryLabelList)
            create categoryLabel c
            add categoryLabel c to categoryLabelList
        end if
        J ← categorize p by ART_a network
        if (categoryLabelList[nodePointer_J] ≠ c)
            temporarily increase vigilance
            J ← categorize p by ART_a network
            reset vigilance
        end if
        updateWeights (w_J)
        nodePointer_J ← index (categoryLabel in categoryLabelList)
    end learn
end for
```

Algorithm 10.1. The CategoryART learning algorithm.

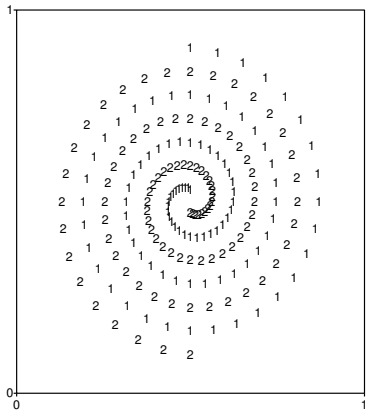

Figure 10.4. Two spirals in the plane. Each spiral consists of 97 points.

succeeded by constructing a special 2-5-5-5-1 network with each node connected to all nodes in subsequent layers. The system had 138 trainable weights. With their fastest algorithm they needed at least 8000 epochs to complete the task, i.e. each of the 194 points in the training set responds to within 0.4 of its target output value, equal to 0 on the '1' spiral and equal to 1 on the '2' spiral. The spirals of the benchmark each make three complete turns in the plane and consist of 97 points, as is illustrated by figure 10.4. The coordinates of the points $(x_n^{(1)}, y_n^{(1)})$ and $(x_n^{(2)}, y_n^{(2)})$ of the two spirals are

given by

$$\begin{aligned} x_n^{(1)} &= 1 - x_n^{(2)} = r_n \sin \alpha_n + 0.5 \\ y_n^{(1)} &= 1 - y_n^{(2)} = r_n \cos \alpha_n + 0.5, \end{aligned} \tag{10.3}$$

where

$$r_n = 0.4 \left(\frac{105 - n}{104} \right), \quad \text{with} \quad n = 1 \ldots 97, \tag{10.4}$$

and

$$\alpha_n = \frac{\pi(n-1)}{16}. \tag{10.5}$$

The 194 points defined above define the training set for CategoryART. The test set also contains 194 points, which lie inbetween the points in the training set. These points are generated by equations (10.3) but now with slightly modified r_n and α_n:

$$r_n = 0.4 \left(\frac{104.5 - n}{104} \right), \quad \text{with} \quad n = 1 \ldots 97, \tag{10.6}$$

and

$$\alpha_n = \frac{\pi(n-0.5)}{16}. \tag{10.7}$$

A trivial solution with CategoryART is obtained by selecting for the vigilance parameter $\rho = 1$. In this case the network learns all patterns in one epoch with 100% correct classification. However, the network uses 194 category nodes for the classification, one node for each training pattern. This amounts to using 970 parameters for the classification: the 194 times four connection weights from the F_2 nodes in the FuzzyART module plus 194 category index pointers to either the first or the second spiral.

In a FuzzyCategoryART we have, in principle, four parameters that determine learning. The first three parameters are determined by the FuzzyART module namely the choice parameter α, the vigilance parameter ρ, and the learning parameter β. The fourth parameter *matchtrack* determines whether match tracking is on or off. The parameters that influence most the number of categories and therefore the number of weights, are the vigilance parameter and the matchtrack parameter. When match tracking is on, the network is capable of raising its vigilance level when a mismatch at the category index level occurs. The most effective strategy to lower the number of categories is to start with match tracking on and a very low vigilance, i.e. $\rho = 0$.

We performed two series of runs with the vigilance parameter increasing from 0 in steps of size 0.02 to 1.0, the learning parameter β fixed at 1, the choice parameter α fixed at 0.001. The first series had match tracking on, the second had match tracking off. The results are displayed in figure 10.5. For all combinations of the parameters the training of CategoryART completed within 15 training epochs. When match tracking was on, the percentage correct classification obtained was always 100%. The left plot in figure 10.5 shows the number of committed nodes as a function of the vigilance level. For vigilance levels smaller than approximately 0.45 the number of committed nodes stays at the very low value of 36. It shows a gradual increase in the number of committed nodes when the vigilance level increases to 0.96, still higher values of the

vigilance level show a steep increase in this number. The maximum, 194, is reached when the vigilance level is equal to 1.0. When match tracking is off, the percentage correct drops to 50% when the vigilance level is reduced, as the right plot in figure 10.5 shows. Because match tracking is off, the number of committed nodes drops much steeper, ultimately to only two committed nodes when the vigilance level drops below 0.4.

This shows that a CategoryART neural network is able to tell two spirals apart with only 36 nodes.

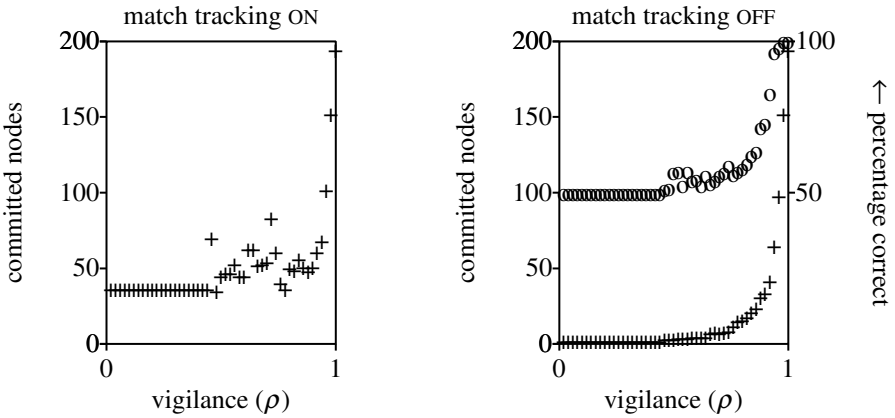

Figure 10.5. FuzzyCategoryART performance figures for the two spirals data set. The figure on the left shows the number of committed nodes as a function of the vigilance parameter when match tracking was active. In this case there was always 100% correct classification. For match tracking off, the figure on the right shows besides the number of committed nodes (+) also the percentage correct classification (o) as a function of the vigilance parameter.

10.6 A real-world test

The two-spirals simulation in the previous section was an example of the potential power of an ART-type neural network in a very special noise-free context with no overlapping classes. We now test CategoryART in a more real-world context where the data are not clear cut. The TIMIT database was introduced in chapter 8 and we try to classify the 20 different vowel classes from this database with a CategoryART type neural network. As can be seen from figure 8.8, in which the mean position of each vowel and its 0.5 σ ellipse are shown, there may be a great deal of overlap between the vowel classes. As input to the CategoryART network the bandfilter values are used. Because the 18 dimensions of the bandfilter measurements are not strictly necessary for the neural net processing, we use the same nine-dimensional training and test sets

as we were used in the previous chapter.

In order to gain more insight into the vigilance and learning rate parameters in training CategoryART with data with overlapping categories, we systematically vary both parameters. We train with set M-L-S-9, the male summary data from the training part of TIMIT, and test with set M-T-A-9, the male summary data from the test part. Table 9.1 indicates that these data sets contain 6008 and 13889 entries, respectively. We vary the vigilance parameter ρ in seven steps from a value of 0.7 to a value of 1.0 and the learning rate parameter β in five steps from 0.05 to 1.0. These 35 different parameter combinations are tested with match tracking *on* as well as with match tracking *off*. With match tracking *on*, the network artificially increases the vigilance level until a matching node has been found (or created). The choice parameter α is kept constant at a value of 0.001. This amounts to $70 = 7 \times 5 \times 2$ possible combinations of parameters in the training process: one could say that we have trained 70 different CategoryART's. In the training of each classifier the data were presented 20 times to the network (20 epochs).[2] After each training session, the fractions correct of the CategoryART were determined on the training and test sets. The tests for each of the 70 parameter settings were repeated 10 times and the results from these repetitions were averaged. The results for match tracking *off* are displayed in figure 10.6. In figure 10.6a the fraction correct is displayed as a function of the vigilance parameter ρ for the training set. We note a steady increase in fraction correct as the vigilance increases, leading to correct classification of all items in the training set when $\rho = 1$. In fact, we can obtain any fraction correct on the training set by properly choosing the vigilance level.

In figure 10.6b the combined effect of the learning rate and the vigilance on the fraction correct is displayed for the training set. We show the fraction correct as a function of the learning rate parameter with the seven different vigilance levels labelled from 1 to 7 (the lowest vigilance, $\rho = 0.7$, corresponds to number 1, the highest vigilance, $\rho = 1.0$, corresponds to number 7). The fraction correct on the training set seems to be a monotonically increasing function of the learning rate. We also note from the figure that the vigilance parameter has a much larger effect on the fraction correct than the learning rate parameter. Figure 10.6c shows the generalizing properties of the network. In the scatterplot we show the fraction correct for the test set versus the fraction correct for the training set at the seven different vigilance levels. The dotted line shows where the performances on the training and test sets are equal. The plot clearly shows that for the first five vigilance levels the performances on the training set and the test set are almost equal and increase monotonically with vigilance level. At vigilance level 6, where $\rho = 0.95$, the fraction correct levels off to a value of roughly 0.52. The value of 0.52 seems to be the maximum performance on the test set. Even if the fraction correct on the training set is at an absolute maximum, i.e. at a value of 1, the maximum attained fraction correct on the test set is 0.51.

In figure 10.6d we show the number of committed nodes as a function of the

[2] Only for the smallest learning rate, $\beta = 0.05$, the 20 epochs were necessary. For all other values the network was trained within fewer epochs. Huang et al. (1995) give results about possible combinations of ρ and β. However, most of their rules apply to situations where either patterns are binary, or fast learning is on, or both.

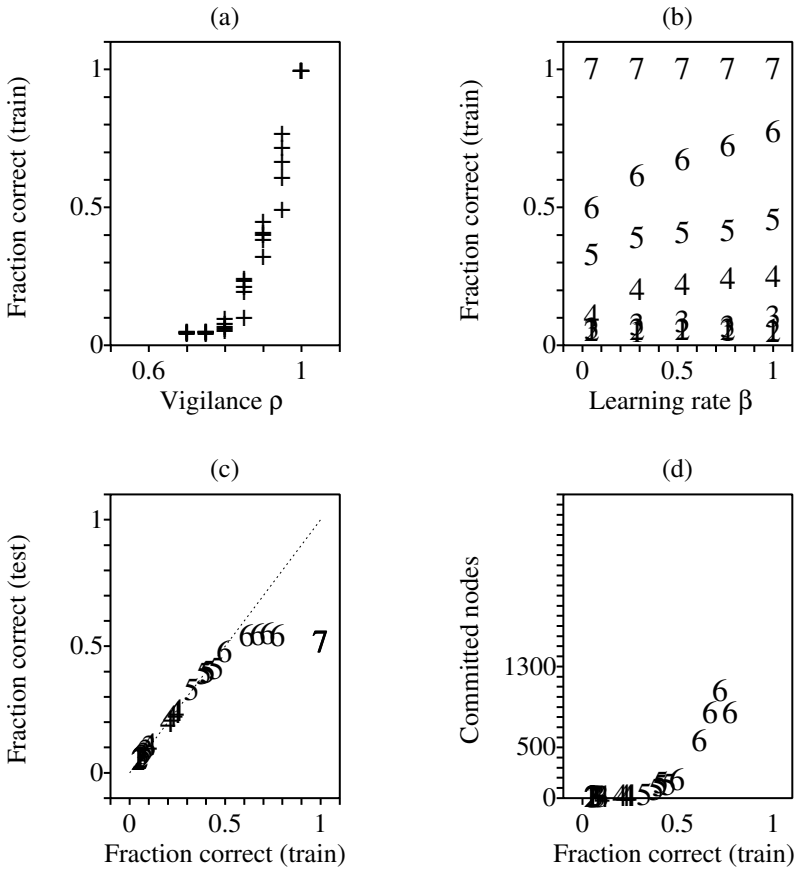

Figure 10.6. Summary of CategoryART training with TIMIT vowel data when match tracking is *off*. (a) Fraction correct classification of the training dataset as a function of the vigilance parameter ρ. (b) Fraction correct classification of the training dataset as a function of the learning rate parameter β. The seven different vigilance levels have been numbered from low to high with numbers 1 to 7, respectively. (c) Fraction correct classification of the test dataset as a function of the fraction correct classification of the training dataset. (d) Number of committed nodes as a function of the fraction correct classification. Vigilance level 7, i.e. $\rho = 1$ is left out since the number of committed nodes is equal to the number of items in the training set, i.e. 6008.

fraction correct for different values of the vigilance parameter. We have left out the seventh level since the number of committed nodes for $\rho = 1$ equals the number of items in the training set, 6008. When we compare these fractions correct with the numbers displayed in table 9.2 for discriminant classification, we must conclude that the generalization capabilities of the CategoryART are not too impressive with this kind of data. For example, the discriminant classifier scores 0.663 fraction correct on the (summary) test set, which is a clearly better result than 0.52. Also the feedforward neural network shows better results, as was shown by the '0' symbols in figure 9.10a when the number of hidden nodes exceeds four or five.

It is clear that the CategoryART exemplar-based network does not show strong generalisation properties under these circumstances. This is confirmed by figure 10.6d, which shows the number of committed nodes as a function of the fraction correct. We see a rapid increase in the number of committed nodes when the fraction correct exceeds 0.5, i.e. for high vigilance levels. This rapid increase is a manifestation of a too detailed representation of the training set by the neural net and results in low compression ratios. For example, if $\rho = 0.95$ the number of committed nodes varies between 500 and 1000. Giving a training set with 6008 items, the resulting compression ratio, i.e. the number of training items divided by the number of committed nodes, lies approximately between twelve and six and on average we need between 25 and 50 exemplars per class.

For completeness, we show in figure 10.7 the same plots as in figure 10.6 but now with match tracking *on*. These plots do not show the smooth and continuous behaviour of figure 10.6. For example, the fraction correct as a function of the vigilance ρ suddenly jumps to a much higher level for $\rho \approx 0.95$. Also, if we want to get past 0.5 fraction correct, the number of committed nodes increases dramatically.

The fact that CategoryART does not perform as well with the vowel database as with the two spirals example, must be attributed solely to the fact that the vowel classes show a considerable amount of overlap. For example, in section 5.3.1 we used the iris data which is four-dimensional and which shows almost no overlap between the three iris classes. With a CategoryART we can easily obtain a fraction correct of 1.0 with only six committed nodes using the following training parameters: $\rho = 0.7$, $\beta = 1$, *numberOfEpochs=10* and match tracking *on*. This data set has 150 items, so we have a compression factor of 25 and on average only two exemplars for each class.

10.7 Conclusions

The preliminary performance of the CategoryART neural network for the task was to tell two spirals apart was good. It was capable of learning this reasonably complex task in a very short time. For the vowel recognition task with a lot of overlapping classes, the performance was not at par with a discriminant analysis or a feedforward neural net. The latter two showed superior results with significantly fewer parameters. As for the number of parameters, Carpenter, Milenova & Noeske (1998) have devised a distributed variation of ARTMAP that reduces the number of committed nodes considerably. However, the classification performance of this new type of network was

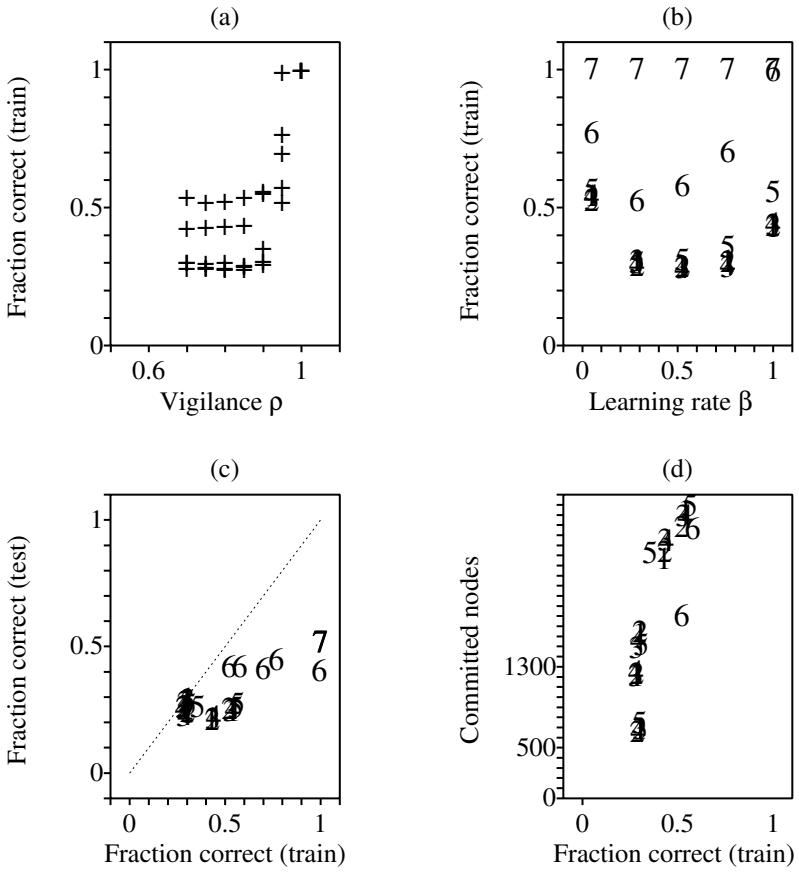

Figure 10.7. Summary of CategoryART training with TIMIT vowel data when match tracking is *on*. (a) Fraction correct classification of the training dataset as a function of the vigilance parameter. (b) Fraction correct classification of the training dataset as a function of the learning rate parameter. The seven different vigilance levels have been numbered from low to high. (c) Fraction correct classification of the test dataset as a function of the fraction correct classification of the training dataset. (d) Number of committed nodes as a function of the fraction correct classification of the training dataset.

not improved. We must conclude that for data with overlapping classes this ARTMAP type of network is not very well suited. What has been advertised as their strength, i.e. correctly classifying individual data points of a class positioned in the middle of data points belonging to another class, has become their weakness. For the batch processing that we perform, with noisy overlapping classes, we prefer discriminant analysis because of its simple algorithmic structure. And, as will be shown in the next chapter, discriminant analysis can be made adaptive as well.

Chapter 11

Adaptive speaker normalization for vowels with TIMIT[‡]

Abstract

In this chapter we present an adaptive speaker normalization procedure for vowels. The model is found to reproduce the difference in human vowel recognition performance for stimuli presented in blocked and mixed speaker context although the model does not incorporate any knowledge at all about speaker identity. The model is independent of the representation of the vowels and we have tested it on bandfilter and formant frequency representations of vowels.

[‡]This chapter is a modified version of Weenink (2001).

11.1 Introduction

In chapter 2 we described listening experiments that showed that subjects, when confronted with vowel-like stimuli from different speakers, show better recognition performance if successive stimuli come from the same speaker than if the speaker identity varies very often. These two experimental conditions are called blocked and mixed speaker context, respectively. Most of the time the mixed/blocked effect is not a large effect, only a few percent, but the effect is consistent and always statistically significant.

In previous chapters a number of vowel normalization procedures have been tested. In chapters 6 and 9 supervised feedforward neural nets were used. After proper training these neural nets were able to recognize vowels reasonably well. By making only the biases variable, we were able to have them adapt to the vowels of a new speaker by proper training. As for example table 6.3 on formant data and figure 9.10 show, this turned out to be successful. However, two disadvantages of the feedforward neural nets are that they are supervised and that the adaptation process needs a lot of training material when compared with human listeners. Because of the way these neural nets are constructed we need at least one training item for each output class. This means that only after proper training with the new, possibly deviant items the net is able to recognize them. Basing network topology on unsupervised topologies didn't help: the recognition performance of CategoryART, which was explored in chapter 10, was not very satisfying.

A successful adaptation method has to adapt faster than the methods discussed above. In order to adapt fast a method should not need one item for each class. Adaptation can be made fast if we either use symmetries or allow local differences to have global consequences. Exploitation of symmetries means that we take advantage of for example the same vowel height as occurs in /o/ and /e/. It implies however that we dispose of a database of known symmetries. We will not pursue the use of symmetries here. Instead, we have developed a model in which a local difference possibly induces a non-local change.

11.2 An adaptive speaker normalization procedure

A rather simple model in which a local difference has global consequences is one in which not only one class adapts to a new stimulus but more classes adapt at the same time. This can be accomplished, for example, by moving, instead of only one reference, (parts of) the whole frame of reference. The behaviour of our adaptive approach can be described as follows: when an unknown vowel enters, we first try to establish its identity. If its identity can be guessed reliably and if the position of this vowel deviates from its class mean, we then move *all* references a little bit in parallel to the difference vector in the direction of the newly identified vowel. In this way the references are not constant but shift along in vowel space.

In this section we will explain the details of this general idea with a specific model, based on a vowel representation with formant frequencies, that accepts the first two

formant frequencies of a vowel as input and produces one of the twelve Dutch vowel classes as output. The pre-assumptions of the model are that the average positions of the twelve vowel classes in the formant plane are known as well as the covariance structure of the formant space. These assumptions guarantee that a proper distance in the formant plane can be calculated. The working of this speaker-adaptive classifier can be visualised with the help of figure 11.1, which displays twelve Dutch vowels in the formant plane.

1. An unknown vowel's F_1 and F_2 are presented to the classifier. The position of the unknown vowel is shown with a question mark in figure 11.1a.

2. The distances of the unknown vowel to the reference vowels are determined. The arrows in figure 11.1b give an impression of these "distances" to the twelve references. The lengths of these arrows in general do not equal the real distances as calculated in this model. The "real" distances are Mahalanobis distances and depend on a covariance matrix.[1]

3. The shortest distances is determined. The shortest distance is shown in figure 11.1c with a solid arrow.

4. It is tested whether the distance meets a tolerance criterion, because we want to make sure that the unknown is "close enough" to the reference. We have implemented "close enough" as a criterion on the probability of group membership. In section 3.5.6 we have quantified this, as equation (3.24) testifies.

5. If the "close enough" criterion is satisfied, all references will be moved parallel to the direction of the difference vector as is indicated by the arrows in figure 11.1e. If the item is not "close enough" the reference system stays in place. However, the unknown is still classified as a member of the closest class.

If during the matching process the "close enough" criterion is satisfied, the reference system can shift its position. However, we do not know beforehand how far we should move the references. Therefore, we have parametrized this movement with a parameter α such that the new positions $\mathbf{x}'_i$ of references i become $\mathbf{x}'_i = \mathbf{x}_i + \alpha\mathbf{d}$, where $\mathbf{x}_i$ is the current position of reference i and $\mathbf{d}$ is the difference vector of the unknown with the "close enough" reference. The parameter α may in principle be any real number. We first note that for negative values the new reference position moves further away from the unknown position and therefore the distance of the new reference to the unknown increases. This is not a very interesting case and we will only consider positive values of α. When α increases from 0 to 1 the new reference moves closer towards the unknown, matching it exactly if α equals 1. For α greater than 1 the position of the new reference starts moving away from the position of the unknown. The larger α, the larger the shift of the positions of the references. It is clear that some bounds on α have to be established to guarantee the stability of the positions of the references. First of all we want our new reference to be closer to the unknown than the

[1] See for example equation (3.26) for a distance function when only the pooled covariance matrix is available.

current reference. If we include the limiting case where the distance stays the same, α has to be in the interval $[0, 2]$. However, in order to maintain a stable reference system that does not jump around from one input to the next input, the $[0, 2]$ interval is too large. Probably some value in the interval $[0, 1]$ will do. In the next section we will show some results for varying values of α.

11.3 Test with formant data

11.3.1 Introduction

As a first application of the model we will test it with the logarithmically transformed formant frequency data for the twelve Dutch vowels from 50 men and 25 women that we used before in chapters 3, 5, 6 and 7. The adaptive classifier discussed in the previous section has been implemented by making some modifications to the discriminant classifier that was described in section 3.3. Instead of using the fixed averages for each class, we keep running averages. The amount of change of these running averages will then be governed by two parameters, the step size α, which determines the amount of change, and the probability level p, which determines if there will be any change at all. Because preliminary runs showed that the value of p is less critical than the value of α, in what follows we only show results for p at the fixed value of 0.5.

11.3.2 Blocked versus mixed speaker condition

In figure 11.2 we show the results of a series of runs for blocked and mixed speaker conditions for different values of α. The results for blocked and mixed speaker conditions are indicated with a '+' and a '0' symbol, respectively. The sets used to train and test the classifier are indicated by the pairs of M and W symbols at the top of each subfigure, respectively. M is the set with data from the 50 male speakers and W is the set with data from the 25 female speakers.

We will now discuss this figure in detail and we will start by showing how the results in part (a) of figure 11.2 were obtained. We started by training the discriminant classifier with the male data to establish the references and the covariance structure in the vowel formant space. The order in which the data are presented does not matter, only the global covariances and averages do. For the testing of the adaptive procedure, presentation order does matter. We have devised therefore two different data permutation schemes, one for the blocked speaker condition and one for the mixed speaker condition. The male data set has 50 speakers, with twelve vowels per speaker and three formant frequencies per vowel, i.e. 50 blocks of dimensionality 12×3. In the *blocked* speaker condition, we first randomly permute the 50 blocks and then randomly permute the twelve items within each block. In this way, we always present the data from each speaker, one after the other, to the classifier but with a different vowel ordering for each speaker. In the *mixed* speaker condition, we first use an interleaving permutation algorithm, i.e., we form new blocks of size twelve by taking the first data item from the block of the first speaker, then the second item from the block of the second speaker, the third item from the third speaker, and so on, until the first

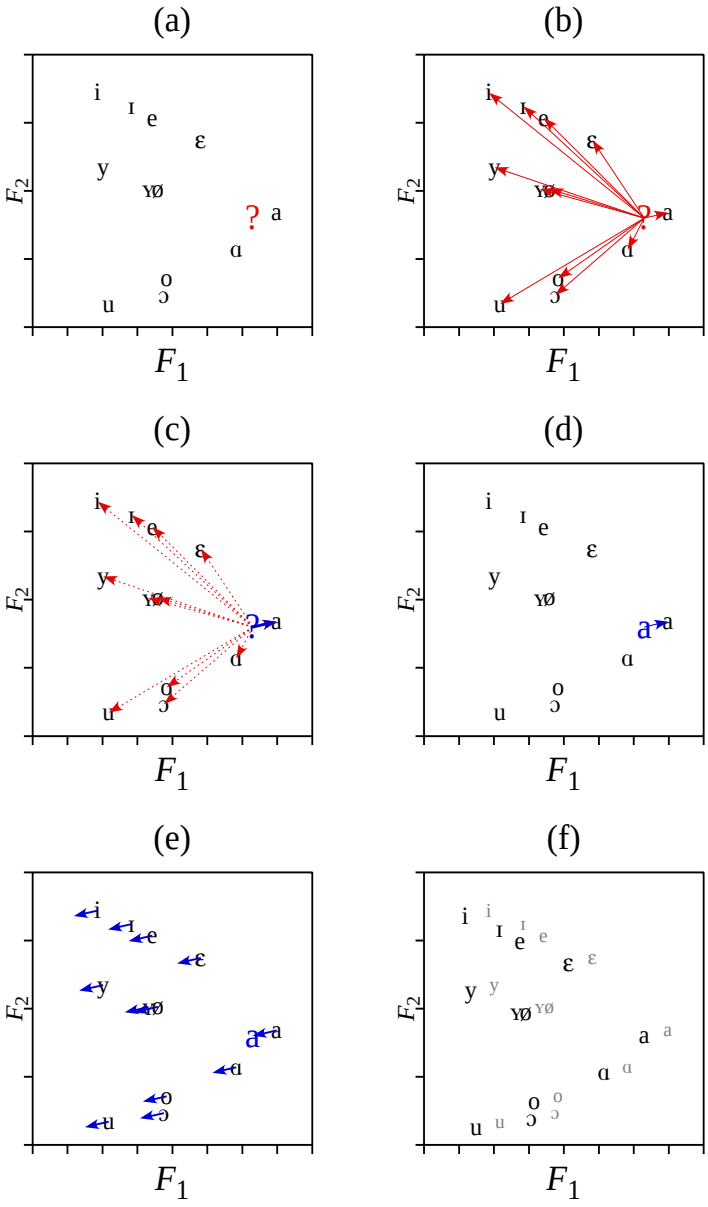

Figure 11.1. The adaptive vowel classifier at work. (a) An unknown vowel '?' is presented. (b) Distances are determined. (c) Find shortest distance. (d) Label as best match. (e) Move references ($\alpha = 1$). (f) The new references (old references in grey).

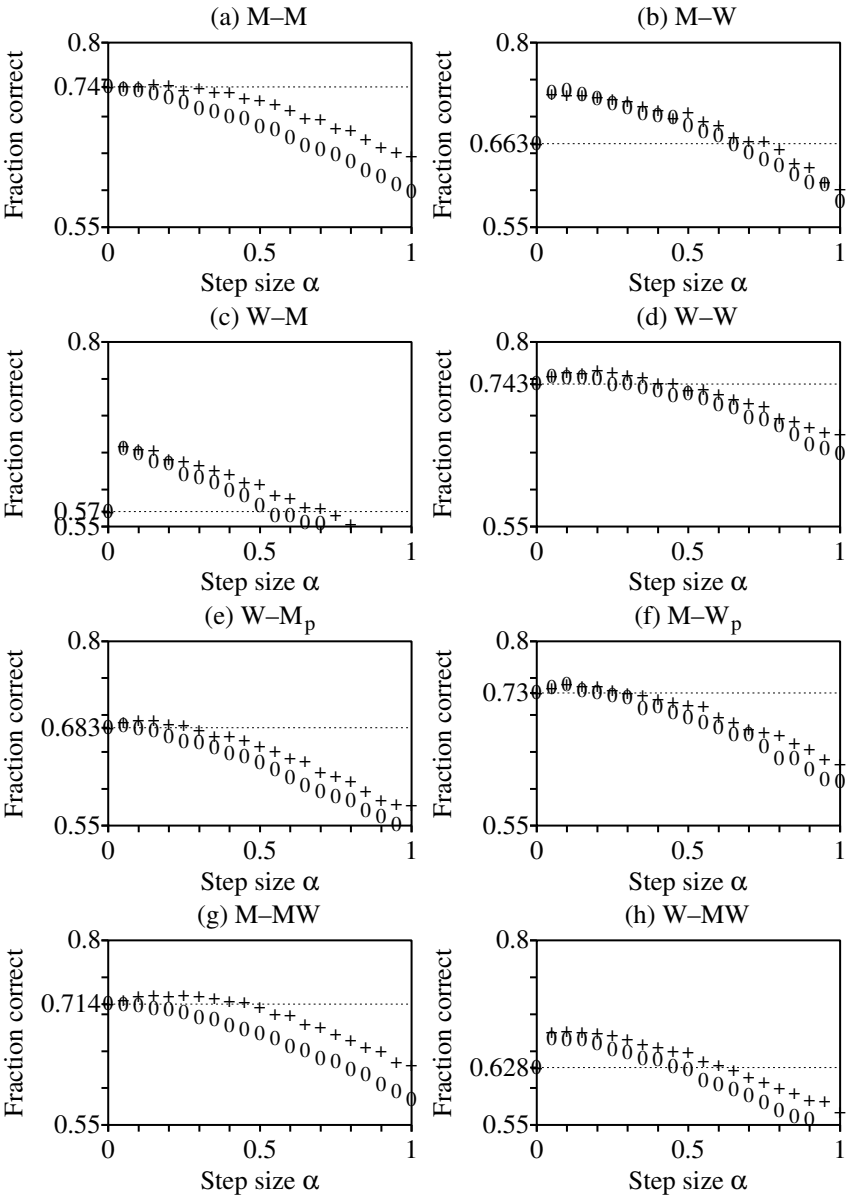

Figure 11.2. Results of the adaptive vowel classifier for blocked (+) and mixed (0) speaker conditions as a function of the step size α. The formant frequency data sets used to train and test the classifier are indicated by the pairs of M and W symbols at the top of each subfigure. M and W denote the sets with formant frequency values from 50 males and 25 females, respectively. For each value of α in the blocked and the mixed speaker conditions each result displayed in the figure is the average of 50 different random permutations of a data set (see text).

new block is full. The first item of the second new block will then be the first item in the 13^{th} old block, and so on. When we are at the end of block 50, at the second vowel, we start again with the first speaker at the third item, and so on. In this way we can always guarantee that the next item is from another speaker and each new block contains all twelve vowels. Next, in the same way as discussed before, we randomly permute these blocks and the items within a block. Of course, because of this last permutation it occasionally may happen that the last item in a block and the first item in the next block were equal. We have not explicitly excluded this possibility.

The fractions correct were determined in the following way. The data set for the blocked speaker condition, in the way described above, were presented to the trained classifier, after which its fraction correct classification was determined. Next the data set for the mixed speaker condition, permuted in the way described above, were presented to the classifier, and again its fraction correct classification was determined. These two measurements were repeated 50 times, every time with different permutations of the data set in the blocked and the mixed speaker condition, and the average fractions correct were determined and collected in a table together with the value of α. The step size α was first varied from 0.0 to 1.0 in small steps of 0.05, making a total of 21 different values for the step size. This means that for each subfigure we have performed $2100 = (21 \times 50 \times 2)$ random permutations of the test data set. We note the following phenomena in the figures.

- If $\alpha = 0$, the adaptation is effectively non-existent and the results will be the same as with the standard discriminant classifier. This fraction correct, the *baseline*, is indicated with a dotted horizontal line in each figure. For example, in part (a) of figure 11.2 the baseline is at 0.74 fraction correct.

- All subfigures (a)–(g) show a difference in fraction correct between mixed and blocked speaker condition. The difference is small for small values of the step size α but it increases when α increases. The blocked speaker condition always shows a higher fraction correct than the mixed speaker condition. This is remarkable: the algorithm does not know anything about different speakers, it does not know when a new speaker starts, it only knows its current reference system and the current item it has to process. Nevertheless it is able to use the blocked speaker context to its advantage to perform somewhat better than in a mixed speaker condition. In this way it qualitatively reproduces the results of human listeners who also perform somewhat better in a blocked speaker condition, as has been shown in chapter 2.

- We expect the difference between blocked and mixed speaker condition to be small at small step sizes because, since we only have twelve vowel items for each speaker, the maximum displacement that can be reached within these twelve items is only small. For example, given that the displacement of the speaker centroid of the current speaker has a distance of one unit to the centroid of the previous speaker, then if $\alpha = 0.01$, the maximum movement after twelve steps is 0.12 units into the direction of the current speaker's centroid. Despite the small displacements at small α-values, we still see fraction correct scores somewhat above the baseline.

- Adaptation helps a lot when training and test sets differ, i.e. in figures 11.2b and 11.2c where we have trained with the male and female data sets, respectively, and tested with the female and male data sets, respectively (M-W and W-M), we note a relatively large difference between the baseline and actual fractions correct in the blocked and mixed speaker conditions. We see a sudden jump in fraction correct from the baseline at $\alpha = 0$ to much higher values for α values that differ from 0, even for very small α's.

11.3.3 Visualisation of the dynamics

The workings of the adaptive algorithm given in the previous section can also be visualised in a dynamic fashion. In the classification of one vowel datum, the adaptation algorithm moves all references by the same vector. If for each vowel item we keep track of this displacement vector, we can actually follow the whole process by drawing this displacement vector for each classified vowel. If we were to show the displacements of all vowel references, we would find twelve subdrawings with the same relative movements. We therefore show the displacements only once, with respect to the overall speaker centroid. This is done in figure 11.3 for four different values of α. To avoid visual clutter, we have displayed the path for the first ten female speakers in the blocked speaker condition. This path is constructed from $120 = 10 \times 12$ possible displacement vectors for the individual vowel data. To be able to track the behaviour of the individual speakers, we have drawn the speaker number at the position at the path after all the twelve vowels of that speaker were processed. To get even more insight in the movements of the individual speakers, we also show the centroid positions of the speakers, with respect to the overall centroid of the training data, with larger numbers in grey. The overall centroid, averaged over the 50 males, is displayed with a capital M, this is the starting position of the path. The relative position of the overall female test data centroid is indicated with a capital W. The figures make very clear that even for small values of α, the fractions correct are well above the baseline: we see that the path roughly makes displacements in the direction of each speaker's centroid. The global effect of these local movements will be a movement in the direction of the overall centroid of the female speakers. In figure 11.3a we see that for the small step size $\alpha = 0.025$, the path is very smooth but the twelve items from the first speaker are not sufficient to reach the goal. The algorithm needs the data of several speakers to cover the distance from M to W. When the step size increases, the distance from M to W can be mastered much faster, however at the cost of the path becoming less smooth: much wilder jumps occur, as figures 11.3b, 11.3c, and 11.3d testify. We are confronted here with a stability-plasticity tradeoff: larger values for α make the system more adaptive to a new speaker; however, the references move more wildly, resulting in less stability.

11.3.4 Mixing male and female data

In order to investigate the effects of mixing male and female data, we have put the adaptive algorithm to some further tests. The practice so far has always been to sep-

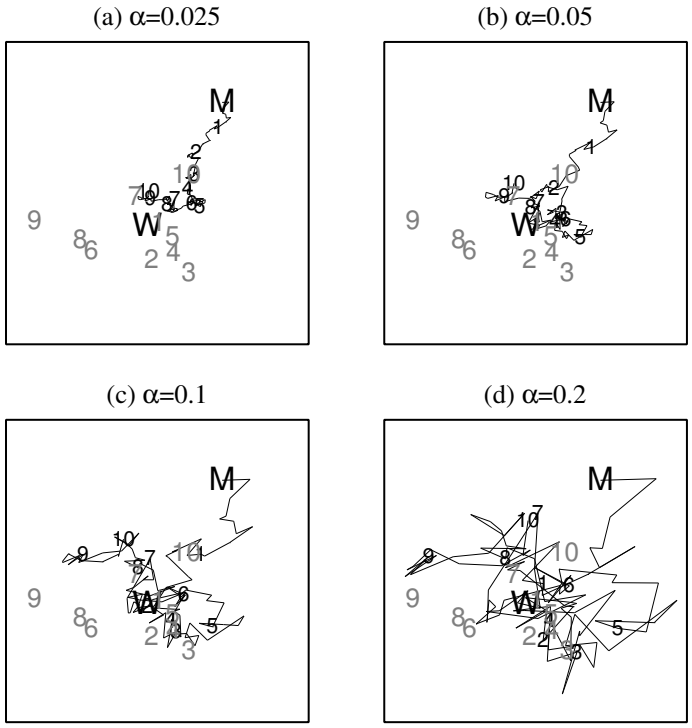

Figure 11.3. The adaptation path for the first ten female speakers with respect to the male reference space for different values of the step size α. The speaker centroids for the ten female speakers are indicated in grey with the larger number symbols. The capital M and W are the averages of 50 male and 25 female speaker centroids, respectively. The smaller numbers that occur on the path show the relative reference position after the data for the twelve vowels of that speaker have been processed.

arate male and female speaker groups and test with either the male or the female speaker group. What will happen if we mix both data sets in the tests, i.e. in the blocked speaker condition we still present the data for each speaker successively, but now a female speaker might follow a male speaker and vice versa? We have done this by appending the 50 males and 25 females data sets, making one data set with 75 speakers. We have taken the imbalance between the number of male and female speakers for granted and the applied permutations in the blocked and mixed speaker condition were just like the ones described.

The results of mixing the male and female data sets are displayed in figures 11.2g

and 11.2h. The text at the top of these figures, M-MW and W-MW, indicates that the difference between the two figures is the M and W data sets used for training. Both figures make clear that also when mixing male and female data, the adaptive model functions well. The fractions correct are above the baseline and the blocked speaker condition always shows a better fraction correct than the mixed speaker condition.

11.3.5 Comparison with Procrustes transform

In section 9.3.3 we introduced the Procrustes transform, a structure-preserving transform. 'Structure preserving' means that the transform is only allowed to rotate, reflect, translate or scale the data. This transform is obviously very general and we can apply it to formant data as well. Our adaptive algorithm has only a translation available to adapt to another speaker (group). We expect that the Procrustes transform, which has many more parameters to modify, constitutes the maximum that can be attained in "adaptability".

Figure 11.4 shows the mean vowel positions of Dutch vowels from male and female speakers before and after Procrustes transforms. The small and large symbols show the vowel positions averaged over the 50 male and 25 female speakers, respectively. The open endpoints of the solid lines show the vowel positions after a Procrustes transform that optimally matches the female averages to the male averages. The open endpoints of the dotted lines show the result of a matching where Procrustes transforms were calculated for each individual speaker separately. The capital M and W are the averages of 50 male and 25 female speaker centroids, respectively. Since only three dimensions are involved now, we can try to analyze the parameters of the Procrustes transform to interpret its behaviour. For the transform that tries to match the female data to the male data, the matrix involved is almost diagonal, the smallest element on the diagonal being 0.998. This means that rotations are very small and may be ignored. The only parameters left are the translation vector and the scaling factor, which are $(0.56, 0.35, 0.31)$ and 0.86, respectively. The net effect of this Procrustes transform is therefore: the female speakers' centroid is put on the male speakers' centroid and the vowel space is shrunk by a factor of 0.86. This scaling operation is not the same operation as a vocal tract length scaling because the latter operates from the origin whereas our scaling operates from the centroid. In figures 11.2e and 11.2f the fractions correct for the Procrustes transformed data are shown. Again each symbol in the plot is the average of 50 random permutations of the corresponding data sets, a procedure that was explained in section 11.3.2. If we compare results of the Procrustes transformed data with the corresponding unprocessed data in figures 11.2c and 11.2b, we see that the global behaviour is approximately the same but the baseline scores in 11.2e and 11.2f are higher, a clear indication that the Procrustes transform has really helped to get the test data of the test speaker group closer to the reference group.

11.4 Test with bandfilter data

In this section we will put the adaptive model, which has been extensively tested with formant frequency data in the previous sections, to a similar test but now with

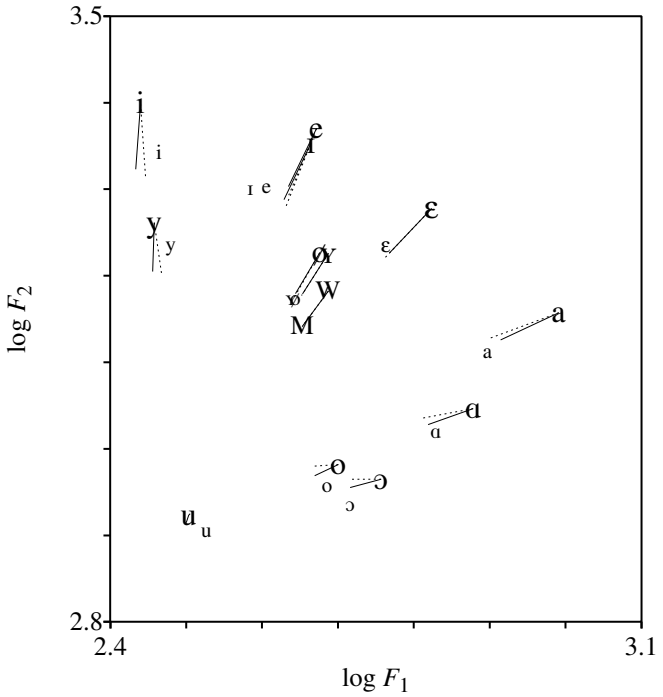

Figure 11.4. Mean vowel positions of vowels from male and female speakers before and after a Procrustes transform. The small and large symbols show the vowel positions averaged over 50 male and 25 female speakers, respectively. The open endpoints of the solid lines show the vowel positions after a Procrustes transform that optimally matches the female averages to the male averages. The open endpoints of the dotted lines show the result of a matching where a Procrustes transform was calculated for each individual speaker separately.

bandfilter data. In order to test the adaptive behaviour, we want to limit the number of vowel representations per speaker, i.e., we want maximal diversity but minimal repetition. The summary data sets that were discussed in section 9.2, i.e. the data sets with maximally 20 vowel means for each speaker, conform to this requirement. We therefore use the male and female training sets M-L-S and F-L-S, respectively, to train two discriminant classifiers. As test sets we used the corresponding male and female test sets, M-T-S and F-T-S, respectively. We also performed tests in which we tested with the whole opposite gender data sets, F-S and M-S, respectively.

For the blocked speaker condition we used the same randomization scheme as was discussed above: we first randomly permuted the speaker data and then randomly permuted the data for each speaker. The actual implementation of this permutation scheme was somewhat more complicated than with the formant data because the

blocks are not of equal size since not all speakers produced all 20 possible vowels. For the mixed speaker condition we randomly permuted the data. The results of these tests are displayed in figure 11.5, where we have indicated the blocked and mixed speaker conditions with a '+' and a '0' symbol, respectively, just like we did in figure 11.2. Figures 11.5a and 11.5d, where training and test set belong to the same speaker category behave just like the corresponding formant data sets, i.e. the blocked speaker condition shows a higher fraction correct than the mixed speaker condition. When training and test sets belong to different speaker categories, the fractions correct do not show any improvement at all, as can be seen in figures 11.5b and 11.5c. Clearly, a simple translation in bandfilter space cannot help to mask the differences in male and female bandfilter spectra. One can see from figure 9.2 in the previous chapter, that a translation of the female average bandfilter spectra does not make them equal to the male average spectra.

In figure 11.5f we show the results of applying a Procrustes transform to the female data set before we apply the adaptive algorithm. The transform in this case was the one that optimally matches the female average spectra to the male spectra. In figure 11.5e we also applied a Procrustes transform, but now the one that matched the average male data to the average female data. Although the fractions correct are lower than the numbers obtained when test and training data sets belong to the same speaker category, applying a Procrustes transform more than doubles the fractions correct compared to not applying it.

A comparison of bandfilter data, figure 11.5, with formant frequency data, figure 11.2, shows that there is a noticeable effect in fraction correct for the mixed and blocked speaker conditions. It seems that this effect is always present for formant frequency data. For the bandfilter data, there is no effect when training and test sets belong to different speaker categories, as figures 11.5b and 11.5c show. It is also clear that when training and test sets belong to different speaker categories, the fractions correct for the formant frequency data show much better scores. Clearly, the formant representation is superior in this respect. As the speech material from which the formants were extracted was very homogeneous we have only marginal variation in formant frequency values.

11.4.1 Stressed vowels

In section 8.3.2 we have shown how to determine vowel stress from lexical stress by means of a dynamic time warping algorithm, and in figure 8.2 we showed an example of the result of applying this procedure. Stressed vowels, in general, are produced more carefully than unstressed vowels (see for example Van Bergem (1995)). Stressed vowels also tend to be longer in duration than unstressed vowels.[2] Because of these facts, we might expect better machine classification for stressed vowels.

In table 11.2 we present results on classification with stressed vowels. We will discuss this table in the next subsection. In table 11.1, set up in the same way as table 9.1, we show the number of vowels associated with the different conditions.

[2] In TIMIT the average durations of the stressed and unstressed vowels are 0.110 s and 0.071 s, respectively.

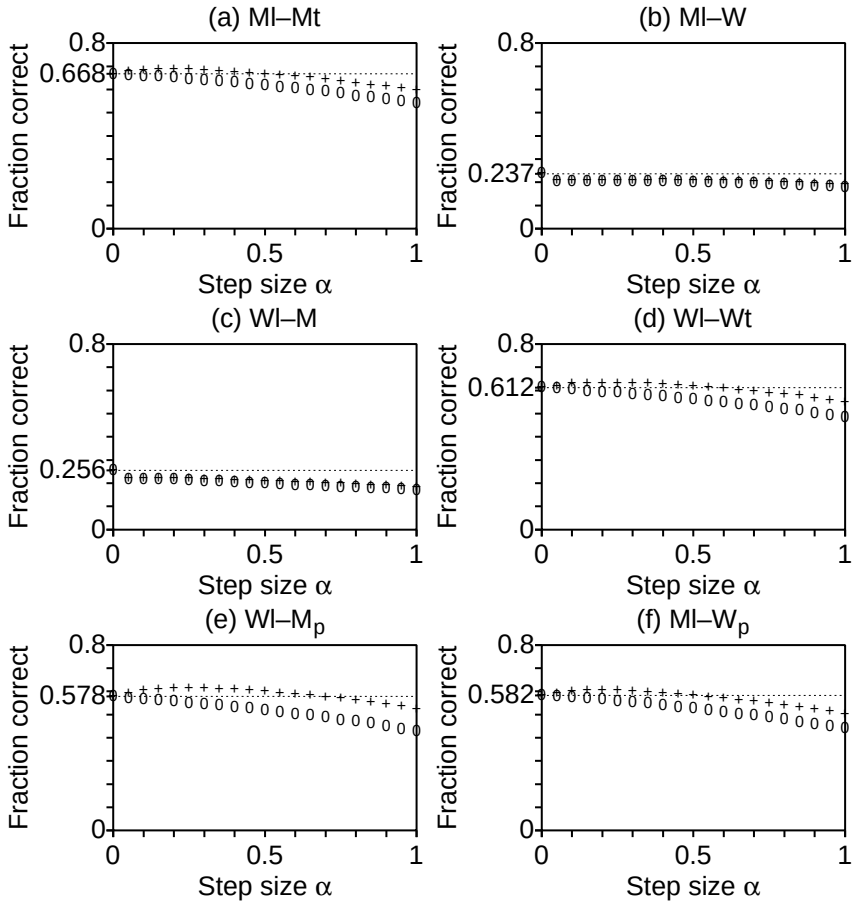

Figure 11.5. Results of the adaptive vowel classifier for blocked (+) and mixed (0) speaker conditions as a function of the step size α. The bandfilter data sets used to train and test the classifier are indicated by the pairs of M and W symbols at the top of each subfigure. M and W denote the sets with bandfilter values from subset from the 438 male and 192 female speakers, respectively. For each value of α in the blocked and the mixed speaker conditions, each result displayed is the average of 50 different randomly permutions of a data set (see text).

The '+' symbol marks stessed vowels. All numbers in the table are smaller than the corresponding numbers in table 9.1 since the stressed vowels form a subset of the total set of vowels. For instance, the last line in this table reads as follows. In the summary data, collected for the stressed vowels, we have 10621 entries. The total number of stressed vowels is 49562. These numbers were 11579 and 78374 in table 9.1. For the

Table 11.1. Summary of the naming scheme for stressed vowels. For further explanation see text.

Name	Entries	Name	Entries	Speakers	Description
M-L-S+	5500	M-L-A+	25706	326	Males, training part
M-T-S+	1926	M-T-A+	8845	112	Males, test part
M-S+	7426	M-A+	34551	438	M-L-· + M-T-·
F-L-S+	2268	F-L-A+	10622	136	Females, training part
F-T-S+	927	F-T-A+	4389	56	Females, test part
F-S+	3195	F-A+	15011	192	F-L + F-T
MF-S+	10621	MF-A+	49562	630	M-·-· + F-·-·

summary data, the average number of vowels per speaker was reduced from 18.4 to 16.9 when selecting only stressed vowels (this is not shown in table 11.1).

11.4.2 Dynamic spectra

In order to investigate if we can improve the vowel recognition scores any further, we tried to incorporate some of the dynamics of speech into the analysis. We have done this by using three spectral representations per vowel segment rather than just one from the middle of the vowel. As was described in section 8.6 three bandfilter spectra were determined per vowel: at the centre of the vowel and 25 ms before and after the centre position. In figure 11.6 we have plotted the relevant plots from a principal component analysis on the 54-dimensional (dynamic) bandfilter data. To determine the principal components for the joint male and female space, we have used the F-S data set with summary data for the 192 female speakers, together with the data from the first 192 speakers from M-S. In this way we have approximately the same amount of male and female data to determine principal components from. This number of speakers is large enough for a proper coverage of the most important characteristics of the male speaker material. We could not apply the same procedure that we used to produce figure 9.1. Using only 20 male and 20 female vowel centroids leads to an underdetermined system of equations when the data vectors are 54-dimensional. We therefore opted for principal components to reduce the number of dimensions. The positions of the male and female vowel centroids in the principal component plane as displayed in figure 11.6a are almost indistinguishable from the ones displayed in figure 9.1a.

Concerning the cumulative fractional eigenvalues as displayed in figure 11.6c, we notice that after the first three eigenvalues that explain a 0.74 fraction of the variance, the variance explained by the rest of the eigenvalues seems to increase only slowly. For example, the individual contributions of the first three eigenvalues are 0.44, 0.18 and 0.12 , respectively, while the next three eigenvalues explain fractions of only 0.05,

0.04 and 0.02. To explain 0.95 fractional variance we need 13 principal components, to explain 0.98 fractional variance we need eight components in addition for only a 0.03 added fractional variance. This may still lead to considerable data reduction since the original vectors were 54-dimensional.

The possibility for data reduction is confirmed by the first two eigenvectors in figure 11.6b: each eigenvector consists of three almost identical parts and each part is very similar to the corresponding eigenvector that is displayed in figure 9.1b. This is all very reassuring because it tells us that the static analyses that we have performed up to this section, cover the character of the vowel very well. However, having two extra frames of probably transitional information on a vowel may help in a classification task because these two frames do contain extra information. Dusan (2005) in a study on the nature of acoustic information in the identification of coarticulated vowels observes seven acoustic cues that may be exploited. Although he uses nine vowels in only three VC and CV contexts, where V is one of /iy, ih, eh, ae, aa, ao, ah, uh, uw/ and C is one of /b, sh, m/, his trajectories in the parameter space clearly show that vowel identity is distributed all over the vowel's durational interval.

We do not need 13 principal components to cover 0.95 fraction of the variance in the single frame case as, figure 9.1c shows.

Table 11.2 shows that adding dynamic information for the TIMIT data significantly increases the fraction correct of the discriminant classifier. For example, the first line shows that training the classifier with the male dynamic data training set, M-L, and testing with dynamic M-T, increases the fraction correct from 0.484 to 0.557 as compared to the static case. For summary data the dynamic results are even more eminent. Row three shows a fraction correct of 0.836 for dynamic data versus 0.668 for static data. Limiting the data to only stressed data, i.e. the four bottom rows in the table, shows distinct results depending on whether we use summary data or not. On the one hand it seems to improve fractions correct somewhat if all the available data are used. This can be seen by comparing the lines M-L and F-L for all the data with the lines M-L+ and F-L+ for the stressed data. These results confirm our expectation. On the other hand, the fractions correct worsen for the summary data. For example, comparing the lines with M-L-S and M-L-S+ shows that when the static training and test sets are different, the fraction correct for stressed summary data, 0.626, is actually somewhat lower than that for all the summary data, 0.668. A possible explanation for these opposite tendencies is that because of the fact that stressed vowels are produced more carefully, they show somewhat more within-speaker variation due to context. Table 11.2 also shows that the extrinsic speaker normalization procedure, subtracting the difference between the average of a speaker's vowels and the average of all the speakers' vowels in the male/female group, increased the fractions correct in all conditions.

11.5 Discussion

The algorithm presented in this chapter and tested both with formant frequency data and bandfilter data was successful in adapting to different speakers. The algorithm

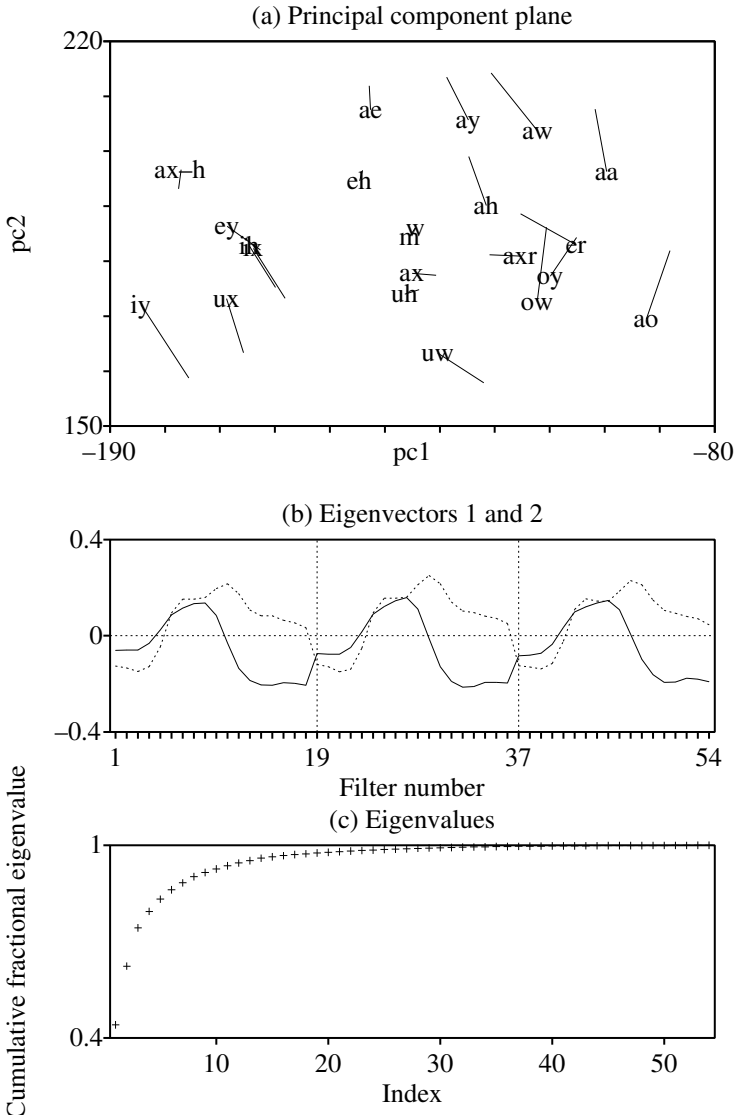

Figure 11.6. Characteristics of male and female vowel centroids in a common PCA eigenspace.
(a) Vowel centroids. The average male vowel centroids are labelled. Each label is the starting point of a solid line that ends at the female vowel centroid. The points labelled ‘m’ and ‘w’ are the average male and female spectra, respectively.
(b) The first and second eigenvectors drawn with a solid and a dotted curve, respectively.
(c) The cumulative eigenvalues as fractions of the total sum of the eigenvalues.

Table 11.2. Static and dynamic discriminant classification of TIMIT vowels with bandfilter data. The first column indicates the vowel sets from which training and test sets are selected: 'All' 78374 vowels, the 11579 'Summary' set, the 49562 'Stressed' vowels and the 10621 'Stressed summary' (see also tables 9.1 and 11.1). The second column shows the selected training sets. The next two columns show the classification scores on the training (L) and the test (T) sets for static data, i.e. one bandfilter data frame per vowel. The last two columns show the classification scores for the dynamic data. The numbers within parentheses show the fractions correct after speaker normalization. This normalization for each male/female speaker was done by subtracting the difference between the average of a speaker's vowels and the average of all the speakers' vowels in the male/female group. For example, the last line in the table reads as follows: a discriminant classifier trained with the (2268) stressed female vowels in the *training* part of TIMIT shows a fraction correct of 0.610 when tested with the same set and 0.588 when tested with the (927) stressed female vowels in the test part. When the three frames of bandfilter data are used for training and testing these score are 0.828 and 0.777, respectively.

		Static test data		Dynamic test data	
Data	Learn	L	T	L	T
All	M-L	0.473 (0.503)	0.484 (0.506)	0.560 (0.584)	0.557 (0.581)
	F-L	0.464 (0.493)	0.461 (0.486)	0.563 (0.590)	0.547 (0.571)
All	M-L-S	0.659 (0.766)	0.668 (0.773)	0.844 (0.899)	0.836 (0.890)
Summary	F-L-S	0.620 (0.739)	0.612 (0.736)	0.838 (0.887)	0.797 (0.866)
Stressed	M-L+	0.498 (0.530)	0.509 (0.537)	0.607 (0.631)	0.600 (0.627)
	F-L+	0.490 (0.519)	0.489 (0.513)	0.607 (0.635)	0.594 (0.623)
Stressed	M-L-S+	0.630 (0.734)	0.626 (0.733)	0.821 (0.872)	0.801 (0.856)
Summary	F-L-S+	0.610 (0.732)	0.588 (0.711)	0.828 (0.879)	0.777 (0.840)

actually shows a different performance when vowel data were presented in blocked and mixed speaker conditions. This was remarkable because the classifier has no concept at all of speakers, and therefore no information about any speaker was used. These differences between the blocked and mixed speaker conditions were produced by only presenting the data in different orders to the classifier. Clearly, there must be (small) correlations within the vowel data of each speaker that can be exploited by the classifier as they are exploited by human listeners. With formant frequencies as input, the classifier was able to perform well even if male and female speakers were mixed. For bandfilter data this was not the case.

An issue with any adaptive system is stability versus plasticity (Grossberg, 1980). Stability was reached in our system because, first of all, we only allow small movements of the references and, secondly, the input to the classifier did not contain very noisy data. However, just like feedforward neural networks there is no internal protec-

tion against deviant data and a carefully crafted data set can move the references in any direction we want. Despite these objections the adaptive classifier performed well and was able to explain differences between the blocked and mixed speaker conditions. We do not claim that the adaptive classifier is the real model of vowel perception: a real model has to be more flexible and cannot be restricted to vowel perception only. A real model also has to cope with within-speaker variation, which as the study of Sun & Deng (1995) showed, is even larger than the between-speaker variation that our model tries to cope with.

11.6 Conclusion

In this chapter we have successfully tested a new speaker-adaptive normalization algorithm. What makes this algorithm special is the fact that local differences can have global effects: a small local difference can move the complete reference system instead of just a single reference as in most other models. As the classifier does not use any speaker-specific knowledge, it can only use the information present in the vowel. By exploiting the small correlations within the vowels of each speaker when these vowels are presented in succession, we showed that the algorithm performed better on vowels presented in a blocked speaker condition than on vowels presented in a mixed speaker condition. The algorithm therefore was able to reproduce, at least qualitatively, the results from listening experiments where stimuli were presented in mixed and blocked speaker conditions.

Chapter 12

Discussion and future research

Abstract

In this final chapter we present an overview of the results obtained. Furthermore, the tools that have been developed and incorporated in the PRAAT program are summarized. We will further discuss the different vowel identification, speaker normalization and data parameterization methods that have been used in this thesis. We will end this chapter by giving an outlook into the future.

12.1 Introduction

In most daily-life situations speech communication seems to be straightforward and we are barely aware of the robustness of our speech recognition capabilities. Large variations exist in the physical realizations of linguistically identical utterances by different speakers. A human being is capable of handling all these variations in a very robust way. Exactly these large variations in the physical signal limit automatic speech recognition systems, although in subdomains, when many sources of variation have been eliminated, impressive results have been obtained.

In this study we have concentrated on the topic of *vowel* identification by humans and machines. Many aspects that trouble the automatic recognition of speech also come into play in the field of vowel identification. We investigated the vowel parts in relatively noise-free read-aloud speech. Despite this limitation, several sources of variability still exist in this material. First of all there is the variability in speaker context, i.e. male and female speakers of different ages, including children. We have used speech from two different languages, Dutch and American-English. The Dutch databases were small in size. They were analyzed in several ways. We have used the formant frequencies from 50 males and 25 females that were measured by Pols et al. (1973) and Van Nierop et al. (1973). We have further used the formant frequencies and bandfilter values from the speech of ten men, ten women and ten children that were measured by Weenink (1986a). These Dutch databases were very homogeneous with respect to the vowel's consonant context. They only contained vowels spoken in isolation, in /h_t/ context and in /p_t/ context. By contrast the TIMIT American-English acoustic database has a lot more variation in its acoustic data: first of all the number of speakers was an order of magnitude greater, i.e. 630, of which 438 were males and 192 were females. Secondly, the variation within the language was greater since it contained sentences from eight American-English dialect regions. In the third place, there was more variation in the context of the vowels because they were realized in different consonantal contexts in sentences. Furthermore, because of the large number of speakers and dialect regions, intonation, speaking style and spaking rate varied, which also influences vowel variation.

In order to quantify the contributions of various identifiable sources of variation in the TIMIT database, Sun & Deng (1995) performed a hierarchically structured analysis of variance (ANOVA) on a spectral representation with mel-frequency cepstral coefficients. The main contributing factors to the total variance they found were "differences between the phoneme units" with 34% explained variance, "difference within phonemes due to different phonemic context" with 28% explained variance, "difference within phoneme due to dynamics" with 16.5% explained variance, and "difference within phonemes due to different speakers" with 12% explained variance. We see from these numbers that the largest single contributing factor to the total variance was the difference between the phonemes, as it should be. However, it is also clear that phonetic context and dynamics contributed more than speaker variance. Another ANOVA on individual phonemes confirmed the observation that contextual variation within each individual phoneme dominated these other factors. This makes it clear that the effect that we try to model, "difference within phonemes due to different speakers"

or in other words "between-speaker variability" is not the main player in the phoneme variability field. Despite the fact that the amount of variation due to phonemic context is greater than the amount of variation due to speaker context, enough between-speaker variability remains to make it an interesting subject to study, especially since in our study we have added the blocked versus mixed speaker conditions as an extra source of variation.

12.2 Human vowel identification

As the introduction already noted, human speech and vowel recognition are very robust. Under normal listening conditions, humans hardly make any vowel identification errors. The only way to force humans to make recognition errors is to make the task more difficult. This can be done by adding noise, or, for example, by limiting the durations of the vowel segments to 50 ms. In chapter 2, we have described eight listening experiments on vowel perception in a mixed and a blocked speaker condition. The stimuli in seven out of eight experiments were vowel segments that were reduced to 50 ms duration. The results of these experiments were summarized in table 2.2 and clearly show that the subjects, when presented with vowel-like stimuli from many speakers, obtained better identification scores if they were familiar with the voice of the speaker than if they were not. The differences in identification scores between stimuli presented in blocked and mixed speaker contexts were between 3.2 and 8.5%. The conclusion of this type of experiments must be that, in one way or another, subjects are able to adapt to the vowels of each particular speaker. This adaptation to a new speaker is relatively fast and of the order of a few syllables. Kakehi (1992) showed that within five-syllable Japanese stimuli subjects adapted to a new speaker. Subjects are able to exploit correlations between vowels produced by the same speaker by only hearing a few of them.

12.3 Machine recognition of vowel segments

In order to explain how subjects might exploit the correlations between vowels, we have tested a number of data reduction, machine recognition and speaker adaptation techniques. The techniques that we have used were introduced in the first chapter and subsequently discussed and elaborated in later chapters. We will review them again and discuss the results obtained.

12.3.1 Principal component analysis

The principal component analysis technique, PCA, has been used extensively in this book. The mathematical techniques and its implementation in the PRAAT program have been discussed in chapter 3. Its most valuable aspect in this thesis has been the representation of multidimensional data in low-dimensional spaces since our main interest is not in data reduction per se. For example, in the analyses of the TIMIT vowel data our measurements consisted of 18-dimensional bandfilter spectra.

By showing the first two principal components of these spectra we were able to better grasp what goes on. In figure 8.8 we used PCA to represent, in a two-dimensional space, the 18-dimensional bandfilter spectra of 20 vowels averaged over 326 male speakers. By making use of two distinct linear combinations of the 18 numbers in each spectrum the first two principal components account for 68% of the variance in the data. The first principal component accounts for 52% of the variance, while the second accounts for 16%. The figure shows that considerable overlap exists between certain vowel categories.

By showing a principal component representation at three different moments in time in the vowel, figure 8.9 demonstrates that some vowels show considerable variation over the time span of their duration. The three moments were at 25 ms before the midpoint, at the midpoint itself and at 25 ms after the midpoint. This figure allowed us to discuss the merits of some of the vowel subsets that were used for analyses in the literature. PCA also enabled us to represent the differences between the bandfilter spectra of vowels for male and female speakers very clearly. Figure 9.1 shows male and female vowel centroids in a common eigenspace.

12.3.2 Procrustes transform

A Procrustes transform preserves the structure of a data set by leaving all relative distances between the data points intact. This can be achieved by only permitting dilation, translation, rotation and reflection operations. By using a Procrustes transform the spectral differences between male and female bandfilter spectra, as were displayed in figure 9.1, could be reduced to a large extent.

12.3.3 Discriminant analysis

Discriminant analysis has proven to be an invaluable technique because it can function as a reference system in vowel segment classification. It constitutes the baseline against which other classifiers can be compared. Given a labelled data set, we have implemented in PRAAT a very fast, reliable and numerically stable algorithm that learns how to optimally represent this data set. The mathematical techniques and the implementation of discriminant analysis in the PRAAT program have been discussed in chapter 3.

In chapter 9 we have used discriminant analysis to investigate differences in TIMIT vowel classification between male and female speakers. We have shown that there are substantial differences between the vowel spectra of male and female speakers and that these differences could not be localized but were spread out along the entire frequency axis, as figure 9.5 clearly demonstrates. As table 9.2 demonstrates, large differences in classification performance result if the training and the test sets for discriminant classification do not belong to the same speaker category. We also showed in section 9.3.3 that these differences could for a large part be overcome by applying a Procrustes transform to the data of one speaker category to make them more similar to the data of the other speaker category.

12.3.4 Feedforward neural nets

In chapter 4 we discussed several classification aspects of feedforward neural nets and in chapter 5 some cost functions for these nets. This type of neural nets is a valuable addition to learning systems. We have first tested feedforward neural nets in chapter 6 in a vowel classification task based on formant frequency values. We learned that their classification possibilities are somewhat superior to the baseline discriminant classifier if a sufficient number of hidden nodes was used. This can easily be seen by comparing the discriminant classifier entry W25, which stands for formant frequency data of 25 female speakers, in table 6.1 with the neural net classification results displayed in table 6.2. An advantage of these nets when used as a classifier is that they can learn *on line* by presenting one item to be learned at a time. This contrasts to a discriminant classifier that needs all the data together in one batch. A slight disadvantage, however, is the iterative learning process, which can take a long time.

We have also tried feedforward neural nets in order to model the blocked versus mixed speaker condition effect by letting only biases adapt to a new speaker. The rationale and the geometrical interpretation were given in sections section 6.2 and section 6.3, respectively. This model was successful in classifying inputs that were formant frequency values as well as bandfilter values. The results, before and after adaptation, were shown in tables 6.3 and 6.4 for formant data, while for bandfilter data results were shown in figure 9.10. Although successful for classification, the model had several disadvantages, besides the ones already mentioned above, which makes it inappropriate to function as a model for human (vowel) perception. First of all the model is supervised during learning, and also during the adaptation process it stays that way. We know that human perception definitely must be based on unsupervised learning: the speech-learning child does not constantly have a little supervisor in her head. Another drawback is that the adaptation process is too slow given the topology of the feedforward neural nets that we have used. These nets basically need at least one input exemplar for each vowel to adapt while also the vowel's label needs to be known. Therefore feedforward neural nets, although an interesting subject for study, are not able to properly model human perception.

12.3.5 Canonical correlation analysis

Canonical correlation analysis quantifies correlations between two sets of measurements on the same object. This technique was introduced in chapter 7. We have used this technique in section 9.3.2 to investigate the relation between fundamental frequency and the bandfilter spectrum in the same vowel spectrum. We saw that more than 50% of the variance in the fundamental frequency could be explained by the bandfilter spectrum.

12.3.6 CategoryART

Another type of neural network model that we have tested was based on adaptive resonance theory. ART neural network models try to find a compromise between *stability* and *plasticity*, i.e., on the one hand we want plasticity, a neural network that

learns new events, on the other hand we want stability, a neural network that does not forget what it has learned. Grossberg has coined this the *stability-plasticity dilemma*.

In chapter 10 we discussed our exemplar-based CategoryART implementation. CategoryART appeared to be a very powerful learning machine for topologically complex classes like the two spirals problem under noise-free conditions. It is also capable of one-shot learning and therefore training times show off very favourable when compared with feedforward neural nets. However, it appeared not to be a successful vowel classifier: because of its lack of data compression it was oversensitive to noise. Its strength, i.e. being able to classify a single item of a class in a cloud of items belonging to another class, turned out to also be its weakness. If vowel classes overlap, category proliferation results and generalisation is poor. In real life vowel classes tend to overlap, no matter what representation has been chosen.

12.3.7 Adaptive vowel normalization

The model that we have developed in chapter 11 was based on an adaptive discriminant classifier and showed adaptive behaviour that at least qualitatively approximated human vowel perception results for the blocked versus the mixed speaker conditions. The nice thing in this model was that better classification in the blocked speaker condition was obtained than in the mixed speaker condition without any supervision, i.e. with no information about speaker identity given to the classifier.

12.3.8 Comparison of vowel classification scores with the literature

In the present study a number of vowel classification experiments were performed. We used two representations for the vowels, a formant frequency representation and a bandfilter representation. The databases for these vowel representations differed considerably. The formant frequency measurements that we used were obtained with operator intervention from relatively small Dutch databases, as was already discussed in section 12.1. We have used formant frequency values only to illustrate certain aspects of the models and algorithms that we have discussed in this thesis. A representation of vowels with formant frequencies is very intuitive because (a) it is a low-dimensional representation and scatter plots of the first and second formant frequency look appealing, at least for most phoneticians; (b) most of the time formants have a simple spectral analog; and (c) formants can easily be recognized in the oscillogram and in the spectrogram, for many vowels.

For American-English the formant frequency measurements of Peterson & Barney (1952) are often used as a dataset in classification benchmark tests. We have done no benchmarking on this dataset. We have measured formant frequency values for all TIMIT vowels at the vowel centres by standard LPC means. We have plotted in figure 12.1 the (medians of) the first and the second formant frequencies for the 20 vowel classes. Solid lines connect the median values of the same vowel for male and female speakers. In section 12.4 we will discuss ways to improve automatic formant frequency measurements.

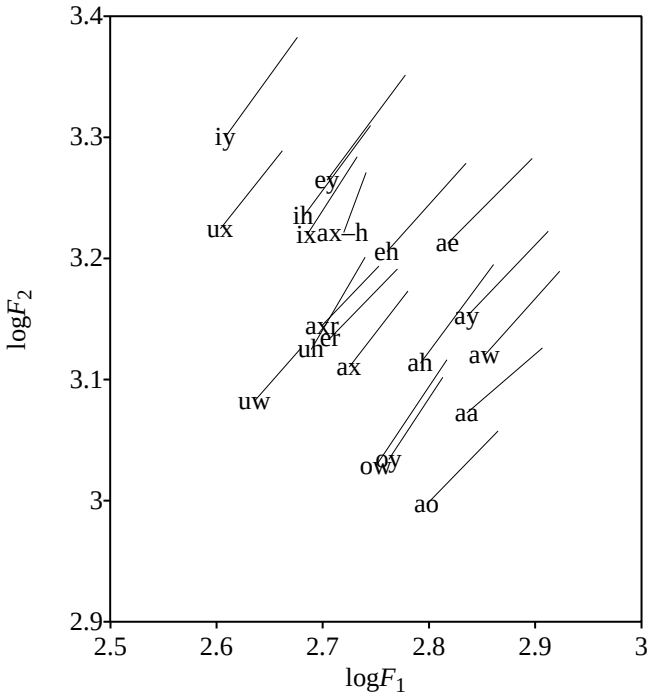

Figure 12.1. Scatter plot of first formant versus second formant frequency the vowels in TIMIT. Male vowel formant frequency *medians* for 20 American-English vowel categories are labelled. Female vowel formant frequency medians are at the endpoints of the lines that start at the labelled male vowel positions.

Most emphasis in our data presentation has been on bandfilter analysis, as all vowels in TIMIT have been analysed in this manner. The TIMIT acoustic phonetic corpus has served as a testing benchmark through the years for many scientists. A great deal of new speech analysis, segmentation and recognition tools were tried on this dataset first because of the availability of hand labelling at the phoneme level. This makes it relatively easy to obtain percentage correct scores. We present in table 12.1 an overview of some of the classification scores that can be found in the literature. We have not attempted to be complete in this respect but have selected a number of papers that present landmark results.

Meng & Zue (1991) compare signal representations for phonetic classification of 13 vowels. They investigate whether there are any advantages in extracting acoustic attributes by directly using the spectral information for classification or whether intermediate sets of linguistic units like distinctive features are advantageous. The spectral representations used in Meng & Zue's study are based on the auditory model of Senef

Table 12.1. Overview of TIMIT classification results. For more details see text.

Reference	Method	Specifics	Set	%Correct
Meng & Zue (1991)	MLP	Straight	mz13	64.5
	MLP	Attributes	mz13	64.1
	MLP	Att. + Features	mz13	63.6
	MLP	Features	mz13	63.6
Cole et al. (1992)	Humans	isolation	t16	54.8
		context	t16	65.9
		prompt	t16	55.3
Zahorian et al. (1993)	MLP	DCT-1	t16	53.5
	BPP MLP	DCT-2		72.4
Zahorian et al. (1997)	MLP	DCT-1	t16	53.5
Halberstadt et al. (1997)	MGM	DCT-2 + mod	kfl14+sv	74.3
	MGM	DCT-2 + mod	kfl39	79.0
Clarkson et al. (1999)	BPP SVM	196 dim (3-4-3)	kfl39	76.3
Salomon et al. (2002)	BPP SVM	195 dim	40	70.6
Chen et al. (2003)	MLP/HMM	TMLP	kfl39	69.0
	MLP/HMM	TMLP+PLP	kfl39	73.2
Gutkin et al. (2004)	MLP	Features	kfl39	60.3
Choueiter et al (2005)	diag. MGM	Wavelets	kfl14	69.5
			kfl39	76.8
Weenink	CategoryART	pc-9 + sum	t20	52.0
Weenink	discriminant	BF18+males+static	t20	48.4
		BF18females +static	t20	46.1
		BF18+males+dyn	t20	55.7
		BF18+females+dyn	t20	54.7

mz13: 13 vowel classes {`iy`, `ih`, `eh`, `ey`, `ae`, `aa`, `ah`, `ao`, `ow`, `uh`, `uw`, `ux`, `er`}, see Meng & Zue (1991)
kfl14: 14 vowel classes `iy`, {`ih`, `ix`}, `eh`, `ae`, {`ah`, `ax`, `ax-h`}, {`uw`, `ux`}, `uh`, {`aa`, `ao`}, `ey`, `ay`, `oy`, `aw`, `ow`, {`er`, `axr`}, see Lee & Hon (1989)
kfl14+sv: the 14 vowel classes in kfl14 + {{`l`, `el`}, `r`, `w`, `y`}
t16: 16 vowel classes `iy`, `ih`, `eh`, `ae`, `aa`, `aw`, `ay`, `ah`, `ao`, `oy`, `ow`, `uh`, `ux`, `er`, `ax`
kfl39: clustering of 61 TIMIT phonemes into 39 classes, see Lee & Hon (1989)
t20: all 20 vowel classes

(1988) and consist of three averaged 40 spectral mean-rate-response coefficients, at half Bark distances apart, that characterize the start, centre and end third part of the vowel. They use speaker normalization by shifting the spectrum down linearly by the median pitch of a speaker. They show that this model has advantages over a Mel-frequency cepstral coefficient representation, especially when tokens get corrupted by

noise. To obtain acoustic attributes and features they use neural networks with one hidden layer of 32 units. Neural networks are also used for all the classifications made in their study. The acoustic attributes are intended to function as the *acoustic correlates* of the *distinctive features* High, Low, Tense, Back, Round and Retroflex. Since neither these acoustic correlates are known beforehand, nor their variability across speakers and phonetic environments, the authors propose general property detectors with free parameters that have to be determined by statistical and data-driven methods that search the parameter space for optimal classification performance. Their most important property detector is based on the spectral centre of gravity and its amplitude within a specific frequency region. The free parameters in this case are the lower and upper frequencies of this region. By splitting the data in a +feature and −feature part they find these upper and lower limits by maximizing a measure of class separability, i.e Fisher's Discriminant Criterion (which is the quotient of the difference of the two class means and the sum of the within-class variances). In the attribute extraction procedure they perform speaker normalization by shifting spectra down linearly on the Bark scale by an amount that depends on the speaker's median pitch. The four main classification tests in this study are all performed with neural networks. Straight one-step classification from the spectral coefficients results in a 64.5% correct classification of the 13 vowels. The two-step process, first extracting acoustic attributes and then classification, resulted in almost the same percentage correct, 64.1%. The three-step process, using the acoustic attributes to calculate a distinctive feature representation followed by classification, results in 63.5% correct. Finally, the other two-step process, extracting distinctive features directly from the spectral coefficients followed by classification results in 63.6% correct. These results show that these 13 vowels can be classified reasonably well and that using acoustic attributes did not weaken the classification performance and needed only half the number of computations. The authors hope that "the compactness and descriptive power of distinctive features may enable us to describe contextual influence more parsimoniously, and thus to make more effective use of training data".

Cole & Muthusamy (1992) examined the ability of human listeners to classify vowel sounds into 16 different categories. For every one of the 168 speakers in the test part of TIMIT, the 16 vowel sounds were excised from the sound files according to the information in the label files, resulting in 2688 vowel sounds. A group of eight subjects participated in ten one-hour sessions. In each session the subjects listened to each randomly selected vowel as often as they wanted before giving a response by moving the cursor on a computer screen over one of the 16 words with the vowel sound and clicking the mouse. The subjects received feedback on the correct response and had the option of listening to the sound again after making the response. On average, subjects scored 54.8% correct. In a second experiment the vowel segment together with preceding and following phoneme were presented to eight other subjects with an identical experimental setup. The context improved the percentage correct to 65.9% on average. In a third experiment Cole & Muthusamy investigated whether familiarity with the speaker's voice had any influence on the subject's response. "Don't ask me" from the sa2-sentences was presented with a gap of 300 ms before the vowel in isolation. A small increase to 55.3% correct was found. This experiment shows

that the agreement between the expert labellers and listeners is not all 100% at the segmental level. In their words "there is in fact no *correct* phonetic labelling of an utterance. ... the same segment of speech is heard differently when presented with different amounts of surrounding context; and knowledge of the speaker and the words spoken influence phonetic labelling."

Zahorian, Nossair & Norton (1993) use two different representations of the vowels. In the first representation, labelled as DCT-1 in table 12.1, only one 20-ms analysis frame at the centre of a vowel is used to obtain vowel phone parameters. DCT-1 consists of the first 25 coefficients of the discrete cosine transform (DCT) of the logarithmically scaled Fourier magnitude spectrum, using log amplitude scaling and a bilinear frequency warping that mostly resembles a Bark scale. Only spectral components between 75 and 6000 Hz were used in this computation. The obtained coefficients are essentially cepstral coefficients. In the second representation, labelled as DCT-2 in table 12.1, phones are represented by 300-ms length segments centred at the midpoint of each labelled phone. For each of 30 equally-spaced frames of 20 ms duration the first 15 DCT coefficient were calculated as outlined above. These 450 numbers were further reduced by again applying a DCT but now in the time-domain instead of the frequency domain. A special Kaiser-window time warping function was used to put emphasis at the centre of the 300 ms segment. These 120 coefficients were further reduced to a set of 58 that showed the highest discriminative power. They obtain 53.5% and 72.4% correct classification with a binary-pair partitioning (BPP) neural network. Binary-pair partitioning is a special case of group partitioning. Instead of using one big classifier to distinguish between M classes, it uses $M(M-1)/2$ binary classifiers to make an M-way decision. Each binary decision is made between a pair of classes. Thus, there are $M-1$ decisions relevant for each class in a set of M. For classification, these decisions are combined to produce an overall decision. The purpose of this type of partitioning is to improve accuracy and to reduce training time. In a single M-class neural network classifier training time increases exponentially with the number of classes while in a BPP training time increases only as ${}^2\log M$.

Zahorian, Silsbee & Wang (1997) use a somewhat modified DCT-2 procedure to classify all phonemes in TIMIT. The differences with the procedure described above for DCT-2 is that they first apply a pre-emphasis on the sound segment with a second-order filter that approximates the inverse of an equal-loudness contour. The windowing function has changed from a Kaiser window to a Hamming window and an 80 Hz wide spectral smoother is applied to the Fourier magnitude spectrum before the frequency-domain DCT is calculated. The time step for the Fourier analyses was reduced to two milliseconds. Zahorian et al. further significantly simplified the coefficient reduction step by using only twelve DCT coefficients to code the frequency domain in each of the analysis frames, and by using only 5 DCT coefficients to code these twelve trajectories in the time domain, resulting in a very efficient 60-dimensional representation for each phone. They do not report separate vowel classification scores but overall on the kfl39 set they obtain an impressive 76.5% correct phoneme score. The kfl39 phoneme set is a clustering of the 61 different phoneme classes that occur in TIMIT into 39 classes, as was first described by Lee & Hon (1989).

Halberstadt & Glass (1997) also use a segment-based approach to speech recogni-

tion and extend the analysing framework of Zahorian et al. (1997) that was discussed above. First of all they added duration and average pitch to the measurements, making a 62-dimensional parameter vector for vowel and semivowel measurements. This resulted in a 74.3% correct classification on vowels/semivowels. Instead of using this homogeneous feature set by applying the same analysis and feature extraction procedure for all phonemes, they argue that differences between phoneme classes also imply differences in the parameter extraction process. For example, the time resolutions needed to differentiate a stop from a nasal are significantly different. In their study they use analysis window lengths of 10.0 and 28.5 ms for these broad classes, respectively. This increases the score with 0.8% to 85.2% correct for nasals. Other modifications were also made such as, for example, adding time derivatives for stops. They achieved a final result of 79.0% correct for the kfl39 phonemes on the core test set. The authors used a mixture Gaussian classifier with maximally twelve full-covariance kernels per phone. Simply stated, mixture models use more than one combination of an average vector and a covariance matrix to model one phone. The mixture is obtained through a combination of supervised and unsupervised learning. Supervision is provided by the class labels. A quadratic discriminant classifier as was discussed in chapter 3 uses one vector of averages and one covariance matrix for each class. In our usage of this classifier we have always split the data into male and female speakers, i.e. we used a mixture model where the mixing was predetermined. A mixture model with two kernels per phone would do this modelling by an iterative unsupervised training method.

Hazen & Halberstadt (1998) reach error rate reductions up to 12% by a process they call aggregation, and others call voting. Train a number of independent classifiers on the same problem and aggregate their outputs by averaging or by using some kind of majority vote system (Carpenter et al., 1992).

Clarkson & Moreno (1999) use binary-pair partitioning Support Vector Machines (SVM, see Vapnik (1995)). An SVM is a *binary* classification machine that tries to optimally separate two classes with a hyperplane. In order to cope with the kfl39 set with 39 phoneme classes, they have to use 741 ($= 39 \times (39-1)/2$) binary SVM's as a classification front-end in a one-versus-one scheme. To encode the variable length phone segments they partioned a phone into a start, middle and end segment with relative durations of 0.3, 0.4 and 0.3, respectively. For each segment they used a 39-dimensional vector. Each representative was obtained by averaging the 39-dimensional analyses vectors in each segment. An analysis vector was obtained by calculating every 10 ms the 13-dimensional Mel-frequency cepstral coefficients plus their first and second derivatives of a 25.5 ms Hamming-windowed part of the sound. Two additional vectors were added that represented 40-ms windows centred at the start and the end of a phone. The log-duration of the phone was also added, adding up to one 196-dimensional representation for each variable length phone. Non-linear separation can be obtained by mapping the data to higher-dimensional spaces. *Kernel functions* can realize these mappings very elegantly. By using a Gaussian kernel function they obtained a 76.3% correct score on the kfl39 set.

SVM's were also used in the study of Salomon, King & Osborne (2002). However, unlike the study above, in which entire phonemes were classified, their system operates in a frame-wise manner. The SVM's in this study are used in a hybrid sys-

tem as the front end to estimate the posterior phone conditional probability density function (pdf) over the phone set on a frame-by-frame basis. This pdf is then decoded into a word sequence. The input for the SVM's consists of 13 Mel-frequency cepstral coefficients with first and second derivatives and two context frames on each side summing to $(13 + 13 + 13) * (2 + 1 + 2) = 195$ components. In a BPP scheme they report a 70.6% framewise accuracy for 40 phones.

Chen, Chang & Sivadas (2003) use a combination of neural networks and Hidden Markov Models (HMM) to incorporate temporal and contextual information. The neural networks provide likelihoods for the HMM recognizer. In contrast to the standard approach of making spectral cross-sections in time, they also use an approach pioneered by Hermansky & Sharma (1999) in which they analyse long stretches of speech, i.e. of the order of 0.5 to 1 s, for patterns within individual critical bands. They obtain a 69% correct score on the kfl39 set. By combining the traditional approach, spectral cross-sections in time by means of perceptual linear prediction (Hermansky, 1990), with the new approach by using frame-wise posterior multiplication, they obtain an even better 73.2% correct score on kfl39.

Gutkin & King (2004) use five multi-valued features in order to devise a structural representation of speech. Their approach is motivated by the fact that a symbolic space is well suited for capturing and exploiting structural properties of speech that other models like HMM's fail to capitalize on. This could be an advantage since a structural description is very efficient for phenomena like assimilation: structural models do not need an enumeration of cases. Gutkin & King use the following 5 multi-valued features: front-back, place of articulation, manner of articulation, roundness and voicing. The total number of values for all features in their system sum up to 25. Neural networks are trained to discover the features. Their output is quantized to obtain a structural representation with symbols: the number of symbols for a feature value depend on the quantization level. The speech sound is transformed into 20 streams of symbols. Clustering of the templates is done by a variant of k-means and distances between strings like the Levenshtein distance (Gusfield, 1997). Gutkin & King obtain 60.3% correct classification on the kfl39 set.

Choueiter & Glass (2005) use a wavelet and filterbank framework for context-independent phonetic classification. They use specially designed filterbanks with wavelet filters whose bandwidths are similar to critical bands and whose centre frequencies are equally spaced on a Mel scale. For classification they use diagonal Gaussian mixture models with a maximum of 96 models per phone. They obtain 69.5% correct for the kfl14 vowels and 76.8% correct for the total kfl39 set.

The above results show that significant results have been obtained on the TIMIT classification benchmark. Our own classifications do not show as impressive results for several reasons. First of all, our view of the vowel was rather static, most of the time we used the results from a single analysis frame for classification. In the second place, we only used a simple linear discriminant classifier, i.e. we did not use vowel-specific covariance matrices in our classification process. By reducing the number of vowels we could have obtained a few percentages higher score. The emphasis in our investigation has not been on obtaining high recognition scores per se but on using simple classifiers as an illustration of our ideas on speaker normalization.

12.4 Tool development motivation

A substantial part of this thesis has been concerned with the analysis of vowel data. In the course of these analyses a number of tools have been developed and integrated into the PRAAT program. We have implemented the possibility to read TIMIT sound and label files. Tools, such as principal component analysis, discriminant analysis and canonical correlation analysis are of course also part of statistical packages like R (R-project, 2006) and SPSS (SPSS, 2006). However, we have tried to make these analyses more user-friendly by representing the results of these analyses as real objects in the PRAAT program. By representing the results as visible objects, the user can select the object and, by the way PRAAT's user interface is organized, can immediately see what are possible further options with these results. Some examples may clarify this. Obtaining principal components in PRAAT is the result of a two-step process. In the first step the user selects the dataset and chooses the `To PCA` command. A new `PCA` object appears. This object contains the new directions, i.e eigenvectors, in the rotated coordinate space that are optimal with respect to variance projection. In the second step the user selects the `PCA` object and the dataset together and chooses the command to actually calculate the principal components of the dataset. Representing the result of the analysis as a separate object with no bindings to the dataset, has an additional advantage: we can re-use the object to project another, properly dimensioned, dataset into the same space. In this way two datasets can be more easily compared by visualizing them in a low-dimensional space. We have used this, for example, in figure 9.1, where we showed characteristics of male and female vowel centroids in the same principal component space. This separation of data and data analysis has also been used for discriminant analysis, canonical correlation analysis and for neural networks and turned out to be very intuitive.

Tool development will continue and new tools will be implemented in the PRAAT program in the near future. As part of a STEVIN programme we will add several new possibilities to PRAAT. We mention the following planned additions:

Klatt synthesizer A very high quality formant-based speech synthesizer that is used for generating artificial speech. The synthesis is determined by a parameter file in which at regular time intervals the values for, among others, fundamental frequency, formant frequencies and bandwidths are given (Klatt & Klatt, 1990). The Klatt synthesizer is often used as a reference synthesizer

Sound-follows-mouse For didactic and demonstration purposes, a straightforward vowel generator of the type *sound-follows-mouse* is needed. By moving around the pointer in the plane formed by the first two formants, a sound with the formant frequencies at the current pointer position will be generated whenever the left mouse button is pressed. The positions of the Dutch vowels could be displayed as a background for the plane in which the pointing device moves.

Formant and bandwidth manipulation For developing theories about speech perception and synthesis, accurate manipulation of speech parameters is necessary. A graphical formant frequency and bandwidth editor in which the user can di-

rectly manipulate the needed parameters to generate speech has distinctive advantages.

Improved formant frequency measurements Many phoneticians rely on formant frequency measurements for their investigations. Formant frequencies and bandwidths are very hard to measure in real speech. Ongoing efforts are put into new analysis and tracking methods. See recent articles by among others De Wet, Weber, Boves, Cranen & Bourlard (2004), Fulop & Fitz (2006), Deng & Acero (2006), Mustafa & Bruce (2006), Deng, Ward, Beddoes & Hodgson (2006) and Gazor & Rashidi (2006). The current formant frequency measurements in PRAAT are based on LPC-analysis performed by standard algorithms available in the literature. The formant frequencies derived by these LPC-analysis algorithms are not always reliable. As Meng & Zue (1991) state: "...formant frequencies, which are obtained through heuristic methods and can lead to catastrophic measurement error." However, these measurements could be improved by incorporating knowledge gained from robust statistics (Lee, 1988). Our own preliminary tests indicate substantial improvements in formant frequency and bandwidth accuracy on artificially generated signals plus significantly less variation in these measurements with varying analysis window position. Of course, this does not immediately guarantee large measurement stability improvements for real speech signals. However, if we can improve significantly the precision of the formant frequency and bandfilter measurements on test signals, this mere fact should give us more confidence in the newer robust analysis method as compared to the standard methods. No known methods exist that always give the "correct" formant frequency values. Therefore making some of the current methods more robust with respect to analysis window position, has high priority. This is useful in all research where formant frequencies are used. Especially for high-pitched voices this is potentially very interesting, because it can be used, for instance, in basic phonetic research on the development of vowel spaces for young children (Pols, Lyakso, Van der Stelt, Wempe & Zajdó, 2006). Another improvement in formant frequency estimation that we will try to implement is based on combining formant frequency measurements with bandfilter data. This would operate with predetermined anchor points for which both types of measurements are accurately known. Then by means of canonical correlation analysis, or by a more predictive variant, redundancy analysis (Van den Wollenberg, 1977), we establish predictive relationships between these two measurements.

The plans discussed aboved clearly show that tool development is an ongoing process and that we will continue to implement new tools in the PRAAT program. We will serve not only our current user group of speech investigators, we like to make our tools useful also to students of language. In the project "Speak good Chinese" the PRAAT program will provide automatic feedback to humans in the process of learning the tone system of Mandarin.

Summary

In this thesis we describe a model for speaker-adaptive vowel identification.

In a series of listening experiments we have investigated how well listeners recognize vowels presented in a mixed and in a blocked speaker condition. In the mixed speaker condition, with each vowel, the listener is confronted with the voice of a random speaker; in the blocked speaker condition, the listener tries to recognize a series of vowels produced by the same speaker. The experiments show that listeners identify vowels better in the blocked speaker condition. The adaptation to a speaker typically happens fast, within approximately five stimuli.

If we want to simulate the listeners' behavior, we need algorithms that also adapt fast. Standard adaptation algorithms are too slow because they adapt only locally, i.e. they need at least one example for each vowel class. The algorithm that we have developed adapts globally. Based upon for example formant frequencies each vowel has a unique position in the vowel system of a language. Instead of modifying one vowel reference at a time, our algorithm modifies the complete system by translating the reference positions of all vowels over the same distance and in the same direction.

Our speaker-adaptive vowel identification algorithm proceeds as follows. A mean reference system is set up on the average positions of the vowels of a large number of speakers. For each vowel that is presented, the vowel class is determined through checking which reference position is closest. The new vowel is accepted as being of the same class as its closest reference. Next a confidence measure is calculated based on the distance between the reference position and the new input. This distance is a vector. If the confidence level is high enough, the system will be adapted. Adaptation means that all current references are shifted parallel to the distance vector. A shift parameter determines by how much the newly classified vowel modifies the reference system. This parameter is important: too large a shift makes the system unstable. For the next input the algorithm will use the adapted reference system.

In a blocked speaker condition, the mean reference system will eventually approximate the reference system of our speaker. Identification of vowels of the speaker will occur more accurately after only a small number of inputs.

The algorithm treats the vowel systems of different speakers as shifted versions of one underlying vowel system. Measurements indicate that a speaker does indeed

have a set structure in his vowel system. Speakers of the same language have a similar structure in the reference positions of their vowels.

Incorporation of our algorithm in a standard discriminant classifier, makes the classifier show a better vowel-identification score in a blocked speaker context than in a mixed speaker context *without any knowledge of speakers at all*. The classifier reproduces the trend of the listening experiments described above.

Samenvatting

In dit proefschrift verkennen we een model voor klinkerherkenning dat zich kan aanpassen aan de spreker en dan beter scoort.

In een serie luisterexperimenten hebben we onderzocht hoe goed luisteraars klinkers herkennen als ze worden aangeboden van verschillende sprekers door elkaar of per spreker. In de eerste situatie wordt de luisteraar per klinker geconfronteerd met de stem van een willekeurige spreker; in de tweede situatie probeert de luisteraar een reeks klinkers te herkennen van een en dezelfde spreker. De experimenten tonen aan dat luisteraars klinkers beter identificeren als ze per spreker worden aangeboden. Deze aanpassing aan de spreker gebeurt gewoonlijk snel, na ongeveer vijf stimuli.

Als we het gedrag van een menselijke luisteraar willen nabootsen, hebben we verwerkingsstappen nodig die snel kunnen leren. Standaard algoritmes voor aanpassing zijn te langzaam omdat ze eerst van elke klinkerklasse een voorbeeld nodig hebben: ze passen lokaal aan. Wij hebben een algoritme ontwikkeld dat globaal aanpast. Op basis van bijvoorbeeld formantfrequenties heeft elke klinker een eigen positie binnen het klinkersysteem van een taal. In plaats van elke keer de verwijzing naar een enkele klinker te wijzigen, kan ons algoritme het complete systeem aanpassen door de posities van alle klinkers te verplaatsen over dezelfde afstand en in dezelfde richting.

Dit zijn in grote lijnen de verwerkingsstappen in ons algoritme bij de herkenning van klinkers. Als uitgangspunt wordt een klinkersysteem gehanteerd dat is opgebouwd uit de gemiddelde posities van de klinkers van een groot aantal sprekers. Van een nieuwe stimulus wordt de klasse bepaald door de klinker te vergelijken met de al aanwezige posities. De nieuwe klinker krijgt de klasse toegedicht van de positie die het dichtst in de buurt ligt. Er wordt een zekerheidsmaat berekend op basis van de afstand tussen deze naaste positie en de nieuwe input. Die afstand heeft een lengte en een richting, het is een vector. Als het zekerheidsniveau hoog genoeg is, wordt het klinkersysteem aangepast. Aanpassing houdt in dat alle aanwezige klinkerposities worden verschoven parallel aan de afstandsvector. Een verschuivingsparameter bepaalt de mate waarin het systeem wordt gewijzigd. Deze parameter is belangrijk: een te grote verschuiving maakt het systeem instabiel. Voor een volgende input gebruikt het algoritme het aangepaste klinkersysteem.

Bij een aanbod per spreker, zal het uitgangssysteem gaan lijken op het systeem

van de spreker. De klinkers van de spreker worden beter herkend na slechts enkele voorafgaande stimuli.

Het algoritme behandelt de klinkersystemen van verschillende sprekers als verschoven versies van een onderliggend klinkersysteem. Metingen wijzen uit dat een spreker inderdaad een vaste structuur kent in zijn klinkersysteem. Sprekers van dezelfde taal hebben eenzelfde structuur in de posities van hun klinkers.

Na de inbouw van ons algoritme scoort een standaard klassificeermachine beter in klinkerherkenning bij een aanbod per spreker dan bij een gemengd aanbod *zonder enige kennis van de sprekers*. De machine bereikt ongeveer hetzelfde resultaat als de mensen in de hierboven beschreven luisterexperimenten.

Curriculum Vitae

David Weenink (1953) graduated in physics in 1981 at the Katholieke Universiteit Nijmegen, now Radboud University. He entered the field of phonetics in that same year, first as assistant investigator and later as assistant professor at IFA, the Institute of Phonetic Sciences of the University of Amsterdam. Recently, he founded the company SpeechMinded dedicated to speech processing by computer. The research presented in this thesis started with a grant of ZWO, the Dutch organisation for Scientific Reseach.

Appendix A

Formant frequencies from 10 men, 10 women and 10 children

The following tables show the first three formant frequencies for ten men, ten women and ten children. This dataset was described in chapter 2. These formant frequency values are also made public in the PRAAT program.

Table A.1. Formant frequency values of ten male speakers.

Vowel		F_1	F_2	F_3		F_1	F_2	F_3
u		335	748	2332		308	709	2222
a		787	1372	3313		758	1244	2658
o		488	911	2362		510	907	2532
ɑ		671	1005	2822		645	1004	2604
ø		472	1585	2344		454	1388	2308
i	*M*1	314	2149	3215	*M*2	303	1984	2835
y		293	1791	2177		314	1620	2111
e		454	1982	2673		460	1748	2442
ʏ		396	1624	2374		449	1489	2279
ɛ		502	1902	2632		580	1679	2383
ɔ		435	669	2886		461	760	2632
ɪ		393	2120	2694		429	1888	2502
u		343	719	2107		322	590	2146
a		850	1328	2299		818	1333	2312
o		532	937	2044		489	845	2157
ɑ		721	1135	2077		708	1126	2302
ø		481	1438	2203		422	1518	2116
i	*M*3	275	2081	2882	*M*4	268	2189	2950
y		270	1705	2068		298	1581	2162
e		464	1949	2536		420	2075	2495
ʏ		390	1307	2177		406	1505	2133
ɛ		645	1854	2488		550	1905	2383
ɔ		519	818	1975		438	703	2392
ɪ		377	2044	2638		400	2151	2870
u		282	656	2355		275	560	2397
a		732	1356	2562		696	1262	2778
o		482	797	2531		419	831	1824
ɑ		723	1036	2456		674	1113	2455
ø		466	1484	2139		422	1599	2319
i	*M*5	255	2390	2787	*M*6	281	2324	3124
y		271	1642	2154		291	1585	2160
e		441	2020	2392		426	2174	2786
ʏ		454	1516	2178		419	1493	2242
ɛ		601	1858	2315		560	1941	2722
ɔ		462	662	2852		395	705	2802
ɪ		353	2114	2507		358	2269	2936
u		315	703	2304		361	730	2256
a		807	1478	2390		921	1409	2909
o		471	838	2391		495	961	2405
ɑ		713	1218	2466		642	988	2308
ø		504	1520	2268		505	1546	2443
i	*M*7	302	2255	2887	*M*8	285	2288	2817
y		341	1738	2246		341	1705	2291
e		400	2198	2551		492	2102	2778
ʏ		467	1639	2227		451	1482	2657
ɛ		587	2024	2516		628	1777	2903
ɔ		531	837	2338		500	714	2782
ɪ		379	2183	2614		364	2126	2907
u		358	700	2672		277	592	2493
a		855	1311	2701		753	1306	2617
o		557	856	2727		392	692	2512
ɑ		685	1075	2677		642	991	2707
ø		502	1385	2112		357	1675	2105
i	*M*9	318	2252	2846	*M*10	258	2280	3350
y		348	1504	2129		248	1846	2117
e		509	1941	2732		357	2163	2624
ʏ		499	1496	2725		388	1570	2175
ɛ		582	1920	2773		581	1865	2572
ɔ		524	759	2784		490	718	2757
ɪ		422	2186	2816		367	2129	2731

Table A.2. Formant frequency values of ten female speakers.

Vowel		F_1	F_2	F_3		F_1	F_2	F_3
u		305	842	2366		307	738	2529
a		938	1580	2953		841	1504	2712
o		606	1090	2422		524	1067	2510
ɑ		819	1327	2819		902	1183	2603
ø		579	1630	2501		443	1754	2607
i	*W*1	291	2540	3225	*W*2	277	2439	3407
y		312	2065	3737		303	1762	2460
e		559	2245	2846		484	2044	2593
ʏ		503	1826	2518		464	1655	2556
ɛ		770	2137	2919		495	2296	3204
ɔ		557	986	2677		505	938	2641
ɪ		506	2270	3040		490	2371	3198
u		309	627	3029		327	586	2463
a		1028	1498	2715		692	1382	2432
o		600	948	2684		540	956	2599
ɑ		893	1136	2844		596	1023	2750
ø		536	1799	2601		512	1674	2458
i	*W*3	275	2664	3174	*W*4	271	2299	3260
y		282	1391	2520		291	1737	2236
e		506	2482	2846		471	2057	2729
ʏ		501	1746	2529		397	1658	2422
ɛ		783	2027	2606		593	2122	2779
ɔ		785	918	2956		520	690	2828
ɪ		511	2320	2882		369	2263	2903
u		448	830	2687		325	737	2290
a		890	1574	2965		833	1333	2256
o		595	1109	2686		577	944	2651
ɑ		806	1386	2591		674	1084	2851
ø		502	1529	2716		502	1546	2326
i	*W*5	374	2580	3181	*W*6	337	2060	2841
y		436	2134	2626		289	1669	2169
e		484	2323	3025		491	1950	2674
ʏ		482	1900	2710		427	1708	2346
ɛ		595	1897	2865		548	2071	2727
ɔ		570	1010	2614		462	764	2600
ɪ		494	2323	2989		416	2217	2769
u		313	578	2457		334	729	3332
a		1047	1578	2848		1008	1573	2932
o		608	904	2724		521	935	2760
ɑ		890	1294	2732		964	1246	3112
ø		508	1747	2799		487	1799	2393
i	*W*7	268	2873	4075	*W*8	296	2597	3571
y		287	1982	2793		306	1781	2415
e		489	2529	3111		490	2642	3494
ʏ		549	1728	2844		509	1887	2596
ɛ		606	2078	2980		787	2326	2992
ɔ		549	855	2793		629	864	3024
ɪ		440	2590	3090		425	2636	3314
u		322	661	2721		324	730	2644
a		898	1628	2770		966	1695	2890
o		457	857	2582		534	954	2542
ɑ		801	1260	2911		820	1256	3064
ø		456	1671	2381		480	1917	2670
i	*W*9	276	2495	3372	*W*10	290	2648	3176
y		283	1779	2211		291	1981	2701
e		435	2363	2866		469	2406	3008
ʏ		463	1630	2306		484	1753	2718
ɛ		637	2130	3029		693	2128	2991
ɔ		589	821	2729		636	923	2757
ɪ		440	2385	3007		501	2358	3058

Table A.3. Formant frequency values of ten child speakers.

Vowel		F_1	F_2	F_3		F_1	F_2	F_3
u		353	758	3269		359	682	2948
a		1129	2249	3404		1006	2166	3263
o		663	1017	3075		688	1135	2930
ɑ		964	1576	3186		746	1135	1926
ø		636	2083	3328		570	2015	3155
i	*C*1	337	2960	3595	*C*2	346	3069	3573
y		357	2389	3185		324	2025	3104
e		621	2839	3614		611	2770	3648
ʏ		650	2336	3284		632	1990	3042
ɛ		765	2755	3636		893	2639	3566
ɔ		627	976	3256		631	795	3537
ɪ		586	2833	3595		636	3025	3709
u		414	828	2908		315	812	3357
a		1150	2116	3205		954	2010	3291
o		659	1201	3015		604	1179	3659
ɑ		890	1303	3003		981	1701	3374
ø		534	1997	3009		613	1991	3392
i	*C*3	330	3010	3746	*C*4	326	3235	3935
y		398	2298	2940		331	2099	3096
e		551	2786	3442		613	2763	3710
ʏ		537	1968	3032		581	2085	3288
ɛ		641	2662	3410		622	2459	3616
ɔ		609	936	3139		621	960	3701
ɪ		547	2965	3673		547	2747	3618
u		417	981	3610		380	798	3316
a		1122	1930	3068		1154	1920	3519
o		741	1302	3381		656	1118	3296
ɑ		1028	1611	2973		1031	1690	3163
ø		628	1929	3197		679	2035	3683
i	*C*5	363	3235	4069	*C*6	361	3082	3782
y		384	2255	2992		374	2042	3206
e		643	2603	3382		646	2820	3970
ʏ		608	2105	3177		604	2162	3603
ɛ		712	2351	3498		706	2785	3942
ɔ		697	1017	3077		456	759	3321
ɪ		527	2698	3458		499	2899	3934
u		339	850	4356		354	1085	3082
a		914	2090	3113		1192	1792	2922
o		576	1156	3681		822	1180	2931
ɑ		865	1567	3326		993	1424	2941
ø		580	2146	3340		608	1817	2909
i	*C*7	313	3090	4039	*C*8	323	3169	3625
y		330	2431	3032		358	2138	3160
e		577	2589	2778		608	2707	3369
ʏ		589	2150	3208		444	2170	3044
ɛ		616	2448	3856		718	2479	3627
ɔ		619	919	3570		617	990	3027
ɪ		539	3015	3905		499	2831	3465
u		363	1050	3239		352	979	2988
a		1163	1707	3188		1060	2079	2706
o		830	1289	3145		753	1174	3005
ɑ		963	1293	3046		884	1432	2806
ø		673	2339	3167		644	1775	3005
i	*C*9	346	2729	3370	*C*10	326	2807	3612
y		345	2344	3103		329	2082	2791
e		677	2561	3320		569	2415	3239
ʏ		580	2191	3234		592	2069	2889
ɛ		897	2463	3338		709	2460	3314
ɔ		666	1094	3269		596	973	2690
ɪ		541	2568	3297		561	2612	3330

Bibliography

Adank P. (2003): *Vowel normalization: A perceptual-acoustic study of Dutch vowels*, Ph.D. thesis, Katholieke Universiteit Nijmegen.

Adank P., Van Hout R. & Smits R. (2004): "An acoustic description of the vowels of Northern and Southern Standard Dutch", *J. Acoust. Soc. Am.* **116**, 1729–1738.

Amari S. (1990): "Mathematical foundations of neurocomputing", *Proc. IEEE* **78**, 1443–1463.

Anderson E. (1935): "The irises of Gaspé peninsula", *Bulletin of the American Iris Society* **59**, 2–5.

Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A. & Sorensen D. (1999): *LAPACK users' guide*, 3rd edn., Society for Industrial and Applied Mathematics, Philadelphia, PA.

Assmann P.F., Nearey T.M. & Hogan J.T. (1982): "Vowel identification: Orthographic, perceptual, and acoustic aspects", *J. Acoust. Soc. Am.* **71**, 975–989.

Bai Z. & Demmel J.W. (1993): "Computing the generalized singular value decomposition", *SIAM J. Sci. Comput.* **14**, 1464–1486.

Boersma P. (1993): "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound", *Proc. Institute of Phonetic Sciences University of Amsterdam* **17**, 97–110.

Boersma P. (1998): *Functional Phonology: Formalizing the interactions between articulatory and perceptual drives*, Ph.D. thesis, University of Amsterdam.

Boersma P., Escudero P. & Hayes R. (2003): "Learning abstract phonological from auditory phonetic categories: An integrated model for the acquisition of languag-specific sound categories", in *Proc. ICPhS*, Barcelona, Spain, pp. 1013–1016.

Boersma P. & Weenink D.J.M. (2006): "Praat: doing phonetics by computer (version 4.3.31)", [Computer program], URL http://www.praat.org/.

Borg I. & Groenen P. (1997): *Modern multidimensional scaling: Theory and applications*, Springer Series in Statistics, Springer.

Bourlard H. & Wellekens C.J. (1989): "Speech pattern discrimination and multilayer perceptrons", *Computer Speech and Language* **3**, 1–19.

Buiting H.J.A.G. (1986): "SESAM, Speech Editing System AMsterdam", IFA report nr. 70.

Carpenter G.A. & Grossberg S. (1987a): "ART 2: Stable self-organization of pattern recognition codes for analog input patterns", *Applied Optics* **26**, 4919–4930.

Carpenter G.A. & Grossberg S. (1987b): "A massively parallel architecture for a self-organizing neural pattern recognition machine", *Computer Vision, Graphics, and Image Processing* **37**, 54–115.

Carpenter G.A. & Grossberg S. (1990): "ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures", *Neural Networks* **3**, 129–152.

Carpenter G.A. & Grossberg S. (eds.) (1991): *Pattern recognition by self-organizing neural networks*, The MIT Press.

Carpenter G.A., Grossberg S., Markuzon N., Reynolds J.H. & Rosen D.B. (1992): "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps", *IEEE Trans. on Neural Networks* **3**, 698–712.

Carpenter G.A., Grossberg S. & Reynolds J.H. (1991a): "ARTMAP: Supervised real-time learning and classification of non-stationary data by a self-organizing neural network", *Neural Networks* **4**, 565–588.

Carpenter G.A., Grossberg S. & Rosen D.B. (1991b): "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system", *Neural Networks* **4**, 759–771.

Carpenter G.A., Milenova B.L. & Noeske B.W. (1998): "Distributed ARTMAP: A neural network for fast distributed supervised learning", *Neural Networks* **11**, 793–813.

Carpenter G.A. & Tan A.H. (1995): "Rule extraction: From neural architecture to symbolic representation", *Connection Science* **7**, 3–27.

Chen B., Chang S. & Sivadas S. (2003): "Learning discriminative temporal patterns in speech: Development of novel TRAPS-like classifiers", in *Proc. Eurospeech*, pp. 853–856.

Choueiter G.F. & Glass J.R. (2005): "A wavelet and filter bank framework for phonetic classification", in *Proc. ICASSP*, pp. 933–936.

Clarkson P. & Moreno P. (1999): "On the use of support vector machines for phonetic classification", in *Proc. ICASSP*, pp. 585–588.

Cole R.A. & Muthusamy Y.K. (1992): "Perceptual studies on vowels excised from continuous speech", in *Proc. ICSLP 1992*, vol. 2, pp. 1091–1094.

De Wet F., Weber K., Boves L., Cranen B. & Bourlard H. (2004): "Evaluation of formant-like features on an automatic classification task", *J. Acoust. Soc. Am.* **116**, 1781–1792.

Dechovitz D. (1977): "Information conveyed by vowels: A confirmation", *Haskins Laboratory Status Report* **SR-51/52**, 213–219.

Dehon C., Filzmoser P. & Croux C. (2000): *Robust methods for canonical correlation analysis*, Springer-Verlag, Berlin, pp. 321–326.

Den Os E.A., Boogaart T.I., Boves L. & Klabbers F. (1995): "The Dutch polyphone corpus", in *Proc. Eurospeech*, vol. 1, Madrid, pp. 825–828.

Deng H., Ward R.K., Beddoes M.P. & Hodgson M. (2006): "A new method for obtaining accurate estimates of vocal-tract filters and glottal waves from vowel sounds", *IEEE Trans. on Audio, Speech, and Language Processing* **14**, 445–455.

Deng L. & Acero A. (2006): "Tracking vocal tract resonances using a quantized nonlinear function embedded in a temporal constraint", *IEEE Trans. on Audio, Speech, and Language Processing* **14**, 425–434.

Diehl R.R., McCusker S.B. & Chapman L.S. (1981): "Perceiving vowels in isolation and in consonantal context", *J. Acoust. Soc. Am.* **69**, 239–248.

Dusan S. (2005): "On the nature of acoustic information in identification of coarticulated vowels", in *Proc. Interspeech 2005*, Lisbon, pp. 2449–2452.

Elman J.L. & Zipser D. (1988): "Learning the hidden structure of speech", *J. Acoust. Soc. Am.* **83**, 1615–1626.

Fisher R.A. (1936): "The use of multiple measurements in taxonomic problems", *Ann. Eugenics* **7**, 179–188.

Fujisaki H. & Kawashima T. (1968): "The roles of pitch and higher formants in the perception of vowels", *IEEE Trans. on Acoustics, Speech, and Signal Processing* , 73–77.

Fulop S. & Fitz K. (2006): "Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications", *J. Acoust. Soc. Am.* **119**, 360–371.

Furui S. (1986): "On the role of spectral transition for speech perception", *J. Acoust. Soc. Am.* **80**, 1016–1025.

Gazor S. & Rashidi R. (2006): "Adaptive maximum windowed likelihood multicomponent AM-FM signal decomposition", *IEEE Trans. on Audio, Speech, and Language Processing* **14**, 479–491.

Golub G.H. & Reinsch C. (1970): "Singular value decomposition and least squares solutions", *Numer. Math.* **14**, 403–420.

Golub G.H. & Van Loan C.F. (1996): *Matrix computations*, 3rd edn., The Johns Hopkins University Press.

Gray Jr A.H. & Markel J.D. (1976): "Distance measures for speech processing", *IEEE Trans. on Acoustics, Speech, and Signal Processing* **24**, 380–391.

Grossberg S. (1976): "Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors", *Biological Cybernetics* **23**, 121–134.

Grossberg S. (1980): "How does a brain build a cognitive code?", *Psychological Review* **87**, 1–51.

Grossberg S. (1986): "The adaptive self-organization of serial order in behavior: Speech, language and motor control", in E. Schwab & H. Nusbaum (eds.), *Pattern Recognition by humans and machines, Volume I: Speech perception*, Academic Press, Inc., pp. 187–294.

Grossberg S. (1991): "Nonlinear neural networks: Principles, mechanisms, and architectures", in Carpenter & Grossberg (1991), pp. 35–109.

Grossberg S. (1998): *The link between brain learning, attention, and consciousness*, Tech. Rep. CAS/CNS-TR-97-018, Boston University.

Gusfield D. (1997): *Algorithms on strings, trees and sequences: Computer science and computational biology*, Cambridge University Press.

Gutkin A. & King S. (2004): "Structural representation of speech for phonetic classification", in *Proc. International Conference on Pattern Recognition*, pp. 438–441.

Halberstadt A.K. & Glass J.R. (1997): "Heterogeneous acoustic measurements for phonetic classification", in *Proc. Eurospeech*, Rhodes, pp. 401–404.

Hampshire J.B. & Waibel A.H. (1990): "A novel objective function for improved phoneme recognition using time-delay neural networks", *IEEE Trans. on Neural Networks* **1**, 216–228.

Hazen T. & Halberstadt A.K. (1998): "Using aggregation to improve the performance of mixture Gaussian acoustic models", in *Proc. ICASSP*, pp. 653–656.

Hermansky H. (1990): "Perceptual linear predictive (PLP) analysis for speech", *J. Acoust. Soc. Am.* **87**, 1738–1752.

Hermansky H. & Sharma S. (1999): "Temporal patterns (TRAPS) in ASR of noisy speech", *Proc. ICASSP* , 289–292.

Hillenbrand J., Getty L.A., Clark M.J. & Wheeler K. (1995): "Acoustic characteristics of American English vowels", *J. Acoust. Soc. Am.* **97**, 3099–3111.

Hotelling H. (1936): "Relations between two sets of variates", *Biometrika* **28**, 321–377.

Hrycej T. (1992): *Modular learning in neural networks: A modularized approach to neural network classification*, John Wiley & Sons, Inc.

Huang J., Georgiopoulos M. & Heileman G.L. (1995): "Fuzzy ART properties", *Neural Networks* **8**, 203–213.

Hult G. (1989): "Some vowel recognition experiments using multilayer perceptrons", *STL-QPSR* **1**, 125–130.

Johnson D.E. (1998): *Applied multivariate methods for data analysts*, Duxburry Press.

Jones R.D., Lee Y.C., Qian S., Barnes C.W., Bisset K.R., Bruce G.M., Flake G.W., Lee K., Lee L.A., Mead W.C., O'Rourke M.K., Poli M.K. & Thode L.E. (1990): *Nonlinear adaptive networks: A little theory, a few applications*, Tech. Rep. LA-UR-91-273, Los Alamos National Laboratory.

Juang H.B. & Katagiri S. (1992a): "Discriminative learning for minimum error classification", *IEEE Trans. on Speech Proc.* **40**, 3043–3054.

Juang H.B. & Katagiri S. (1992b): "Discriminative training", *J. Acoust. Soc. Jpn (E)* **13**, 333–339.

Kakehi K. (1992): "Adaptability to differences in Japanese monosyllabic perception", in Y. Tokhura, E. Vatikiotis-Bateson & Y. Sagisaka (eds.), *Speech perception, speech production, and linguistic structure*, Tokyo, OHM, pp. 135–142.

Kamm C.A., Kane-Esrig L.A. & Burr D.J. (1989): "Comparing performance of spectral distance measures and neural network methods for vowel recognition", *Computer Speech and Language* **3**, 21–34.

Kämmerer B.R. & Küpper W.A. (1990): "Experiments for isolated-word recognition with single and two-layer perceptrons", *Neural Networks* **3**, 693–706.

Klatt D.H. & Klatt L.C. (1990): "Analysis, synthesis, and perception of voice quality variations among female and male talkers", *J. Acoust. Soc. Am.* **87**, 820–857.

Komori T. & Katagiri S. (1992): "GPD training of dynamic programming-based speech recognizers", *J. Acoust. Soc. Jpn (E)* **13**, 341–349.

Koopmans-van Beinum F. (1980): *Vowel contrast reduction. An acoustic and perceptual study of Dutch vowels in various speech conditions*, Ph.D. thesis, University of Amsterdam.

Kurinami K. & Sujiyama M. (1992): "An optimization technique for speaker mapping neural networks using minimal classification error criterion", *J. Acoust. Soc. Jpn (E)* **13**, 419–427.

Ladefoged P. & Broadbent D.E. (1957): "Information conveyed by vowels", *J. Acoust. Soc. Am.* **29**, 98–104.

Lamel L.F., Kassel R.H. & Seneff S. (1986): "Speech database development: Design and analysis of the acoustic-phonetic corpus", in *Proc. DARPA Speech Recognition Workshop*, pp. 100–109.

Lamel L.F. et al. (1994): "The translanguage English database (TED)", in *Proc. ICSLP 1994*, pp. 1795–1798.

Lang K.J. & Witbrock M.J. (1988): "Learning to tell two spirals apart", *Proc. Connectionist Models Summer School* , 52–59.

LeCun Y., Bottou L., Orr G.B. & Müller K.R. (1998): "Efficient backprop", in G.B. Orr & K.R. Müller (eds.), *Neural networks: Tricks of the trade*, vol. 1524 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 9–53.

Lee C.H. (1988): "On robust linear prediction of speech", *IEEE Trans. on Acoustics, Speech, and Signal Processing* **36**, 642–649.

Lee K.F. & Hon H.W. (1989): "Speaker-independent phone recognition using hidden Markov models", *IEEE Trans. on Acoustics, Speech, and Signal Processing* **37**, 1641–1648.

Lindblom B.E.F. (1963): "Spectrographic study of vowel reduction", *J. Acoust. Soc. Am.* **35**, 1773–1781.

Lippmann R.P. (1987): "An introduction to computing with neural nets", *IEEE ASSP Magazine* **4**, 4–22.

Macchi M.J. (1980): "Identification of vowels spoken in isolation versus vowels spoken in consonantal context", *J. Acoust. Soc. Am.* **68**, 1636–1642.

Makhoul J., Schwartz R. & El-Jaroudi A. (1989): "Classification capabilities of two-layer neural nets", in *Proc. ICASSP*, Glasgow, Great Brittain, pp. 635–638.

McCulloch N. & Ainsworth W.A. (1988): "Speaker independent vowel recognition using a multilayer perceptron", *Proc. of Speech'88* , 851–857.

Meng H.M. & Zue V.W. (1991): "Signal representation comparison for phonetic classification", in *Proc. ICASSP*, Toronto, Canada, pp. 285–288.

Minsky M. & Papert S. (1969): *Perceptrons: an introduction to computational geometry*, MIT Press.

Mirchandani G., Cao W. & Bosworth B. (1989): "Efficient implementation of neural nets using an optimal relationship between numbers of patterns, input dimension and hidden nodes", in *Proc. ICASSP*, Glasgow, Great Brittain, pp. 2521–2523.

Molau S. (2003): *Normalization in the acoustic feature space for improved speech recognition*, Ph.D. thesis, Rheinish-Westfälischen technische Hochschule Aachen, URL http://www-i6.informatik.rwth-aachen.de/PostScript/InterneArbeiten/Molau_NormAcousticFeatureSpaceImprovedSR_Dissertation_14Feb2003.pdf.

Morin T.M. & Nusbaum H.C. (1990): "Perceptual learning of vowels in a neuromorphic system", *Computer Speech and Language* **4**, 79–126.

Mustafa K. & Bruce I.C. (2006): "Robust formant tracking for continuous speech with speaker variability", *IEEE Trans. on Audio, Speech, and Language Processing* **14**, 435–444.

Nearey T.M. (1989): "Static, dynamic, and relational properties in vowel perception", *J. Acoust. Soc. Am.* **85**, 2088–2113.

Nearey T.M. (1997): "Speech perception as pattern recognition", *J. Acoust. Soc. Am.* **101**, 3241–3254.

Niles L., Silverman H., Tajchman G. & Bush M. (1989): "How limited training data can allow a neural network to outperform an 'optimal' statistical classifier", in *Proc. ICASSP*, Glasgow, Great Brittain, pp. 17–20.

Nocedal J. & Wright S.J. (1999): *Numerical optimization*, Springer Series in Operations Research, Springer-Verlag.

Nooteboom S.G. & Doodeman G.J.N. (1980): "Production and perception of vowel length in spoken sentences", *J. Acoust. Soc. Am.* **67**, 276–287.

Peterson G.E. & Barney H.L. (1952): "Control methods used in a study of the vowels", *J. Acoust. Soc. Am.* **24**, 175–184.

Pols L.C.W., Lyakso E., Van der Stelt J.M., Wempe T.G. & Zajdó K. (2006): "Vowel data of early speech development in several languages", in *Proc. ISCA Tutorial and Research Workshop on Multilingual Speech and Language Processing*, URL http://www.unistel.co.za/multiling2006.

Pols L.C.W., Tromp H.R.C. & Plomp R. (1973): "Frequency analysis of Dutch vowels from 50 male speakers", *J. Acoust. Soc. Am.* **53**, 1093–1101.

Pols L.C.W., Van der Kamp L.J.T. & Plomp R. (1969): "Perceptual and physical space of vowel sounds", *J. Acoust. Soc. Am.* **46**, 458–467.

Press W.H., Teukolsky S.A., Vetterling W.T. & Flannery B.P. (1996): *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edn., Cambridge University Press.

R-project (2006): URL http://www.r-project.org/.

Rakerd B., Verbrugge R.R. & Shankweiler D.P. (1984): "Monitoring for vowels in isolation and in a consonantal context", *J. Acoust. Soc. Am.* **76**, 27–31.

Rumelhart D.E., Hinton G.E. & Williams R.J. (1986): "Learning internal representations by error propagation", in J.A. Anderson & E. Rosenfeld (eds.), *Neurocomputing: Foundations of Research*, MIT Press, pp. 675–695.

Salomon J., King S. & Osborne M. (2002): "Framewise phone classification using support vector machines", in *Proc. ICSLP*, pp. 2645–2648.

Schläfli L. (1950): *Gesammelte mathematische Abhandlungen*, vol. 1, Birkhäuser, Basel.

Schroeder M.R. (1977): "Recognition of complex acoustic signals", in T.H. Bullock (ed.), *Life Sciences Research Report 5 (Dahlem Konferenzen)*, Abakon-Verlag, Berlin, pp. 323–328.

Sekey A. & Hanson B.A. (1984): "Improved 1-Bark bandwidth auditory filter", *J. Acoust. Soc. Am.* **75**, 1902–1904.

Senef S. (1988): "A joint synchrony/mean rate model of auditory speech processing", *Journal of Phonetics* **16**, 55–76.

Shikano K. & Itakura F. (1992): "Spectrum distance measures for speech recognition", in S. Furui & M.M. Sondhi (eds.), *Advances in Speech Signal Processing*, Marcel Dekker, Inc., pp. 419–452.

Slawson A.W. (1968): "Vowel quality and musical timbre as functions of spectral envelope and fundamental frequency", *J. Acoust. Soc. Am.* **43**, 87–101.

SPSS (2006): URL http://www.spss.com/.

Stevens K.N. & House A.S. (1963): "Perturbation of vowel articulations by consonantal context: an acoustic study", *J. Speech Hear. Res.* **6**, 111–128.

Strange W. (1989): "Evolving theories of vowel perception", *J. Acoust. Soc. Am.* **85**, 2081–2087.

Strange W., Verbrugge R.R., Shankweiler D.P. & Edman T.R. (1976): "Consonant environment specifies vowel identity", *J. Acoust. Soc. Am.* **60**, 213–224.

Sulter A.M. & Schutte H.K. (1994): "Groningen corpus", Speech Processing Expertise Centrum (SPEX), Leidschendam, The Netherlands.

Sun D.X. & Deng L. (1995): "Analysis of acoustic-phonetic variations in fluent speech using TIMIT", in *Proc. ICASSP*, Detroit, USA, pp. 201–204.

Van Alphen P. (1992): *HMM-based continuous-speech recognition: systematic evaluation of various system components*, Ph.D. thesis, University of Amsterdam.

Van Bergem D.R. (1986): "The influence of acoustic context on the identification of vowels in pVt utterances. A study on speaker normalization", IFA report nr. 88.

Van Bergem D.R. (1995): *Acoustic and lexical vowel reduction*, Ph.D. thesis, University of Amsterdam.

Van Bergem D.R., Pols L.C.W. & van Beinum F.J.K. (1988): "Perceptual normalization of the vowels of a man and a child in various contexts", *Speech Communication* **7**, 1–20.

Van den Wollenberg A.L. (1977): "Redundancy analysis: an alternative for canonical correlation analysis", *Psychometrika* **42**, 207–219.

Van Heuven V. & Van Houten E. (1985): "De klinkers in het Nederlands van Turken", *Forum der Letteren* , 201–213.

Van Nierop D.J.P.J., Pols L.C.W. & Plomp R. (1973): "Frequency analysis of Dutch vowels from 25 female speakers", *Acustica* **29**, 110–118.

Van Son R.J.J.H. & Pols L.C.W. (1990): "Formant frequencies of Dutch vowels in a text, read at normal and fast rate", *J. Acoust. Soc. Am.* **88**, 1683–1693.

Van Son R.J.J.H. & Pols L.C.W. (2001): "Phoneme recognition as a function of task and context", *Proc. Institute of Phonetic Sciences University of Amsterdam* **24**, 27–38.

Vapnik V. (1995): *The Nature of Statistical Learning*, Springer, N.Y.

Verbrugge R.R., Strange W., Shankweiler D.P. & Edman T.R. (1976): "What information enables a listener to map a talker's vowel space?", *J. Acoust. Soc. Am.* **60**, 198–212.

Waibel A., Hanazawa T., Hinton G., Shikano K. & Lang K.J. (1989): "Phoneme recognition using time-delay neural networks", *IEEE Trans. on Acoustics, Speech, and Signal Processing* **37**, 328–339.

Watrous R.L. (1991): "Context-modulated vowel discrimination using connectionist networks", *Computer Speech and Language* **5**, 341–362.

Watrous R.L. (1993): "Speaker normalization and adaptation using second-order connectionist networks", *IEEE Trans. on Neural Networks* **4**, 21–30.

Weenink D.J.M. (1985): "Formant analysis of Dutch vowels from 10 children", *Proc. Institute of Phonetic Sciences University of Amsterdam* **9**, 45–52.

Weenink D.J.M. (1986a): "The identification of vowel stimuli from men, women, and children", *Proc. Institute of Phonetic Sciences University of Amsterdam* **10**, 41–54.

Weenink D.J.M. (1986b): "QQ, een programma voor analyse, resynthese en herkenning van klinkersegmenten", IFA report nr. 82.

Weenink D.J.M. (1991): "Aspects of neural nets", *Proc. Institute of Phonetic Sciences University of Amsterdam* **15**, 1–25.

Weenink D.J.M. (1993): "Vowel classification with neural nets: a comparison of cost functions", *Proc. Institute of Phonetic Sciences University of Amsterdam* **17**, 1–11.

Weenink D.J.M. (1996): "Adaptive vowel normalization and the TIMIT acoustic phonetic speech corpus", *Proc. Institute of Phonetic Sciences University of Amsterdam* **20**, 97–110.

Weenink D.J.M. (1997): "Category ART: A variation on adaptive resonance theory neural networks", *Proc. Institute of Phonetic Sciences University of Amsterdam* **21**, 117–129.

Weenink D.J.M. (1999): "Accurate algorithms for performing principal component analysis and discriminant analysis", *Proc. Institute of Phonetic Sciences University of Amsterdam* **23**, 77–89.

Weenink D.J.M. (2001): "Vowel normalizations with the TIMIT acoustic phonetic speech corpus", *Proc. Institute of Phonetic Sciences University of Amsterdam* **24**, 117–123.

Weenink D.J.M. (2003): "Canonical correlation analysis", *Proc. Institute of Phonetic Sciences University of Amsterdam* **25**, 81–99.

Weenink D.J.M. & Pols L.C.W. (1999): "Multi-speaker vowel classification with adaptive neural networks", in *Proc. ICPhS*, vol. 3, San Francisco, USA, pp. 1633–1636.

Weenink D.J.M. & Wempe T.G. (1986): "Communicatie tussen een Apple IIe en vier Commodore Vic 20's", IFA report nr. 83.

Wendahl R.W. (1959): "Fundamental frequency and absolute vowel identification", *J. Acoust. Soc. Am.* **31**, 109–110(A).

Zahorian S.A., Nossair Z.B. & Norton C.A. (1993): "A partitioned neural network approach for vowel classification using smoothed time/frequency features", in *Proc. Eurospeech*, pp. 1225–1228.

Zahorian S.A., Silsbee P. & Wang X. (1997): "Phone classification with segmental features and a binary-pair partitioned neural network classifier", in *Proc. ICASSP*, pp. 1011–1014.

Zue V.W. & Seneff S. (1988): "Transcription and alignment of the TIMIT database", *Proc. Second Meeting on Advanced Man-Machine Interface through Spoken Language* .

Index