Institute of Phonetic Sciences,
University of Amsterdam,
Proceedings 16 (1992), 1-26.

DIMENSIONALITY AND CORRELATION OF OBSERVATION VECTORS IN HMM-BASED SPEECH RECOGNITION

Xue Wang, Louis F.M. ten Bosch and Louis C.W. Pols

Abstract

The dimensionality of, and correlation within and between, the observation vectors in the front-end of a speech recognizer are manipulated. The effect of such manipulations, and of applying multiple codebooks, on the recognition scores, is studied experimentally with a DDHMM (discrete-density hidden Markov model)-based system. The manipulations are performed with transformations of both principal component analysis (PCA) and linear discrimination analysis (LDA). Subsequent recognition is performed with the transformed vectors with full and reduced numbers of components. Comparison is made between the recognition scores with single and multiple (two) codebooks. The two vectors for the two codebooks are transformed from the filter-bank spectrum and its frequency derivative, and the one large vector for the single codebook is transformed from a combination of the spectrum and the derivative. The results show that the multiple-independent-codebook approach leads to higher recognition scores than the use of a single codebook.

1. Introduction

The first part of an HMM-based speech recognizer, including all the acoustic processing, is considered to be the *front-end*, in our terminology. In the front-end of a speech recognizer, the acoustic speech signal is transformed into, and represented by, one or more *observation vectors*, as a function of time. The present study searches for efficient and justified front-end representations, as reflected by the performance of the recognizer.

1.1 General possibilities of front-end representation

The general process in a DDHMM front-end is as follows. Several multi-dimensional observation vectors are encoded with either one, or more codebooks[1]. Encoding by each codebook (no matter with how many vectors) generates one *observation*

[1]A codebook design process, also known as clustering or codebook training, is regarded as being outside of the front-end, where the codebook is obtained by a separate program with a corpus of 'training' vectors. The term *vector quantization* refers to both codebook design and encoding processes. In the context of *encoding*, the codebooks are assumed to exist. However, in this study, the codebook design process will also be explored.

sequence, which is a 1-dimensional sequence of codeword indices[2]. Therefore, in a DDHMM system, a *single observation* approach is also referred to by *single codebook*, whereas *multiple observation* by *multiple codebook*, respectively. Note the different terms for observation vector and observation sequence. A 'single observation approach' refers to the sequence, instead of the vectors. Therefore, a single observation approach can deal with many observation vectors. In order to avoid confusion, we prefer to use 'codebook' instead of 'observation', when we refer to the approach of concern, since in a DDHMM system, codebooks are always used.

The general purpose of acoustic modeling at the front-end is to preserve essential information from the speech signal for subsequent use by the rest of the recognizer. It is then conceptually true that a good choice of the kind of observation vector (or a combination of several vectors) is vitally important for the overall performance of the recognizer. However, due to the particular methods with which the observation vectors are used in an HMM-based recognizer, the relationship between the choice of vectors on the one hand, and the recognition scores on the other hand, is not straightforward. The factors which cause such a complication have been introduced with different technical arguments in mind that tried to achieve improved implementations, yet the relations between these independently developed techniques have been rarely studied. For example, in systems where several different kinds of vectors are used together under a framework of multiple-independent-codebooks, it is assumed or implied by the HMM training and recognition algorithms, that these vectors are statistically independent. However, such an assumed condition is unfortunately not met by actual speech data (Lee et al., 1990). Regardless of this serious violation to the assumption, the multiple-codebook approach is still in wide use. This is generally based on other arguments such as less total distortion introduced by multiple codebooks than by a single large codebook (Lee, 1989).

The multiple-codebook approach possesses another advantage over the single-codebook one, namely that it takes more items from the different front-end observation vectors into the training procedure. Therefore, in the multiple-codebook approach, more (useful) information may be embedded into the HMM models (see e.g. formula (1) in the next section and the accompanying text). Generally, this would result in a better quality of the models (given e.g. the total number of training iterations and amount of training data), when using certain training procedures (the 'maximum likelihood' in particular, see e.g. Lee, 1989), which would in turn lead to improved recognition performance. Hence, on the one hand, we have the case of multiple codebooks with advantages of less encoding distortion and more training items, and on the other hand, we have the case of a single codebook with the merit of being based on fewer improper assumptions about the statistics of the data. The conflicting factors with the two cases make it difficult to give a direct answer as which one should give a superior overall performance. Although it is reported in the literature (e.g. Lee, 1989) that the multiple-codebook approach leads to better results, the comparison has not been made between the two cases using exactly the same (number of) vectors, but rather, between the cases with and without some *additional* observation vectors. An experimental comparison of these two approaches with respect to the recognition scores, under carefully controlled conditions, will be the topic of this paper.

In practice, some acoustic observation vectors, such as the filter-bank outputs and LPC analysis parameters, and the total energy of the frame (all the above known as *basic* features), are chosen based on the existence of efficient algorithms and the experience in other fields of speech technology. Other vectors, such as the time- and fre-

[2]The term *codeword* refers to the multi-dimensional vector representing the center of a cluster, whereas *codeword index* refers to a (one-dimensional) integer pointing to a codeword in a codebook.

quency-derivatives of the basic features, are chosen based on intuitions that they may provide some extra information in the speech signal. In an application, several of these vectors can be used together, under a framework of multiple-independent-codebooks (Lee, 1989). However, although all these vectors are meaningful physical measures, the choice of (the combination of) these vectors is usually not based on any consideration of their statistical correlation[3]. Therefore, when we use some algorithms in the speech recognizer that requires certain statistical properties of the observation vectors, this triggers a need for an investigation of the data. As a matter of fact (see section 3), there exist strong correlation, both between parameters within a vector (*within-vector correlation*), and between parameters of different vectors (*between-vector correlation*). The within-vector correlation concerning LPC-based cepstrum vectors is minimized by the algorithms calculating them (Huang et al., 1990). However, there is a rather strong within-vector correlation in the filter-bank parameters, intrinsic from the fact that the speech production organ is not a synthesizing vocoder composed of *independent* band-pass filters.

In addition to the direct use of the observation vectors obtained from signal processing procedures (called *original* in our presentation), linear transformations of these vectors into new vectors may serve as alternatives. The output of the front-end is in general a representation of the selected speech features. A linear transformation just produces another representation of the same features, thus the transformation is included in the front-end. We can nevertheless distinguish such a front-end from the conventional ones by calling it a *transformed front-end*. The transformed representation carries essentially the same information (if all the components are used) as the original representation, while the former, as having included a post-processing after conventional front-end, may provide better justification to the algorithms of the rest of the recognizer. In this study we investigate two linear transformations, namely, Principal Component Analysis (PCA) and Linear Discrimination Analysis (LDA). PCA and LDA both transform the original vectors into orthogonal dimensions (the parameters on these dimensions being called *components*), whereas the directions of these components are defined by the PCA based on maximal explanation of the variance in the data vectors by the lower-dimensional components, and by the LDA based on maximal discrimination by the lower-dimensional components between the clusters of the vectors representing different phonetic classes.

It would be more insightful if the comparison between single- and multiple-codebook approaches is made not only at full dimensionality (i.e. using all components of a vector), but also at various reduced dimensionalities (using only a selected sub-set[4] of components). A further technical limit cast by the approach of DDHMM (which is the setup of the current study) is that in comparing the single- and multiple-codebook approaches, the finite accuracy of the codebook encoding procedure would introduce less serious problems if the total number of dimensions is not too large. In other words, a comparison made at a lower dimensionality would be more sensible than at the full dimensionality. Furthermore, from an engineering point of view, it is interesting to investigate ways of reducing the number of parameters at the front-end, while keeping the performance of the whole system intact. However, it will be favorable to perform the dimensional reduction in a transformed domain, instead of the original

[3]Mathematically, statistical dependence and correlation are not the same concepts; but in our treatment, since both concepts are applied to real-world data where both properties describe more or less the similar situation, the two terms are used interchangeably.

[4]Generally, any sub-set of components can be used. But, due to the particular optimality of PCA and LDA (see section 3), and for the purpose of the present study, always a certain number of contiguous lower-dimensional components are selected.

domain. This is mainly for two reasons:

1. Each component in the original domain carries nearly the same amount of information, therefore leaving out some original components will lose too much information. In a transformed domain, on the other hand, information is compressed to the lower-dimensional components, therefore higher-dimensional components can be easily left out;
2. Each of the components in the transformed domain, especially the lower-dimensional one, explains a particular spectral feature, e.g. the energy fluctuation or tilting (Pols, 1977). Therefore, retaining or leaving out components in the transformed domain will keep us informed about the *sort* of information manipulated.

Since the higher-dimensional components in the transformed domain have relatively smaller amplitude and show irregular patterns, it is anticipated that they carry merely noise. Therefore, reducing the number of components in the transformed vectors will not only optimally reduce the data size at the front-end, but also remove the noisy part seen from the rest of a recognizer (i.e. excluding this noisy part may even lead to a higher score than otherwise using all information).

1.2. Choice of observation vectors to manipulate

In this sub-section, we explain the reasons for the choice of particular types of observation vectors to be manipulated in the present study.

Basically, the linear transformation can be applied either to a single vector, e.g. the spectrum (denoted X^P), to remove the within-vector correlation; or to a large vector composed of two vectors (by concatenating their components), to remove also between-vector correlation. Since the spectrum and its frequency derivative, or slope (denoted X^L) are totally correlated (see Appendix 1), the variance-covariance matrix of the composed vector is of reduced rank, therefore it is not possible to apply a PCA on that composed vector. For that reason, it only makes sense to apply transformation to X^P and X^L *separately*. On the other hand, however, since the time-derivative of the spectrum ΔX^P is not totally correlated with X^P, the variance-covariance matrix of the composed vector between them is of a full rank. It is then possible to apply a PCA on the composed vector of X^P and ΔX^P, which would also remove the between-vector correlation. Yet since the correlation between X^P and ΔX^P is smaller than that between X^P and X^L, the effect of the improper independence assumption between vectors would be accordingly less tangent. Therefore, in the current study, we choose to investigate the situation with the strongest correlation, namely, between X^P and X^L, to emphasize the difference in performance with single- and multiple-codebook approaches.

It has to be stated, however, that removing the within-vector correlation is mainly meaningful for a CDHMM (Continuous-density HMM) system. An explanation is as follows. In a (conventional) CDHMM recognizer, the acoustic observation vectors are modelled by some probability distribution functions (PDFs), such as a multi-variate Gaussian. Although no codebook is needed, the parameters in such PDFs should be estimated from a certain corpus of training data. To accurately model the multi-variate Gaussian distribution, not only the mean and the variance in each component, but also the covariances between the components, should be estimated with some time-consuming algorithm. PCA-transformed vectors have zero-covariance if modelled with a Gaussian, therefore they provide an efficient and accurate data representation to CDHMM systems. In a DDHMM system, on the other hand, the existence of within-vector correlation does not affect the accuracy of modeling, because no assumption has been made on the statistical properties of the data vectors ***when*** applying codebook encoding process. After the codebook encoding process, all the training and recogni-

tion algorithms that follow (considering now only one vector, thus only the within-vector correlation) only see a sequence of 1-dimensional codebook indices, regardless whether the vectors are correlated or not. In this sense, therefore, the linear transformation applied to single vectors separately, in our DDHMM system, only provides a means of manipulating the dimensionality.

For a clear summary of the factors that will determine which setups will or will not be considered in this study, with respect to insight they may provide, their technical feasibility and some relations between them, we list them in the following table (Table 1). Only filter-bank related parameters are considered.

	X^P and X^L		X^P and ΔX^P	
transformation setup	separate	composed	separate	composed
original within-vector correlation	strong		strong	
original between-vector correlation	very strong (totally correlated)		weak	
possibility to remove correlation with PCA	yes	no	yes	yes
vector(s) to apply PCA	separate	doesn't apply	separate	composed
removing correlation meaningful for:	CDHMM	doesn't apply	CDHMM	CDHMM DDHMM
possibility of comparing 1 and 2 codebooks	provided	doesn't apply	provided	not provided
			one more in between	
possibility of dimensional manipulation	provided	doesn't apply	provided	provided
within-vector correlation after PCA	removed	doesn't apply	removed	removed
between-vector correlation after PCA	reduced	doesn't apply	reduced	removed
violation of independence assumption when using 2 codebooks	remain	doesn't apply	remain	removed
difference of between-vector correlation between 1 and 2 codebooks	not seen	doesn't apply	not seen	seen, but small

Table 1. Relations and effect of factors concerned. From the sixth row down, all entries refer to PCA-transformed vectors. Both within- and between-vector situations apply to 1 and 2 codebook setups.

One asymmetric feature not very clearly seen from Table 1 is, that the possible comparison that *can* be made between the 1- and 2-codebook cases for X^P and X^L is different from that for X^P and ΔX^P. For X^P and X^L, since a PCA on the composed vector is not possible, both the 1- and the 2-codebook cases will apply PCA on the separate vectors of X^P and X^L, respectively, while for the 1-codebook case, the X^P and X^L

vectors after PCA manipulation will be composed into one vector. It will be seen (Section 4) that in the comparison between the 1- and 2-codebook cases, exactly the same vectors (the same extent of reduction in both within- and between-vector correlations) will be used, only in the 1-codebook case, the X^P and X^L vectors are handled in one codebook. This corresponds to the first column from the left in the table. The insight will be a *pure* effect of single- or multiple-codebook. For X^P and ΔX^P, on the contrary, since it *is* possible to perform PCA on the composed vector, it is desirable to do so, to remove also the between-vector correlation (although it is also possible to do the same manipulation as for X^P and X^L). However, after doing so, the large vector transformed from the composed vector should only be used in a 1-codebook case, since all the components of this large vector are from both X^P and ΔX^P, therefore it is not possible to split them into any two meaningful vectors and use them in 2 codebooks. As such, the comparison between the 1- and 2-codebook cases should be made with *not* exactly the same vectors: in the 2-codebook case with separate PCAs, only the within-vector correlations are removed, while the between-vector correlation is not (completely) removed; in the 1-codebook case, both within- and between-vector correlations are removed. The manipulation between X^P and ΔX^P would correspond to the third and fourth columns in the table, for the 2- and 1-codebook cases, respectively. The insight that might be obtained would be, a *composite* effect of multiple-codebook and between-vector correlation, or a *complete* effect of the violation of the independence assumption. Although the manipulation between X^P and ΔX^P is not applied in this study, we can see that it also has some advantages. In summary, the manipulation between X^P and ΔX^P provides one more possibility for comparison than that between X^P and X^L, but this is not a very fair one.

This paper is organized as follows. In section 2, the relevant aspects of the recognizer used in the present experiments are briefly described. Section 3 gives a detailed description of data transformation of PCA and LDA, and gives an analysis on the correlation among the parameters, before and after the transformations are applied. In section 4, (1) three different ways of codebook design, and (2) 1- or 2-codebook encoding procedures, using the vectors transformed in the same way, are described. Also some topics in training of the codebooks are mentioned. Section 5 explains the training of the HMMs and recognition tests using the different front-end transformations, and different remaining number of components. Next, the recognition scores are given, and a few comparisons are made. The last section concludes with a general discussion. Appendix 1 gives an analysis of the ranks of the variance-covariance matrices S's of the composed vector of X^P and X^L, and that of X^P and ΔX^P, respectively. In Appendix 2, the effect of diagonalization of the separate S's of X^P and X^L with both PCA and LDA is explained, and the calculation formulae for the composed S's of the separately transformed vectors are given in terms of the original S's.

2. The speech recognizer

We used a recognizer developed at our institute (van Alphen, 1992) to perform the experiments. It is an HMM-based continuous speech recognizer with a vocabulary size of 230 Dutch words. The basic HMM models correspond to 39 context-free phone-like units (PLUs), including a basic Dutch phoneme set and some non-verbal sound-categories. The training data consist of 100 sentences in a fictitious banking task, each sentence spoken three times by a single male speaker. The (sentence) utterances used for recognition tests consist of an independent set (another utterance) of the first 34 out of the 100 sentences, spoken again by the same speaker. The training is performed with the standard Baum-Welch re-estimation algorithm with extension to access multiple codebooks, using the forward/backward recursion to reach a maxi-

mum likelihood of the observation sequences of the training *sentence* utterances given the parameters of the *PLU* models. The recognition is performed with the Viterbi algorithm (see e.g. Juang and Rabiner, 1992).

The utterances were sampled at 16 kHz and a 15-band FIR filter bank analysis was performed and the band values were subsequently converted into decibels: $X^P = \{x_i^P\}$, where $1 \le i \le 15$. These vectors were obtained at 8 msec frame intervals. The slope vectors $X^L = \{x_i^L\}$ were derived as the difference between the next-but-one band values: $x_i^L = x^P_{i+2} - x^P_i$, where $1 \le i \le 13$. The data corpus D used for vector quantization of all codebooks consisted of 32768 vectors[5] from the first 76 of the 300 training sentence utterances. Since the acoustic data in the 76 sentences were sufficiently representative of the whole available data, we did not select frame vectors randomly from all the utterances.

In the codebook design procedure, the whole corpus D represented by a certain kind of vector (e.g. the original full-dimensional X^P or its various transformed vectors) is used to obtain one codebook. Then any incoming vector (either in training or testing data) is represented by one codeword in the codebook, called an observation: $O^v(t)$, at time t, where v refers to a certain kind of vector. One of the formulae involved in the forward/backward training procedure reads

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1)\, a_{ij} \sum_{\forall v} b_{ij}^{v}(O^{v}(t)), \qquad (1)$$

where α is the forward probability, and the HMM parameters a and b are, respectively, the transition and observation probabilities before the current iteration of the re-estimation. This α and the backward probability β are used to calculate the new a and b. As can be seen from the inner summation in (1), all different kinds of observations involved in a particular front-end take part in the calculation. Therefore, for a front-end with multiple-codebook of both X^P and X^L, it sums over two terms, which implies that more statistics of the data has been taken into calculation than a single-codebook front-end, for which the inner summation has only one term.

3. Transformation of the vectors

The orthogonal basis for PCA transformation is defined by the eigenvectors of the (sample) variance-covariance matrix[6] S of the whole corpus D of either X^P or X^L vector. Let us denote the data vector of an observation frame as $X^P = (x_1^P, x_2^P, \ldots b_M^P)^T$, $M = 15$, and $X^L = (x_1^L, x_2^L, \ldots b_M^L)^T$, $M = 13$, respectively, where T denotes transpose. Then[7] $S = E\{[X - \mu][X - \mu]^T\}$, where E denotes the sample mean over D and $\mu =$

[5]The total number of vectors is limited by the algorithm.

[6]When S is calculated from real data as $S = (1/N)\Sigma^N_{i=1}(X_i-\mu)(X_i-\mu)^T$, where no theoretical probability density function (pdf) is assumed or known, S is called a *sample* variance-covariance matrix, and $\mu = (1/N)\Sigma^N_{i=1}X_i$ is called a *sample* mean. If the pdf $p(X_i)$ is available, the theoretical variance-covariance matrix would be calculated as $S = \Sigma^N_{i=1}(X_i-\mu)p(X_i)(X_i-\mu)^T$. In both cases, the matrix can be written as $S = E[(X-\mu)(X-\mu)^T]$, only the former is referred to with an extra adjective 'sample' (see e.g. Johnson, et al. 1988). We replace $(N - 1)$ by N for simplicity, as N is very large in our present case (c.f. Pols, 1977).

[7]From now on we drop the superscripts P and L whenever the discussion is the same for X^P and X^L, unless it is necessary to distinguish them.

$E\{X\}$. The (column) eigenvectors $V = (v_1, v_2, ..., v_M)$ satisfy $SV = V\Lambda$, with eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_M)$, and $\lambda_1 > \lambda_2 > ... > \lambda_M \geq 0$. The j-th principal component is then $y_j = v_j^T X$, $(1 \leq j \leq M)$. The eigen-system arranged in this way guarantees that y_1 explains most of the variance in D, y_2 explains most of the remaining variance, and so on. Therefore, using only the first $L \leq M$ components in the front-end will be optimal in a particular sense of data reduction, namely, maximally reserving the variance by a limited number of parameters (Pols, 1977).

In the LDA transformation (Cooley et al., 1971), two matrices instead of one are made from the same D. Firstly the whole D is partitioned into K non-overlapping classes D_k, each corresponding to a phoneme, and to one HMM model. The membership of each frame vector is assigned to one of 39 phoneme classes using the segmentation information in the hand-labeled training sentences. Then the *Among-class* matrix U is calculated as

$$U = \sum_{k=1}^{K} R_k (\mu_k - \mu)(\mu_k - \mu)^T, \tag{2}$$

where R_k is the number of vectors in class k, μ_k is the centroid of the class k, and μ is the *grand centroid* of D, i.e. the sample mean as in the PCA transformation. The *Within-class* matrix W is calculated as

$$W = \sum_{k=1}^{K} \sum_{l \in D_k} (X_{kl} - \mu_k)(X_{kl} - \mu_k)^T, \tag{3}$$

where D_k is the cluster of class k, X_{kl} denotes those vectors belonging to class k, and l refers to individual vectors. Whereas U is an indication of dispersion of the classes, W expresses the dispersion of individual vectors within each class and pooled over all the classes.

The LDA transformation is then based on the solution of eigen-equation $(W^{-1}U)V = V\Lambda$. The treatment is the same as in PCA, while the V and Λ will be different. Furthermore, since the matrix $(W^{-1}U)$ is not symmetric, contrary to the S in PCA, the algorithm for obtaining the eigen-system is different. The eigenvectors will also be arranged in descending order of the corresponding eigenvalues, and after transforming the vectors into the component space, also the first $L \leq M$ components will be used in the front-ends.

A geometric view on PCA and LDA may help to understand the two transformations. In PCA, the direction of the first component lies along the direction with most of the variance of the original data corpus, i.e. the longest axis of the multi-dimensional ellipse of the whole data cluster. The second component finds its direction both perpendicular to the first one, and along the second longest axis. The direction of any higher-dimensional component is found perpendicular to all the lower-dimensional ones, and along the next longest axis. Therefore, by using only some number of lower-dimensional components to represent the data, the information lost is minimal. An analysis of PCA is given in Okamoto (1969) where it is pointed out that, although the optimal properties in terms of either variance explanation or of information measure (in terms of entropy) are not the same, these two merits are both possessed by PCA. Therefore, in this sense, more variance corresponds to more information.

In LDA, on the other hand, firstly the whole data cluster is not treated as a whole, but is partitioned into classes. Then, the direction of the first component is found along which the discrimination function between all the classes will be maximal. It is

straightforward to imagine two classes of data of equal-sized circle shape in a 2-dimensional space, where the first component should be along the direction of the straight line through the centers of the two circles. The projection of the data points onto this line will provide more separation of the data points from the two classes than with any other possible line. This is to say, that using the first component alone to represent the data will reserve a maximal power of discrimination between the two classes. For cases with more classes and in a space with more dimensions, and if the shapes of the clusters of all classes are not regular, the operation is more complicated and the distribution of the data points inside each cluster will also play a role. The eigen-problem formulation provides an optimal solution (Cooley et al., 1971).

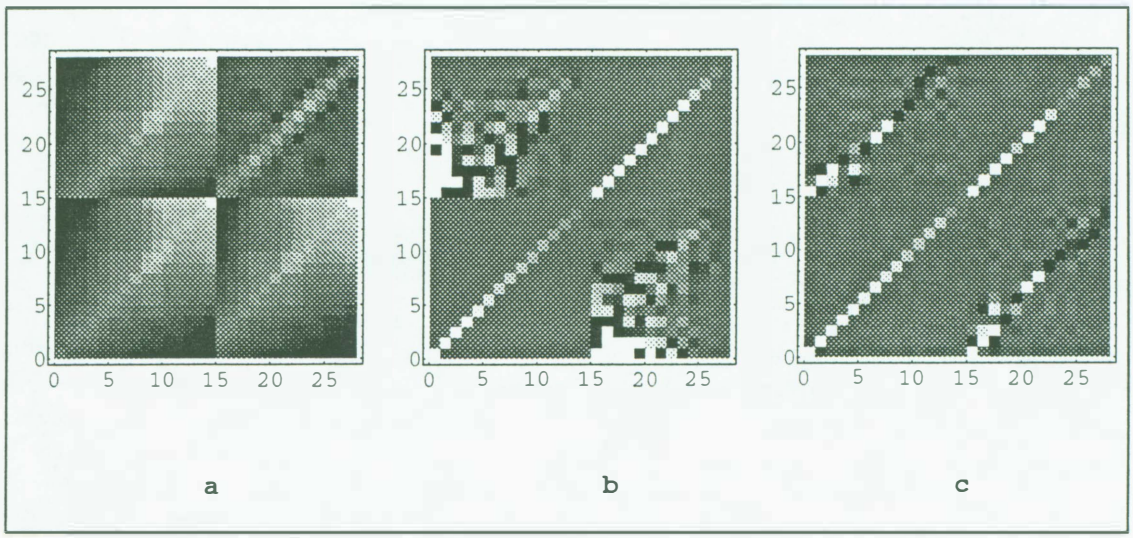

Figure 1. (a) Original, (b) PCA and (c) LDA transformed S matrices. Brighter areas indicate greater positive values. Indices 1-15 are for spectrum, and 16-28 for slope parameters.

We have studied the correlation between the filter-bank components, before and after PCA and LDA transformations. This is depicted in Fig. 1. In (a), the original vectors of both spectrum X^P and slope X^L are used. The lower-left region of the whole square is the variance-covariance matrix of the X^P vectors, where the variance is indicated along the diagonal (starting from the lower-left corner), whereas the off-diagonals are covariances between different parameters. The upper-right region refers to X^L vectors. As can be seen, the within-vector correlation indicated in both the X^P and X^L regions are very strong. Furthermore, it can be seen that the between-vector correlation of X^P and X^L (i.e. the lower-right region) is also very strong. In Fig. 1(b), the variance-covariance matrix is shown of the data vectors Y^P and Y^L, after PCA transformations performed separately for X^P and X^L, respectively (i.e. find the eigen-systems of X^P and X^L independently, and then transform them by their own eigenvectors). In both regions within X^P and within X^L, the off-diagonal entries become zero (Appendix 2). This diagonalization process by two separate transformations also reduces the correlation between X^P and X^L, however it does not remove correlation completely. In Fig. 1(c), the effect of separate LDA transformation is shown by the variance-covariance matrix of the transformed data vectors. Since the LDA is *not* based on the eigen-system of the variance-covariance matrix S of the original data vectors, but rather, based on the eigen-system of $(W^{-1}U)$, it does not fully diagonalize the entries within the two block regions (Appendix 2). However, the general within-vector corre-

lations of both X^P and X^L are reduced. It is also worthwhile to note that the diagonal entries, i.e. the variance, in the LDA-transformed vectors are not arranged in a strictly descending order[8], as it is with PCA. This is because the LDA eigen-system is *not* based on the variance-covariance matrix, yet we *are* calculating the variance-covariance matrix of the LDA-transformed vectors (Appendix 2).

4. The transformed front-ends

For a transformed front-end, firstly each codebook is designed, based on (or 'trained' with) one kind of (or a composition of) data vectors from a certain corpus of training data which is representative for all the incoming acoustic vectors of an application. Different from a conventional codebook encoding process, the training data vectors and the codewords should be represented in a transformed domain. Then each incoming data vector (having undergone the same transformation as the training data) is *encoded* as the codeword which is closest to it according to a certain distance measure[9] (Pols, 1987), also in the transformed domain.

4.1. Three versions of codebook encoding

In our experiments, we want to test the recognizer with front-ends with many different number of dimensions, and under both PCA and LDA transformations. For each such test, we need a different codebook which is trained from the data vectors of a certain dimensionality *and* related to a certain transformation. Furthermore, we want to compare the difference between 1- and 2-codebook cases under the condition of the same dimensionality and transformation. Therefore, many distinct codebooks are required[10]. However, the training of codebooks, as an iterative process, is very time-consuming.

Since those codebooks with reduced dimensionalities are made from certain number of dimensions from the transformed data vector, we may come up with an idea to reverse the order of the processes of transformation and codebook design. For simplicity, we call such a process *truncation*, which takes only a certain number of lower-dimensional components after the transformation, whereas the actual number of remaining dimensions will be discussed in the next section concerning experiments. On the other hand, the term *transformation* will merely refer to the process of changing the representation of either data vectors or codewords (recall footnote 2 that, codewords are also multi-dimensional vectors) into the new space, at *full* dimensionality. To be systematic, we list three different versions of encoding (with their abbreviated

---

[8]It has to be stated that, owing to the limited range of gray levels in the density plot, the three plots in Fig.3 adapt to different range of interests (all linear), to emphasize the distributions of both diagonal and off-diagonal entries. The analysis on the comparison is based on the actual values of these entries, instead of the illustration. The matrices in (b) and (c) have been calculated with the formulae in Appendix 2.

[9]In our study, the Euclidian distance is always used.

[10]Imagine a three-way matrix with (1) type of transform, (2) number of remaining dimension, and (3) number of codebook, respectively, as indices of entries. Along the third index, the two entries correspond to three *different* codebooks: one for the composed vector, and two for the two separate vectors, respectively. Therefore, if we want to test at 5 different dimensionalities, we need 2 x 5 x 3 = 30 different codebooks !

names), by changing the order of the transformation, truncation and encoding processes, on data vectors and on codewords. (a) *explicit*. Each codebook is trained explicitly by using the data after transformation and truncation, whereas the encoding of the incoming vectors undergoes the same procedure; (b) *transformed*. Only one codebook is made from the original data vectors (without transformation), and then the transformation and the truncation are applied to the codewords of the codebook. The incoming data vectors are firstly transformed and truncated, and then encoded with such a codebook; (c) *truncated*. One codebook is made from the transformed, but not truncated, training vectors (i.e. full dimensionality), and then such a codebook is truncated. The incoming data vectors are transformed and truncated and then encoded. Note that all three aforementioned versions only refer to one transformation at a time, i.e. PCA or LDA, and to only one part of the data vectors concerned, e.g. the spectrum X^P or the slope X^L. A schematic diagram of the three versions can be seen in Fig. 3 in the next sub-section 4.2.

The transformed and truncated procedures for obtaining the codebooks, though saving computing time considerably, introduce problems of encoding quality. This is

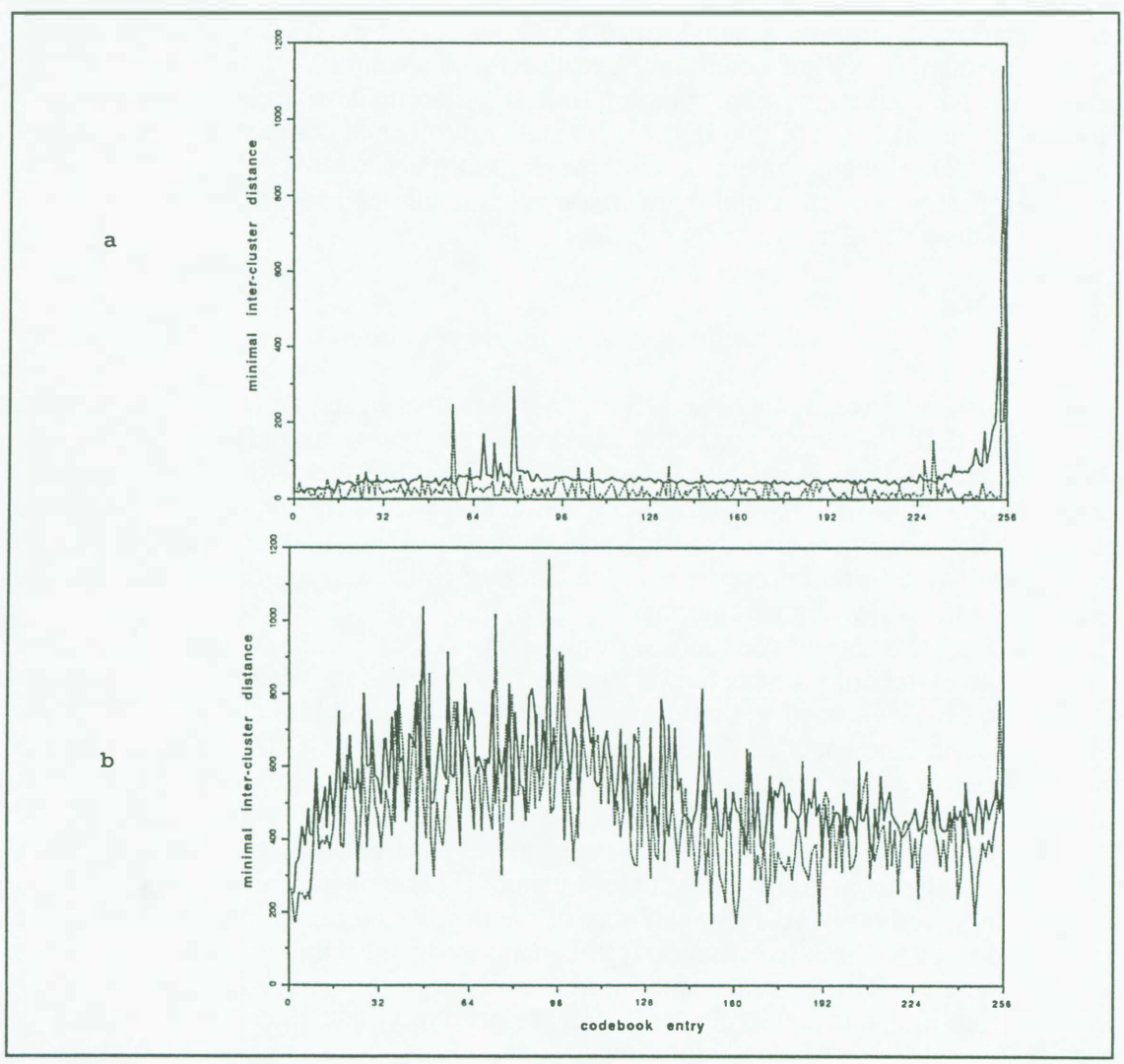

Figure 2. The minimal inter-cluster distance for (a) 1-dimensional and (b) 5-dimensional codebooks. Solid curves are for the explicit, and dotted for the transformed versions, respectively.

mainly due to the fact that such codebooks no longer satisfy completely the two necessary optimal conditions as guaranteed by the explicit procedure, namely, the centroid condition and the nearest-neighbour condition, respectively (see e.g. Gersho and Gray, 1992). As an indication, we give an example of a comparison of one of the quality measures between the explicit and the transformed versions, i.e. the inter-cluster distance (see Fig. 2). This distance is measured between every cluster center and its nearest cluster center, thus a greater value indicates a better quality (clusters being well apart). As can be seen from the figure, for both the 1-dimensional and the 5-dimensional cases, the transformed codebooks show systematic smaller values than the explicit ones, thus a degraded quality[11].

However, these optimal conditions merely specify the quality of the codebook itself, not the quality of the whole encoding process, if some variations are made from the conventional codebook encoding approaches. This is because the codebook design procedure is only a part of the encoding process with transformation: The quality of the codebook, as well as the optimal criterion used in the training algorithm, are defined between the *immediate* input and output of the codebook. For example, the quantization error is measured between the vector $\boldsymbol{Y}$ (which has been transformed and truncated from an original vector $\boldsymbol{X}$) and the encoded codeword $\boldsymbol{Z}$, instead of directly between $\boldsymbol{X}$ and $\boldsymbol{Z}$, which is a quality measure of the whole encoding process including transformation. Therefore, a transformed codebook might even provide better encoding results than a explicit codebook, because the distribution of codewords of the former may have taken more information from the training data (from all components) than that of the latter (an 'explicitly' exact smaller number of components). An analytical comparison of the goodness of the three encoding procedures turns out to be difficult. Therefore, an experimental comparison is made, under a few selected conditions, and the results will be shown in section 5.

4.2. Single versus multiple codebooks

We compare between single and multiple codebooks in the front-end. It has to be emphasized that, the single codebook is made of the *two* (manipulated) data vectors from spectrum X^P and slope X^L. This should not be confused with the conventional setup where a single codebook refers to the situation with only *one* data vector. In the multiple codebook case, also two data vectors are used, but they are handled by two codebooks. The different procedures in the cases of single and multiple codebooks are described in the following paragraphs.

In the case of multiple codebooks, firstly both vectors X^P and X^L are used to train two separate codebooks, respectively. Then the incoming data vectors, either of training (for the HMMs), or of the unknown ones for recognition, are encoded with these two codebooks. The encoded observation sequences $O^P(t)$ and $O^L(t)$ are used either to train the HMMs using formulae including (1), or to be used in recognition by the related algorithms. Since the encoding process is done completely *separately* for X^P and X^L, the whole procedure for the three versions is straightforward. This is because, everything in the front-end (that is, before using (1)) is being manipulated for X^P and X^L separately, and no items from half way of the processing are combined. Any of the three version procedures works nearly the same as in the explicit version, only the order of the transformation, truncation, and encoding are swapped. Here we obtain two observation sequences $O^P(t)$ and $O^L(t)$, respectively, and they are dealt with by formula (1) and all the related algorithms.

[11] For higher-dimensional cases, however, the difference is less obvious to see.

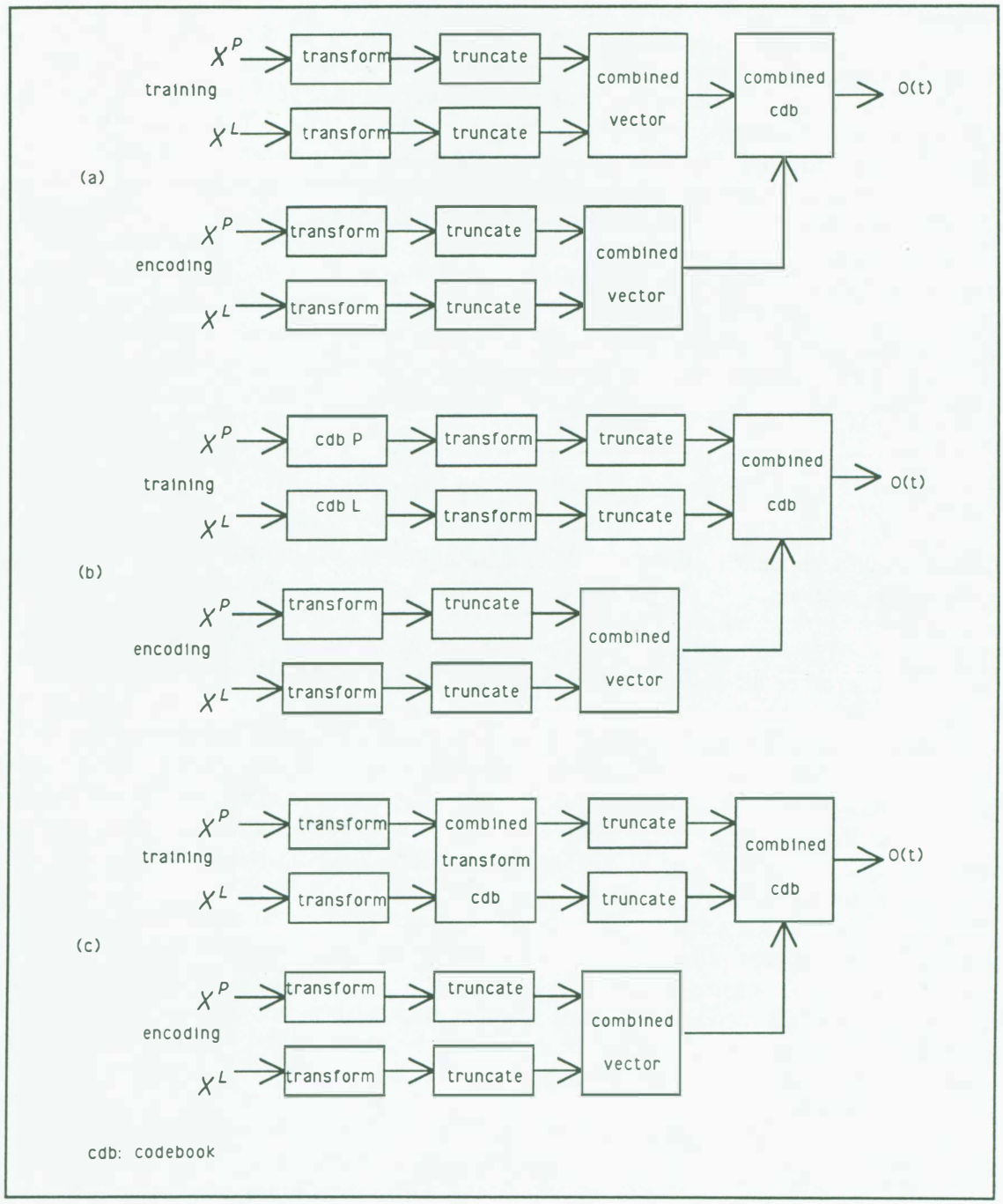

Figure 3. (a) explicit, (b) transformed, and (c) truncated versions of front-end encoding with a single codebook.

On the other hand, in the front-ends with one codebook, the transformed and truncated items (training data vectors, incoming unknown vectors, and codewords) from the spectrum and the slope[12] are always combined into one large item, by concatenating the dimensions together. This is done slightly differently for the three versions, particularly in obtaining the codebooks. For clarity, we illustrate them in a figure. In

[12]We do not use notations X^P and X^L to indicate items after any process, in order to avoid confusion.

the explicit version (Fig. 3(a)), vectors X^P and X^L are both firstly transformed and truncated, and then they are concatenated into larger vectors used to train the codebook. In the transformed version (Fig. 3(b)), two codebooks are firstly trained by the original vectors X^P and X^L respectively. These two codebooks[13] are then transformed and truncated and concatenated into one larger codebook. In the truncated version (Fig. 3(c)), firstly the full-dimensional vectors, transformed from both X^P and X^L, are concatenated into a large full-dimensional vector and used to train one large codebook. Then the two parts of this codebook corresponding to X^P and X^L are truncated and concatenated into one codebook. In all the three versions, the encoding of the incoming vectors X^P and X^L are the same, i.e. they are both firstly transformed and truncated and concatenated into a larger vector, and then encoded with one of the three codebooks into *one* observation sequence $O(t)$ (Note the difference from the 2-codebook case). Again, all the aforementioned processes only refer to one transformation (PCA or LDA) and to a particular number of remaining dimensions in the truncation.

5. The recognition experiments

In our system, the size of the codebooks is 256 codewords. In order to see whether this codebook size might be too limited for the codebook using both X^P and X^L vectors (because there is more information than from one vector alone), a pilot test was done with a size of 512. The result, that will not be discussed here, showed no significant difference. Therefore all subsequent experiments used codebooks of one size being 256.

For each of the conditions explained in section 4, the HMMs are trained and recognition tests are performed without using grammar constraints, i.e. each word in the vocabulary has the same probability of occurrence. This is done to emphasize the pure effect of the different front-ends, whereas the improved performance with e.g. a *bigram* grammar may show a ceiling effect of the scores[14] which would make it difficult to clarify the current investigation.

The word percentage correct scores (only counting deletion and substitution errors) and total correct scores (also counting insertion errors) are shown in the following tables. These two scores are indicated in the tables as '%' and 'total %', respectively. The comparisons are made at the same number of total components used, between various cases concerning single- and multiple-codebook, PCA or LDA, etc. An interpretation of the results in Tables 2 through 5 will be given in the conclusion of the next section.

The arrangement of the tables is as follows. In Table 2, a comparison is made between the PCA and LDA transformations. These tests were done with the explicit version of transformation, with two (i.e. multiple) codebooks, at the total dimensionality of 2, 4, 10, 16, and 28 (full), respectively[15]. It can be seen that at the total dimensionality of 10, the increase of scores with the increase of dimensionality has reached a transition point from a steep increase to a saturation. Therefore, all the further com-

[13]Actually, all the multi-dimensional codewords in the codebooks.

[14]For example, by using a bigram grammar which only allows those between-word transitions appearing in all the test sentences, the performance score at a dimensionality of 10 increases to nearly 100 % from about 70 % without a grammar, using our system (van Alphen, 1992).

[15]For all cases, equal numbers of components are taken from X^P and X^L, e.g. 5 from each, to form a total dimensionality of 10; except for the last case, where all 15 X^P and all 13 X^L components are used.

dimensionality			PCA		LDA	
X^P	X^L	total	%	total %	%	total %
1	1	2	15.8	5.3	9.0	6.5
2	2	4	50.0	42.6	45.7	39.4
5	5	10	73.3	68.6	69.9	62.7
8	8	16	77.0	71.1	74.2	66.2
13	15	28	80.0	72.1	78.6	72.1

Table 2. Word recognition scores with PCA and LDA explicit transforms with two codebooks.

conditions		PCA		LDA	
codebook	version	%	total %	%	total %
one ($X^P + X^L$)	explicit	61.2	60.9	66.8	66.5
	transformed	52.8	51.9	55.0	54.4
	truncated	63.4	63.0	63.0	62.4
two (X^P, X^L)	explicit	73.3	68.6	69.9	62.7
	transformed	70.8	65.8	66.5	59.6
	truncated	72.7	66.2	69.9	63.4

Table 3. Word recognition scores of PCA and LDA with three versions and single and multiple codebooks, all at a total dimensionality of 10.

	original		PCA		LDA	
codebook	%	total %	%	total %	%	total %
one ($X^P + X^L$)	70.2	69.9	68.6	68.3	69.6	69.4
two (X^P, X^L)	76.7	70.2	78.0	72.1	78.6	72.1

Table 4. Word recognition scores with full dimensionality, original and explicit transforms.

parisons under other conditions were made with a total dimensionality of 10 only. Table 3 shows the results of both PCA and LDA, with either single (one) or multiple (two) codebooks, using all three versions of transformation, and with a fixed total number of dimensions of 10. Table 4 gives a comparison between the explicit version of both the PCA and LDA transformation with both one and two codebooks at the full dimensionality. This is also compared with the front-end of the original spectrum and

spectrum only (one codebook)		slope only (one codebook)		double spectrum (two codebooks)	
%	total %	%	total %	%	total %
68.0	67.4	66.8	66.5	66.8	57.5

Table 5. Word recognition scores with one codebook of spectrum and slope only, respectively, and double use of spectrum codebook. All are done with original vectors of full dimensions.

codebook	total dimensionality = 10	total dimensionality = 28
one $(X^P + X^L)$	- 153764.0	- 149343.4
two (X^P, X^L)	- 123121.1	- 117463.2

Table 6. Log likelihood with LDA explicit transformation, given the whole training sequences.

slope vectors, to see the pure effect of transformation without truncation[16]. Table 5 gives the results of spectrum-only and slope-only original front-ends, while comparing these with a front-end using the spectrum vectors twice in a 'multiple' codebook way ('double spectrum'). This last comparison is made to investigate whether or not the improved scores with multiple codebook (as in other tables) is solely due to the fact that twice as many training observation sequences are involved, no matter from what kind of vectors they have been derived. It has to be emphasized that, in all the tables, 'one codebook' refers to the case also with *two* observation vectors derived from X^P and X^L, however they are handled with *one* codebook and denoted by $(X^P + X^L)$, whereas the case of two separate codebooks is denoted by (X^P, X^L). In the 'spectrum only' and 'slope only' cases in table 5, however, really *one* vector is used.

In addition to the comparison of the recognition scores, some comparison of the (log) likelihood of the set of HMM parameters, for a given length of the total training data, can be informative. In general, a higher likelihood would mean a better training result (i.e. the trained HMM models are able to model the training data *more likely*), according to the maximum-likelihood criterion of the parameter estimation (such a measure has been used in e.g. Ljolje, Ephraim and Rabiner, 1990 and Deng, 1991). One such example is given for the single and multiple codebook approaches using LDA, with a total dimensionality of 10 or 28 (full). As can be seen from Table 6, the multiple codebook cases are superior, with the two listed dimensionalities. It also can be seen from the table, that the likelihood of the 2-codebook case at the total dimensionality of 10 is higher than that of the 1-codebook case at the total dimensionality of 28. This can be interpreted as that, the HMM models whose (two) observation sequences are from two independent codebooks with a total dimensionality of 10, are more likely to model the speech signal in the training data, than the (other) HMM

[16]Although at the full dimensionality, a linear transformation does not lose any information, the error of vector quantization introduced to the original data vectors and to the transformed ones are in general different. Also, any particular codebook design algorithm is data-dependent, thus even the 'same' data represented in the transformed domain will lead to a codebook, which, after an inverse transformation, is in general different from the codebook obtained directly from the original data. This is because the order of the two processes, i.e. PCA and vector quantization, are in general *not* reversible.

models whose observation sequence is from one codebook with a dimensionality of 28. To put this in a more general interpretation, this means that the existence of a greater number of observation sequences is more helpful than the use of a higher dimensionality. Note that a higher dimensionality by itself also helps. It has to be pointed out that the likelihood values only provide a *relative* comparison between different training procedures with the same training data; The *absolute* values and the scaling, on the other hand, are related to complicated calculations in the training procedure, and give little clear information about the quality of the models as a whole.

6. Conclusion and Discussion

From the experiment results, the following conclusions can be drawn. The attempt in the current study to arrange many different processing conditions makes the conclusions generally valid for similar systems, particularly for DDHMM systems with a comparable vocabulary size.

(1) The total number of dimensions of data vectors can be reduced significantly in the transformed domain (Table 2). This is in agreement with the result of Le Cerf et al., (1992). This result can be loosely[17] attributed to the fact that, both the major part of the information in the speech signal is preserved in the lower-dimensional components, and this information is conveyed in a suitable form for the HMMs;

(2) The very similar scores between the cases with the same (full) dimensionality, but with or without transformations (Table 4), indicate that although the PCA and LDA procedures decorrelate the within-vector correlation, this does not affect the performance of such a system. This is in agreement with our theoretical expectation in the introduction, that the removal of the within-vector correlation is not meaningful for a DDHMM system;

(3) The fact that the score with one codebook is significantly lower than the score with two codebooks at the same total dimensionalities (Table 3 and 4) indicates obviously that the multiple independent codebook approach is superior to the single codebook approach, regardless of the actual residual between-vector correlation after the transformation. This conclusion is true for all the three versions of encoding procedures (Table 3). This generalizes the conclusion in a way, because the three versions can be regarded as using three different kinds of codebooks;

(4) A test with a 'double spectrum' did not give higher scores than the (single) spectrum-only case (Table 5). This indicates that the higher performance of a multiple codebook really comes from the *useful* information from the extra codebook. In other words, it is not just the larger number of observation sequences that improves the recognition; instead, any additional observation sequence should really come from a data vector that is not completely correlated with the first one. In this comparison, the 'double spectrum' case uses two identical vectors (of course totally correlated with each other) in a multiple-codebook setup, and the score is not improved, but even degraded slightly from the single spectrum case. Note that all the other cases with two code-

[17] Even with a conventional full-dimensional original representation, we do not known for sure whether the flow of information in an HMM-based recognizer is strictly supported in a correct way or not. Nevertheless, the performance with a manipulated front-end can only be compared with the original one as reference.

books always lead to higher scores than the 1-codebook cases (conclusion 3);

(5) LDA leads to poorer performance than PCA, except at the full-dimensionality (with only transformation and without truncation, the performance is about the same as with the original vectors). This is due, at least in part, to the fact that the classes used in LDA are *phonemes* while the recognition scores are given in terms of *words*. This mismatch between the unit of HMM models and the unit of recognition items is intrinsic to the structure of all the *continuous* speech recognizer with sub-word HMM models. Nevertheless, this is one of the most general, and difficult problems with such systems, and requires further research;

(6) The transformed front-ends always performed worse than the explicit ones, while the truncated front-ends gave comparable results with the explicit ones. This is probably because the transformed encoding process violates the optimal condition of codebook design more seriously than the truncated process. Recall the (b) and (c) parts of Fig. 3, for the transformed and truncated versions, respectively. We can see that the latter obtains its codebook(s) from the transformed data vectors, whereas the former does this from the original vectors. In this sense (and there may be other reasons, too), the codebook in the truncated version resembles the data representation of the incoming data vectors (which are to be transformed in the same way and then to be encoded) better than the transformed version. This conclusion has a direct implication to applications where calculation efficiency is essential; and

(7) None of the transformed front-ends with reduced dimensionality out-performed the scores of original full-dimensional vectors significantly. Some slightly higher scores e.g. the PCA explicit score of 77.0 % with 16 dimensions (Table 2), than the original score 76.7 % at the full dimensionality (Table 4), may be due to technical reasons, and does not provide general significance. In other words, the transformation techniques in the present study may be useful to reduce the number of parameters and correlation between them, but it is not very useful to *improve* the recognition.

In a DDHMM-based speech recognizer, the discrete front-end has some limitations. Although such front-ends suffer less from the improper modeling assumptions with respect to the statistical distribution of the information source than the continuous front-ends (which make one additional assumption when using parametric models), the finite partitioning of the acoustic space introduces problems of accuracy in representing the original spectra. Linear transformation such as PCA and LDA can find better partitions of the acoustic space in the sense of representing important spectral features with fewer parameters. As the later stages in the recognizer only use the 1-dimensional representation of observation sequences, the saving of parameters has limited significance to the whole system.

It has to be noted that the recognition scores presented in the previous tables are only counts of number of correctly recognized words, regardless of *which* words are actually recognized wrong. As a matter of fact, in some of the cases with similar scores, the error sets are not the same. Therefore, merely comparing the scores only gives a statistical figure, and provides limited insight into the effect of the front-ends. However, comparison of the recognition scores is still the main method used in the field, simply because a high score *is* the ultimate goal for a recognizer to achieve.

Manipulation of the information representation at the front-ends, while judging the quality of the front-ends by means of the performance of the whole recognition system, has certain drawbacks. This mainly has to do with the fact that the operation on the front-ends is isolated from all the algorithms in the rest of the system, which are

kept the same for all the tests. For example, the 'language match factor'[18] is set equal for all the tests, while actually it demonstrates different optimal values for different front-ends. Further work along this line may investigate some conditions in the training and recognition algorithms, which must be set differently for the different front-ends to be optimal.

The multiple codebook approach is investigated in this study with only two codebooks. However, the same analysis and manipulation can be easily extended to systems with more than two codebooks. Accordingly, all the pair-wise between-vector correlations should be taken into account.

The likelihood measure is neither the only measure on the model quality, nor the only criterion that guides the algorithms for estimating the model parameters in the training of HMMs. Although the maximum-likelihood training is the simplest one (for both understanding and implementation), the likelihood is neither directly related to the discrimination power (and more detailed distinction between discrimination at frame, phone, and word levels, see Bourlard, 1992), nor to the actual goal of the system: recognition of the words. Several different estimation algorithms have been developed where other criteria than maximum-likelihood are used (see e.g. Ephraim and Rabiner, 1990). These algorithms generally have a more profound mathematical background, but are also more complicated to implement, than is the maximum-likelihood. It is then obvious that, a higher value of likelihood does not necessarily imply a higher recognition score (a counter example can be found in Ljolje, Ephraim and Rabiner, 1990), if different training algorithms are concerned[19]. (Note that the likelihood is a measure that can be calculated from models obtained with many different training algorithms, while it is used as optimal criterion in the *maximum*-likelihood algorithm.) The research in this field is still quite active, and is generally aiming at solving the problem of the mismatch between the optimum criteria in parameter estimation on the one hand, and the optimal recognition performance on the other hand.

The present study mainly investigated the correlation between the parameters of the front-end vectors (therefore more or less the dependence between them as well), for a few setups which are governed by the technical feasibility of a DDHMM system. However, the correlation between parameters is only one of a few violations of the HMM algorithms. A more serious assumption has been made on the statistical independence between the subsequent observation frames (over time). It must be clarified that a justification of such an assumption cannot be implemented directly with the current technique of HMM systems, both used in this study, and in most of the open literature. Including time derivatives in the vectors, or using context-dependent HMMs may be some of the alternatives to tackle the problem. However, an ultimate solution to such problems may rely on the modification of the modeling algorithms, and eventually the Markovian assumption of the finite-length dependency between the speech frame vectors over time (Juang and Rabiner, 1992). Further research, both theoretical and technical, is required to obtain more insight.

---

[18]It is a pure technical effort artificially put in the recognition algorithm to balance the probabilities of insertion and deletion types of word errors.

[19]The comparison in our study is valid, however, because we are restricted to the maximum-likelihood training algorithm.

Appendix 1. Ranks of S's

In this study, we often use the variance-covariance matrix S of a large vector X composed of two vectors e.g. X^P and X^L, by concatenating their components. An important property, namely, the rank of this large S is useful in the analyses. This rank can be derived by the special relations between the entries s_{jk} of S, which can in turn be derived from the relations between the components x_i^P of X^P and x_i^L of X^L, respectively.

The slope component is a simple linear combination of the spectrum ones,

$$x_i^L = x_{i+2}^P - x_i^P, \qquad (1 \le i \le 13). \tag{A1.1}$$

Let us look at any column in the region corresponding to the right-hand part of the (28 x 28) matrix S. The elements in this region are denoted as s_{jk}, with $j > 15$ and $k \in [1, 28]$. Then, for any such j, there is a corresponding column l in the left-hand region ($l \in [1, 15]$), with a relation $j = l + 15$. For convenience, we denote the relation in (A1.1) by the elements x_l of the large composed vector X, i.e.

$$x_{l+15} = x_{i+2} - x_i, \qquad (1 \le i \le 13). \tag{A1.2}$$

Then we have

$$\begin{aligned} s_{jk} &= E\{(x_j - \mu_j)(x_k - \mu_k)\} \\ &= E\{[(x_{l+2} - x_l) - (\mu_{l+2} - \mu_l)](x_k - \mu_k)\} \\ &= E\{[(x_{l+2} - \mu_{l+2}) - (x_l - \mu_l)](x_k - \mu_k)\} \\ &= s_{l+2,k} - s_{lk}. \end{aligned} \tag{A1.3}$$

This is to say, that the whole column j of S is a linear combination of columns $(l+2)$ and l. In other words, any column in the right-hand region is a linear combination of two columns in the left-hand region, of S. It is obvious also from such a derivation that, even if the slope component is composed of any linear combination of two spectrum components, no matter what coefficients are used, the columns in the right-hand region are all linear combinations of the columns in the left-hand region. Therefore, the whole matrix S has a reduced rank, namely, only 15. In other words, S is singular, therefore, an eigen-problem cannot be applied to the whole matrix S[20].

In order to analyze the rank property of a variance-covariance matrix S^Δ of a large vector X^Δ composed of two vectors X^P and ΔX^P, by concatenating their components, a slightly different step from that used for X^P and X^L should be taken. Because the relation between the components in X^P and ΔX^P spans over time, the particular calculation form of E should be used. Let i denote the time index. Let a constant m denote the time span over which the 'time derivative' is defined (in our system $m = 4$). Then the relation between the components x^P of X^P and Δx^P of ΔX^P is

$$\Delta x_{ji}^P = x_{j,i+m}^P - x_{ji}^P, \qquad (1 \le j \le 15). \tag{A1.4}$$

[20]With some computer programs, however, still a set of 28 'eigen-vectors' can be found, but those higher-dimensional eigen-values are very small, only reflecting some round-up errors.

where j is the same for both x^P and Δx^P. The elements s^{Δ}_{jk} of the variance-covariance matrix S^{Δ} are calculated as

$$\begin{aligned} s^{\Delta}_{jk} &= \frac{1}{N}\sum_{i=1}^{N}(\Delta x_{ji}^{P} - \Delta\mu_j^P)(x_{ki}^{P} - \mu_k^P) \\ &= \frac{1}{N}\sum_{i=1}^{N}(x_{j,i+m}^{P} - \mu_{j,N+m}^{P})(x_{ki}^{P} - \mu_k^P) - \frac{1}{N}\sum_{i=1}^{N}(x_{ji}^{P} - \mu_{jm}^{P})(x_{ki}^{P} - \mu_k^P), \end{aligned} \qquad \text{(A1.5)}$$

$$(1 \le j,k \le 15).$$

where

$$\begin{aligned} \mu_{j,N+m}^{P} &= \frac{1}{N}\sum_{i=N+1}^{N+m} x_{ji}^{P}, \\ \mu_{jm}^{P} &= \frac{1}{N}\sum_{i=1}^{m} x_{ji}^{P}, \end{aligned} \qquad (1 \le j \le 15), \qquad \text{(A1.6)}$$

are both summed over only m items, at the beginning and the end of the whole sequence of length N, respectively. It can be argued that since $m \ll N$, the two μ's in A1.6 are much smaller than x, therefore they can be neglected from A1.5. Further, because the two factors in the first term of A1.5 concern *two* different time instances, it is impossible to represent this term with components at only *one* time instance (e.g. by shifting the index of the summation). Since s^{Δ}_{jk} are entries in the 'right-hand' region of S^{Δ} concerning the covariance between X^P and ΔX^P, this result indicates that the columns in the right-hand region are generally *not* linear combinations of the columns in the left-hand region. This violates the *sufficient* condition for a matrix to be of reduced rank, therefore, we can say that the matrix is not *guaranteed* to be singular.

Whether the matrix S^{Δ} is of full rank or not depends on the real data. Actually, since it is well known that the time correlation in the speech signal carries information, this correlation cannot be constant. Therefore, the correlation between X^P and ΔX^P, as

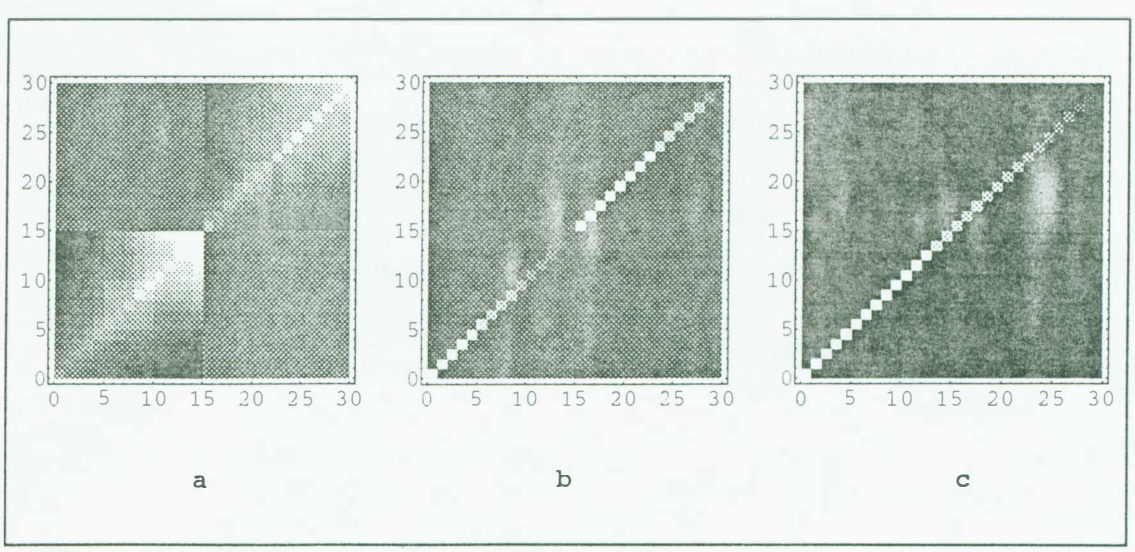

Figure A1.1. (a) Original, (b) separate PCA-transformed and (c) composed PCA-transformed S matrices. Indices 1-15 are for X^P, and 16-30 for ΔX^P.

calculated by summing the cross-correlation at the different time instances over a large N, will be small, if not zero, due to the randomness of the speech data, instead of having systematic relations with the correlation parameters within X^P. A real situation is plotted in Fig. A1.1(a), where the between-vector correlation is very small. A PCA transformation on the large composed vector is possible. Although the front-end manipulation between X^P and ΔX^P would not be performed in this study, we give a comparison between the matrices of the original (Fig. A1.1(a)), the separately transformed (Fig. A1.1(b)), and the composed-transformed (Fig. A1.1(c)) data vectors. The matrices in (b) and (c) are calculated using the formulae in Appendix 2. Note that the matrices in (b) and (c) of this figure have different meaning from those in Fig. 1.

Appendix 2. Diagonalization of S's with PCA and LDA

The transformation by either PCA or LDA from the original vectors $X = (x_1, x_2, ..., x_M)^T$ into $Y = (x_1, x_2, ..., x_M)^T$, where $M = 15$ for spectrum X^P and 13 for slope X^L, changes the variance-covariance matrix S. Obtaining the S of the Y from these vectors directly is straightforward. However, since the calculation of S involves addition and multiplication with a large number of vectors which have undergone transformation, this accumulation greatly decreases the numerical accuracy of S, let alone a heavy calculation. On the other hand, however, we can use the properties of these orthogonal transformations and the intrinsic relation between the X^P and X^L, to obtain some analytical relations and thus simplify the calculation with a higher accuracy. In this appendix, the effect of diagonalization by PCA and LDA will be analyzed, and at the same time, the calculation formulae for S of Y of the vector composed of Y^P and Y^L will be derived in terms of the original S and the transformation matrices.

As in the main text, we will use $V = (v_1, v_2, ..., v_M)$, where each v_i denotes an eigenvector, to represent the transformation matrix, and the superscripts P and L to refer to the operations on spectrum and the slope vectors, respectively. The items referring to PCA and LDA transformations will be indicated by subscripts π and ζ, respectively, whereas the original items before transformation will have no subscripts, except for the indication of individual elements. $\mu = E\{X\}$ is the sample mean, and the superscript T denotes the transpose. All the vectors are column vectors.

The variance-covariance matrix S of the full-dimensional large vectors concatenated from either original X^P and X^L or the transformed Y^P and Y^L can be divided into 4 blocks, namely S^P, S^L, S^{PL} and S^{LP}, for the variance-covariance matrices of P and of L vectors, and the covariance between P and L vectors, respectively. Then we have

$$\begin{aligned} S^P &= E\{[X^P - \mu^P][X^P - \mu^P]^T\}; \\ S^L &= E\{[X^L - \mu^L][X^L - \mu^L]^T\}. \end{aligned} \tag{A2.1}$$

A2.1 PCA transformation

The PCA-transformed P and L vectors are, respectively,

$$\begin{aligned} Y_\pi^P &= V_\pi^{PT} X^P; \\ Y_\pi^L &= V_\pi^{LT} X^L. \end{aligned} \tag{A2.2}$$

Then the S's for the transformed Y^P and Y^L are, using (A2.2) and the fact that the V's are constants with respect to the operation of sample mean E,

$$S_\pi^P = E\{[Y_\pi^P - \mu_{Y_\pi}{}^P][Y_\pi^P - \mu_{Y_\pi}{}^P]^T\} = V_\pi^{PT} S^P V_\pi^P;$$
$$S_\pi^L = E\{[Y_\pi^L - \mu_{Y_\pi}{}^L][Y_\pi^L - \mu_{Y_\pi}{}^L]^T\} = V_\pi^{LT} S^L V_\pi^L. \qquad \text{(A2.3)}$$

Therefore, when we calculate $S_\pi{}^P$ and $S_\pi{}^L$, we can use the original S^P and S^L and the eigenvectors $V_\pi{}^P$ and $V_\pi{}^L$, rather than going back to the data vectors Y^P and Y^L.

We can look at the eigen-equations $S^P V_\pi{}^P = V_\pi{}^P \Lambda_\pi{}^P$ and $S^L V_\pi{}^L = V_\pi{}^L \Lambda_\pi{}^L$, where the eigenvectors are found orthogonal and normalized: $V_\pi{}^P V_\pi{}^{PT} = V_\pi{}^{PT} V_\pi{}^P = I^P$ and $V_\pi{}^L V_\pi{}^{LT} = V_\pi{}^{LT} V_\pi{}^L = I^L$, where I's are identity matrices, respectively. Therefore, we have

$$S_\pi^P = \Lambda_\pi^P,$$
$$S_\pi^L = \Lambda_\pi^L. \qquad \text{(A2.4)}$$

This is to say, that the variance-covariance matrices of the transformed vectors within the P and L blocks are *completely* diagonalized, and the diagonal elements are the eigenvalues of S^P and S^L, respectively.

In order to get a calculation formula for the PL block (thus also the LP block, which is the transpose of the PL block due to the symmetry), we have to look at the individual elements $s_{ij}{}^P$ and $s_{ij}{}^L$ of the S^P and S^L, respectively. The covariance in the PL block is

$$S_\pi^{PL} = E\{[Y_\pi^P - \mu_{Y_\pi}{}^P][Y_\pi^L - \mu_{Y_\pi}{}^L]^T\} = V_\pi^{PT} E\{[X^P - \mu^P][X^L - \mu^L]^T\} V_\pi^L. \qquad \text{(A2.5)}$$

In the right-hand side of (A2.5), the first and the third terms are the eigenvector matrices for S^P and S^L, respectively, while the middle term is the covariances between the parameters from (the original) X^P and X^L, which will be our main concern. Note that the slope parameters are a simple combination of the spectrum parameters i.e.

$$x_i{}^L = x_{i+2}{}^P - x_i{}^P, \qquad (1 \le i \le 13). \qquad \text{(A2.6)}$$

Therefore the X^L parameters can be represented by the X^P parameters as follows. Firstly,

$$X^P = \begin{pmatrix} x_1{}^P \\ x_2{}^P \\ \cdots \\ x_{15}{}^P \end{pmatrix}, \qquad X^L = \begin{pmatrix} x_1{}^L \\ x_2{}^L \\ \cdots \\ x_{13}{}^L \end{pmatrix} = \begin{pmatrix} x_3{}^P - x_1{}^P \\ x_4{}^P - x_2{}^P \\ \cdots \\ x_{15}{}^P - x_{13}{}^P \end{pmatrix}, \qquad \text{(A2.7)}$$

and

$$\mu^P = \begin{pmatrix} \mu_1^P \\ \mu_2^P \\ \cdots \\ \mu_{15}^P \end{pmatrix}, \qquad \mu^L = \begin{pmatrix} \mu_1^L \\ \mu_2^L \\ \cdots \\ \mu_{13}^L \end{pmatrix} = \begin{pmatrix} \mu_3^P - \mu_1^P \\ \mu_4^P - \mu_2^P \\ \cdots \\ \mu_{15}^P - \mu_{13}^P \end{pmatrix}. \tag{A2.8}$$

Then the argument of the E in the middle term of (A2.5) can all be represented with the X^P parameters as

$$\begin{aligned} &(X^P - \mu^P)(X^L - \mu^L)^T \\ &= \begin{pmatrix} x_1^P - \mu_1^P \\ x_2^P - \mu_2^P \\ \cdots \\ x_{15}^P - \mu_{15}^P \end{pmatrix} \begin{pmatrix} (x_3^P - x_1^P) - (\mu_3^P - \mu_1^P) \\ (x_4^P - x_2^P) - (\mu_4^P - \mu_2^P) \\ \cdots \\ (x_{15}^P - x_{13}^P) - (\mu_{15}^P - \mu_{13}^P) \end{pmatrix}^T = \begin{pmatrix} x_1^P - \mu_1^P \\ x_2^P - \mu_2^P \\ \cdots \\ x_{15}^P - \mu_{15}^P \end{pmatrix} \begin{pmatrix} (x_3^P - \mu_3^P) - (x_1^P - \mu_1^P) \\ (x_4^P - \mu_4^P) - (x_2^P - \mu_2^P) \\ \cdots \\ (x_{15}^P - \mu_{15}^P) - (x_{13}^P - \mu_{13}^P) \end{pmatrix}^T . \end{aligned} \tag{A2.9}$$

Noting that the right-hand matrix can be written as the difference of two matrices, and using the calculation for the individual elements s_{ij}^P of the S^P,

$$s_{ij}^P = E\{[x_i^P - \mu_i^P][x_j^P - \mu_j^P]\}, \qquad (1 \le i, j \le 15), \tag{A2.10}$$

we can write the PL covariance as

$$E\{[X^P - \mu^P][X^L - \mu^L]^T\} = \begin{pmatrix} s_{13}^P & s_{14}^P & \cdots & s_{1,15}^P \\ s_{23}^P & s_{24}^P & \cdots & s_{2,15}^P \\ \cdots & \cdots & \cdots & \cdots \\ s_{15,3}^P & s_{15,4}^P & \cdots & s_{15,15}^P \end{pmatrix} - \begin{pmatrix} s_{11}^P & s_{12}^P & \cdots & s_{1,13}^P \\ s_{21}^P & s_{22}^P & \cdots & s_{2,13}^P \\ \cdots & \cdots & \cdots & \cdots \\ s_{15,1}^P & s_{15,2}^P & \cdots & s_{15,13}^P \end{pmatrix} = S_{3,15}^P - S_{1,13}^P. \tag{A2.11}$$

The two matrices $S^P{}_{3,15}$ and $S^P{}_{1,13}$ are, the parts which take the columns 3 through 15 and 1 through 13, of the original variance-covariance matrix S^P, respectively. By substituting them into (A2.5), the covariance between the transformed Y^P and Y^L is eventually represented by the variance-covariance matrix S^P of the original X^P and the two transformation matrices for X^P and X^L, respectively:

$$S_\pi^{PL} = V_\pi^{PT}(S_{3,15}^P - S_{1,13}^P)V_\pi^L. \tag{A2.12}$$

A2.2 LDA transformation

The LDA transformations V_ζ^P and V_ζ^L satisfy the eigen-equations $((W^P)^{-1}U^P)V_\zeta^P = V_\zeta^P\Lambda_\zeta^P$ and $((W^L)^{-1}U^L)V_\zeta^L = V_\zeta^L\Lambda_\zeta^L$, respectively. Then, similar to PCA, the LDA-transformed vectors Y^P and Y^L are, respectively,

$$\begin{aligned} Y_\zeta^P &= V_\zeta^{PT}X^P; \\ Y_\zeta^L &= V_\zeta^{LT}X^L. \end{aligned} \tag{A2.13}$$

The variance-covariance matrices of the transformed Y^P and Y^L (recall (A2.3)) are

$$\begin{aligned} S_\zeta^P &= V_\zeta^{PT}S^PV_\zeta^P; \\ S_\zeta^L &= V_\zeta^{LT}S^LV_\zeta^L. \end{aligned} \tag{A2.14}$$

Solving S^P and S^L out of the PCA eigen-equations, respectively, we have

$$\begin{aligned} S^P &= V_\pi^P\Lambda_\pi^PV_\pi^{PT}; \\ S^L &= V_\pi^L\Lambda_\pi^LV_\pi^{LT}. \end{aligned} \tag{A2.15}$$

Substituting (A2.15) into (A2.14), we obtain the variance-covariance matrices of the LDA-transformed Y^P and Y^L, represented by the diagonal eigenvalue matrices Λ_π of the PCA transformation:

$$\begin{aligned} S_\zeta^P &= V_\zeta^{PT}V_\pi^P\Lambda_\pi^PV_\pi^{PT}V_\zeta^P; \\ S_\zeta^L &= V_\zeta^{LT}V_\pi^L\Lambda_\pi^LV_\pi^{LT'}V_\zeta^L. \end{aligned} \tag{A2.16}$$

Since in general the eigenvectors obtained from PCA and LDA are not the same, the two transformations do not compensate completely. Therefore, although the middle terms Λ_π's are diagonal, after the two transformations V_π and V_ζ, the S_ζ^P and S_ζ^L are generally *not* diagonal.

The calculation of the covariance in the *PL* region is exactly the same as in PCA, only the π (in A2.12) should be replaced by ζ. In summary, the calculation formulae will be A2.3 (or A2.4), A2.12 and A2.14, respectively.

References

Alphen, P. van (1992): *HMM-Based Continuous Speech Recognition: Systematic Evaluation of Various System Components*, Ph.D. thesis, Univ. Amsterdam, the Netherlands.

Bourlard, H. (1992): "Continuous Speech Recognition: from Hidden Markov Models to Neural Networks", *Proceedings EUSIPCO-'92*, Brussels, published in: Vandewalle, J, Boite, R, Moonen, M. and Oosterlinck, A. (eds.): *Signal Processing VI: Theories and Applications*, Elsevier Science Publishers b.v., 63-70.

Cerf, P. le, Compennolle, D. van & Diest, M. van (1992): "Reduction Techniques for Frames and Frame Dimensions in Automatic Speech Recognition", *Proceedings EUSIPCO-'92*, Brussels, published in: Vandewalle, J, Boite, R, Moonen, M. and Oosterlinck, A. (eds.): *Signal Processing VI: Theories and Applications*, Elsevier Science Publishers b.v., 371-374.

Cooley, W. W. & Lohnes, P. R. (1971): *Multivariate Data Analysis*, John Wiley & Sons Inc. New York.

Deng, L. (1991): "The Semi-Relaxed Algorithms for Estimating Parameters of Hidden Markov Mod-

els", *Computer Speech and Language*, **5**: 231-236.
Ephraim, Y. & Rabiner, L. R. (1990): "On the Relations between Modeling Approaches for Speech Recognition", *IEEE Trans. Information Theory*, **36(2)**: 372-380.
Gersho, A. & Gray, R. M. (1992): *Vector Quantisation and Signal Compression*, Kluwer Academic Publ. Boston.
Huang, X.D., Ariki, Y. & Jack, M. A. (1990): *Hidden Markov Models for Speech Recognition*, Edinburgh Univ. Press.
Johnson, R. A. & Wichern, D. A. (1988): *Applied Multivariate Statistical Analysis*, Prentice-Hall International Edition.
Juang, B.-H. & Rabiner, L. R. (1992): "Issues in Using Hidden Markov Models for Speech Recognition", in: Furui, S. and Sondhi, M. M. (eds.), *Advances in Speech Signal Processing*, Marcel Dekker, Inc. New York, 509-553.
Lee, K.-F. (1989): *Automatic Speech Recognition: the Development of SPHINX system*, Kluwer Academic Publ. Boston.
Lee, K.-F., Hou, H.-W. & Reddy, R. (1990): "An Overview of the SPHINX Speech Recognition System", *IEEE Trans. ASSP*. **38(1)**: 35-45.
Ljolje, A., Ephraim, Y. & Rabiner, L. R. (1990): "Estimation of Hidden Markov Model Parameters by Minimizing Empirical Error Rate", *Proceedings IEEE ICASSP*, 709-712.
Okamoto, M. (1969): "Optimality of Principal Components", in: Krishnaiah, P. R. (ed.) *Multivariate Analysis-II*, Academic Press, New York, 673-685.
Pols, L. C. W. (1977): *Spectral Analysis and Identification of Dutch Vowels in Monosyllable Words*, Ph.D. thesis, Free University Amsterdam, the Netherlands.
Pols, L. C. W. (1987): "Distance Measures: Physical and Perceptual Aspects", *Proceedings of the Institute of Phonetic Sciences*, Amsterdam, the Netherlands, **11**: 59-66.