INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 3, 263–276, 2000

A Comparative Study of Several Feature Transformation and Learning Methods for Phoneme Classification

ANDRÁS KOCSOR, LÁSZLÓ TÓTH, ANDRÁS KUBA, KORNÉL KOVÁCS, MÁRK JELASITY, TIBOR GYIMÓTHY AND JÁNOS CSIRIK
Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and of the University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1, Hungary
kocsor@inf.u-szeged.hu

Received ; Accepted

Editors: Gábor Olaszy and Daryle Gardner-Bonneu

Abstract. This paper examines the applicability of some learning techniques for speech recognition, more precisely, for the classification of phonemes represented by a particular segment model. The methods compared were the IB1 algorithm (TiMBL), ID3 tree learning (C4.5), oblique tree learning (OC1), artificial neural nets (ANN), and Gaussian mixture modeling (GMM), and, as a reference, a hidden Markov model (HMM) recognizer was also trained on the same corpus. Before feeding them into the learners, the segmental features were additionally transformed using either linear discriminant analysis (LDA), principal component analysis (PCA), or independent component analysis (ICA). Each learner was tested with each transformation in order to find the best combination. Furthermore, we experimented with several feature sets, such as filter-bank energies, mel-frequency cepstral coefficients (MFCC), and gravity centers. We found LDA helped all the learners, in several cases quite considerably. PCA was beneficial only for some of the algorithms, and ICA improved the results quite rarely and was bad for certain learning methods. From the learning viewpoint, ANN was the most effective and attained the same results independently of the transformation applied. GMM behaved worse, which shows the advantages of discriminative over generative learning. TiMBL produced reasonable results; C4.5 and OC1 could not compete, no matter what transformation was tried.

Keywords: speech and phoneme recognition, feature space transformations, discriminative and generative learning

1. Introduction

Automatic speech recognition is a special pattern classification problem in which one of the pattern's dimensions is time. Speech signals show very specific dynamic variations along this axis and thus require dedicated recognition techniques. One approach is to segment the speech signal into its supposed building blocks (e.g., phonemes), recognize these separately, and then combine the recognition scores for the whole signal. Because of the difficulties of segmentation, however, HMM, in which utterances are processed in small uniform chunks (called frames), became the dominant technology instead, and their time variability is handled by a neat probabilistic structure.

Lately, HMM has received a lot of criticism over its time modeling capabilities, and there have been efforts toward generalizations that work with phonetic segments rather than with frames. One goal of this study is to evaluate several possible learning methods for a particular phoneme model of such a probabilistic segmental recognizer. A special reason for this was that

we wanted to test certain classification techniques that are well-known in the machine learning community but are rarely seen in speech literature.

The other aim of this article is to study the effects of several feature vector transformation methods on the learning algorithms. These techniques, namely PCA, LDA, and ICA, are well-known among speech researchers but are quite rarely used.

The statistical learning methods employed in classification problems are called either discriminative or generative, depending on what they model. Discriminative models describe the common feature space of all the classes and focus on discriminating one class from another. They do this either by finding proper parameters for a set of separating surfaces of a given type (parametric modeling) or by representing the classes with elements and distance metrics (nonparametric modeling). In our paper C4.5, OC1, TiMBL, and ANN represent the class of discriminative learners.

According to Bayes' law, the conditional probability $P(C \mid \mathbf{x})$ of a class C for a vector $\mathbf{x}$ can be obtained from the formula

$$P(C \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C)P(C)}{P(\mathbf{x})}. \tag{1}$$

Thus, instead of modeling $P(C \mid \mathbf{x})$ directly as discriminative models do, another possibility is to estimate the class-conditional probabilities $P(\mathbf{x} \mid C)$ for each class separately. This is the so-called generative modeling approach. Although it may seem a disadvantage that a priori probabilities $P(C)$ also have to be estimated, this decomposition is actually very useful in speech recognition because "acoustic" and "language" models can then be handled separately. From the techniques studied in our article, HMM and GMM belong to the class of generative learners.

The structure of this article is as follows. First, we briefly describe the acoustic features that were used in the experiments. Then we examine the feature transformation methods and afterward the learning algorithms we applied. In the final part of this article, we discuss aspects of the experiments, especially the corpus, the test cases, and, of course, the results. We close with a few ideas about how the phoneme classifiers might be used as a part of a speech recognition system.

2. Acoustic Features

In the following we describe the feature extraction techniques that were used in our tests. For each test, a certain subset of these features was chosen. The only exception was the HMM recognizer, which used its own features (for details see Sections 5.1.1 and 6.3).

2.1. Critical Band Log-Energies

Before feature extraction, the energy of each word was normalized. After this the signals were processed in 512-point frames (23.2 milliseconds), in which the frames overlapped by a factor of 0.75. Fast Fourier transform was applied on the frames. After that, critical band log-energies (CBLE) were approximated by the use of triangular weighting functions. Twenty-four such filters were used to cover the frequency range from 0 to 11,025 Hz, the bandwith of each filter being 1 bark. The energy values were then measured on a logarithmic scale.

2.2. Mel-Frequency Cepstral Coefficients

In order to incorporate the most common preprocessing method, that is, MFCC into our features, we made additional tests after taking the discrete cosine transform (DCT) of the CBLEs calculated previously. The test used the first 16 coefficients (including the zeroth one). Noticed that because the spectrum has already been smoothed by the critical band filters, the calculation of the cepstrum does not fulfil its original task of removing the effect of pitch. Instead, its supposed benefit is the decorrelation of features. In fact, it can be proved that the DCT approximates the PCA quite closely for most signals (Akansu and Haddad, 1992), so it is worth comparing the classification results for MFCC with CBLEs plus PCA.

2.3. Formant Band Gravity Centers

In addition to the preceding experiments, we also wanted to do some with more phonetically based features, such as formants. However, because we had no reliable formant tracker, we instead used formant band gravity centers (FBGC) as a crude approximation for formants (Albesano et al., 1999). The gravity centers were calculated from the power spectrum in the four frequency bands [200 Hz, 1400 Hz], [1000 Hz, 3000 Hz], [2500 Hz, 4000 Hz], and [3000 Hz, 11025 Hz]. The formula for the gravity center $\mathcal{G}(a, b)$ of a band [a,b] is

$$\mathcal{G}(a, b) = \frac{\int_a^b f\, S(f)\, df}{\int_a^b S(f)\, df}, \tag{2}$$

where $S(f)$ denotes the power spectrum.

Because the gravity centers can give misleading values at parts that have no clear formant structure (e.g., silence), the "spreading ratio" $\mathcal{D}(a, b)$ of the gravity center was employed as a kind of measure for the strength or reliability of the formant. We defined this as the ratio of the deviance of the spectrum in intensity and frequency. That is,

$$\mathcal{D}(a,b) = \frac{\mathcal{D}_i(a,b)}{\mathcal{D}_f(a,b)} = \frac{\sqrt{\frac{1}{b-a}\int_a^b S^2(f)\,df - \left(\frac{1}{b-a}\int_a^b S(f)\,df\right)^2}}{\sqrt{\frac{\int_a^b f^2 S(f)\,df}{\int_a^b S(f)\,df} - \mathcal{G}^2(a,b)}}. \tag{3}$$

Thus the four gravity centers and four spreading ratios gave eight additional features.

3. Segmental Features

Although HMM is the most widely used technology for speech recognition, it is not without its critics when it comes to its phoneme modeling abilities. The most important criticisms against HMM as a phoneme model are that (Ostendorf, 1996b)

- its duration modeling abilities are poor
- because it works with uniform-sized frames, it does not permit the incorporation of long-term (segmental) measurements
- it assumes that the frames that belong to a given state are independent.

One possible way of overcoming these limitations is to work within a segmental framework like that of Glass et al. (1996) or Ostendorf et al. (1996a). Because working with variable-sized segments instead of frames introduces many new problems, switching to such an approach requires very strong justification. One convincing proof would be if we found phoneme models that not only were intuitively more appealing but also led to better classification results. Recently, great efforts have been made to find good segmental phoneme models (Fukada et al., 1997; Ostendorf, 1996b; Gales and Young, 1993; Gish and Ng, 1993).

Compared with these sophisticated techniques we followed a very simple procedure, the idea having been taken from Halberstadt (1998). For each feature we took the average of the frame-based measurements for the first quarter, the central part, and the last quarter of the phoneme. In this way we obtained $3n$ features for each phoneme, where n is the number of the features for one frame. Our reasons for modeling the segments this simple way were twofold. On the one hand, we needed to describe each phoneme with the same number of features; otherwise, the discriminative learners could not have been applied at all. On the other hand, we wanted to use features that were very closely related to the original spectrum and learn what the transformations being studied would produce from them.

In HMM the duration of the phonemes is modeled only implicitly: the usual three-state left-to-right model remains in a state with an exponentially decaying probability, which is quite a poor approximation of how the length of phonemes (or rather, phoneme thirds) is distributed in the real world. In segmental models phonemic duration can be modeled explicitly, which we think is especially important in languages such as Hungarian, in which most of the phonemes have a "short" and a "long" version (that is, duration has a distinctive role). Our experiments showed that adding duration to the segmental feature set indeed increased classification accuracy, so duration was used in all of our experiments. However, our statistical measurements pointed out that the duration of the phonemes has a huge scatter. This means that speaking rate normalization would be very beneficial for recognition.

4. Feature Vector Transformations

Before executing a learning algorithm, additional vector space transformations may be applied on the extracted features. The role of these methods is twofold. First, they may improve classification performance, and second they may also reduce the dimensionality of the data. This is because these techniques search for a transformation that emphasizes more important features and suppresses or even eliminates less desirable ones.

Although some nonlinear extensions of these methods have been presented in recent years, we restrict our investigations to their linear versions. From these linear PCA, ICA, and LDA will be dealt with in this article.

Without loss of generality we will assume that the original data set lies in $\mathbb{R}^n$, and that we have l elements $\mathbf{x_1}, \ldots, \mathbf{x_l}$ in the training set and t elements $\mathbf{y_1}, \ldots, \mathbf{y_t}$ in the testing set. After applying a feature vector transformation method, the new data set lies in $\mathbb{R}^m$ ($m \leq n$), the transformed training and testing vectors being denoted by $\mathbf{x}'_1, \ldots, \mathbf{x}'_l$ and $\mathbf{y}'_1, \ldots, \mathbf{y}'_t$, respectively.

We search for an optimal (in some cases orthogonal) linear transformation $\mathbb{R}^n \rightarrow \mathbb{R}^m$ of the form $\mathbf{x}'_\mathbf{i} = \mathbf{A}^\top \mathbf{x}_\mathbf{i}$, $i \in \{1, \ldots, l\}$, noting that the precise definition of optimality can vary from method to method.

All transformation methods mentioned previously use the following general unified strategy for obtaining the $n \times m$ matrix $\mathbf{A}$:

- The column vectors of $\mathbf{A}$, denoted by $\mathbf{a_1}, \ldots, \mathbf{a_m}$ (determined successively in some methods) are supposed to be normalized.
- The algorithm uses a differentiable method-dependent objective function $\tau(): \mathbb{R}^n \rightarrow \mathbb{R}$ for selecting optimal linear directions (orthogonality is also assumed in some methods).

Actually, with the linear transformation $\mathbf{A}$ we select a new (possibly orthogonal) basis for representing the data. To obtain a particular basis we first choose a real-valued objective function $\tau()$ that serves as a measure for selecting the proper directions. A feature vector transformation method searches for m optimal directions. One way of obtaining these is to look for unit vectors that form the stationary points of $\tau()$. Intuitively, if larger values of $\tau()$ indicate better directions and the chosen directions needs to be independent in some ways, then choosing stationary points that have large values is a reasonable strategy.

In the following subsections we describe the PCA, LDA, and ICA algorithms one by one. In the literature there are several different approaches for discussing the addressed methods (Battle et al., 1998; Comon, 1994; Fukunaga, 1989; Hyvärinen, 1997, 1998; Tipping and Bishop, 1997). Our primary goal here was to describe the principles of each in a broad and comparative way. That is why we chose to deviate slightly from the way these transformations are usually presented.

4.1. Linear Principal Component Analysis

PCA is a ubiquitous technique for data analysis and dimension reduction. Normally in PCA,

$$\tau(\mathbf{a}) = \frac{\mathbf{a}^\top \mathbf{C} \mathbf{a}}{\mathbf{a}^\top \mathbf{a}}, \tag{4}$$

where $\mathbf{C}$ is the sample covariance matrix

$$\mathbf{C} = \frac{1}{l} \sum_{i=1}^{l} (\mathbf{x_i} - \mu)(\mathbf{x_i} - \mu)^\top, \quad \mu = \frac{1}{l} \sum_{i=1}^{l} \mathbf{x_i}. \tag{5}$$

Practically speaking, Eq. (4) defines $\tau(\mathbf{a})$ as the variance of the $\{\mathbf{x_1}, \ldots, \mathbf{x_l}\}$ n-dimensional point-set projected onto the $\mathbf{a}$ vector. Therefore, this method prefers directions having a large variance. It can be shown that stationary points of Eq. (4) correspond to the right eigenvectors of the sample covariance matrix $\mathbf{C}$, where the eigenvalues form the corresponding optimum values. If we assume that the eigenpairs of $\mathbf{C}$ are $(\mathbf{c_1}, \lambda_1), \ldots, (\mathbf{c_n}, \lambda_n)$ and $\lambda_1 \geq \cdots \geq \lambda_n$, then the transformation matrix $\mathbf{A}$ will be $[\mathbf{c_1}, \ldots, \mathbf{c_m}]$, in other words, the eigenvectors with the largest m eigenvalues. Because the sample covariance matrix $\mathbf{C}$ is symmetrical positive semidefinite, the eigenvectors are orthogonal and the corresponding real eigenvalues are nonnegative. After this orthogonal linear transformation the dimensionality of the data will be m, and the sample $\mathbf{x}'_i = \mathbf{A}^\top \mathbf{x_i}$, $i \in \{1, \ldots, l\}$ represented in the new orthogonal basis will be uncorrelated; in other words, its covariance matrix $\mathbf{C}'$ is diagonal. The diagonal elements of $\mathbf{C}'$ are the m dominant eigenvalues of $\mathbf{C}$.

In our experiments, m (the dimensionality of the transformed space) was chosen to be the smallest integer for which

$$\frac{\lambda_1 + \cdots + \lambda_m}{\lambda_1 + \cdots + \lambda_n} > 0.99 \tag{6}$$

holds. There are many other alternatives, however, for finding a reasonable m. Because PCA behaves very sensitively when the magnitude of the components in the feature vector are significantly different, the data were first standardized, where the mean vector of the training data became the zero vector and the deviance of each component became 1.

4.2. Linear Discriminant Analysis

The goal of LDA is to find a new (not necessarily orthogonal) basis for the data that provides the optimal separation between groups of points (classes). The class label of each point is supposed to be known beforehand. Let us assume that we have k classes and an indicator function $f() : \{1, \ldots, l\} \rightarrow \{1, \ldots, k\}$, where $f(i)$ gives the class label of the point $\mathbf{x_i}$. Let l_j ($j \in \{1, \ldots, k\}$, $l = l_1 + \cdots + l_k$) denote the number of vectors associated with label j in the data. The function $\tau(\mathbf{a})$ is similar to that employed in PCA:

$$\tau(\mathbf{a}) = \frac{\mathbf{a}^\top \mathbf{B} \mathbf{a}}{\mathbf{a}^\top \mathbf{W} \mathbf{a}}, \tag{7}$$

where $\mathbf{W}$ is the within-class scatter matrix, and $\mathbf{B}$ is the between-class scatter matrix. Here the within-class scatter matrix $\mathbf{W}$ shows the weighted average scatter of the covariance matrices $\mathbf{C_j}$ of the sample vectors having label j:

$$\mathbf{W} = \sum_{j=1}^{k} \frac{l_j}{l} \mathbf{C_j}, \tag{8}$$

$$\mathbf{C_j} = \frac{1}{l_j} \sum_{f(i)=j} (\mathbf{x_i} - \mu_\mathbf{j})(\mathbf{x_i} - \mu_\mathbf{j})^\top, \quad \mu_\mathbf{j} = \frac{1}{l_j} \sum_{f(i)=j} \mathbf{x_i} \tag{9}$$

and the between-class scatter matrix $\mathbf{B}$ represents the scatter of the class mean vectors, $\mu_\mathbf{j}$ around the overall mean vector μ:

$$\mathbf{B} = \sum_{j=1}^{k} \frac{l_j}{l} (\mu_\mathbf{j} - \mu)(\mu_\mathbf{j} - \mu)^\top. \tag{10}$$

The value of $\tau(\mathbf{a})$ is large when its nominator is large and its denominator is small. Therefore the within-class averages of the sample projected onto $\mathbf{a}$ are far from each other, and the variance is small in each of the classes. The larger the value of $\tau(\mathbf{a})$ is, the farther the classes are spaced out and the smaller their spreads are.

Much like in PCA, it can be shown that stationary points of (7) correspond to the right eigenvectors of $\mathbf{W}^{-1}\mathbf{B}$, where the eigenvalues form the corresponding optimal values. As in PCA, we again select the m eigenvectors with the greatest real eigenvalues. Since $\mathbf{W}^{-1}\mathbf{B}$ is not necessarily symmetrical, the number of the real eigenvalues can be less than n. Moreover, the corresponding eigenvectors will not necessarily be orthogonal.

In our tests with LDA, the eigenvectors belonging to the dominant eigenvalues of $\mathbf{W}^{-1}\mathbf{B}$ were chosen as basis vectors for the transformed space, but only $k-1$ of them at most. Although there exist statistical tests for finding the optimal number of dimensions for the new feature space, we followed this simple method during our investigations. We should point out, however, that the ideal circumstances for LDA, those needed to use its full power, are the following:

- All classes have an identical covariance matrix $\mathbf{C_j}$
- Each class can be represented by a single Gaussian distribution

4.3. Independent Component Analysis

ICA (Comon, 1994) is a useful feature extraction technique, originally developed in connection with blind source separation. The goal of ICA is to find directions along which the distribution of the sample set is the least Gaussian. The reason for this is that along these directions the data are supposedly easier to classify. Several measures can be used to assess whether distribution is non-Gaussian. We always choose from those ones which are nonnegative and give zero for the Gaussian distribution. A useful measure of being non-Gaussian is negentropy, but obtaining this quantity via its definition is computationally very difficult. Fortunately, some simpler easily computable approximations exist for the negentropy of a variable y with zero mean and unit variance, for example,

$$\mathbf{J}(y) \approx (E[G(y)] - E[G(\nu)])^2, \tag{11}$$

where $G(): \mathbb{R} \to \mathbb{R}$ is an appropriate doubly differentiable contrast function, $E()$ denotes the expected value, and ν is a standardized Gaussian variable. Three conventionally used contrast functions are G_1, G_2, and G_3:

$$\begin{aligned} G_1(y) &= y^4 \\ G_2(y) &= \log(\cosh(y)) \\ G_3(y) &= -\exp\left(-\frac{1}{2}y^2\right) \end{aligned} \tag{12}$$

It is worth noting that in Eq. (11) $E(G(\nu))$ is a constant, its value depending on the contrast function G. For instance, in the case of $G_1()$ its value is 3.

Hyvärinen (1997, 1998) proposed a fast iterative algorithm called FastICA that uses these contrast functions. This method defines the functional $\tau()$ used for the selection of the base vectors of the transformed space by replacing y with $\mathbf{a}^\top\mathbf{x}$ in the negentropy functions earlier:

$$\tau_G(\mathbf{a}) = (E(G(\mathbf{a}^\top\mathbf{x})) - E(G(\nu)))^2. \tag{13}$$

Before running FastICA, however, some preprocessing steps need to be performed.

Centering. An essential step is to shift the original sample set $\mathbf{x_1}, \ldots, \mathbf{x_l}$ with its mean μ to obtain a set

$\tilde{\mathbf{x}}_\mathbf{1}, \ldots, \tilde{\mathbf{x}}_\mathbf{l}$, with a mean of $\mathbf{0}$:

$$\tilde{\mathbf{x}}_\mathbf{1} = \mathbf{x}_\mathbf{1} - \mu, \ldots, \tilde{\mathbf{x}}_\mathbf{l} = \mathbf{x}_\mathbf{l} - \mu \tag{14}$$

Whitening. The goal of this step is to transform the $\tilde{\mathbf{x}}_\mathbf{1}, \ldots, \tilde{\mathbf{x}}_\mathbf{l}$ samples via an orthogonal transformation $\mathbf{Q}$ into a space where the covariance matrix

$$\hat{\mathbf{C}} = \frac{1}{l} \sum_{i=1}^{l} \hat{\mathbf{x}}_\mathbf{i} \hat{\mathbf{x}}_\mathbf{i}^\top \tag{15}$$

of the points $\hat{\mathbf{x}}_\mathbf{1} = \mathbf{Q}\tilde{\mathbf{x}}_\mathbf{1}, \ldots, \hat{\mathbf{x}}_\mathbf{l} = \mathbf{Q}\tilde{\mathbf{x}}_\mathbf{l}$ is the unit matrix.

With the PCA discussed earlier we can transform the covariance matrix into a diagonal form, the elements in the diagonal being the eigenvalues of the original covariance matrix. Thus, it only remains to transform each diagonal element to 1. This can be done by dividing the normalized eigenvectors of the transformation matrix by the square root of the corresponding eigenvalue.

Consequently, the whitening procedure can be computed this way:

$$\mathbf{Q} := [\tilde{\mathbf{c}}_\mathbf{1}, \ldots, \tilde{\mathbf{c}}_\mathbf{n}]^\top \begin{bmatrix} \frac{1}{\sqrt{\tilde{\lambda}_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{\tilde{\lambda}_2}} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \frac{1}{\sqrt{\tilde{\lambda}_n}} \end{bmatrix}, \tag{16}$$

where the eigenpairs of the matrix

$$\tilde{\mathbf{C}} = \frac{1}{l} \sum_{i=1}^{l} \tilde{\mathbf{x}}_\mathbf{i} \tilde{\mathbf{x}}_\mathbf{i}^\top \tag{17}$$

are $(\tilde{\mathbf{c}}_\mathbf{1}, \tilde{\lambda}_1), \ldots, (\tilde{\mathbf{c}}_\mathbf{n}, \tilde{\lambda}_n)$.

Of course, as in the case of PCA, we can do a dimension reduction as well if we use only the first m dominant eigenvalues and corresponding eigenvectors for the construction of $\mathbf{Q}$.

After centering and whitening the following statements hold:

- For any normalized $\mathbf{a}$ the mean of $\mathbf{a}^\top \hat{\mathbf{x}}_\mathbf{1}, \ldots, \mathbf{a}^\top \hat{\mathbf{x}}_\mathbf{l}$ is 0, and its variance is 1. In fact we need this because Eq. (11) requires that y has a 0 mean and variance of 1, and so because of the substitution $y = \mathbf{a}^\top \hat{\mathbf{x}}$, $\mathbf{a}^\top \hat{\mathbf{x}}$ must also have this property.
- For any matrix $\mathbf{R}$ the covariance matrix $\hat{\mathbf{C}}_\mathbf{R}$ of the transformed points $\mathbf{R}\hat{\mathbf{x}}_\mathbf{1}, \ldots, \mathbf{R}\hat{\mathbf{x}}_\mathbf{l}$ remains the unit matrix if and only if $\mathbf{R}$ is orthogonal, since

$$\begin{aligned} \hat{\mathbf{C}}_\mathbf{R} &= \frac{1}{l} \sum \mathbf{R}\hat{\mathbf{x}}_\mathbf{1} (\mathbf{R}\hat{\mathbf{x}}_\mathbf{1})^\top \\ &= \mathbf{R}\left(\frac{1}{l} \sum \hat{\mathbf{x}}_\mathbf{1} \hat{\mathbf{x}}_\mathbf{1}^\top \right) \mathbf{R}^\top = \mathbf{R}\mathbf{I}\mathbf{R}^\top = \mathbf{R}\mathbf{R}^\top \end{aligned} \tag{18}$$

Actually, FastICA is an approximate Newton iteration method that finds such an orthogonal basis for the centered and whitened data, where the values of the measure for being non-Gaussian, $\tau_G()$, for the base vectors are large. Note that because the data remain whitened after an orthogonal transformation, ICA can also be considered as an extension of PCA.

In the experiments we applied a dimension reduction during the whitening procedure. The value of m was chosen as in PCA, and for the contrast function we used G_1.

4.4. Summary

Stated briefly, the most important properties of the techniques just discussed are that.

PCA concentrates on those independent directions with the largest variances.

LDA prefers those directions along which the class centers are far away and the average variance of the classes is small.

ICA besides keeping the directions independent chooses ones along which the measure of being non-Gaussian is large.

Figure 1 demonstrates the effects of PCA, LDA, and ICA on a two-dimensional (artificially generated) set of points consisting of three classes.

Naturally when we applied a certain transformation on the training set before learning, we applied the same transformation on the test data during testing.

5. Learning Methods

The following sections briefly present the generative and discriminative learning techniques applied in the experiments.

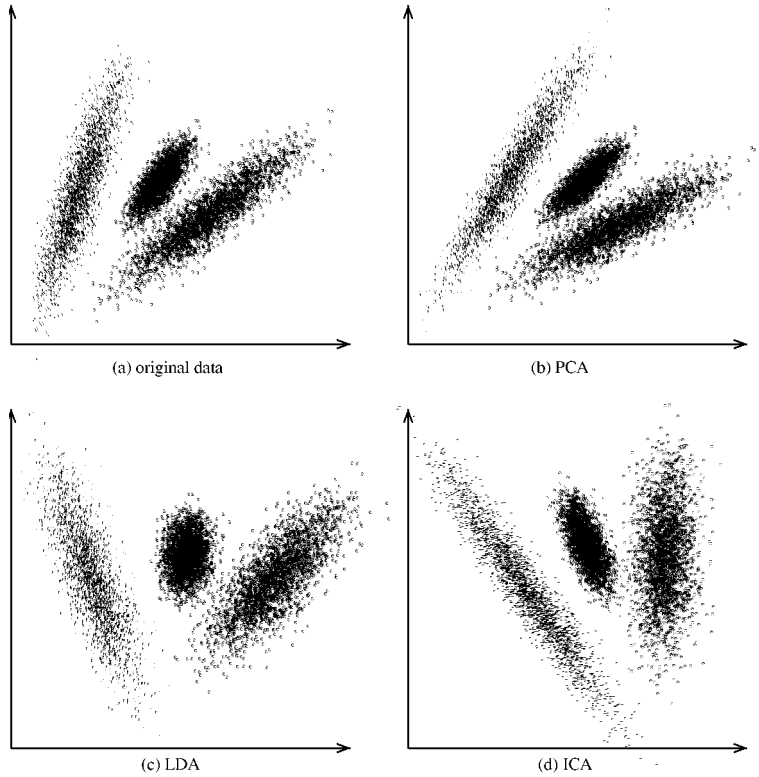

Figure 1. The original data and their transformations.

5.1. Applied Generative Learners

5.1.1. Hidden Markov Model. HMM is currently the dominant technology in speech recognition (Rabiner and Juang, 1993). This is why in the tests the HMM was trained on its "standard" features and not on those used in all the other experiments. The intention behind this was to have a reference point for the current state-of-the-art technology by which to judge things.

The HMMs for the experiments were trained using the FlexiVoice speech engine (Szarvas et al., 2000). The system used a feature vector of 34 components that consisted of 16 Linear Prediction Coding-derived cepstrum coefficients plus the frame energy and the first derivatives of these. The frame size was 30 milliseconds, whereas the stepsize was 10 milliseconds.

One model was assigned to each of the phonemes; that is, 28, 11, and 5 models were trained for the groupings *grp1*, *grp2*, and *grp3*, respectively (see Section 6.1). The phoneme models were of the three-state strictly left-to-right type; that is, each state had one self-transition and one transition to the next state. In each case, the observations were modeled using a mixture of four Gaussian components with diagonal covariance matrices. The models were trained using the Viterbi training algorithm.

5.1.2. Gaussian Mixture Model. GMM (Alder, 1994; Duda and Hart, 1973) assumes that the class-conditional probability distribution $p(\mathbf{x} \mid C)$ can be well-approximated by a distribution of the form

$$f(\mathbf{x}) = \sum_{i=1}^{k} c_k \mathcal{N}(\mathbf{x}, \mu_i, \mathbf{C_i}), \tag{19}$$

where $\mathcal{N}(\mathbf{x}, \mu_i, \mathbf{C_i})$ denotes the multidimensional normal distribution with mean μ_i and covariance matrix $\mathbf{C_i}$, k is the number of mixtures, and c_i are nonnegative weighting factors that sum to 1.

Unfortunately, there is no closed formula for the optimal parameters of the mixture model. Normally, the expectation-maximization (EM) algorithm (Alder, 1994; Dempster et al., 1977) is used to find proper parameters, but it guarantees only a locally optimal solution. This iterative technique is very sensitive to initial parameter values, so we used k-means clustering

(Rabiner and Juang, 1993) to find a good starting parameter set. Since *k*-means clustering again guaranteed finding only a local optimum, we ran it 15 times with random parameters and used the one with the highest log-likelihood to initialize the EM algorithm. After experimenting, the best values for the number of mixtures *k* was found to be 3, 7, and 20 for the three groupings used in the tests (see Section 6.1).

In all cases, the covariance matrices were forced to be diagonal, because training full matrices would have required much more training data (and computation time as well).

5.2. Applied Discriminative Learners

5.2.1. C4.5. C4.5 (Quinlan, 1993) is based on the well-known ID3 tree learning algorithm. It is able to learn predefined discrete classes from labeled examples. The result of the learning process is an axis-parallel decision tree. This means that during the training, the sample space is divided into subspaces by hyperplanes that are parallel to every axis but one. In this way, we get many *n*-dimensional rectangular regions that are labeled with class labels and organized in a hierarchical way that can be encoded into the tree. Because C4.5 considers attribute vectors as points in an *n*-dimensional space, using continuous sample attributes naturally makes sense.

For knowledge representation, C4.5 uses the "divide and conquer" technique, which means that regions are split during learning whenever they are insufficiently homogeneous. Splitting is done by axis-parallel hyperplanes, and thanks to this learning is very fast. Thus, the greatest advantage of the method is time complexity. In the worst case it is $O(dn^2)$, where d is the number of features and n is the number of samples. Unfortunately, the simple learning strategy in certain cases results in a huge number of regions that are needlessly split.

In general, three main cases should be mentioned in which C4.5 faces serious challenges:

- **Nonrectangular regions.** Even with reasonable feature space transformations (see Section 4), the phoneme classes are found in nonrectangular regions. To achieve the desired accuracy, C4.5 should increase the number of regions without limit, but some reduction is inevitable because of time and space considerations. No matter how carefully it is done, the reduction of tree size increases misclassification rate.
- **Poorly separated regions.** When the algorithm divides the search space, its goal is to create near-homogeneous regions. Also, early splits determine the direction toward which the whole procedure progresses. However if the samples are scattered or noisy (e.g., classes that are distributed randomly along certain axes) there is scarce guidance for the initial divisions. The overall result is a set of numerous regions (in other words, large decision trees) with a substantial number of misclassifications, although only a few well-placed regions would ensure the same accuracy.
- **Fragmented regions.** Irrelevant attributes force C4.5 to create too many regions as the examples become mixed along one or more axes. The consequence is errors similar to those we mentioned earlier.

5.2.2. OC1. The oblique classifier 1 (OC1) algorithm (Murthy et al., 1994) learns by creating *oblique* decision trees. The advantages and drawbacks are similar to those of C4.5, although in many cases OC1 produces better results. Having more freedom when splitting regions not surprisingly increases accuracy and decreases tree size. Then some nonrectangular regions become efficiently learnable.

OC1 chooses oblique splits through its perturbation algorithm performing random jumps. There are two corollaries of this. One is that it eliminates the problem of early splits being so crucial. The other is that it ensures an efficient algorithm, the time complexity being of $O(dn^2 \log n)$. This is only $\log n$ times more than that seen in C4.5.

5.2.3. TiMBL. TiMBL (Daelemans et al., 1999) is a memory-based learner that means a new example is evaluated by using up the previous examples stored in the memory. Because no rule or decision is made before the actual classification, this approach is called *lazy learning*. Typically, this kind of machine learning has a very short training time but the classification of new data takes rather a long time. The storing and processing of millions of examples can also be serious handicap. TiMBL is based on IB1, which is a version of the *k*-nearest neighbor algorithm with a special difference metric. TiMBL has many advanced tools to get the most out of the *k-NN* approach. Information theory is applied to both the difference metrics and the attribute weighting. Measuring the information gain ratio of different attributes yields valuable information about

useful and information-poor attributes as well. In this way, irrelevant features can be skipped over. Because of the underlying strategy, however, evidently redundant features run the risk of being overweighted, which may corrupt the classification by excessively dominating the metric.

TiMBL has a tree storing model as a solution for the time and space problems mentioned before. Training examples are stored in a treelike manner to decrease both computational time and memory required for data storage.

One of the key problems is that IB1 is designed to run on discrete features. Generally speaking, it can deal with numerical features only by making them discrete.

5.2.4. Artificial Neural Networks. ANN (Schürmann, 1996) now count among the conventional pattern recognition tools, so we will not describe them here. In the experiments, we used the most common feed-forward multilayer perceptron network with the backpropagation learning rule. The number of neurons in the hidden layer was set to be three times the number of features (this value was chosen empirically based on preliminary experiments). Training was stopped when, for the last 20 iterations, the decrease in the error between two consecutive iteration steps stayed below a certain threshold.

6. Experimental Results

The classification techniques were compared using a relatively small[1] corpus that consists of several speakers pronouncing Hungarian numbers. More precisely, 20 speakers were used for training and 6 for testing, and 52 utterances were recorded from each person. The ratio of male and female talkers was 50% : 50% in both the training and the testing sets. The recordings were made using a cheap commercial microphone in a reasonably quiet environment, at a sample rate of 22050 Hz. The whole corpus was manually segmented and labeled. Because the corpus contained only numbers, we had occurrences of only 32 phones, which is approximately two thirds of the Hungarian phoneme set. Because some of these labels represented only allophonic variations of the same phoneme, some labels were fused, and so we actually worked with a set of 28 labels. We made tests as well with two other groupings, where the labels were grouped into 11 and 5 classes, respectively, based on phonetic similarity. We had two good reasons for doing experiments with these gross phonetic classes. First, this way we could increase the number of training examples in each class and inspect the effects of this on the learning algorithms. Second, our speech recognition system has a first-pass, in which the segments are classified into gross phonetic categories only.

Hence we had three phonetic groupings, denoted by *grp1*, *grp2*, and *grp3* from this point on. With the first grouping, the number of occurrences of the different labels in the training set was between 40 and 599. This value was between 120 and 1158 for the second grouping and between 716 and 2158 for the third grouping.

6.1. Evaluation Method

The task of pattern classification is to map a given feature vector $\mathbf{x}$ to one of the classes C. The standard machine learning algorithms we tried (C4.5, OC1, TiMBL) do just this. That is, they return a label (i.e., a class) for each test vector. With these methods, the calculation of the recognition rate is quite straightforward.

The learning methods that model the a posteriori probabilities $P(C \mid \mathbf{x})$ return a probability value for each class C given a test vector $\mathbf{x}$. The so-called Bayes' decision rule states (Schürmann, 1996) that the optimal way of converting these values to a class label is to choose the class with the maximum a posteriori probability. We used this rule to calculate the classification error for these techniques.

Finally, some of the learning techniques (HMM and GMM) model the class-conditional probabilities $P(\mathbf{x} \mid C)$. From this, $P(C \mid \mathbf{x})$ can be obtained by employing Eq. (1). Thus, according to the Bayes' decision rule, we have to choose that class for which $P(\mathbf{x} \mid C)P(C)$ is maximal. ($P(\mathbf{x})$ is independent of C and so can be omitted.) Instead of doing this, we did not multiply by the factor $P(C)$ in the evaluation, because handling this probability traditionally belongs to the language model. Also, preliminary tests showed that multiplication with $P(C)$ led only to marginal improvements, clearly because the relative frequencies of the phonemes were quite well-balanced.

6.2. Experiments

The experiments were performed on five feature sets as described later. Because all sets contained duration, we do not mention it separately.

Set1 consisted of the MFCC coefficients, because these are the most commonly used features. To have the opportunity of studying the importance of the cosine transform, we also made tests on the filter bank energies themselves (*Set3*). By augmenting *Set1* and *Set3* with the gravity center features, we acquired two further sets, *Set2* and *Set4*. We had expected the addition of these phonetics-based features to lead to a slight improvement.

Last, the largest set (*Set5*) contained all the features, that is, filter bank energies, MFCC coefficients, and gravity centers. Our interest was in seeing whether the transformations could effectively select the important ones and in finding out whether mixing all the features would confuse the learning algorithms.

The same experiments were carried out on the three phoneme groupings *grp1*, *grp2*, *grp3*, all the learning methods being tested not just on each set but with each transformation technique. The only exception was HMM, which we used as a comparison with the current "standard" technique, so it used its own feature extraction method (see Section 5.1.1), but of course, with the same training and test corpus.

Tables 1 through 3 show the recognition accuracies for *grp1*, *grp2*, and *grp3*. The columns show the five feature sets, and the rows correspond to the applied transformation methods (including "none"). The numbers in the diagonal correspond to the recognition accuracies of TiMBL, C4.5, OC1, ANN, and GMM, respectively.

6.3. Discussion

When inspecting the results, the first thing one notices is that only the neural net could attain the performance of the HMM, all the other learners produce an error rate of 1.5 to 2 times bigger. We attribute this to the very simple segment model that used only the feature averages for the three preselected segment parts. Actually, the fact that the neural net could achieve the results of the HMM in spite of this drastic data reduction clearly showed the advantages of discriminative learning over generative types.

The drawbacks of our primitive segment modeling become even more apparent when we compare

Table 1. Recognition accuracies for *grp*1 (28 phonemes).

TiMBL C4.5 OC1 ANN GMM	**MFCC**	**MFCC + FBGC**	**CBLE**	**CBLE + FBGC**	**all**
	79.96	78.60	80.02	79.61	79.96
	61.20	66.00	65.20	68.00	65.90
none	65.13	70.51	69.74	69.50	69.89
	87.89	87.12	**90.84**	88.06	88.65
	79.20	75.18	74.14	74.23	74.35
	83.27	82.86	83.39	82.62	x
	65.50	70.10	66.10	77.10	x
LDA	72.93	72.10	71.69	72.64	x
	87.65	86.50	87.35	86.35	x
	86.76	85.82	85.82	85.99	x
	76.54	75.53	81.86	78.54	77.26
	50.80	48.90	60.60	63.10	56.10
PCA	60.52	60.34	69.21	66.96	65.54
	86.29	87.53	85.05	88.00	88.95
	83.57	82.51	85.17	81.62	83.39
	74.35	75.35	74.76	75.00	76.59
	53.30	43.20	54.70	34.20	37.60
ICA	54.43	49.53	54.91	44.92	46.93
	85.82	87.65	86.41	87.94	88.89
	80.44	76.77	79.55	76.60	79.49

For comparison, HMM scored 90.66% for this grouping. The maximum is typeset in bold.

Table 2. Recognition accuracies for *grp2* (11 phonetic categories).

		MFCC	**MFCC + FBGC**	**CBLE**	**CBLE + FBGC**	**all**
	TiMBL	87.05	85.83	88.29	87.76	87.82
	C4.5	77.00	78.70	77.70	75.70	76.90
	OC1	79.26	78.37	82.89	80.85	82.11
	ANN	88.89	89.83	90.96	91.49	91.43
	GMM	84.28	82.45	83.92	83.39	85.22
	TiMBL	84.81	85.57	82.86	85.52	x
	C4.5	80.40	84.10	81.90	82.60	x
LDA	OC1	83.92	83.92	83.33	84.16	x
	ANN	90.07	89.73	89.72	89.78	x
	GMM	89.01	88.87	88.53	88.42	x
	TiMBL	81.67	82.32	87.05	86.28	84.57
	C4.5	60.50	61.90	75.50	74.30	68.80
PCA	OC1	71.10	72.52	77.07	78.61	73.64
	ANN	87.65	89.13	**91.73**	91.25	90.84
	GMM	87.17	87.77	87.77	87.88	87.23
	TiMBL	80.73	81.20	82.38	82.26	82.15
	C4.5	53.20	53.20	66.30	48.00	49.30
ICA	OC1	66.19	63.53	68.85	62.47	57.45
	ANN	89.80	89.83	89.01	89.83	91.49
	GMM	85.58	82.74	85.82	83.22	83.63

For comparison, HMM scored 95.27% for this grouping. The maximum is typeset in bold.

the results of the HMM with those of the GMM, because HMM also uses Gaussian mixtures, but in a more refined way. This comparison obviously indicates that we must look for a more sophisticated segment representation later on.

As for the other algorithms, from the results it seems that C4.5 and OC1 are quite unsuitable for the task of phoneme recognition, at least in this form. Nevertheless, for *grp3* (the case of gross phonetic categories), they worked reasonably well, and their fast learning may be a justification for their use in this case.

Finally TiMBL (namely, the IB1 algorithm) worked quite well, and it appears that in the case of sparse training data it may work better than the parametric learners. Its drawback however, is its long evaluation time.

As regards the transformation techniques, first we should explain the reason for the xs in the "all" column of each table. In these cases, we could not apply LDA, because the matrix **W** was ill-conditioned because of the redundancy of the features. Hence, selecting features for LDA requires a certain degree of providence. Even so, LDA proved the most beneficial for all the learners.

PCA fared slightly worse than LDA did, clearly because it works in a class- independent way. It proved the most useful for the GMM, because of its decorrelating effects. C4.5 and OC1 behaved quite unpredictably, not being flexible enough to exploit the benefits of PCA. The neural net was able to attain virtually the same results, independent of the transformation applied. We should mention, however, that its convergence was much faster after LDA.

Last, ICA did not fulfil our expections because it improved on PCA only in a few cases (and in fact usually resulted in a much worse performance, especially in the case of C4.5 and OC1). To learn precisely why, we plan to make further investigations and also to look at some other versions of ICA.

On examining the features, the first thing to notice is that each of the learners performed the same or better with the filter bank energies (*Set3*) than they did with the corresponding cosine-transformed version (*Set1*). The only exception was the GMM, for which the

Table 3. Recognition accuracies for *grp3* (5 gross phonetic categories).

TiMBL / C4.5 / OC1 / ANN / GMM	**MFCC**	**MFCC + FBGC**	**CBLE**	**CBLE + FBGC**	**all**
	92.84	92.43	93.38	92.90	92.90
	85.90	86.10	87.50	83.50	87.20
none	86.23	87.06	88.48	87.94	88.01
	93.03	92.85	**95.69**	95.63	94.56
	91.63	89.87	91.90	91.49	91.73
	89.36	89.45	88.88	91.43	x
	89.00	91.30	88.50	91.80	x
LDA	90.43	90.96	89.13	91.90	x
	92.69	93.50	92.49	93.20	x
	91.43	92.32	91.37	92.97	x
	87.76	89.36	91.07	91.13	89.24
	71.10	73.90	85.40	91.90	79.00
PCA	81.97	83.16	88.12	90.96	88.42
	92.73	93.14	95.21	94.80	94.98
	92.55	92.08	92.08	93.38	93.79
	85.40	87.29	88.06	87.76	86.70
	68.90	63.80	60.90	62.20	61.30
ICA	74.53	77.84	79.14	75.71	74.23
	91.73	92.06	94.40	93.56	94.27
	90.43	89.01	90.25	90.54	89.30

For comparison, HMM scored 96.75% for this grouping. The maximum is typeset in bold.

decorrelating effect of the DCT proved beneficial—but the PCA was always much better.

Another thing we realized was that the gravity center features did not bring about any general improvement. The main exception was *grp*3, in which they helped in many cases. It seems that they are fairly useful in identifying gross phonetic categories but not consistent enough for classifying phonemes.

With the feature set "all," we found that in general it was neither better nor worse than the other sets. It seems that although using all the features together did not confuse the learners, it did not significantly help them either.

Finally, we mention that the previous conclusions were drawn from visual inspection of the results. For a rigorous justification of our impressions we also run significance tests. More precisely, paired two-sided t-tests were applied at the 5% significance level. These tests confirmed that filter bank energies, LDA, and ANN were the best feature set, feature space transformation method, and learning technique, respectively.

7. Beyond the Phoneme Level

Up to this point we have been concerned only with phonetic classification. That is, in our experiments we supposed that the start and end points of the phonemes were known and that the classifiers return a "hard label." However, when using the phonetic classifiers in a speech recognizer, the phonetic boundaries are not actually known, and the phoneme models are supposed to return probabilities. As regards the former problem, finding "the" correct phonetic segmentation of an utterance (if there is such thing at all) is still unsolved. So if we insist on working with segments, the best we can do is to try many possible segmentations, assign scores to them, and pick the best one according to some evaluation criterion. If the evaluation means mapping probabilities to the segmentations, then the best one means the most probable one—and we arrive at a probabilistic framework (Glass et al., 1996).

Other authors arrived at a segmental probabilistic structure in a different way. They have been trying to

find models that overcome the limitations of the HMM, yet keep its advantages. These authors aim at presenting a mathematically unified framework in which the HMM is just a special case (Ostendorf et al., 1996a). A survey of these probabilistic segmental recognizers is far beyond the scope of this article, but we suppose that our phonetic classifiers will be used in such a recognition system. For a description of our segmental recognizer see Kocsor et al. (1999a; 1999b).

The second problem we mentioned earlier is that the usual speech recognition frameworks (be they frame-based or segmental) expect the acoustic module to return probabilities and not hard labels. This clearly does not cause a problem for the methods that model the class-conditional or the a posteriori probabilities, but we have to do something in the case of those nonparametric models that return only a class label. Some of these algorithms can be modified to return probabilities, but some of them cannot. In the latter case, a possible solution is to train a set of classifiers on randomly chosen subsets of the training data and approximate the probability of a class based on the votes of these. Although this seems plausible, we are unaware of any such theoretical study in the machine learning literature.

8. Conclusion

As regards the features, we must conclude that in general filter-bank energies work just as well as MFCCs, so the computation time would be better spent on LDA or PCA than on DCT, because the former could give more significant improvement in recognition accuracy. The gravity center features helped only in the case of the discrimination of gross phonetic categories, so we now have to look for other techniques if we want to use phonetic information.

Using the transformations we found that LDA was the most convincing because it was beneficial for all learners. PCA was useful only for some of the learning algorithms; ICA improved the results only rarely and proved adverse with some of the methods. We plan to conduct more experiments with ICA, because it probably needs to be applied with more care.

With the learners, we found discriminative learning superior to the generative type. ANN was the most convincing disciminative technique and worked best on the filter-bank energies without a transformation. The second best technique was GMM, but this requires PCA (or LDA) for a good performance. TiMBL behaved reasonably well; C4.5 and OC1 could match the others only in the recognition of gross phonetic categories. None of the latter three techniques could really make use of the feature tranformations.

Finally, we have to mention that only the ANN reached the accuracy of the HMM. This means that we have to look for more sophisticated segment modeling for our segmental speech recognizer.

Acknowledgments

The HMM system used in our experiments (Szarvas et al., 2000) was trained and tested by Máté Szarvas at the Department of Telecommunications and Telematics, Technical University of Budapest. We greatly appreciate his help, which was indispensible in making this study complete. We also thank David Curley for scrutinizing and correcting the article from a linguistic point of view. This work was supported by the OTKA Grant No. T25721.

Note

1. Our reason to employ such a limited database was that we insisted on working with Hungarian, and there simply was no larger (segmented) corpus of Hungarian available at the time of this writing. However, one of our priorities for the future is to conduct additional testings on a larger database.

References

Akansu, A.N. and Haddad, R.A. (1992). *Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets*. New York: Academic Press.

Albesano, D., De Mori, R., Gemello, R., and Mana, F. (1999). A study on the effect of adding new dimensions to trajectories in the acoustic space. In *Proceedings of EuroSpeech'99*, Budapest, Hungary, pp. 1503–1506.

Alder, M.D. (1994). *Principles of Pattern Classification: Statistical, Neural Net and Syntactic Methods of Getting Robots to See and Hear*. http://ciips.ee.uwa.edu.au/~mike/PatRec.

Battle, E., Nadeu, C., and Fonollosa, J.A.R. (1998). Feature decorrelation methods in speech recognition. A comparative study. In *Proceedings of ICSLP'98*, Sydney, Australia.

Comon, P. (1994). Independent component analysis, A new concept? *Signal Processing*, *36*:287–314.

Daelemans, W., Zavrel, J., Sloot, K., and Bosch, A. (1999). TiMBL: Tilburg Memory Based Learner version 2.0 Reference Guide, ILK Technical Report - ILK 99-01, Computational Linguistics, Tilburg University, The Netherlands.

Dempster, A.P., Laird, N.M., and Rubin, D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. Ser. B*, *39*(1):1–38.

Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. New York: Wiley & Sons.

Fukada, T., Sagisaka, Y., and Paliwal, K.K. (1997). Model parameter estimation for mixture density polynomial segment models. In *Proceedings of ICASSP'97*, Munich, Germany, pp. 1403–1406.

Fukunaga, K. (1989). *Statistical Pattern Recognition*. New York, Academic Press.

Gales, M. and Young, S. (1993). Segmental hidden Markov models. In *Proceedings of EuroSpeech'93*. Berlin, Germany, pp. 1579–1582.

Gish, H. and Ng, K. (1993). A segmental speech model with applications to word spotting. In *Proceedings of ICASSP'93*. Minneapolis, MN, pp. 447–450.

Glass, J., Chang, J., and McCandless, M. (1996). A probabilistic framework for feature based speech recognition. In *Proceedings of ICSLP'96*. Philadelphia, PA, pp. 2277–2280.

Halberstadt, A.K. (1998). *Heterogeneous Measurements and Multiple Classifiers for Speech Recognition*, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, Ph.D. thesis.

Hyvärinen, A. (1997). A family of fixed-point algorithms for independent component analysis. In *Proceedings of ICASSP*, Munich, Germany.

Hyvärinen, A. (1998). New approximations of differential entropy for independent component analysis and projection pursuit. In *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, pp. 273–279.

Kocsor, A., Kuba, A., Jr., Toth, L., Jelasity, M., Gyimothy, T., and Csirik, J. (1999a). A segment-based statistical speech recognition system for isolated/continuous number recognition. In *Software Technology, Fenno-Ugric sumposium FUSST'99*, Sagadi, Estonia, pp. 201–209.

Kocsor, A., Kuba, A., Jr., and Toth, L. (1999b). An overview of the OASIS speech recognition project. In *Proceedings of ICAI'99*, Eger-Noszvaj, Hungary.

Murthy, S.K., Kasif, S., and Salzberg, S. (1994). A system for induction of oblique decision trees. *J. Artificial Intelligence Research*, *2*:1–32.

Ostendorf, M. (1996a). From HMMs to segment models: Stochastic modeling for CSR. In C.-H., Lee, F.K., Soong, and K.K., Paliwal (Eds.), *Automatic Speech and Speaker Recognition*, Advanced Topics, New York, Kluwer Academic, pp. 185–211.

Ostendorf, M., Digalakis, V.V., and Kimball, O.A. (1996b). From HMMs to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Trans. Speech Audio Proc.*, *4*(5):360–378.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*, San Mateo, California, Morgan Kaufmann.

Rabiner, L. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*, Englewood Cliffs, NJ, Prentice Hall.

Schürmann, J. (1996). *Pattern Classification, A Unified View of Statistical and Neural Approaches*. New York: Wiley & Sons.

Szarvas, M., Mihajlik, P., Fegyó, T., and Tatai, P. (2000). Automatic recognition of Hungarian: Theory and practice. elsewhere in this issue.

Tipping, M.E. and Bishop, C.M. (1997). *Probabilistic Principal Component Analysis*, Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, Birmingham, United Kingdom.