TR - H - 196

A Study of Cepstrum Optimization by Discriminative Feature Extraction. - DFE implementation details -

Alain Biem **Shigeru Katagiri**
(ATR Interpreting Telecommunication Research Labs.)

1996. 5. 27

ATR人間情報通信研究所
〒619-02 京都府相楽郡精華町光台2-2 ☎ 0774-95-1011

ATR Human Information Processing Research Laboratories
2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
Telephone: +81-774-95-1011
Facsimile: +81-774-95-1008

A Study of Cepstrum Optimization by Discriminative Feature Extraction. -DFE implementation details-

Alain BIEM & Shigeru KATAGIRI†
ATR Human Information Processing Research Laboratories
†ATR Interpreting Telecommunications Research Laboratories

Abstract

This report discusses application of Discriminative Feature Extraction (DFE) to speech recognition. The report is an opportunity to discuss DFE implementation on speech recognizers. The implementation process is viewed in detail in the context of filter-bank optimization. Choice of learning rate parameters, smoothing of the loss, optimization methodology are described for an accurate application to speech recognition. As illustration, application of DFE to cepstrum optimization is studied, for a multi-speaker vowel recognition task. It is shown that DFE-based cepstrum are more robust than conventional Mel-scale based cepstrum coefficients.

Contents

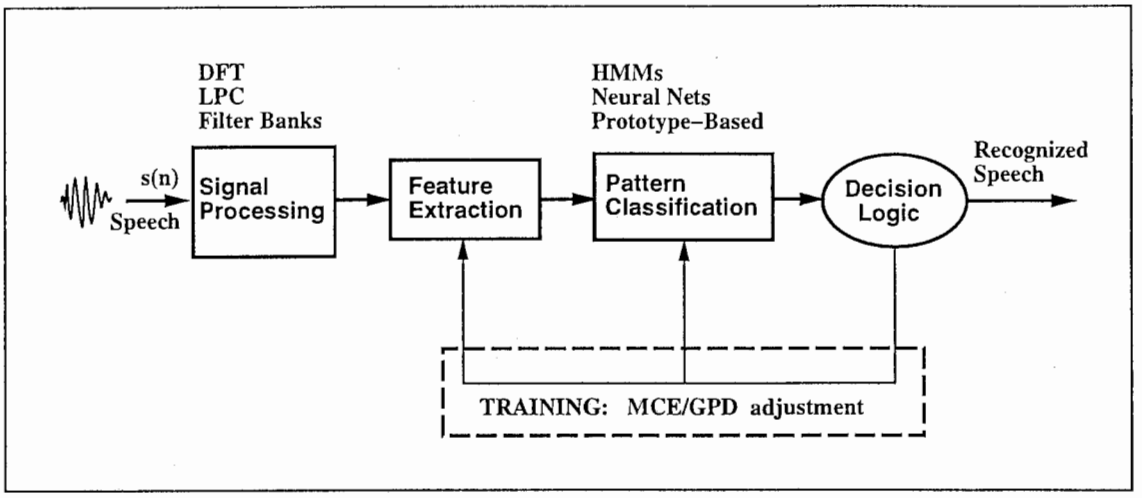

Figure 1: Block Diagram of DFE-based Speech recognizer Design

1 Introduction

The discriminative feature extraction approach was previously introduced as a method to efficiently extract features for achieving minimum error at the output of the recognizer. However, basic characteristics, such chain rule learning or choice of parameters were left undetailed. This report is intended to provide these information which might be needed to efficiently implement DFE on speech recognizer. As illustration of DFE, we discuss DFE application to filter-bank based cepstrum aiming at japanese vowel s recognition using the prototype based minimum error classifier [4].

Cepstrum coefficients, either based on filter bank or LPC model of speech, have been widely used in speech recognition. The popularity of cepstral coefficients is mainly due to the fact that they provide a good compactness of information by appropriate decorrelation of features. This enables the use of simple metrics such the Euclidean metric as measure of similarity with prototype patterns.

For a filter bank-based cepstrum, performance depends on appropriate design of the filter bank. In previous work [3], we proposed a method for designing various filter bank parameters aiming at minimum classification error. The filter bank outputs were log energies that fed a classifier based on an Euclidean distance measure. Here, we present an extension of this method by performing the distance measure on the cepstral domain.

The report is organized as follows: In section II, we describe the Discriminative Feature Extraction method (DFE). This includes discussion on t the selection of parameters such as learning rates, loss function, the optimization methodology and the interaction between the classification module and the feature extraction module during learning. In Section III, we discuss DFE application to cepstrum optimization. Implementation details using PBMEC are shown in section IV. Details of filter bank optimization scheme are presented in section V. Finally, section VI describes the application of DFE-based cepstrum to vowel recognition.

2 Discriminative Feature Extraction

2.1 Formalization

Let us consider the task of recognizing speech pattern of K categories, $\{C_1, \ldots, C_k, \ldots, C_K\}$. The categories could be phonemes, syllables, words or sentences. We assume our pattern recognizer to be a modular system, consisting of a front-end feature extractor and a back-end classifier. A desirable approach to the overall design of the recognizer must link the two modules appropriately.

Let us assume that the speech signal S is expressed as a sequence of frames $\boldsymbol{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t \ldots, \mathbf{x}_T\}$ where $\mathbf{x}_t$ is a vector containing time domain or spectral domain information, prior to any data reduction process. The elements in $\mathbf{x}_t$ consist of raw speech representation and are still containing spurious or redundant information which must be selected according to the task. This selection is to be carried out by the feature extraction process.

The feature extraction process is a transformation $\mathcal{F}_\Theta(\boldsymbol{X}) = \boldsymbol{Y}$, with parameter Θ, which maps $\boldsymbol{X}$ to a corresponding feature pattern $\boldsymbol{Y} = \{\mathrm{y}_1, \ldots, \mathrm{y}_t, \ldots, \mathrm{y}_T\}$; here, we assume that the feature extraction performs on a frame-by-frame basis. Thus, $\mathrm{y}_t = \mathcal{F}_\Theta(\mathrm{x}_t)$ is the corresponding feature pattern of the frame x_t. The components of y_t are simply the outputs of the feature extraction.

In classical pattern recognition, the classifier $C(\cdot)$ takes $\boldsymbol{Y}$ as input and operates according to the following decision rule:

$$C(\boldsymbol{Y}) = C_i \qquad \text{if} \quad i = \arg\min_k g_k(\boldsymbol{Y}; \Lambda) \tag{1}$$

where $g_k(\boldsymbol{Y}; \Lambda)$ is a discriminant function which estimates the degree to which $\boldsymbol{Y}$ belongs to C_k: distances and probabilities (likelihoods) are widely used discriminant functions [2]. Λ is the parameter set of the classifier, which provides an accurate tuning of the decision rule. Here, we assume that the decision rule is based on the minimum value of the discrimination function.

Thus, classical methods assume s for a separate design of the classifier $\{g_k(\boldsymbol{Y}; \Lambda)\}$ and the feature extractor $\mathcal{F}_\Theta(\cdot)$. However, by substituting $\boldsymbol{Y}$ by $\mathcal{F}_\Theta(\boldsymbol{X})$ in (1), we have an integrated recognizer design defined by a set of discriminant functions $\{g_k(\mathcal{F}_\Theta(\cdot); \Lambda) = g_k((\cdot); \Phi)\}$ with equivalent parameter set Φ. We can now work directly with a recognizer defined by a super parameter set Φ. Accurate tuning of the parameter set Φ, which leads to an optimal overall recognizer, could not be achieved when assuming the separability $\Phi = (\Theta, \Lambda)$. Thus, in DFE-based pattern recognition, the recognizer decision is made by the following rule:

$$C(\boldsymbol{X}) = C_i \qquad \text{if} \quad i = \arg\min_k g_k(\boldsymbol{X}; \Phi). \tag{2}$$

The Discriminative feature extraction method implies the use of discriminative training as a mean for optimizing the overall recognizer. In Discriminative training, the target is to maximize the degree of separability between categories, instead of maximizing the ability of a model to represent a given category. For this purpose, DFE mainly relies on the MCE/GPD formulation. Within this framework, 3 points must be to be clearly defined:

- A Misclassification measure.
- A smooth approximation to the 0-1 step wise error function.
- The optimization scheme.

A typical DFE-based speech recognizer is illustrated in Fig. 1.

2.2 Misclassification measure

Given the observation pattern $\boldsymbol{X} \in C_k$, the feature extractor transforms $\boldsymbol{X}$ into a more compact representation $\boldsymbol{Y}$. The discrimination ability of the recognizer is estimated by the use of a misclassification measure $d_k(\boldsymbol{Y}; \Lambda)$ which depends on the set of discriminant functions $\{g_k(\boldsymbol{Y}; \Lambda)\}_{k \in \{1,\ldots,K\}}$. The misclassification measure emulates the classification decision in scalar values: A positive value means a misclassification and a negative value implies correct classification.

For instance, considering the classification rule of (1), a misclassification measure for the spectral pattern $\boldsymbol{Y}$ could be defined as follows:

$$d_k(\mathcal{F}_\Theta(\boldsymbol{Y}); \Lambda) = d_k(\boldsymbol{Y}; \Lambda) = g_k(\boldsymbol{Y}; \Lambda) - \left[\frac{1}{M-1} \sum_{j \neq k} \{g_j(\boldsymbol{Y}; \Lambda)\}^\eta \right]^{\frac{1}{\eta}}, \tag{3}$$

where η is a positive number which controls the relative contribution of the competing categories in the decision process.

2.3 Loss definition

The loss (cost) of the decision of assigning $\boldsymbol{X}$ to category C_k, denoted by $\ell_k(\boldsymbol{X}, \Phi)$, is a function of the misclassification measure $d_k(\boldsymbol{Y}; \Lambda)$ and is a smooth approximation of the minimum error cost function, e.g 0-1 step function. The smoothness of the loss is required by the use of a gradient-based optimization process during learning. Many approximations exist and we here present few basic choices.

Sigmoidal loss

The sigmoid is widely used function in feed-forward neural network as the neuron transfer function. Its shape is shown in Fig. 2. It is defined as

$$\ell(d) = \frac{1}{1+\exp(\alpha d)}, \quad \alpha \geq 0 \tag{4}$$

$$\ell'(d) = \alpha \ell(d)(1-\ell(d)) \tag{5}$$

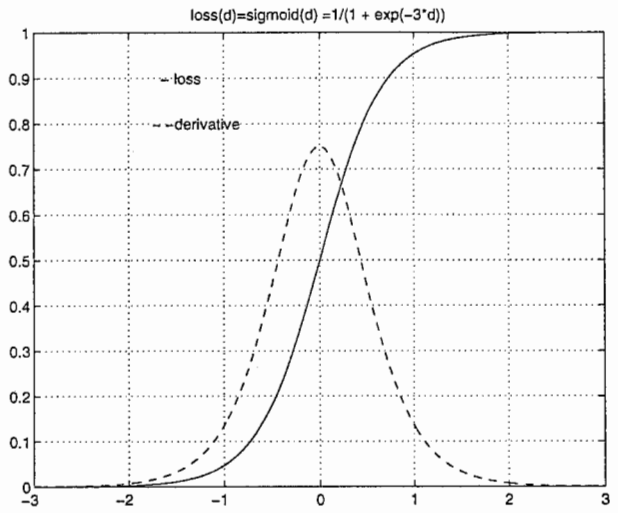

Figure 2: Sigmoid loss function and its derivative.

Exponential

The exponential-based approximation is not as widely used as the the sigmoidal loss function but it is a good approximation to the 0-1 step wise function. Its shape is shown Fig.3 and it is defined as

$$\ell(d) = (1-\exp(-\alpha d))1(d>0), \quad \alpha \geq 0 \tag{6}$$

$$\ell'(d) = \alpha \exp(-\alpha d)1(d>0) \tag{7}$$

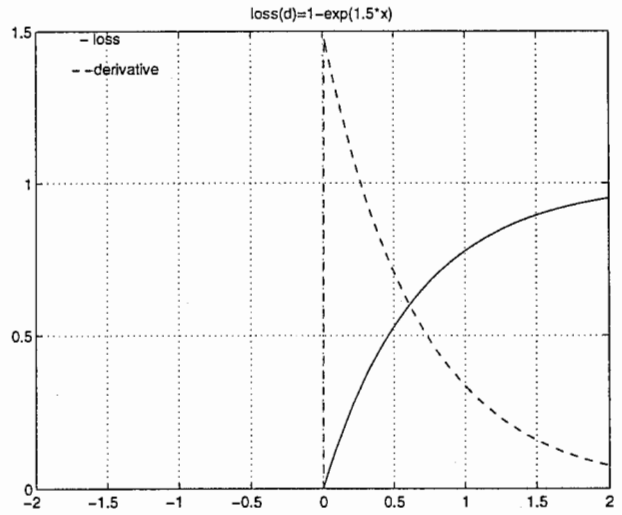

Figure 3: Exponential loss function and its derivative.

Error function

A loss function based on the error function erf(·) is shown in Fig. 4 and its definition is

$$\ell(d) = \frac{1}{2}\mathrm{erf}(\alpha d) + \frac{1}{2}, \qquad \alpha \geq 0 \tag{8}$$

$$\ell'(d) = \frac{\alpha}{\sqrt{\pi}} \exp(-(\alpha d)^2) \tag{9}$$

where the error function is defined as

$$\mathrm{erf}(d) = \frac{2}{\sqrt{\pi}} \int_d^{\infty} \exp(-\nu^2) d\nu \tag{10}$$

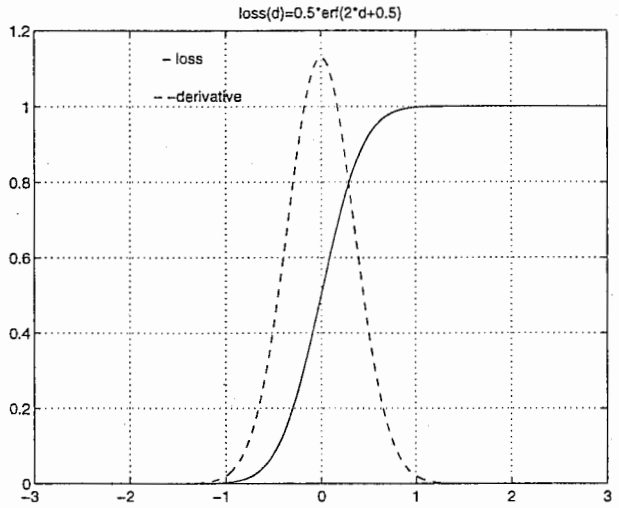

Figure 4: Error-function-based loss and its derivative.

2.4 Optimization methodology

The target in the DFE paradigm is to find the optimal values of both Θ and Λ. Thus DFE treats both Θ and Λ as adjustable parameters, while the original MCE/GPD implementation treats only Λ as adjustable parameters for recognition. Any optimization method could be used for this purpose (a batch-type or adaptive optimization algorithm). In this paper, we consider the use an adaptive, descent optimization which updates, the recognizer parameters Θ and Λ every time one training sample is given.

The training aims at reducing the ultimate error measure, i.e., the expected loss

$$\mathcal{L}(\Phi) = E_{\boldsymbol{X}}\left[\ell_k(\boldsymbol{X}, \Phi)\right] \tag{11}$$

$$= E_{\boldsymbol{X}}\left[\ell_k(\mathcal{F}_\Theta(\boldsymbol{X}), \Lambda)\right], \tag{12}$$

where it is assumed that the expectation for our observation sample space exists. The probabilistic descent theorem then guarantees that asymptotically (infinite repetition), the following adaptive adjustment

$$\Phi[\tau+1] = \Phi[\tau] - \epsilon_\tau \mathbf{U} \frac{\partial \ell(X, \Phi)}{\partial \Phi} \tag{13}$$

will lead to at least a local optimal status of Φ (a status of Φ which realizes a local minima of $\mathcal{L}(\Phi)$), where $\mathbf{U}$ is a positive-definite matrix, ϵ_τ is a small, monotonically-decreasing, positive number, called the *learning rate*, and $\Phi[\tau]$ denotes a status of Φ at iteration τ. The adjustment rule of (13) includes the use of a chain rule of differential calculus for adjusting the feature extractor module.

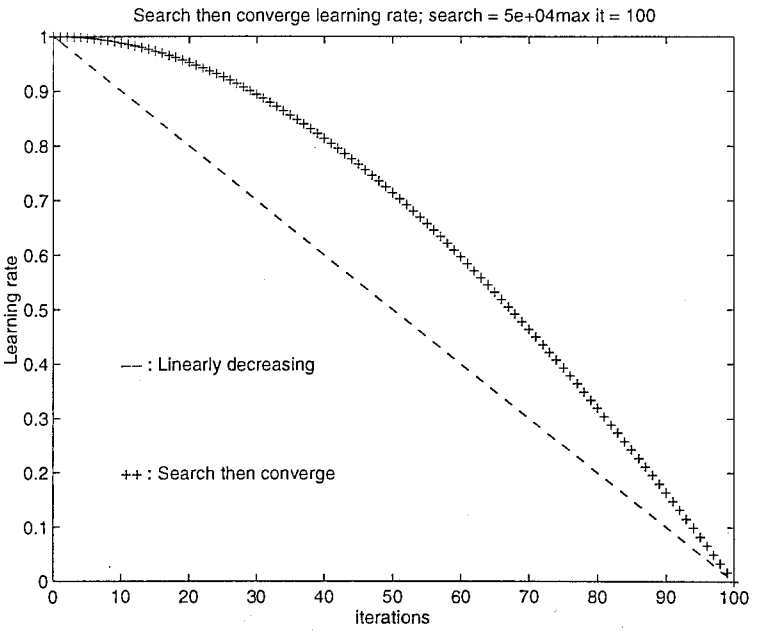

Figure 5: Different types of learning rates.

2.4.1 Choice of the learning rate

In gradient descent learning, performance usually depends on an accurate choice of the learning rate. In theory, the learning should obey the stochastic constraints:

$$\sum_{\tau=1}^{\infty} \epsilon_\tau = \infty \tag{14}$$

$$\sum_{\tau=1}^{\infty} \epsilon_\tau^2 < \infty, \tag{15}$$

for leading to the minimization of the expected loss.

In practice, infinite training is not possible. Similar to other MCE/GPD applications, DFE simply tries to reduce the classification errors over the given finite training pattern set. Various learning rate strategies can be used for this purpose. One classical example is to choose

$$\epsilon_\tau^{(1)} = \frac{\epsilon_o}{a\tau + b}, \tag{16}$$

where the parameters a and b are chosen such that

$$\epsilon_\tau^{(1)} = \epsilon_o \left(1 - \frac{\tau}{\mathcal{N}}\right) \tag{17}$$

for a finite number of iterations $\mathcal{N}$.

Sometimes, a fast learning is needed at the first iterations. Thus, Darken [1], proposed the "search-then-converge learning rate", which is defined for an infinite number of iterations as

$$\epsilon_\tau^{(2)} = \epsilon_o \frac{1 + \frac{a}{\epsilon_o}\frac{\tau}{\tau_o}}{1 + \frac{a}{\epsilon_o}\frac{\tau}{\tau_o} + \tau\frac{\tau^2}{\tau_o{}^2}}. \tag{18}$$

This learning scheme obeys the following properties:

$$\epsilon_\tau^{(2)} \sim \epsilon_o \quad \text{when} \quad \tau \ll \tau_o \tag{19}$$

$$\epsilon_\tau^{(2)} \sim \frac{a}{\tau} \quad \text{when} \quad \tau \gg \tau_o \tag{20}$$

which means that the learning rate is approximately constant (equal to ϵ_o) for iterations which are much smaller that τ_o and decreases linearly at rate a for iterations which are far greater than τ_o. For a finite

number of $\mathcal{N}$ iterations, this learning scheme could be approximated by

$$\epsilon_\tau^{(3)} = \frac{\epsilon_\tau^{(2)} - \epsilon_{\mathcal{N}}^{(2)}}{\epsilon_0^{(2)} - \epsilon_{\mathcal{N}}^{(2)}}. \tag{21}$$

Fig.5 shows the behavior of this two types of learning rates for fixed number of iterations.

2.4.2 Modular optimization

The learning rule of (13) shows the interaction between the classifier's parameter and feature extractor's parameter. However, in practice, if both parameters are different in nature, a straightforward use of (13) may lead to instability (no immediate convergence). It might be useful, in light of better comprehension of the interaction of the feature extractor process to run a selective optimization scheme according to the type of parameters. This is done through an appropriate choice of the positive definite matrix $\mathbf{U}$ according the type of parameters to be optimized. That is, let $\mathbf{U} = [\mathbf{U1}, \mathbf{U2}]$ such that (13) is replaced by

$$\Lambda[\tau+1] = \Lambda[\tau] - \epsilon_\tau \mathbf{U1} \frac{\partial \ell(\boldsymbol{Y}, \Lambda)}{\partial \Lambda} \tag{22}$$

$$\Theta[\tau+1] = \Theta[\tau] - \rho_\tau \mathbf{U2} \frac{\partial \ell(\boldsymbol{Y}, \Lambda)}{\partial \Theta} \tag{23}$$

where ϵ_τ and ρ_τ are small positive numbers, representing the classifier learning rate and the feature extractor learning rate, respectively.

Selective optimization enables to use different convergence speed for the classifier parameters and the feature extractor parameters which would permit more adaptability and interaction between the two set of parameters. The relation between ϵ_τ and ρ_τ is crucial for accurate learning. The use of $\rho_t = f(\epsilon_t)$ where $f(\cdot)$ is a monotonic mapping called the *modulating function* is advisable. This is equivalent to an adaptive learning speed between the feature extractor and the classifier. However, the use of higher order functions might also be appropriate. Few suggestions are:

- $\rho_\tau = \alpha p_n(\epsilon_\tau)$ where $\alpha < 1$.
- $\rho_\tau = \alpha \log(p_n(\epsilon_\tau) + 1)$.

where $p_n(\epsilon_\tau) = \epsilon_\tau q_n(\epsilon_\tau)$ and q_n is a n-th order polynomial with $n = 0$ or $n \geq 2$.

Another approach is to train, separately, classifier and feature extractor (separate optimization). Within this framework, one can train first the classifier and then feature extractor or vice-versa.

The two methods (selective optimization and separate optimization) are merely equivalent in term of convergence but may not lead to the same minima in the parameter space. The selective optimization is convenient in practice for a straightforward optimization of the overall recognizer. It enables the achievement of a more flexible interaction between the feature extractor and the classifier as well as accelerates the overall training convergence. That is, in this framework, one can control the convergence speed and convergence nature for each of the module. However, the modulating function $f(\cdot)$ which sets the link between the two sets of parameters during learning should be carefully chosen. Separate optimization is a convenient way to use DFE on already optimized classifiers in order to optimize the feature extractor. It provides a convenient way to adaptively optimize the feature extractor as data becomes available. This case of training each module separately seems to contradict the DFE concept, but it may provide a practical way to easily retrain (or adapt) pre-designed status of either the feature extractor or the classifier.

2.5 Classifier and feature extractor interaction

The basic rule when optimizing the feature extractor by DFE is given in (23). Let θ represents any optimizable parameter of the feature extractor. Let $\boldsymbol{X}$ be any given training token of known category. As said before, $\boldsymbol{X}$ is a sequence of raw frames $\mathbf{x}_t = [x_{t,f}]^T$. Let $\boldsymbol{Y}$ be the corresponding feature pattern, which is also the input to the classifier. $\boldsymbol{Y}$ is also a sequence of frame $\mathbf{y}_t = [y_{t,i}]^T$, for $i \in \{1, \ldots, L\}$. θ will adjusted by the following iterative process:

$$\theta[\tau+1] = \theta[\tau] - \rho_\tau \mathbf{U2} \delta\theta \tag{24}$$

where

$$\delta\theta = \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial \theta} \tag{25}$$

$$= \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{L} \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial y_{t,i}} \frac{\partial y_{t,i}}{\partial \theta}. \tag{26}$$

The term $\frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial y_{t,i}}$ depends on the structure of the classifier and represents the derivative of the classifier error according to the i-th feature of frame t. The set of $\frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial y_{t,i}}$ is called *the classifier's input derivatives* since it characterizes the derivative of the loss versus the input to the classifier. The second term $\frac{\partial y_{t,i}}{\partial \theta}$ is the derivative of the i-th feature of frame t versus the feature extractor parameter θ. The set $\frac{\partial y_{t,i}}{\partial \theta}$ is called *the feature extractor output-derivatives* because it characterizes the derivative of the output of the feature extractor versus the feature extractor parameter. Thus, feature extraction module optimization is done on a frame-by-frame basis.

Thus, in the modular framework of DFE, the classifier will participate in DFE training by providing the input-derivatives to the feature extraction module. These derivatives depend only on the classifier. The feature extractor will use these s derivatives to update itself. This process is described in detail in the next sections.

3 DFE-based cepstrum representation design

3.1 Filter bank-based cepstrum

In filter bank modeling of the speech spectrum, the cepstrum is computed at the output of the filter bank. In this paper, we focus on a filter bank model simulated on the DFT domain by weighting of the DFT bins with the magnitude frequency response of the filter. Thus, for a sequence of speech vectors $\boldsymbol{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_\mathcal{T}\}$ in which $\mathbf{x}_t = [x_{t,1}, \ldots, x_{t,f}, \ldots, x_{t,F}]^T$ is the magnitude spectrum of the frame (short time window position); $x_{t,f}$ represents the f-th element of frame-vector[1]. F is the maximum frequency. A Q-channel filter bank model transforms each $\mathbf{x}_t$ into a lower dimensional vector $\mathbf{e}_t = [e_{t,1}, \ldots, e_{t,c}, \ldots, e_{t,Q}]^T$ such that an output feature $e_{t,c}$ is the windowed log energy of the c-th channel:

$$e_{t,c} = \log_{10}\left(\sum_{f \in B_c} \theta_c(f) x_{t,f}\right), \quad \text{for } c = 1, \ldots, Q, \tag{27}$$

where B_c represents the channel interval and $\theta_c(f)$ the weighting at frequency f provided the c-th filter.

From the vector of log energies, the cepstrum vector $\mathbf{c}_t = [c_{t,1}, \ldots, c_{t,i}, \ldots, c_{t,L}]^T$ is computed via a discrete cosine transform:

$$c_{t,i} = \sum_{c=1}^{Q} e_{t,c} \cos\left(\frac{i\pi}{Q}(c - 0.5)]\right), \tag{28}$$

for $i = 1, \ldots, L$, where L is the number of cepstral coefficients. Here the cepstrum vectors are the inputs to the classifiers (the cepstrum vector $\mathbf{c}_t$ is having the role of the feature vector $\mathbf{y}_t$ of session II).

3.2 Filter bank representation

For practical requirement and commodity of gradient-based optimization, the magnitude response $\theta_c(f)$ of the c-th filter is constrained to a Gaussian-form:

$$\theta_c(f) = \varphi_c \exp\left(-\beta_c \left(p(\gamma_c) - p(f)\right)^2\right), \tag{29}$$

for $c = 1, \ldots, Q$, where the trainable parameters $\beta_c > 0$ and γ_c determine bandwidth and center frequency, and φ_c is the trainable "gain" parameter in the c-th channel. $p(f)$ maps the linear frequency f onto the perceptual representation. For instance, a Mel scale mapping will produce Mel cepstral coefficients (MFCC). The block diagram of cepstrum optimization by DFE is shown in Fig. 6.

[1] Concretely, f is an integer describing the sequence of DFT bins

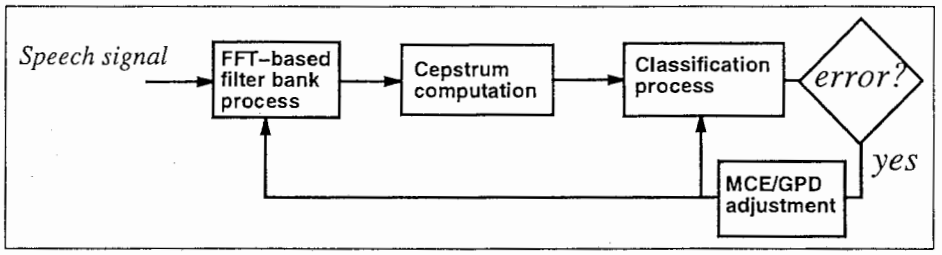

Figure 6: Block diagram of DFE-based cepstrum optimization.

DFE-optimized cepstrum coefficients (DFCC) are designed by appropriate optimization of center frequency, bandwidth and gain. To find the relevant parameter and reduce the complexity thereof, each parameter could be adjusted independently while others are fixed. Thus, center frequency-optimized cepstrum coefficients (C-DFCC), bandwidth-optimized DFCC (B-DFCC), gain-optimized DFCC (G-DFCC) and weighting-optimized DFCC (W-DFCC) could be designed by such a method. Here the term "weighting" refers to optimizing each frequency weight without keeping the Gaussian constraint. For generating a globally efficient model, a simultaneous optimization of center frequency, bandwidth and gain (CBG-DFCC) could be carried out.

4 Pbmec-based implementation

4.1 Recognizer structure

The recognizer used in the following is the Prototype-Based Minimum Error Classifier (PBMEC) structure described in [4]. It is a finite state machine, similar to a Hidden Markov Model (HMM) but with use of Lp-norm of distances instead of probabilities, which embeds a DTW procedure to provide the final score of an input pattern. Thus, the experiments described throughout this paper could easily be extendible to a HMM-based speech recognizer, using Viterbi decoding.

A category (phoneme/word/sentence) is represented as a string of phonetic models, in which each phonetic model consists of a concatenation of sub-phonemic states. Each state is assigned a number of references vectors similar to mean vectors within an HMM-state.

Concretely, we are given a finite set of P phonetic models, i.e,

$$\Lambda = \{\lambda_1, \ldots, \lambda_j, \ldots, \lambda_P\}, \quad 1 \leq j \leq P \tag{30}$$

where λ_j is composed of a set of prototypes vectors distributed among the states of the model:

$$\lambda_j = \{\mathbf{r}_{j,s,m}\} \begin{array}{l} 1 \leq s \leq S \\ 1 \leq m \leq M. \end{array} \tag{31}$$

$\mathbf{r}_{j,s,m}$ represents the m-th prototype vector of the s-th state of model λ_j and $r_{j,s,m,i}$ is the i-th component of $\mathbf{r}_{j,s,m}$. S is the total number of states. The number of reference vector per states is M.

The distances between an input spectral frame-vector $\mathbf{x}$ to state s of category j is an Lp-norm of distances defined as

$$D_{j,s}(\mathbf{x};\Phi) = \left\{ \sum_{m=1}^{M} \sigma(\mathbf{c}, \mathbf{r}_{j,s,m})^{-\nu} \right\}^{-\frac{1}{\nu}} \tag{32}$$

where $\sigma(\mathbf{c}, \mathbf{r}_{j,s,m}) = (\mathbf{c} - \mathbf{r}_{j,s,m})(\mathbf{c} - \mathbf{r}_{j,s,m})^T$ is the Euclidean distance between $\mathbf{c}$ and $\mathbf{r}_{j,s,m}$. $\mathbf{c}$ is the cepstral representation of $\mathbf{x}$ and ν is a positive constant.

The discriminant function $g_k(\boldsymbol{X};\Phi)$ for each string category k is the sum of states-distances along possible DTW paths for that category:

$$g_k(\boldsymbol{X};\Phi) = \left\{ \sum_{\psi_k=1}^{\Psi} \mathcal{D}_{\psi_k}(\boldsymbol{X};\Phi)^{-\mu} \right\}^{-\frac{1}{\mu}} \tag{33}$$

where $\mathcal{D}_{\psi_k}(\boldsymbol{X};\Phi)$ is ψ_k-th path distance as found by the DTW procedure in recognizing category k. Ψ is the maximum number of paths considered.

For a large ν, (33) is reduced to

$$g_k(\boldsymbol{X};\Phi) = \sum_{t=1}^{\mathcal{T}} D_{\psi_{k,\lambda}(t),\psi_{k,s}(t)}(\mathbf{x}_t;\Phi), \tag{34}$$

which is the sum over the best DTW path ψ_k (from now on, ψ_k will label the best DTW path of category k). $\psi_{k,\lambda}(t)$ is the current phonetic model at time t and $\psi_{k,s}(t)$ is the current state at time t along the ψ_k DTW path of the string category k.

4.1.1 Misclassification measure and loss

For a pattern $\boldsymbol{X}$ of category $\mathcal{C}$, the misclassification measure $d_{\mathcal{C}}(\boldsymbol{X};\Phi)$ is here defined as

$$d_{\mathcal{C}}(\boldsymbol{X};\Phi) = 1 - \frac{\left\{ \frac{1}{K-1} \sum_{k\neq\mathcal{C}}^{K} g_k(\boldsymbol{X};\Phi)^{-\xi} \right\}^{-\frac{1}{\xi}}}{g_{\mathcal{C}}(\boldsymbol{X};\Phi)} \tag{35}$$

where K denotes the total number of categories (the vocabulary size). This misclassification measure is believed to be more robust since e the range of the misclassification measure values is narrower than the ones provided by the form given in (3). See [4] for further explanations.

A large ξ value was used which means that only the best incorrect path was considered. Thus, the definition of equation (35) could be simplified to

$$d_{\mathcal{C}}(\boldsymbol{X};\Phi) = 1 - \frac{g_{\mathcal{W}}(\boldsymbol{X};\Phi)}{g_{\mathcal{C}}(\boldsymbol{X};\Phi)} \tag{36}$$

where $g_{\mathcal{W}}$ corresponds to the best incorrect category. The loss is a piece wise linear approximation to the 0-1 step wise function as shown in Fig. 7.

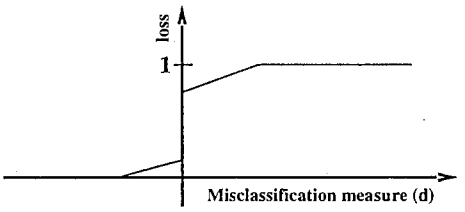

Figure 7: A piecewise loss approximation of the ideal loss function

Note that the misclassification measure used here performs learning at the string level. During learning, the gradients of the errors are propagated back to adjust reference vectors within each state, down to the parameters of the front-end filter bank. The next session will be detailing this phase.

4.2 DFE-based design

DFE-based cepstrum design is optimizing various parameters of the filter bank while using a cepstral distance measure in the classification process. If Λ denotes the set of parameter of the classifier and Θ the parameter set of the filter bank, the overall recognizer parameter $\Phi = \{\Theta, \Lambda\}$ is adaptively updated after presentation of each pattern $\boldsymbol{X}$ aiming at minimizing a smooth error count measure. The classifiers parameters consist of the set of prototypes for each phonetic model, i.e, $\Lambda = \{\lambda_j\}$, $j \in \{1,\cdots,P\}$. The filter bank parameters are composed of the set of weighting, center frequency and gain, i.e, $\Theta = \{\phi = \{\gamma_c, \beta_c, \varphi_c, \theta_c(f)\}\}$, for $c \in \{1,\cdots,Q\}$.

Given a training token $\boldsymbol{X}$, belonging to a known category (words/phoneme/sentence), the adaptation rule for the classifiers parameters is

$$r_{j,s,m,i}[\tau+1] \quad = \quad r_{j,s,m,i}[\tau] - \epsilon_t \mathbf{U} 1 \delta r_{j,s,m,i} \tag{37}$$

where the reference vector increment $\delta r_{j,s,m,i}$ is

$$\delta r_{j,s,m,i} = \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial r_{j,s,m,i}} . \tag{38}$$

The feature extractor proceeds in two steps. For each frame $\mathbf{x}_t = [x_{t,f}]^T$ (for $f = \{1, \ldots, F\}$) of the power spectrum signal $\boldsymbol{X}$, first, a vector of log energies $\mathbf{e}_t = [e_{t,c}]^T$, for $c = \{1, \ldots, Q\}$, is calculated. Secondly, the cepstrum vector $\mathbf{c}_t = [c_{t,i}]^T$, for $i = \{1, \ldots, L\}$), is generated. The adjustment of filter bank parameters must consider these two steps. Let ϕ be any parameter of the filter bank. Since ϕ has a physical meaning, we need to make sure that its value remain positive. This is done by using the transformation

$$\phi = \exp(\overline{\phi}) \tag{39}$$

and then performing the following adjustment:

$$\overline{\phi}[\tau + 1] = \overline{\phi}[\tau] - \rho_\tau \mathbf{U2} \delta \overline{\phi} \tag{40}$$

where

$$\delta \overline{\phi} = \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial \overline{\phi}} \tag{41}$$

$$= \sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{L} \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial c_{t,i}} \frac{\partial c_{t,i}}{\partial \overline{\phi}} \tag{42}$$

Then the key calculations are the classifier's input-derivatives $\frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial c_{t,i}}$ which depend on the classifier structure and the feature extractor's output derivatives $\frac{\partial c_{t,i}}{\partial \overline{\phi}}$ which characterize the feature extractor process. In the next sections, we provide the details of the input-derivatives for PBMEC and feature extraction's output derivatives when using Gaussian-shaped filters.

4.3 PBMEC-input derivatives

In the PBMEC context, each input frame is compared against a set of a reference vectors within a state of a certain phonetic model as decided by the DTW procedure along a certain category. Given a spectral pattern $\boldsymbol{X}$, if $\psi_{k,\lambda}(t)$ and $\psi_{k,s}(t)$ are the current phonetic model and the current state along the ψ_k DTW path at time t of category k, then, according to (34),

$$\frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial c_{t,i}} = \sum_{k=1}^{K} \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial D_{\psi_{k,\lambda}(t), \psi_{k,s}(t)}(\mathbf{x}_t; \Phi)} \sum_{m=1}^{M} \frac{\partial D_{\psi_{k,\lambda}(t), \psi_{k,s}(t)}(\mathbf{x}_t; \Phi)}{\partial \sigma(\mathbf{c}, \mathbf{r}_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m})} \frac{\partial \sigma(\mathbf{c}, \mathbf{r}_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m})}{\partial c_{t,i}} \tag{43}$$

From (32), we have

$$\frac{\partial \sigma(\mathbf{c}, \mathbf{r}_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m})}{\partial c_{t,i}} = - \frac{\partial \sigma(\mathbf{c}, \mathbf{r}_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m})}{\partial r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i}} . \tag{44}$$

Thus,

$$\frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial c_{t,i}} = \sum_{k=1}^{K} \sum_{m=1}^{M} - \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i}} \tag{45}$$

$$= - \sum_{k=1}^{K} \sum_{m=1}^{M} \delta r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i} \tag{46}$$

where $\delta r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i} = \frac{\partial \ell(\boldsymbol{X}, \Phi)}{\partial r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i}}$ is the increment of the i-th component of the m-th reference vector of the current state at time t of path ψ_k defining category k (See [4] for further information). Thus, for PBMEC, the input derivatives are simply the negative sum over the increment of all reference vectors along DTW paths. This is due to the use of the Euclidean distance as a frame-level distance. If one uses the form of the misclassification measure given in (36), only the correct and the best incorrect path needed to be considered.

5 Filter bank parameters adaptation

Here, we focus on expanding the feature extractor's output derivatives $\frac{\partial c_{t,i}}{\partial \overline{\phi}}$. These derivatives do not depend on the back-end classifier, but only on the structure of the filter bank. Thus, the formula displayed in this section could be used on any recognizer which uses the filter bank model described in this paper.

Taking account of the cepstrum transformation, we have

$$\frac{\partial c_{t,i}}{\partial \overline{\phi}} = \sum_{c=1}^{Q} \frac{\partial c_{t,i}}{\partial e_{t,c}} \frac{\partial e_{t,c}}{\partial \overline{\phi}} \tag{47}$$

$$= \sum_{c=1}^{Q} \cos\left(\frac{i\pi}{Q}(c-0.5)\right) \frac{\partial e_{t,c}}{\partial \overline{\phi}} . \tag{48}$$

The above computation is only needed because of the cepstrum transformation and is not necessary if one uses log energies as input to the classifier. The key steps for filter bank optimization is the calculation of $\frac{\partial e_{t,c}}{\partial \overline{\phi}}$, for each channel c. The general chain rule is

$$\frac{\partial e_{t,c}}{\partial \overline{\phi}} = \sum_{f=1}^{F} \frac{\partial e_{t,c}}{\partial \theta_c(f)} \frac{\partial \theta_c(f)}{\partial \overline{\phi}} . \tag{49}$$

This signifies that, first, one needs to compute the derivative $\frac{\partial e_{t,c}}{\partial \theta_c(f)}$ for each frequency f, before expanding to $\frac{\partial \theta_c(f)}{\partial \overline{\phi}}$ according to the type of parameter $\overline{\phi}$. According to (27),

$$\frac{\partial e_{t,c}}{\partial \theta_c(f)} = \frac{x_{t,f}}{\log(10)\mathcal{E}_c(\mathbf{x}_t)} \tag{50}$$

where

$$\mathcal{E}_c(\mathbf{x}_t) = \sum_{f \in B_c} \theta_c(f) x_{t,f} = \exp_{10}(e_{t,c})$$

is the output energy of frame $\mathbf{x}_t$ in the c-th channel (without the log transformation).

Now, let us expand $\frac{\partial \theta_c(f)}{\partial \overline{\phi}}$ according to the parameter $\overline{\phi}$.

5.1 Filter bank weight adjustment

Let us consider the case in which ϕ is the weight of a given channel $\hat{c}$ at frequency $\hat{f}$ ($\phi = \theta_{\hat{c}}(\hat{f})$). This is the "weighting" optimization scheme where the filter bank weight are adjusted without keeping the Gaussian constraint. For $\theta_{\hat{c}}(\hat{f}) = \exp(\overline{\theta_{\hat{c}}(\hat{f})})$, we have

$$\frac{\partial \theta_c(f)}{\partial \overline{\theta_{\hat{c}}(\hat{f})}} = \chi(c,\hat{c})\, \chi(f,\hat{f})\, \theta_c(f) \tag{51}$$

where

$$\chi(a,b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{otherwise} \end{cases} \tag{52}$$

5.2 Center frequency adjustment

Let ϕ represent the center frequency of a channel $\hat{c}$. In the center optimization case, since the perceptual mapping function is monotonic, it is easier to consider the center frequency in the perceptual domain. Thus $\phi = \tilde{\gamma}_{\hat{c}} = p(\gamma_{\hat{c}})$, where $\tilde{\gamma}_{\hat{c}}$ represents the center frequency of channel $\hat{c}$ in the perceptual domain. For $\tilde{\gamma}_{\hat{c}} = \exp(\overline{\tilde{\gamma}_{\hat{c}}})$ and from (29), it follows that

$$\frac{\partial \theta_c(f)}{\partial \overline{\tilde{\gamma}_{\hat{c}}}} = -2\beta_c \left(\tilde{\gamma}_c - p(f)\right) \theta_c(f)\, \tilde{\gamma}_c\, \chi(c,\hat{c}). \tag{53}$$

5.3 Bandwidth adjustment

Here ϕ is the parameter $\beta_{\hat{c}}$ of channel $\hat{c}$-th. Let $\beta_{\hat{c}} = \exp(\overline{\beta_{\hat{c}}})$. From (29), it follows that ,

$$\frac{\partial \theta_c(f)}{\partial \overline{\beta_{\hat{c}}}} = -\beta_c \left(\tilde{\gamma}_c - p(f)\right)^2 \theta_c(f)\, \chi(c, \hat{c}) \tag{54}$$

5.4 Gain adjustment

ϕ is the parameter $\varphi_{\hat{c}}$ of channel $\hat{c}$-th. For $\varphi_{\hat{c}} = \exp(\overline{\varphi_{\hat{c}}})$,

$$\frac{\partial \theta_c(f)}{\partial \overline{\varphi_{\hat{c}}}} = \theta_c(f)\, \chi(c, \hat{c}). \tag{55}$$

Thus, the derivative of the weight versus the log of the gain within a channel is equal to the weight.

The summary of filter bank optimization by DFE is given in the Appendix.

6 Application to vowel segments recognition

6.1 Task and Classifier structure

The task is to recognize the 5-class Japanese vowel uttered under various contexts by several speakers. A database of 500 sentences spoken by 5 speakers (3 males and 2 females) was used to extract 1750 tokens for training and 1750 token for testing. The training body and the testing body was balanced among the speakers and the vowel-category.

The speech signal was digitized at 12kHz and stored at 16 bits. A Hamming window of 21ms was used for the production of the spectral frames, prior to filter bank analysis and cepstrum computation.

6.2 Experimental conditions and results

In the following experiment, the original filter, before training was based on the Mel scale. The Mel scale maps the perceived frequency of a pure tone onto a linear scale. Hence, various analytical approximations of the original Mel scale has been derived from psychoacoustic experiments, all of them being approximatively linear below 1 kHz and logarithmic above 1kHz. In this paper, we used the following approximation, as provided by [5]:

$$m(f) = 2595 \log_{10}(1 + \frac{f}{700}). \tag{56}$$

It is common practice to choose a lower number of cepstrum coefficients as compared to the number of filters. This is motivated by the fact that a higher number of filters permits a better resolution of spectral characteristics and a lower number of cepstral coefficients will reduces the dimensionality of the feature. However, reducing the feature vector dimensionality also means that fewer information are given to the recognition process. .

For investigation, two kind of feature representation was tested. The first representation (representation I) uses the same number of filters and cepstrum (Q = 16, L = 15). 15 cepstrum coefficients is due to the fact the 16-th cepstral components is zero. The second representation (representation II) uses 20 filters with 10 cepstral coefficients (Q = 20, L = 10). In both cases, the filters were initially aligned in the Mel scale. A K-means algorithm was ran to select the initial set of PBMEC's prototypes, prior to MCE/GPD (no training of feature extraction module) or DFE training (simultaneous training of classifier and feature extractor). MCE/GPD training was carried out as baseline for MFCC testing. DFE training produced the various DFCC. DFCC was produced in a segment classification basis, using only one state of PBMEC with 1 and 3 prototypes per category in the first representation case and 1 prototype per category in the second representation case.

The best results achieved across the optimization methods (Full optimization and Selective optimization) are summarized in Table 1 and Fig. 8 for representation I and in table 2 for representation II.

In representation I, C-DFCC, W-DFCC and CBG-DFCC show the best results so far in the testing set. In particular, the performance of C-DFCC are rather close to the performance of CBG-DFCC. The meaning of these results is that center frequency optimization (spacing) contributes the most in the quality of cepstral features. W-DFCC put an emphasis on single optimum frequency which has resulted into the best of the DFCC in the training set.

In representation II, the best performance is achieved by C-DFCC in the testing set, although W-DFCC and CBG-DFCC show the best results in the training set. Thus, when looking across the two types of representations, the center frequency optimization contributes the most to the performance.

types of features	1 prototype/class		3 prototypes/class	
	train	test	train	test
k-means	29.77	27.14	25.31	23.71
MFCC	13.1	15.5	11.82	14.05
C-DFCC	12.4	14.3	11.14	13.37
B-DFCC	12.0	14.9	10.97	13.65
G-DFCC	13.0	14.9	11.6	13.7
W-DFCC	10.4	14.3	10.7	12.9
CBG-DFCC	12.4	14.2	11.02	13.54

Table 1: Experimental results using Mel-based DFCC in terms of error rates in the vowel recognition task for Q=16 and L = 15.

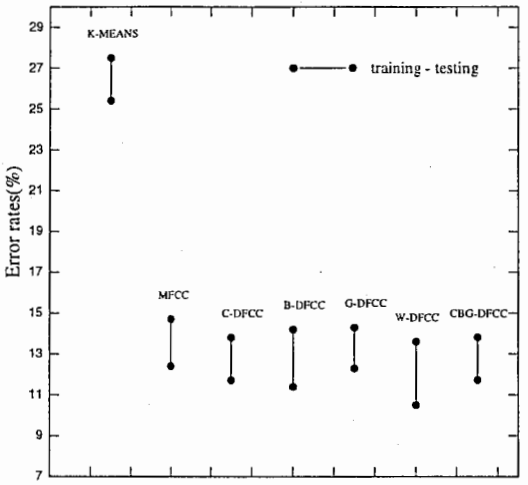

Figure 8: Error rate averaged over classifier size for various cepstral parameters for representation I. K-means labeled the performance of the original K-means algorithm before Minimum Classification Error training.

types of features	train	test
k-means	21.02	22.34
MFCC	13.08	15.5
C-DFCC	12.17	14.5
B-DFCC	12.08	14.6
G-DFCC	13.3	15.2
W-DFCC	11.88	14.9
CBG-DFCC	11.9	14.9

Table 2: Experimental results using Mel-based DFCC in terms of error rates in th vowel recognition task for Q=20 and L= 10.

For both representation, it could be noticed that optimizing any parameter of the filter bank model showed an improvement of the system performance. The best improvement are achieved when optimizing

the center, the "weighting" and the main three parameters at the same time. Optimizing the gain showed less improvement of the original model

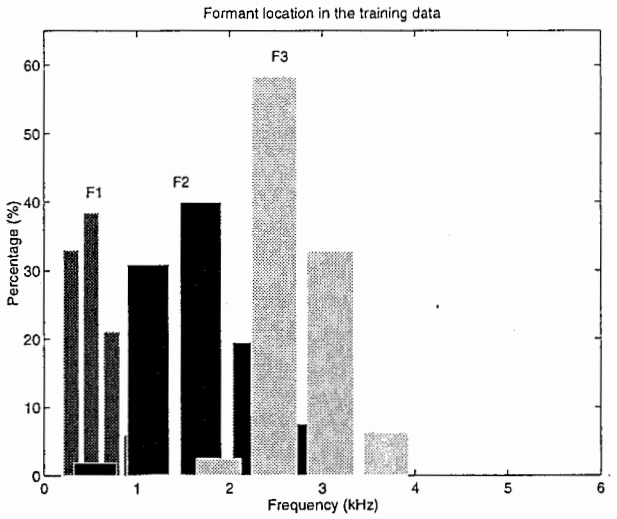

Figure 9: Formant location in the traing data as found by an LPC-based root finding algorithm.

6.3 Feature extraction process analysis

Formants are essential feature in recognizing vowels. For a deeper analysis of DFCC, we will analyze how the resulting filter bank model performs formants resolutions. Figure 9 shows an histograms of formants location in the train database as found by an LPC-based root finding method. It can be seen that F1 formants are mostly located in the region below 1 kHz, whereas most F2 formants are contained in the region between 1 and 3 kHz. 90 % F3 formants spans the region between 2.5 and 3.5 kHz.

6.3.1 Center frequency optimization

Figure 10 illustrates the modified filter bank after center frequency optimization by DFE for the two representation.

It could be seen that for both representations, modifications have occurred in the 1.5 kHz region (F2 domain), where some filters have moved closer for an emphasis of this region. The same observation stands for the filters in the 2.5 kHz region. More noticeable is the shift toward lower value of the center frequency to accentuate the 4kHz region, where F3 formants are located.

It seems that the spacing of the filters have been modified for a better resolution of the formants independently of the cepstral representation. This may explain the better performance achieved by this model for both representations.

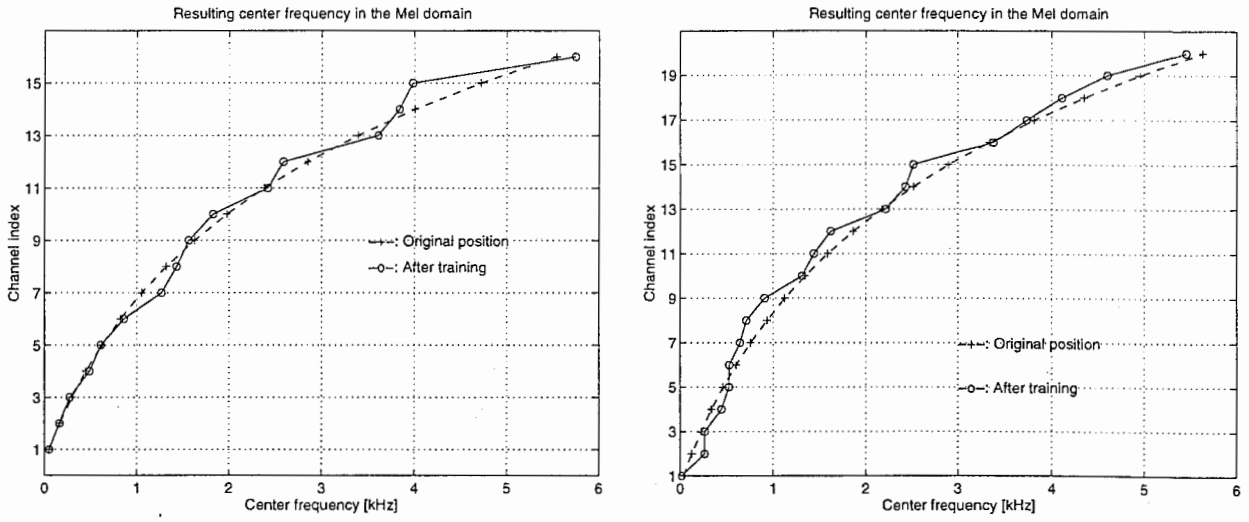

Figure 10: **Left**: Resulting Mel-scale based filter bank center optimization task for L= 15 and Q = 15. **Right**: Resulting Mel-scale based filter bank center optimization task for L= 20 and Q = 10. Each channel is plotted versus its modified center.

6.3.2 Bandwidth optimization task

Optimization of the filter bank through the parameter β_c will maximize or minimize the contribution of those regions which may be important for minimum error. Note that, the center frequency are *fixed*, thus optimization is done on pre-chosen perceptually based regions. For better visualization of the bandwidth, we introduce the notion of "corresponding bandwidth value" (CBW) of a channel. The CBW of a channel is the interval in the *Hertz* domain between the two frequencies whose weighting is half of the weighting at the center frequency of the channel.

Figure 11 shows the CBW versus the center frequencies of the filter bank model for the two representations. In general, most filters above the 3 kHz region have seen a significant change of their bandwidth value. In contrast to the original model, the obtained bandwidth values are not a monotonic function of the center frequencies. Note that a decrease in CBW value signifies that the filter tend to select fewer frequencies in the channel.

In representation I, the 13th and the 14th filters have recorded a decrease of the CBW of more than 200Hz but compensated by the 15th and 16th filters whose bandwidths have recorded an increase of their CBW of more than 300Hz.

In representation II, the 15th filter have increased its CBW value of almost 2kHz, fully covering the 3kHz region, in detriment of the surrounding filters.

These models leads to good performance on the training set (12.0%) while showing a relatively poor generalization on the testing set:14.9% for representation I and 14.6% for representation II.

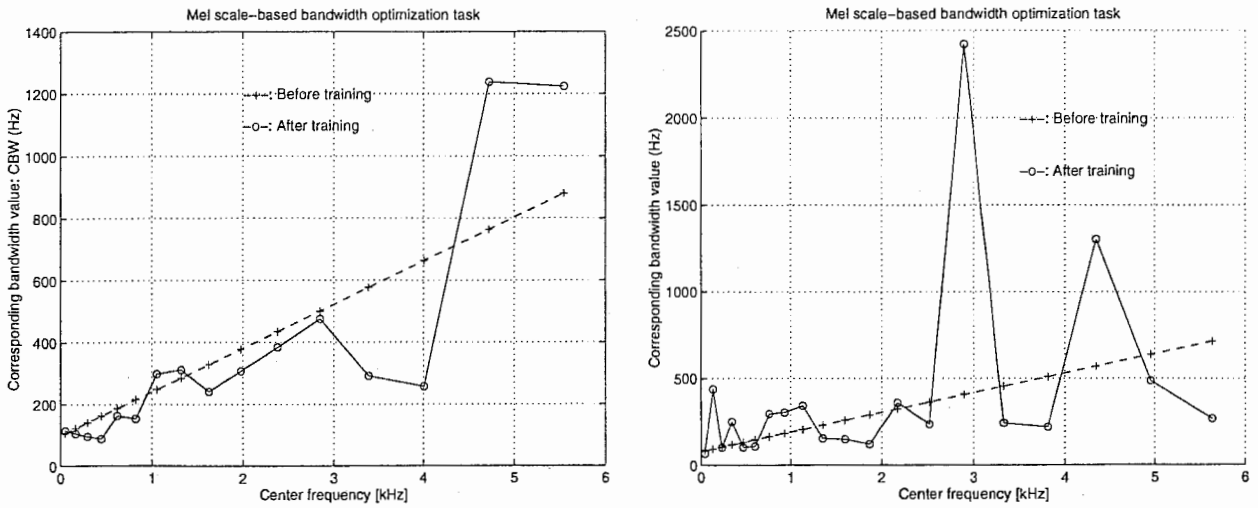

Figure 11: **Left**: Resulting Mel-scale based filter bank bandwidth optimization task for Q = 16 and L = 15. **Right**: Resulting Mel-scale based filter bank bandwidth optimization task for Q = 20 and L = 10. The CBW is plotted versus center frequency in each channel.

6.3.3 Gain optimization task

The gain optimization puts an emphasis on those outputs relevant to minimum error.

Figure 12 shows the value of the gain (the parameter φ_c) versus the center frequency in each channel for both representations. It can be seen that both representations show a similar shape although the value range is different: both accentuate and de-accentuate similar regions of the spectrum.

Filters in the region below 1.5 kHz frequency region have shown a small decrease of the gain parameter. While the filters above the 4kHz have shown significant increase of their gain value. Note that only high value of the gain should have a noticeable effect on the performance of the system because of the use of the logarithm in calculating the log energies which tends to attenuate the small modifications of the gain value. The sharp increase of the gain in the region above 4kHz may be similar to a pre-emphasis which accentuates higher frequencies.

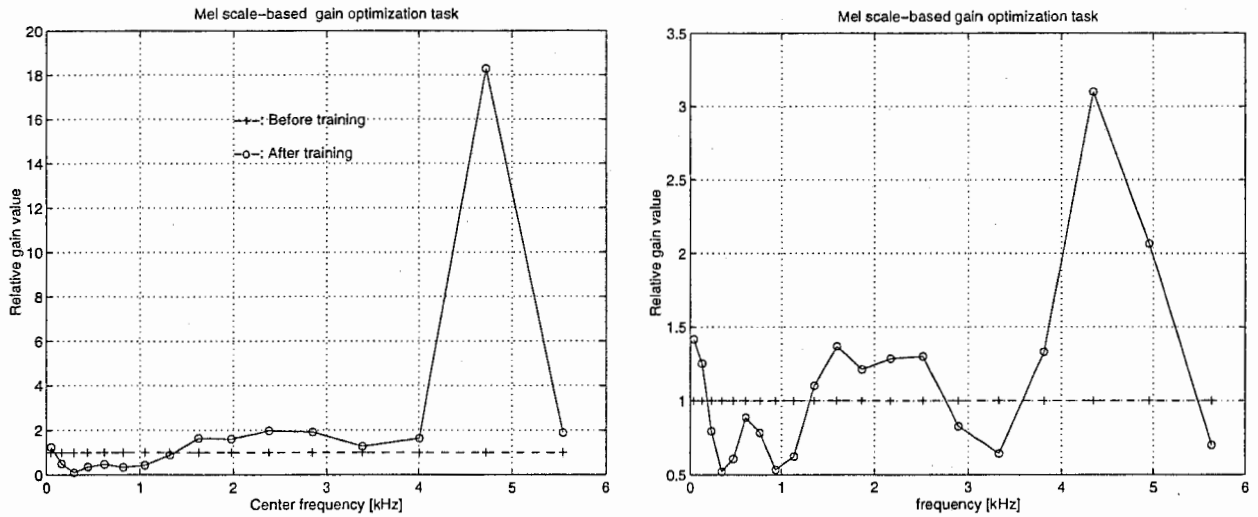

Figure 12: **Left**: Resulting Mel-scale based filter bank gain optimization task for Q= 16, L = 15. **Right**: Resulting Mel-scale based filter bank gain optimization task for Q = 20, L = 10. The valure of φ_c are plotted against the center frequencies in the channels.

6.3.4 Weighting optimization task

As said, before each parameter $\theta_c(f)$ can be optimized independently without keeping the Gaussian constraint.

As can be seen from Fig. 13, the resulting model did not keep the Gaussian form of the frequency response. One can notice that within certain channel, several optimum frequency have been emphasized. The fewer number of channel in representation I has resulted in filters with higher energies than filters in representation II. In representation I, filters in the region below 1.5kHz have not shown any significant increase of the weighting.

The elections of multiple optimum frequencies within a channel may, in a sense, be compared to the search for the *the best* within-channel optimum frequency as in center optimization case. However. the higher number of free parameters may inhibit generalization.

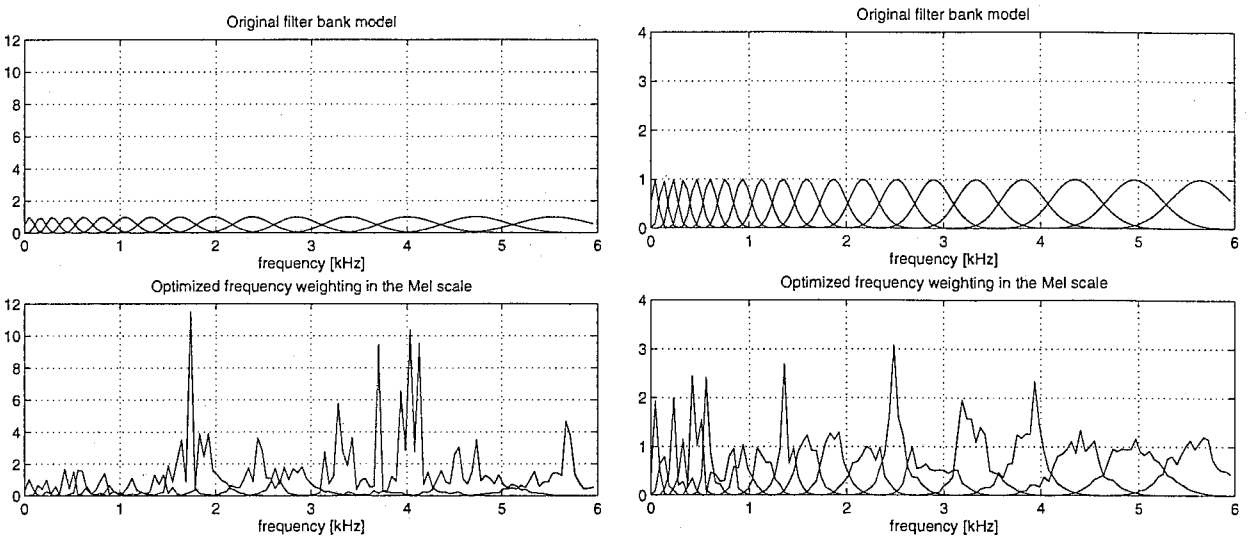

Figure 13: Resulting Mel-scale based filter bank weighting optimization. **Left**: Resulting Mel-scale based filter bank weighting optimization task for Q= 16, L = 15. **Right**: Resulting Mel-scale based filter bank weighting optimization task for Q = 20, L = 10.

6.3.5 Simultaneous optimization of center frequency, bandwidth and gain

A global optimization of the filter bank maybe achieved by optimizing at the same time, the center, the bandwidth, the gain. This enables an efficient interaction between the three parameters while reducing the degree of liberty which may lead to the over training dilemma.

Hence, in representation I, the best results on the testing set have been achieved so far by this model: 14.2% on the testing set with similar result in the training set with the center optimization case. When looking at the DFE-adjusted models of Figure 14, one can see that the center frequencies have more contribution in the modification of the filter bank. Thus, the spacing of the filter clearly puts an emphasis on well-defined region: the 1-2 kHz region (F2 domain), the 2.5 kHz region covered by 11th filter (F2 region) and 12th filter and the 3.5 -4 kHz region (F3 domain). One should note that , a similar filter spacing was achieved in the center optimization task. The bandwidth and the gain have remained relatively stable.

It seems that, when optimizing all the parameters at the same time, DFE focuses on finding the optimal spacing for minimum error by center frequency modifications and put less emphasis on bandwidth and gain modification *when using the Mel scale.*

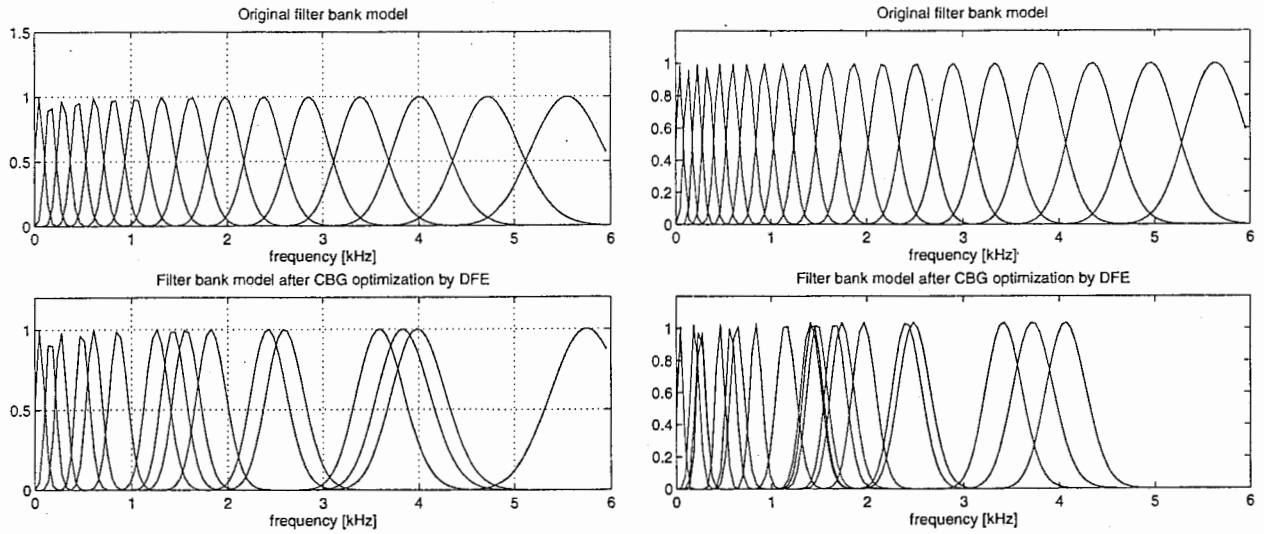

Figure 14: Resulting Mel-scale based filter bank CBG optimization task. **Left**: Resulting Mel-scale based filter bank CBG optimization task for Q= 16, L = 15. **Right**: Resulting Mel-scale based filter bank CBG optimization task for Q = 20, L = 10.

6.4 Using DFCC on standard recognizer

One may wonder how DFCC behaved in classically trained systems (i.e non-DFE trained system). Hence, we performed an experiment in which the baseline recognizer, used in the previous experiment, was trained using CBG-DFCC on the same vowels recognition task. CBG-DFCC were generated d using the filter bank obtained from the simultaneous optimization of center frequency, bandwidth and gain. Figure 15 gives an idea of MFCC and CBG-DFCC representations of the same utterance. DFCC representation seems to show a more stable representation across the frames than the MFCC.

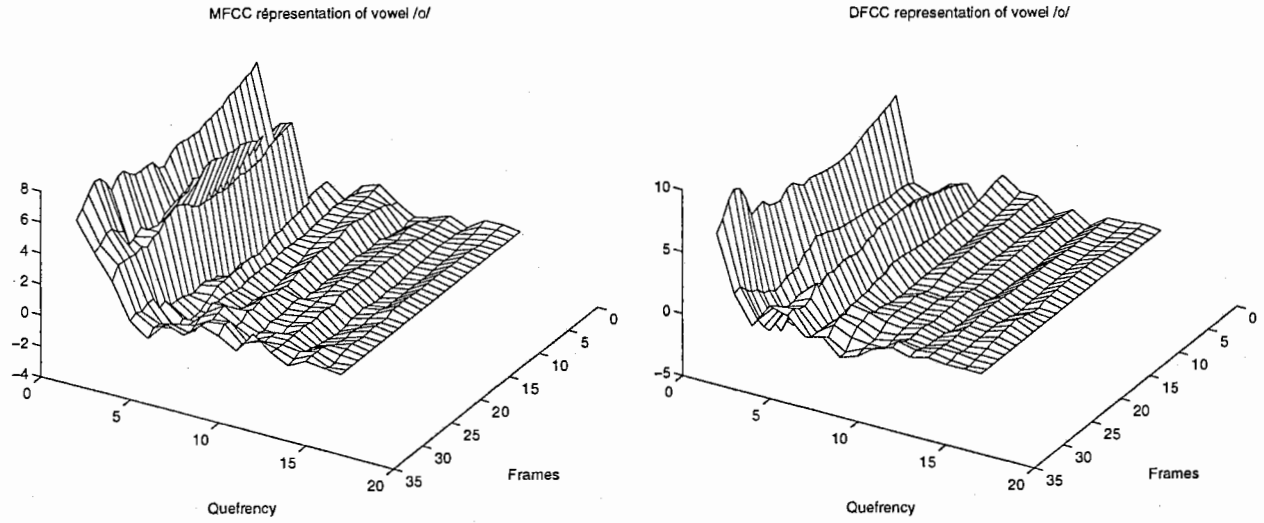

Figure 15: Cepstral representation of an utterance of vowel /o/ using MFCC and CBG-DFCC. **Left**: MFCC representation.**Right**: CBG-DFCC representation.

Table 3 illustrates the performance of CBG-DFCC on MCE/GPD-trained standard recognizers. The recognizer here is the same as the baseline (one prototype/class) where the input pattern are CBG-DFCC without adaptation of the filter bank parameters (static-CBG-DFCC).

Static-CBG-DFCC	train	test
	12.4	14.8

Table 3: Performances of CBG-DFCC in the vowels recognition task using a classical MCE/GPD trained system.

It can be seen that Static-CBG-DFCC performs better than MFCC on both the training and testing set. Moreover, as can been seen from Table 1, it achieved a similar performance on the training set with the CBG-DFCC (12.4%) but with a poorer generalization (14.8%) as compared to CBG-DFCC (14.2%). These results suggest that using DFE-extracted features on classical recognizer may increase the performance although a DFE-based recognizer enables a better integration of the feature extractor and classifier.

6.5 Full optimization and selective optimization

A DFE-based speech recognizer requires an accurate tuning of parameters for optimality. In classical MCE/GPD-based optimization scheme, the decreasing learning rate ϵ_τ is chosen as

$$\epsilon_\tau = \epsilon\left(1 - \frac{\tau}{\mathcal{N}}\right)$$

where $\mathcal{N}$ is total number of iterations and ϵ is the learning at the beginning of the training set. Needless to say, ϵ_τ determines the performance of the system, given a fixed number of iterations $\mathcal{N}$.

In previous sections, we have proposed the use of the selective optimization for an efficient adaptability and interactivity of the two modules. Full optimization (use of the same learning rate for feature extractor and classifier) is the standard method. How the two methods of optimization behave when adapting center frequency, bandwidth, gain at the same time, is shown in Figure 16. It can be seen that, the selective optimization realizes better performance (i.e lower error rate) in both the training set and the testing while showing a more stable curve across a wide range of the learning rate ratio. The solution consists of finding the accurate learning rate ratio between the feature extractor an the classifier. However, the performance depends on the task as well as the optimized parameter. In fact, full optimization can viewed as special case of selective optimization in where the classifier 's and feature extractor's learning rate ratio is equal to 1.

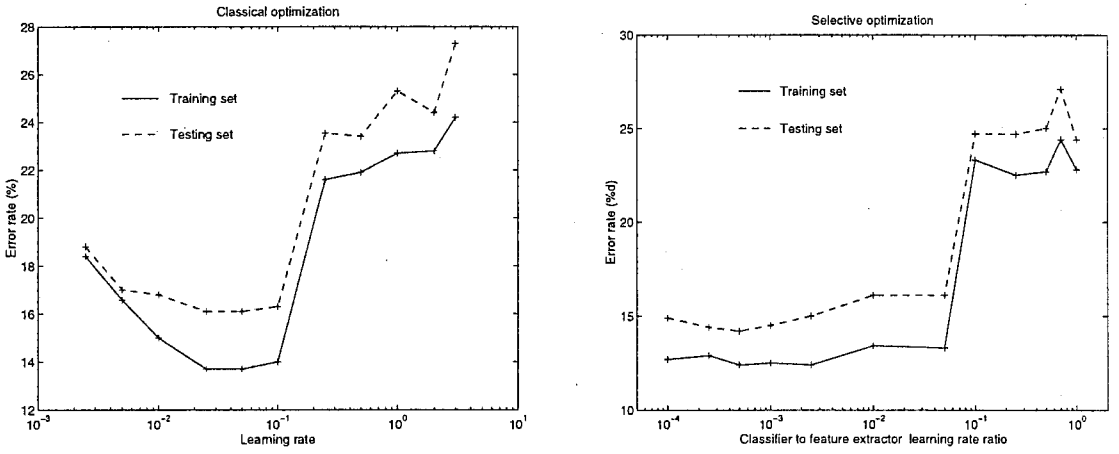

Figure 16: Classical MCE/GPD optimization schemes and Selective optimization. The **left**: performances of the system versus the learning rate when adapting all parameters using the same learning rate for all parameters. **Right**: performance of the system versus the ratio between classifier learning rate and feature extractor learning rate.

7 Conclusion

DFE implementation was described along with an application to cepstrum optimization. The DFE optimization scheme which includes the choice of the learning rate, the use of a selective optimization algorithm according to the type of parameter and an accurate application of the chain rule when optimizing the feature extractor, was shown in detail. For illustration, the method was applied to cepstrum optimization using the prototype-based minimum error classifier. DFE-cepstrum were shown to be more robust than classical Mel-based filter bank cepstral parameters.

References

[1] J. Chang C. Darken and J. Moody. Learning rates schedule for faster stochastic gradient search. In *nnsp*, volume II, pages 3–12, 1992.

[2] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 40(12):3043–3054, Dec 1992.

[3] A. Biem E. McDermott and S. Katagiri. Discriminative filter bank model for speech recognition. In *Eurospeech 95*, volume 1, pages 545–548, Oct 1995.

[4] E. McDermott and S. Katagiri. Prototype-based minimum classification error/ generalized probabilistic descent for various speech units. *Computer Speech and Language*, 8(8):351–368, Oct. 1994.

[5] E. Zwicker and E. Terhardt. Analytical expressions for critical band-band rate and critical bandwidth as a function of frequency. *Journal of the Acoustical Society of America*, 68(5):1523–1525, Dec. 1980.

A Appendix

Filter bank optimization by DFE

For a pattern $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_t \ldots, \mathbf{x}_T\}$ where each $\mathbf{x}_t = [x_{t,1}, \ldots, x_{t,f}, \ldots, x_{t,F}]^T$ is a power-spectrum frame and $\mathbf{e}_t = [e_{t,1}, \ldots, e_{t,c}, \ldots, , e_{t,Q}]^T$, is the corresponding vector of log energies. f represents the FFT bins. We have

$$e_{t,c} = \log_{10}\left(\sum_{f \in B_c} \theta_c(f) x_{t,f}\right),$$

for $c = 1, \ldots, Q$, and

$$\theta_c(f) = \varphi_c \exp\left(-\beta_c \left(\tilde{\gamma}_c - p(f)\right)^2\right),$$

for $c = 1, \ldots, Q$ where $\theta_c(f)$ is the weight at frequency f provided by the c-th filter.

If ϕ is any filter bank parameter, the adaptation rule is

$$\phi[\tau + 1] \quad = \quad \exp\left(\log(\phi[\tau]) - \rho_\tau \mathbf{U} 2 \delta\overline{\phi}\right) \tag{57}$$

with

$$\delta\overline{\phi} = \begin{cases} \displaystyle\sum_{t=1}^{\mathcal{T}} \sum_{i=1}^{L} \sum_{c=1}^{Q} \cos\left(\frac{i\pi}{Q}(c - 0.5)\right) \mathcal{I}_{t,i} \mathcal{O}_{t,c} & \text{if cepstrum transformation of size } L. \\ \displaystyle\sum_{t=1}^{\mathcal{T}} \sum_{c=1}^{Q} \mathcal{I}_{t,c} \mathcal{O}_{t,c} & \text{if no cepstrum transformation.} \end{cases} \tag{58}$$

- $\mathcal{I}_{t,i}$ is the classifier's i-th input derivative of the t-th feature-vector. For PBMEC, we have

$$\mathcal{I}_{t,i} = -\sum_{k=1}^{K} \sum_{m=1}^{M} \delta r_{\psi_{k,\lambda}(t), \psi_{k,s}(t), m, i}.$$

- $\mathcal{O}_{t,c}$ is the feature extractor c-th output derivative of the t-th feature-vector (the derivative of the filter bank output in channel c versus the filter bank parameter $\overline{\phi}$). $\mathcal{O}_{t,c}$ *does not depend* on the classifier.

$$\mathcal{O}_{t,c} = \frac{\partial e_{t,c}}{\partial \overline{\phi}} = \sum_{f=1}^{F} \upsilon_{t,c,f} \xi_{c,f} \tag{59}$$

with

$$\upsilon_{t,c,f} = \frac{x_{t,f}}{\log(10) \mathcal{E}_c(\mathbf{x}_t)} \quad \text{and} \quad \mathcal{E}_c(\mathbf{x}_t) = \sum_{f \in B_c} \theta_c(f) x_{t,f} = \exp_{10}(e_{t,c}), \tag{60}$$

and

$$\xi_{c,f} = \begin{cases} \chi(c, \hat{c})\, \chi(f, \hat{f})\, \theta_c(f) & \text{if } \phi \text{ is the weight } \theta_{\hat{c}}(\hat{f}). \\ -2\beta_c \left(\tilde{\gamma}_c - p(f)\right) \theta_c(f)\, \tilde{\gamma}_c\, \chi(c, \hat{c}) & \text{if } \phi \text{ is the center frequency } \tilde{\gamma}_{\hat{c}} \text{ of channel } \hat{c} \\ & \text{as expressed in the perceptual domain.} \\ -\beta_c \left(\tilde{\gamma}_c - p(f)\right)^2 \theta_c(f)\, \chi(c, \hat{c}) & \text{if } \phi \text{ is the is the parameter } \beta_{\hat{c}} \\ & \text{controlling the bandwidth of channel } \hat{c}. \\ \theta_c(f)\, \chi(c, \hat{c}) & \text{if } \phi \text{ is parameter } \varphi_{\hat{c}} \text{ of channel } \hat{c}\text{-th, which controls the gain.} \end{cases} \tag{61}$$

where

$$\chi(a, b) = \begin{cases} 0 & \text{if } a \neq b \\ 1 & \text{otherwise} \end{cases} \tag{62}$$