TR－H－059

0045

A Minimum Error Approach to Spotting-Based Speech Recognition

Takashi KOMORI

Shigeru KATAGIRI

1994. 2. 24

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　☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

A Minimum Error Approach to Spotting-Based Speech Recognition

Takashi KOMORI and Shigeru KATAGIRI***

* ATR Human Information Processing Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02 Japan

** ATR Interpreting Telecommunications Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-02 Japan

Abstract

Word spotting is a fundamental approach to recognition/understanding of natural, spontaneous spoken language. An overall spotting system, comprising word models and decision thresholds, primarily needs to be optimized to minimize all spotting errors. However, in most conventional spotting systems, the word models and the thresholds are separately and heuristically designed: There has not necessarily been a theoretical basis that has allowed one to design an overall system consistently. This paper introduces a novel approach to word spotting, by proposing a new design method called Minimum SPotting Error learning (MSPE). MSPE is conceptually based on a recent discriminative learning theory, i.e., the Minimum Classification Error learning (MCE)/Generalized Probabilistic Descent method (GPD); it features a rigorous framework for minimizing spotting error objectives. MSPE can be used in a wide range of spotting pattern applications, such as spoken phonemes, written characters as well as spoken words. Experimental results for a Japanese phoneme spotting task clearly demonstrate the promising future of the proposed approach.

1 Introduction

Natural conversation utterances, often including such common phenomena as hesitation, repetition, as well as correction, usually do not observe grammar rules and are difficult to recognize in a precise, word-by-word manner. Therefore, there has long been research concern about the speech understanding aimed at obtaining the meanings of utterances instead of transcribing all acoustical events.

Spotting, especially keyword spotting, is a fundamental technique for this type of speech understanding. A *spotter* (spotting system) attempts to selectively recognize preset target acoustical events such as keywords which are necessary to understand given utterances correctly, by shifting short, spotting decision segments over given utterances. This nature of spotting intrinsically has a risk of producing two kinds of categorical errors: 1) *mis-detection* (deletion error), i.e., failing to detect a target event that actually exists, and 2) *false alarm* (addition error), i.e., mistakenly detecting a target event that actually does not exist. Therefore, spotter design is complex, compared with that for a simple classification of isolated (pre-segmented) utterance patterns; it must primarily consider such complex features in spotting as the localization of spotting and error categorization.

A spotter usually consists of two kinds of adjustable system parameters: 1) event models and 2) decision thresholds. One goal of spotter design must then be to determine the state of all of these system parameters to reduce the spotting errors as much as possible. However, perhaps due to the lack of a theoretical basis, most conventional spotters have been designed in a scratch-of-heuristics manner, with no direct attempt at this error reduction; in particular, event models have been designed to achieve minimum distortion (maximum likelihood) and thresholds have been selected empirically through a trail-and-error repetition of spotting experiments. Obviously then there is no consistency between the actual design procedure and the design goal of minimizing the spotting errors.

To overcome this problem, we propose in this paper a novel design method, called the Minimum SPotting Error learning (MSPE), which allows one to train all of the adjustable system parameters in a manner consistent with the spotting error minimization. A key concept of MSPE is to embed the entire spotting process including an intrinsically-discontinuous operation of counting spotting errors, in a *smooth* (at least first differentiable in adjustable system parameters) functional form. This concept is taken from a recent discriminative learning theory, i.e., the Minimum Classification Error Learning/Generalized Probabilistic Descent Method (MCE/GPD), which has recently been shown to be useful in various pattern classification tasks [1]-[6]. Nevertheless, MSPE is clearly distinct from MCE/GPD in the following two points: In MSPE we newly formalize the above-cited, complicated error evaluation process and MSPE provides an innovated departure from the conventional, scratch-of-heuristics way of designing spotters.

The paper is organized as follows. In the next section, we present some fundamentals for MSPE formalization. In particular, the definition of the cost function being directly related to spotting error measurement is described in detail. In Section 3, we present practical implementation examples for Dynamic Time Warping (DTW) incorporating N-best concepts such as N-best warping path and N-best reference. In Section 4, we attempt to show the utility of the proposed method, by reporting experimental results for Japanese consonant spotting using a simple DTW-based spotter. Section 5 summarizes the paper.

2 Minimum Spotting Error Learning

Spotting can be considered a more general and practical version of classification which is the decision process to assign one of many possible class labels to a given pre-segmented event: It is the process to assign one of the two classes, mis-detection or false alarm, to an event that is automatically detected by shifting a decision position over a given sample. A key of the MSPE formalization is thus to define this complex decision process as a smooth functional form that appropriately reflects the spotting accuracy. Similar to a general spotting concept, MSPE can handle any reasonable sample such as a continuous speech utterance, and any reasonable event such as spoken words or written letters. For clarity of presentation, we shall describe our formalization by focusing on the distance-based spotting of words from continuously-spoken utterances.

2.1 Fundamentals for formalization

We consider a C-class word spotting task; suffix c is used for the class index. A speech utterance $\boldsymbol{x}$ is represented as a variable but finite length sequence of acoustic feature vectors, each of fixed-dimension S; i.e., $\boldsymbol{x} = \{\boldsymbol{x}_1, \ \boldsymbol{x}_2, \ \cdots, \ \boldsymbol{x}_I\}$ where the i-th feature vector (or the i-th frame) of $\boldsymbol{x}$ is $\boldsymbol{x}_i = [x_{i1} \ x_{i2} \ \cdots \ x_{iS}]^T$. A spotter consists of parameters $\boldsymbol{\Lambda} = \{\Lambda^c\}_{c=1}^{C} = \{\boldsymbol{\lambda}^c, h^c\}_{c=1}^{C}$, where $\boldsymbol{\lambda}^c$ is a word model consisting of templates, each having the same representation as the utterance, and h^c is a scalar-variable decision threshold.

The spotting for one class can be independent of that for another class. This feature should be quite useful in executing spotting/design runs because it allows one to use parallel and distributed computations. In the MSPE formalization, we thus use a class-by-class design strategy. In light of this, we focus our discussion in the following on the spotting of a class c word. It is worth noting that, if necessary, one can easily change this class-independency without adversely affecting the MSPE concept.

Since a word is originally a segmental (durational) pattern, spotting is fundamentally performed by comparing an input speech segment and these word models. Clearly, a spotting rule must make such a segment-based decision. We thus specially use in this paper the following decision rule, i.e., introduce a *discriminant function*, denoted by $g_i(\boldsymbol{x}; \boldsymbol{\lambda}^c)$, to indicate the degree to which a class c word exists at the i-th frame of the input $\boldsymbol{x}$: For a class c, if

$$\min_{i \in \Omega_k} g_i(\boldsymbol{x}; \boldsymbol{\lambda}^c) < h^c, \tag{1}$$

one word exists in $\Omega_k = \{i \,|\, p(k) \leq i \leq q(k)\}$ with $p(k)$ and $q(k)$ being the first and final frames, respectively; otherwise, no word exists in Ω_k. The Ω_k is the k-th segment of $\boldsymbol{x}$, called the *k-th block* , and is fundamentally preset by a correct word location. Figure 1 illustrates this spotting decision. Two words to be spotted are given; w_1 and w_2. In the top of the figure, a hatched region indicates the given word segment. The blocks are determined based on the locations of these words. See the block sequence (a) of the figure. There are two kinds of blocks in this sequence: a shaded one and a white one. The shaded blocks correspond to word segments to be spotted and the white blocks correspond to segments not to be spotted. We refer to the shaded block as an *S-block* and the white block as a *non-S-block*. Note that the S-block is set around the last end of a given word segment. This feature is related to a reason why we bother to introduce the blocks in the spotting decision, not making a direct use of the word segments. Detailed discussions about the block definition will be given later.

In the spotting (testing) stage, we make the spotting decision based on (1). In the design stage, we must thus train the spotter parameters, on the premise that we use (1), so that we can increase spotting accuracy as much as possible. Therefore, MSPE is formalized to emulate the above decision process in a smooth function and provides a gradient search-based design algorithm that optimizes the state of the parameters. The formalization, like MCE/GPD, basically consists of three steps.

In the first step, we embody the discriminant function. The form of this function is essentially determined by the selection of the word model and in our case using templates we define $g_i(\boldsymbol{x}; \boldsymbol{\lambda}^c)$ as a generalized distance between $\boldsymbol{\lambda}^c$ and a segment of $\boldsymbol{x}$ with the i-th frame being the final end of DTW matching.

In the second step, we introduce a block-based *spotting measure* $d_k^c(\boldsymbol{x}; \Lambda)$ to indicate the degree to which a class c word is spotted in Ω_k. Among many possibilities, we specially define the following simple example:

$$d_k^c(\boldsymbol{x}; \Lambda) = h^c - \ln \left\{ \frac{1}{I_k} \sum_{i=p(k)}^{q(k)} \exp(g_i(\boldsymbol{x}; \boldsymbol{\lambda}^c))^{-\xi} \right\}^{-1/\xi}, \tag{2}$$

where I_k is the frame length of Ω_k ($I_k = q(k) - p(k) + 1$) and ξ is a positive number. So that the reader understands our formalization concept correctly, let us consider the extreme case that ξ goes to ∞; the spotting measure becomes simpler as

$$d_k^c(\boldsymbol{x}; \Lambda) \approx h^c - \min_{i \in \Omega_k} g_i(\boldsymbol{x}; \boldsymbol{\lambda}^c). \tag{3}$$

Both (2) and (3) clearly show that the spotting measure emulates the above spotting decision in a scalar measurement. A positive value of $d_k^c(\boldsymbol{x};\Lambda)$ implies that the corresponding word exists in Ω_k; a negative value implies that the word does not exist in Ω_k. Note that this measure simulates the spotting decision in a scalar value comparison. As in MCE/GPD, the use of the L_p-norm form is mainly motivated by the following two points: 1) achieving the function's smoothness and 2) making the decision softer (more robust).

In the final step of the formlization, we define a loss, denoted by $\ell_k^c(\boldsymbol{x};\Lambda)$, that directly reflects the spotting result and is controlled by block-based supervision. Taking into account the possibility that there can be two different kinds of spotting errors, mis-detection and false alarm, the loss is defined as

$$\ell_k^c(\boldsymbol{x};\Lambda) = \tilde{\ell}_k^c(d_k^c(\boldsymbol{x};\Lambda)) \tag{4}$$

with

$$\tilde{\ell}_k^c(y) = \begin{cases} \ell^*(y;\alpha_t^c,\beta_t^c) & \text{if } V^c(k)=1 \\ \gamma^c\ell^*(y;\alpha_f^c,\beta_f^c) & \text{if } V^c(k)=0 \end{cases}, \tag{5}$$

where $V^c(k)$ is a supervising signal that indicates one (1) if the class c word actually exists in Ω_k, zero (0) otherwise, α_t^c and α_f^c are positive and negative real numbers, respectively, γ^c is a weight controlling the balance in possibility between mis-detection and false alarm, and $\ell^*(y;\alpha,\beta)$ is a smooth step function such as

$$\ell^*(y;\alpha,\beta) = \frac{1}{1+\exp(\alpha(y-\beta))}. \tag{6}$$

The loss is a function of the spotting result and the supervising signal, and straightforwardly approximates the number of block-based spotting errors in a flexible manner based on γ^c: It approximately takes 1) one (1) for one mis-detection, 2) γ^c for one false alarm, and 3) zero (0) for correct spotting. When γ^c is set to one (1), all of the errors are treated as even. Since the mis-detection is fatal compared with the false alarm, γ^c may sometimes be set less than one (1). By taking account of the significance of these error conditions, one can easily control the design process by changing γ^c.

Since a spotting result must in principle be evaluated over the entire set of possible input utterances, we use in the design stage the following expectation loss consisting of the above loss for an individual decision:

$$L(\Lambda) = E\left[\,\ell_k^c(\boldsymbol{x};\Lambda)\,\right]. \tag{7}$$

Minimizing $L(\Lambda)$ will lead to an approximation of the minimum spotting error situation. This minimization can be done by any reasonable, mathematically-proven search algorithm such as the steepest descent method or simulated annealing. Again, based on the MCE/GPD concept, we specially use in our formalization the adaptive training based on GPD that guarantees the probabilistic convergence. It can easily be shown that the following adjustment lets Λ converge to at least a locally-optimal status in a probabilistic sense:

$$\Lambda(t+1) = \Lambda(t) - \epsilon_t U \sum_{c=1}^{C} \sum_{k=1}^{K} \nabla \ell_k^c \left(\boldsymbol{x}(t); \Lambda(t)\right), \tag{8}$$

where $\boldsymbol{x}(t)$ and $\Lambda(t)$ denote an input token and the parameter status at the t- th learning iteration stage, respectively, ϵ_t is a small positive learning factor which satisfies the stochastic approximation constraints $\sum_{t=1}^{\infty} \epsilon_t \to \infty$ and $\sum_{t=1}^{\infty} {\epsilon_t}^2 < \infty$ [1],[7], K is the number of segments set up in $\boldsymbol{x}(t)$, and U is a positive-definite matrix. Note here that this learning updates all of the trainable spotter parameters, i.e., both the word models and the thresholds, under the single criterion of smooth spotting error counts.

2.2 Block setting

In principle, one spotting decision is required to correspond to each block (each word segment). Nevertheless, in reality, as illustrated in Figure 1, the value of the discriminant function fluctuates continuously and unsteadily due to natural statistical variation, and thus it is rather difficult to meet this requirement; therefore, we obviously need a careful way of setting the blocks, or a more precise way of spotting decisions than the above rule, so that we can handle realistic decision situations as appropriately as possible. There is a clearer need of this consideration in the training stage: the block setting determines spotting accuracy.

Since the (spoken) word spotting is by nature a segment-based decision and our spotter uses DTW-based distance accumulation, the spotting decision is naturally done around the last end of a true word segment. In this sense, S-blocks should be set as short segments, one around each true last end; other segments should be non-S-blocks. Moreover, since the rule is based on the minimum-search operation, a correct spotting position (frame) should basically correspond to a local minimum lower than h^c of the discriminant function. Therefore, in making the decisions, we can naturally focus on local minima, each called an *M-frame*, of the function. For later discussions, we specially refer to the M-frame that indicates a lower discriminant function value than h^c as the *S-frame*. In light of this, let us elaborate here on possible decision situations at a true last end of a word. Consider two exemplars, last end frames i_{e1} and i_{e2}. Ideally, a correct M-frame, or an S-frame, should appear at each of these frames. Nevertheless, this pin-point match is obviously unrealistic. It is thus natural to consider a preset, fixed-length segment around the last end of the word as a segment in which a correct M-frame should appear. We refer to this segment indicating a correct spotting location as a *C-segment*;

we denote the C-segment for i_e by $[i_e - \kappa_1, i_e + \kappa_2]$ with κ_1 and κ_2 being positive constants. We also refer to a segment, other than the C-segment, that should inhibit spotting as the *I-segment*. Figure 1 illustrates an example consisting of two C-segments, one for the i_{e1} frame and one for i_{e2}. Then there may be the following manifold of decision situations:

1. There is no S-frame in a C-segment; this indicates one misdetection (See the C-segment for i_{e2}).
2. There is more than one S-frame in a C-segment; this suggests one correct spotting for the most plausive (in terms of the discriminant function value) S-frame among them and false alarms for the others (See the C-segment for i_{e1}).
3. There is at least one S-frame in an I-segment; this implies that all of these S-frames cause false alarms (See the I-segments).
4. Each of all the situations other than the above three indicates one correct spotting (no error).

An appropriate block setting should reflect the above variety. On this premise, we specially consider in the paper two methods of block setting.

The first method is simply to assign blocks to either C-segments or I-segments; an S-block is set at a C-segment and a non-S-block is set at an I-segment. Figure 1(a) illustrates this method. Note that the concept of *min-max error* lies in this method. Even if there is more than one S-frame in one block, the dominant (lower discriminant function value) S-frames are primarily used for training. This feature of the method is clearer in the extreme case of letting ξ go to ∞; only the most dominant (lowest discriminant function value, or most undesirable) S-frame is used for training (See the first two segments of Figure 1(a)). We refer to this method as the *min-max* method.

The second method is to fundamentally set an S-block around one dominant S-frame in a C-segment; an important point here is that the S-block is set to the segment sandwiched by two frames with local maxima of the discriminant function, each next to the inside S-frame, so that the S-block includes only one S-frame. For convenience, we refer to the frame with local maxima of the discriminant function as the *P-frame*. To deal with possible situations appropriately, the method is decomposed to the following sequential procedures:

1. Compute all local minima (S-frames) and local maxima of the discriminant function.
2. If there is one S-frame in a C-segment, assign an S-block to the segment that includes this S-frame and is bounded by two adjacent P-frames.

3. If there is more than one S-frame in a C-segment, assign an S-block to the segment that includes the S-frame with the lowest discriminant function value and is bounded by two adjacent P-frames(See the fourth block of Figure 1(b)).

4. If there is no S-frame in a C-segment, assign an S-block to the C-segment (See the sixth block of Figure 1(b)).

5. Set a non-S-block to each of all remaining S-frames. In this procedure, the block boundaries set in the above 4th rule are used with priority over the boundaries due to the S-frames in this rule (See the fifth and seventh blocks of Figure 1(b)).

We call this method the *S-frame-based method*. Compared with the min-max method, this S-frame-based method seems to be more direct in counting spotting errors: a spotting decision is fundamentally done at every S-frame. However, since the blocks that should essentially be determined only by training samples are set based on training results, in other words, the training target is affected by the training result, this method is inadequate in a mathematical sense.

Experimental evaluation for these two methods will be given in Section 4.

3 Implementation Details of MSPE Training to DTW-Based Spotter

In this section, we describe implementation details of the MSPE training to a DTW-based spotter which we assumed to use in Section 2. As cited before, our formalization is of the class-by-class strategy, and thus we consider one class spotting, removing the class index c from notations in this section. Implementation examples for other types of spotter structures are given in the Appendix.

In order to show a general means of implementation, we assume to use a multi-template spotter structure. Our word model is then assumed to be composed of a set of B *sub-models*, each consisting of a template (acoustic feature vector sequence) and its corresponding set of covariance matrices; $\boldsymbol{\lambda} = \{\boldsymbol{\lambda}_b\}_{b=1}^{B} = \{\boldsymbol{R}_b, \Sigma_b\}_{b=1}^{B}$, where the b-th sub-model, $\boldsymbol{\lambda}_b$, is a pair comprising template $\boldsymbol{R}_b$ and its corresponding set of covariance matrices Σ_b; $\boldsymbol{R}_b = \{\boldsymbol{r}_{bj}\}_{j=1}^{J_b}$ with J_b being the length of the template; $\boldsymbol{r}_{bj}$ is the j-th S-dimensional acoustic feature vector, $[r_{bj1}\ r_{bj2}\ \cdots\ r_{bjS}]^T$; $\Sigma_b = \{\Sigma_{bj}\}_{j=1}^{J_b}$; and Σ_{bj} is the $S \times S$ covariance matrix that corresponds to $\boldsymbol{r}_{bj}$.

There are many ways to define the discriminant function for this spotter structure. By way of example, we consider a starting-end-free, multi-template, multi-path, DTW-based spotter of which the discriminant function is a *word distance* calculated based on the so-called

N-best (minimum distortion) concept; the word distance consists of M *sub-model distances* ($M \leq B$), each calculated between a segment of an input utterance and one of the M-best sub-models. The sub-model distance consists of N *path distances*, each calculated by accumulating a *frame distance* along one of the N-best Dynamic Programming (DP) matching paths for every corresponding sub-model. The frame distance is calculated between frames, i.e., one frame of an input utterance and its corresponding (DP matching-determined) template frame. Among many possibilities to measure the frame distance, we specially use a general, likelihood-based distance. We assume that the sub-model index b indicates the order of its closeness, in terms of the above distance measure, to the input segment; e.g., $\boldsymbol{\lambda}_1$ is the closest sub-model. Precisely, the discriminant function is then defined in the following step-by-step fashion: First,

$$g_i(\boldsymbol{x};\boldsymbol{\lambda}) = \ln\left\{\frac{1}{\Phi}\sum_{\phi=1}^{\Phi}\exp(D_i^{\phi}(\boldsymbol{x};\boldsymbol{\lambda}))^{-\eta}\right\}^{-1/\eta}, \tag{9}$$

where Φ is the total number of permutations for choosing M sub-models from all possible B sub-models, i.e., $\Phi = B!/(B-M)!$; $D_i^{\phi}(\boldsymbol{x};\boldsymbol{\lambda})$ is the *combined sub-model distance* computed over the M sub-models in the ϕ-th best permutation; and η is a positive constant. Second,

$$D_i^{\phi}(\boldsymbol{x};\boldsymbol{\lambda}) = \sum_{m=1}^{M}\rho_m D_i(\boldsymbol{x},\boldsymbol{r}_{b(\phi,m)};\Sigma_{b(\phi,m)}), \tag{10}$$

where m is an order index in permuting the M sub-models from the B sub- models, $b(\phi,m)$ indicates the m-th sub-model in the ϕ-th best permutation, $D_i(\boldsymbol{x},\boldsymbol{r}_{b(\phi,m)};\Sigma_{b(\phi,m)})$ denotes the sub-model distance due to the $b(\phi,m)$-th sub-model, and ρ_m is a weighting factor that satisfies $\rho_m \geq \rho_{m+1}$. Third,

$$D_i(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b) = \ln\left\{\frac{1}{\Psi}\sum_{\psi=1}^{\Psi}\exp(\Delta_i^{\psi}(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b))^{-\zeta}\right\}^{-1/\zeta}, \tag{11}$$

where Ψ is the total number of permutations for choosing N paths from all possible Θ_{ib} DTW paths, i.e., $\Psi = \Theta_{ib}!/(\Theta_{ib}-N)!$; $\Delta_i^{\psi}(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b)$ is the combined path distance computed along N paths in the ψ-th best permutation for the b-th best sub-model; and ζ is a positive constant. Note that in this description we have specially focused on the b-th sub-model. Fourth,

$$\Delta_i^{\psi}(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b) = \sum_{n=1}^{N}\pi_n\Delta_i(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b,\theta(\psi,n)), \tag{12}$$

where n is an order index in choosing N paths, $\Delta_i(\boldsymbol{x},\boldsymbol{r}_b;\Sigma_b,\theta(\psi,n))$ is the path distance along the $\theta(\psi,n)$-th path of the b-th best sub-model, $\theta(\psi,n)$ indicates the n-th path in the ψ-th best permutation, and π_n is a weighting factor that satisfies $\pi_n \geq \pi_{n+1}$. Finally, the path distance $\Delta_i^{\psi}(x, r_b; \Sigma_b)$ is defined by using the frame distance $\delta(\cdot)$ as

$$\Delta_i(x, r_b; \Sigma_b, \theta) = \frac{1}{J_b} \sum_{j=1}^{J_b} \delta(x_{\check{i}(i,b,\theta,j)}, r_{bj}; \Sigma_{bj}), \tag{13}$$

where $\check{i}(i, b, \theta, j)$ is the frame indicator for x, which corresponds to r_{bj} via the θ-th path, and the frame distance is of the following form

$$\delta(y, z; \Sigma) = (y - z)^T \Sigma^{-1} (y - z), \tag{14}$$

where $y \in \Re^S$, $z \in \Re^S$, and $\Sigma \in \Re^{S \times S}$.

The above definitions are general but actually rather complicated. One reason for this is that we try to incorporate as rigorously as possible the concept of using multiple best candidates, which has been shown to be useful in increasing design robustness, in our implementations. In practical use, nevertheless, one can run various simpler spotting/design procedures by controlling the L_p norm parameters such as ξ and the weights such as γ, and by introducing a practical approximation into the formalization.

One extreme but the most practical way is to let ξ, η, and ζ go to ∞. The above definitions then become simpler as

$$d_k(x; \Lambda) \approx h - g_{i^*}(x; \lambda), \tag{15}$$

$$g_i(x; \lambda) \approx D_i^1(x; \lambda), \tag{16}$$

$$D_i(x, r_b; \Sigma_b) \approx \Delta_i^1(x, r_b; \Sigma_b) \tag{17}$$

where

$$i^* = \underset{p(k) \leq i \leq q(k)}{\arg\min}\ g_i(x; \lambda). \tag{18}$$

Note that (16) and (17) respectively are calculated using only M-best templates and N-best DTW paths based on the fact that $b_m^1 = m$ and $\theta_n^1 = n$. It is worth mentioning here that the idea of using multiple best candidates holds accurately even for this simple implementation: In particular, the combined path distance can be efficiently computed by the A^*-based N-best search algorithm [8] for $N > 1$, or by dynamic programming for $N = 1$.

Moreover, the full use of covariance matrices often complicates the computation and is time-consuming. Simplification should clearly be attempted over these matrices. The use of the following diagonal form

$$\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_S^2). \tag{19}$$

may be a recommendable solution. The frame distance then becomes

$$\delta(y, z; \Sigma) = \sum_{s=1}^{S} \frac{(y_s - z_s)^2}{\sigma_s^2}. \tag{20}$$

GPD requires in principle an infinite run of training adjustment in order to achieve a local optimal status of the parameters. However, the infinite training is obviously unrealistic. Using finite, monotonically-decreasing positive numbers for ϵ_t in a manner similar to Learning Vector Quantization should thus be another simplication for practicability.

4 Experimental Results

To evaluate the fundamental utility on MSPE, we conducted experiments of Japanese consonant spotting. The input speech data consisted of 50 phonetically-balanced Japanese sentences. This set was a part of the ATR phonetically-balanced 503-sentence set. Each sentence was uttered twice in a sound-proof room by one male speaker. Speech was converted to 16 Mel-scale power coefficients (16-dimensional power spectrum) every 5 msec. The acoustic feature vector (frame vector) was a sequence of seven adjacent power spectra; S was 112 ($= 7 \times 16$). Manually-selected labels were used for both training (in particular, block setting) and testing. Each label consists of a transcription (name) and the first- and last-end frames of the corrsponding consonant.

For computational simplicity, we used a simplified implementation of the DTW-based spotter described in Section 3. The spotter actually evaluated used the simplest L_∞ norm form, the top best sub-model for the combined sub-model distance computation, the top best DTW matching path for the combined path distance computation, and the Euclidean distance for the frame distance computation; $\xi = \eta = \zeta = \infty$, $M = N = 1$, $\rho_1 = \pi_1 = 1$, and $\sigma^2_{bjss'} = \delta(s, s')$, where $\delta(\cdot)$ is Kronecker's delta. We also specially used the asymmetric DP path shown in Figure 2.

Each spotter detected the input frame whose distance from the class model was the minimum among nearby frames and was less than the threshold; such a frame was considered to be the last-end frame of a consonant, i.e., the input frame with the index $\hat{\imath}$ satisfying the following condition:

$$d_{\hat{\imath}}(x; \Lambda) < 0 \quad \wedge \quad \hat{\imath} = \underset{\hat{\imath}-\kappa_1 \leq i \leq \hat{\imath}+\kappa_2}{\arg\min} \; d_i(x; \Lambda), \tag{21}$$

where κ_1, κ_2 are prescribed positive constants. In this experiment, these values were set to $\kappa_1 = \kappa_2 = 3$ for all phoneme spotters. The mis-detection was simply evaluated by counting the number of mis-spottings. To evaluate the false alarm rates properly, we used false alarm rate in *false alarm/hour/phoneme* (FA/H/P), i.e., the number of false alarms, normalized by time and the number of phoneme classes.

Since our training is based on gradient search, the parameters had to be initialized effectively. We initialized the templates by modified k-means clustering [9] with $B = 5$ for every phoneme class using phoneme segments extracted by referring to the hand labels cited above. The thresholds were initially set so that the mis-detection rate and false alarm rate would simultaneously be small. Precisely, the threshold initialization was done as follows: First, we investigated the mis-detections and false alarms for 23 different threshold values (10.0, 15.0, 20.0, ..., 120.0). Then, as the initial threshold value $h(0)$, we employed the value among them that minimized the initial average spotting error

$$\tilde{L}(\Lambda) = n_t + \gamma n_f \tag{22}$$

where n_t and n_f denote the total number of mis-detections and false alarms over the whole training data set, respectively.

Two different context conditions were tested; 1) context-closed mode and 2) context-open mode. In the context-closed mode, a data set consisting of one of two utterances for each sentence (Set 1) was used for training; the other utterance (Set 2) was used for testing. Phoneme contexts appearing in them were the same. In the context-open mode, 40 sentences of Set 1 were used for training and the remaining 10 sentences of Set 1 were used for testing. We carried out training and testing for five different training/testing combinations and got the average performance.

In both modes, we ran a 160-*epoch* (epoch = one full presentation of all training sentences) training for each spotter, with $\alpha_t = 0.1$, $\alpha_f = -0.1$, $\beta_t = \beta_f = 0.0$, $\gamma = 0.15,\ 0.5$, and the learning factor $\epsilon_t = \epsilon_0(1 - t/T)$, where $\epsilon_0 = 20.0$ and T is the product of the total number of training sentences and the prescribed maximum epoch (= 160).

We used six consonant classes that rather frequently appeared in the data set, namely, /t/, /k/, /r/, /N/, /s/, and /h/. Table 1 shows the total number of segments for these consonants in the data set for each training condition. In the table, "C" and "O" denote the context-closed mode and the context-open mode, respectively, and "train" and "test" denote the training set and the testing set, respectively. In the "O" column, the sums for all five training/testing sets are showed. Tables 2-7 show spotting results for testing sets obtained with the min-max method and S-frame-based method for each spotter. In the tables, "MDC" and "FAC" denote the total number of mis-detections and false alarms, respectively; "MDR" denotes the mis-detection rate in percentage; "FAR" denotes the false alarm rate in false alarm/hour/phoneme; and in the "mode" column, "C" and "O" indicate the context-closed mode and context-open mode, respectively, and "B" and "A" indicate the situation before and after the MSPE training, respectively. Accordingly, a pair of these notations such as "CB" and "OA" indicates the corresponding combined condition of experiment. In the context-open mode, the sums of errors of the five training/testing sets and the average error rates are showed.

A significant reduction in the number of errors was obtained for /t/, /k/, and /r/, where MSPE training decreased the mis-detection rateand holds the false alarm rate low, both in the context-closed mode and in the context-open mode. Although the improvements were smaller for /N/, /s/, and /h/, the effectiveness of MSPE training was clear.

The performance of the two types of block setting methods do not differ so much. However, the S-frame-based method tends to give more weight to false alarms than to mis-detections, since it conducts adjustment for non-S-blocks more frequently. As the proliferation of false alarms by the min-max method with small γ for /r/, /N/, and /h/ suggests, γ and the learning factor should be selected carefully depending on the phoneme (or word etc.) category and the block setting method.

5 Conclusion

We presented a new design method for word spotting, called the Minimum Spotting Error learning (MSPE), which fundamentally guarantees the minimum spotting error situation in a probabilistic sense through MCE/GPD. MSPE allows us to train all trainable spotter parameters consistently; this key feature implies an innovative departure from conventional, heuristic approaches to spotter design.

The MSPE concept can be applied to any spotting process for even visual patterns as well as spoken words/phonemes (used for the formalization description in the paper). The concept can also be applied to any reasonable spotter system structure. The paper presented the detailed implementation for a multiple sub-model, DTW-based system; implementations for an HMM system and a subspace-projection-based system are summarized in the Appendix.

To show the utility of the proposed method, we conducted spotting experiments on several Japanese phonemes. The experiments were somewhat limited to a particular setting of controllable parameters, e.g., coefficients in the L_p norm form and the number of multiple best sub-model/DP-path selections. However, our results clearly demonstrated a very high utilization possibility for MSPE.

Acknowledgement

The authors would like to thank Dr. Takeshi Kawabata of NTT Basic Research Laboratories and Dr. Hiroyuki Tsuboi of Toshiba Corporation for their valuable comments.

References

[1] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New Discriminative Training Algorithms Based on the Generalized Probabilistic Descent Method," Proc. of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp. 299–308 (1991).

[2] E. McDermott and S. Katagiri, "Prototype-Based Discriminative Training for Various Speech Units," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 417–420 (1992).

[3] W. Chou, B.-H. Juang, and C.-H. Lee, "Segmental GPD Training of HMM Based Speech Recognizer," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 473–476 (1992).

[4] P.-C. Chang and B.-H. Juang, "Discriminative Template Training for Dynamic Programming Speech Recognition," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 493–496 (1992).

[5] T. Komori and S. Katagiri, "Application of a Generalized Probabilistic Descent Method to Dynamic Time Warping-Based Speech Recognition," IEEE, Proc. of ICASSP-92, Vol. 1, pp. 497–500 (1992).

[6] A. Biem and S. Katagiri, "Feature Extraction Based on Minimum Classification Error/Generalized Probabilistic Descent Method," IEEE, Proc. of ICASSP-93, Vol. 2, pp. 275–278 (1993).

[7] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," IEEE Trans. on Signal Processing, Vol. 40, No. 12, pp. 3043–3054 (1992).

[8] F. K. Soong and E.-F. Huang, "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition," IEEE, Proc. of ICASSP-91, pp. 705–708 (1991).

[9] J. G. Wilpon and L. R. Rabiner, "A Modified K-means Clustering Algorithm for Use in Speaker Independent Isolated Word Recognition," AT&T Bell Labs., Technical Memo. 11227-840103-1TM (1984).

[10] H. Tsuboi, Y. Takebayashi, H. Matsu'ura, T. Nitta, and S. Hirai, "Speaker-Adaptive Connected Syllable Recognition Based on the Multiple Similarity Method," IEEE, Proc. of ICASSP-86, pp. 2655–2658 (1986).

[11] Y. Takebayashi, "Speech Recognition Based on the Subspace Method: AI Class-Description Learning Viewpoint," J. Acoust. Soc. Jpn. (E), Vol. 13, No. 6, pp. 429–439 (1992).

[12] H. Tsuboi and Y. Takebayashi, "A Real-Time Task-Oriented Speech Understanding System Using Keyword-Spotting," IEEE, Proc. of ICASSP-92, Vol.1, pp. 197–200 (1986).

[13] K. Aikawa, "Phoneme Recognition Using Time-Warping Neural Networks," J. Acoust. Soc. Jpn. (E), Vol. 13, No. 6, pp. 395–402 (1992).

Appendix

A Implementation to HMM-Based Spotter

The implementation of MSPE to HMM-based spotting frameworks is described below. We treat a J-state hidden Markov model $\{\Pi, A, B\}$ as the class model $\boldsymbol{\lambda}$; Π is a set of initial state probabilities $\{\pi_j\}_{j=1}^{J}$, $A = [a_{i,j}]$ is a state transition probability matrix, and B is a set of output probability density functions $\{b_j(\cdot)\}_{j=1}^{J}$. For notation convenience, we assume $a_{0,j} = \pi_j$ in the following discussion.

In HMM-based spotting frameworks, the discriminant function takes the following form:

$$g_i(\boldsymbol{x}; \boldsymbol{\lambda}) = -\ln \left\{ \frac{1}{\Theta_i} \sum_{\theta=1}^{\Theta_i} P_i(\boldsymbol{x}; \boldsymbol{\lambda}, \theta)^{\eta} \right\}^{1/\eta} \tag{23}$$

where $P_i(\boldsymbol{x}; \boldsymbol{\lambda}, \theta)$ is the *path probability* at the θ-th *best* (largest probability) frame-state matching path among all possible Θ_i paths.

The path probability at the θ-th path is defined as a duration-normalized likelihood:

$$P_i(\boldsymbol{x}; \boldsymbol{\lambda}, \theta) = \left\{ \prod_{l=1}^{L_\theta} a_{\bar{j}(\theta,l-1),\bar{j}(\theta,l)} b_{\bar{j}(\theta,l)}(\boldsymbol{x}_{\bar{i}(\theta,l)}) \right\}^{1/L_\theta} \tag{24}$$

where L_θ is the length of the θ-th path. Note that due to this duration normalization, time-synchronous trellis algorithms populary used in HMM-based systems is not adequate here.

The output probability density function form depends on the HMM type; continuous or discrete. A typical continuous HMM uses

$$b_j(\boldsymbol{y}) = \sum_{m=1}^{M_j} w_{jm} N(\boldsymbol{y}; \boldsymbol{r}_{jm}, \Sigma_{jm}) \tag{25}$$

where M_j is the total number of mixtures of state j, and $N(\boldsymbol{y}; \boldsymbol{r}_{jm}, \Sigma_{jm})$ is a multivariate Gaussian with a mean vector $\boldsymbol{r}_{jm}$ and a covariance matrix Σ_{jm}, and w_{jm} is the normalized mixture weight satisfying $\sum_{m=1}^{M_j} w_{jm} = 1$. On the other hand, a discrete HMM using Fuzzy VQ uses

$$b_j(\boldsymbol{y}) = \sum_{m=1}^{M} u_{jm} f_m(\boldsymbol{y}) \tag{26}$$

where u_{jm} is the output probability satisfying the requirement $\sum_{m=1}^{M} u_{jm} = 1$ and $f_m(\boldsymbol{y})$ is a fuzzy membership function of the m-th VQ code defined as

$$f_m(\boldsymbol{y}) = \frac{\delta(\boldsymbol{y}, \boldsymbol{r}_m)^{-1/(F-1)}}{\sum_{m'=1}^{M} \delta(\boldsymbol{y}, \boldsymbol{r}_{m'})^{-1/(F-1)}} \tag{27}$$

where $\delta(\boldsymbol{y}, \boldsymbol{r}_m)$ is the distance (usually the Euclidian distance) between $\boldsymbol{y}$ and the m-th centroid vector $\boldsymbol{r}_m \in \Re^S$ of the VQ-codebook with size M, and the value $F > 1$ is the degree of fuzziness. Also in the discrete one, the reference vectors can be trained.

Simplification techniques stated in Section 3 can also be applied similarly and to keep some constraints of the HMM parameters during MSPE training, parameter transformations stated in Ref. [3] can be used.

B Implementation to Subspace Method-Based Spotter

We can also apply our training method to spotting based on Iijima's subspace method using multiple similarity [10],[11].

The discriminant function for a multiple similarity-based subspace method is denoted by

$$g_i(x; \boldsymbol{\lambda}) = -\ln \left\{ \frac{1}{\Theta_i} \sum_{\theta=1}^{\Theta_i} \exp(s_i(x; \boldsymbol{\lambda}, \theta))^{\eta} \right\}^{1/\eta} \tag{28}$$

where $s_i(x; \boldsymbol{\lambda}, \theta)$ is the multiple similarity at the i-th input frame along the θ-th *best* (maximum similarity) time warping path among all possible Θ_i paths. The multiple similarity is defined as follows:

$$s_i(x; \boldsymbol{\lambda}, \theta) = \sum_{m=1}^{M} \frac{a_m(\boldsymbol{f}_{i\theta}(x), \boldsymbol{\phi}_m)^2}{a_1 \parallel \boldsymbol{f}_{i\theta}(x) \parallel^2} \tag{29}$$

where $\boldsymbol{f}_{i\theta}(x)$ is the time-normalized static input vector by the θ-th best path whose final frame is the i-th input frame; a_m and $\boldsymbol{\phi}_m$ are the m-th largest eigenvalue and corresponding eigenvector of the covariance matrix of the class, M is the total number of eigenvalues/eigenvectors used to compute the multiple similarity, and (,) denotes dot product operation. There can be many possibilities of time-normalization. One may use a simple linear oor non-linear mapping function for this normalization [12],[13]. By letting

$$\boldsymbol{y}_{i\theta} = \frac{\boldsymbol{f}_{i\theta}(x)}{\parallel \boldsymbol{f}_{i\theta}(x) \parallel} \tag{30}$$

and

$$\boldsymbol{b}_m = \sqrt{a_m} \cdot \boldsymbol{\phi}_m, \tag{31}$$

we simplify the definition of the discriminant function in Eq. (28) to

$$s_i(x; \boldsymbol{\lambda}, \theta) = \sum_{m=1}^{M} \frac{(\boldsymbol{b}_m, \boldsymbol{y})^2}{\parallel \boldsymbol{b}_1 \parallel^2}. \tag{32}$$

To keep satisfying the orthogonality of vector set $\{b_m\}$ during the MSPE training procedure, we use parameter transformation based on Schmidt's nomalization:

$$b_m = \bar{b}_m - \sum_{k=1}^{m-1} \frac{(\bar{b}_m, \bar{b}_k)\bar{b}_k}{\| \bar{b}_k \|^2}. \tag{33}$$

Transformed parameter $\{\bar{b}_m\}$ can be freely modified with keeping the orthogonality of $\{b_m\}$. In MSPE training for this framework, a class model λ represented by $\lambda = \{a_m, \bar{b}_m\}_{m=1}^{M}$ is updated according to Eq. (8).

Table 1: The total number of phonemes in each data set

	C		O	
phon.	train	test	train	test
t	102	102	408	102
k	141	141	564	141
r	117	117	468	117
N	102	103	408	102
s	69	69	276	69
h	66	66	264	66

Table 2: Training result of /t/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	23	8	22.5	122
	CA	2	20	2.0	305
	OB	11	30	10.8	476
	OA	0	23	0.0	365
0.5	CB	23	8	22.5	122
	CA	4	11	3.9	168
	OB	19	19	18.6	302
	OA	3	15	2.9	238

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	23	8	22.5	122
	CA	2	10	2.0	152
	OB	11	30	10.8	476
	OA	3	21	2.9	333
0.5	CB	23	8	22.5	122
	CA	4	3	3.9	45
	OB	19	19	18.6	302
	OA	3	12	2.9	190

Table 3: Training result of /k/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	34	110	24.1	1682
	CA	11	79	7.8	1208
	OB	27	111	19.1	1764
	OA	11	87	7.8	1383
0.5	CB	74	23	52.5	351
	CA	18	19	12.8	290
	OB	48	41	34.0	651
	OA	22	22	15.6	349

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	34	110	24.1	1682
	CA	15	29	10.6	443
	OB	27	111	19.1	1764
	OA	15	39	10.6	620
0.5	CB	74	23	52.5	351
	CA	21	14	14.9	214
	OB	48	41	34.0	651
	OA	23	15	16.3	238

Table 4: Training result of /r/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	68	133	58.1	2034
	CA	2	630	1.7	9638
	OB	57	199	48.7	3163
	OA	13	593	11.1	9428
0.5	CB	94	48	80.3	734
	CA	23	24	19.7	367
	OB	98	22	83.8	425
	OA	38	29	32.5	561

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	68	133	58.1	2034
	CA	18	42	15.4	642
	OB	57	199	48.7	3163
	OA	28	50	23.9	794
0.5	CB	94	48	80.3	734
	CA	43	9	36.8	137
	OB	98	22	83.8	425
	OA	78	6	66.7	116

Table 5: Training result of /N/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	52	62	50.5	948
	CA	10	1112	9.7	17012
	OB	29	118	28.4	1876
	OA	18	1100	17.6	17488
0.5	CB	73	19	70.9	290
	CA	42	77	40.8	1178
	OB	55	46	53.9	731
	OA	49	53	48.0	842

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	52	62	50.5	948
	CA	14	67	13.6	1025
	OB	29	118	28.4	1876
	OA	23	66	22.5	1049
0.5	CB	73	19	70.9	290
	CA	31	32	30.1	489
	OB	55	46	53.9	731
	OA	38	30	37.3	476

Table 6: Training result of /s/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	4	36	5.8	550
	CA	3	31	4.3	474
	OB	9	29	13.0	461
	OA	2	28	2.9	445
0.5	CB	7	16	10.1	244
	CA	9	11	13.0	168
	OB	14	18	20.3	286
	OA	10	8	14.5	127

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	4	36	5.8	550
	CA	3	21	4.3	321
	OB	9	29	13.0	461
	OA	5	26	7.2	413
0.5	CB	7	16	10.1	244
	CA	4	8	5.8	122
	OB	14	18	20.3	286
	OA	8	15	11.6	238

Table 7: Training result of /h/ spotter

(a) Min-max method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	34	30	51.5	458
	CA	8	2170	12.1	33198
	OB	36	41	54.5	651
	OA	8	1077	12.1	17123
0.5	CB	44	5	66.7	76
	CA	21	9	31.8	137
	OB	44	3	66.7	79
	OA	36	11	54.5	292

(b) S-frame-based method

γ	Mode	MDC	FAC	MDR	FAR
0.15	CB	34	30	51.5	458
	CA	19	12	28.8	183
	OB	36	41	54.5	651
	OA	36	20	54.5	317
0.5	CB	44	5	66.7	76
	CA	28	4	42.4	61
	OB	44	3	66.7	79
	OA	44	5	66.7	132

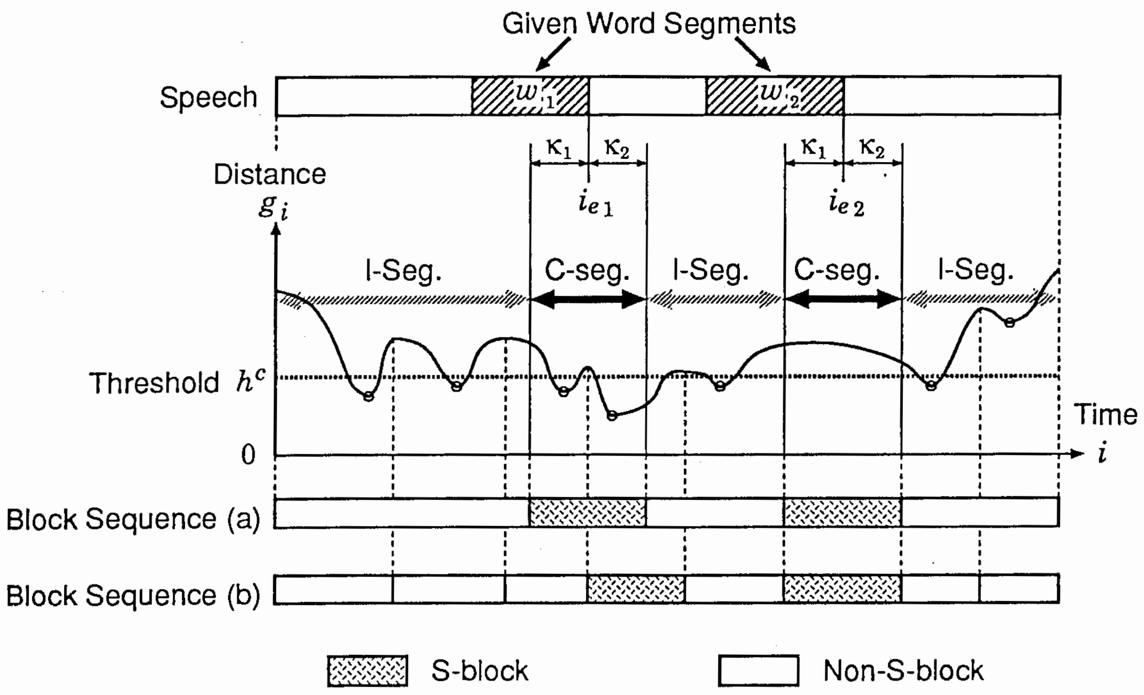

Figure 1: An example of (a) min-max method and (b) S-frame-based method. White circles indicate local minima of discriminant function.

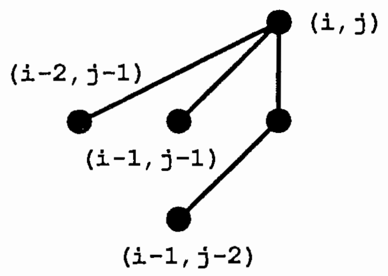

Figure 2: Asymmetric DP path used for starting-end-free dynamic time warping.