7

Algorithms for Joint Evaluation of Multiple Speech Patterns for Automatic Speech Recognition

Nishanth Ulhas Nair and T.V. Sreenivas
Department of Electrical Communication Engineering,
Indian Institute of Science, Bangalore - 560012,
India

1. Introduction

Improving speech recognition performance in the presence of noise and interference continues to be a challenging problem. Automatic Speech Recognition (ASR) systems work well when the test and training conditions match. In real world environments there is often a mismatch between testing and training conditions. Various factors like additive noise, acoustic echo, and speaker accent, affect the speech recognition performance. Since ASR is a statistical pattern recognition problem, if the test patterns are unlike anything used to train the models, errors are bound to occur, due to feature vector mismatch. Various approaches to robustness have been proposed in the ASR literature contributing to mainly two topics: (i) reducing the variability in the feature vectors or (ii) modify the statistical model parameters to suit the noisy condition. While some of the techniques are quite effective, we would like to examine robustness from a different perspective. Considering the analogy of human communication over telephones, it is quite common to ask the person speaking to us, to repeat certain portions of their speech, because we don't understand it. This happens more often in the presence of background noise where the intelligibility of speech is affected significantly. Although exact nature of how humans decode multiple repetitions of speech is not known, it is quite possible that we use the combined knowledge of the multiple utterances and decode the unclear part of speech. Majority of ASR algorithms do not address this issue, except in very specific issues such as pronunciation modeling. We recognize that under very high noise conditions or bursty error channels, such as in packet communication where packets get dropped, it would be beneficial to take the approach of repeated utterances for robust ASR. We have formulated a set of algorithms for both joint evaluation/decoding for recognizing noisy test utterances as well as utilize the same formulation for selective training of Hidden Markov Models (HMMs), again for robust performance. Evaluating the algorithms on a speaker independent confusable word Isolated Word Recognition (IWR) task under noisy conditions has shown significant improvement in performance over the baseline systems which do not utilize such joint evaluation strategy.

A simultaneous decoding algorithm using multiple utterances to derive one or more allophonic transcriptions for each word was proposed in [Wu & Gupta, 1999]. The goal of a

Source: Speech Recognition, Technologies and Applications, Book edited by: France Mihelič and Janez Žibert,
ISBN 978-953-7619-29-9, pp. 550, November 2008, I-Tech, Vienna, Austria

simultaneous decoding algorithm is to find one optimal allophone sequence W^* for all input utterances U_1, U_2,. . ., U_n. Assuming independence among U_i, according to the Bayes criterion, W^* can be computed as

$$\begin{aligned} W^* &= \arg\max_W P(W/U_1, U_2, \ldots, U_n) \\ &= \arg\max_W P(U_1, U_2, \ldots, U_n/W)P(W) \\ &= \arg\max_W P(U_1/W)P(U_2/W)\ldots P(U_n/W)P(W) \end{aligned} \quad (1)$$

where $P(X)$, stands for the probability of the event X occurring.

From an information theoretic approach, consider two speech sequences $U1$ and $U2$. The joint entropy $H(U1, U2)$ will be higher than either of their individual entropies $H(U1)$ or $H(U2)$ [Shannon, 1948]. We know that if $U1$ and $U2$ are completely independent of each other then the joint entropy $H(U1,U2)$ will be equal to $H(U1) + H(U2)$. If they are completely dependent $H(U1,U2) = H(U1) = H(U2)$. When $U1$ and $U2$ come from the same class, there is a degree of correlation between them. Particularly when parts of $U1$ or $U2$ is corrupted, then the joint entropy would have a higher difference with respect to either of the individual entropies. This is because the noise is more random in nature. This applies to > 2 sequences also.. The goal of the pattern recognition task is to exploit this higher information entropy in a maximum likelihood (ML) framework for better recognition.

One direct approach to simultaneous decoding is to use the N-best criteria [Nilsson, 1971, Schwartz & Chow, 1990, Soong & Hung, 1991]. In this, an individual N-best list for each input utterance is generated independently using the N-best search algorithm of statistical decoding. These individual N-best lists are merged and rescored using all the input utterances [Haeb-Umbach et al., 1995]; based on their joint likelihoods the transcriptions are re-ordered. However this solution is suboptimal unless N is very large [Wu & Gupta, 1999]. Simultaneous decoding for multiple input utterances can be done using a modified version of the tree-trellis search algorithm [Soong & Hung, 1991] (the same algorithm was used in [Holter et al., 1998]). A forward Viterbi beam search for each utterance is performed independently, and then a combined backward A^* search [Bahl et al., 1983] for all the utterances is applied simultaneously. A word-network-based algorithm is also developed for simultaneous decoding. This algorithm has been shown to be computationally very efficient [Wu & Gupta, 1999].

Multiple utterances of same speech unit has been typically used in pronunciation estimation. Pronunciation determined using only one recording of a word can be very unreliable. So for more reliability, modeling multiple recordings of a word is used. However commonly used decoding algorithms are not suited to discover a phoneme sequence that jointly maximizes the likelihood of all the inputs.To arrive at the same solution, various alternative techniques have been proposed. One method is to produce recognition lattices individually from each of the inputs, and identify the most likely path in the intersection of these lattices. Another generates N-best hypotheses from each of the audio inputs and re-scores the cumulative set jointly with all the recordings [Singh et al., 2002, Svendson, 2004]. Alternately, the pronunciations may be derived by voting amongst the recognition outputs from the individual recordings [Fiscus, 1997]. While all of these procedures result in outputs that are superior to what might be obtained using only one recorded instance of the word, they nevertheless do not truly identify the most likely pronunciation for the given set of recordings, and thus remain suboptimal. So it is important to jointly estimate the pronunciation from multiple recordings.

Dealing with multiple speech patterns occurs naturally during the training stage. In most, the patterns are considered as just independent exemplars of a random process, whose parameters are being determined. There is some work in the literature to make the ML training process of statistical model, such as HMM, more robust or better discriminative. For example, it is more difficult to discriminate between the words "rock" and "rack", than between the words "rock" and "elephant". To address such issues, there has been attempts to increase the separability among similar confusible classes, using multiple training patterns.

In discriminative training, the focus is on increasing the separable distance between the models, generally their means. Therefore the model is changed. In selective training the models are not forced to fit the training data, but deemphasizes the data which does not fit the models well. In [Arslan & Hansen, 1996, Arslan & Hansen, 1999], each training pattern is selectively weighted by a confidence measure in order to control the influence of outliers, for accent and language identification application. Adaptation methods for selective training, where the training speakers close to the test speaker are chosen based on the likelihood of speaker Gaussian Mixture Models (GMMs) given the adaptation data, is done in [Yoshizawa et al., 2001]. By combining precomputed HMM sufficient statistics for the training data of the selected speakers, the adapted model is constructed. In [Huang et. al, 2004], cohort models close to the test speaker are selected, transformed and combined linearly. Using the methods in [Yoshizawa et al., 2001, Huang et. al, 2004], it is not possible to select data from a large data pool, if the speaker label of each utterance is unknown or if there are only few utterances per speaker. This can be the case when data is collected automatically, e.g. the dialogue system for public use such as Takemaru-kun [Nishimura et al., 2003]. Selective training of acoustic models by deleting single patterns from a data pool temporarily or alternating between successive deletion or addition of patterns has been proposed in [Cincarek et al., 2005].

In this chapter, we formulate the problem of increasing ASR performance given multiple utterances (patterns) of the same word. Given K test patterns ($K \geq 2$) of a word, we would like to improve the speech recognition accuracy over a single test pattern, for the case of both clean and noisy speech. We try to jointly recognize multiple speech patterns such that the unreliable or corrupt portions of speech are given less weight during recognition while the clean portions of speech are given a higher weight. We also find the state sequence which best represents the K patterns. Although the work is done for isolated word recognition, it can also be extended to connected word and continuous speech recognition. To the best of our knowledge, the problem that we are formulating has not been addressed before in speech recognition.

Next, we propose a new method to selectively train HMMs by jointly evaluating multiple training patterns. In the selective training papers, the outlier patterns are considered unreliable and are given a very low (or zero) weighting. But it is possible that only some portions of these outlier data are unreliable. For example, if some training patterns are affected by burst/transient noise (e.g. bird call) then it would make sense to give a lesser weighting to only the affected portion. Using the above joint formulation, we propose a new method to train HMMs by selectively weighting regions of speech such that the unreliable regions in the patterns are given a lower weight. We introduce the concept of "virtual training patterns" and the HMM is trained using the virtual training patterns instead of the

original training data. We thus address all the three main tasks of HMMs by jointly evaluating multiple speech patterns.

The outline of the chapter is as follows: sections 2 and 3 gives different approaches to solve the problem of joint recognition of multiple speech patterns. In section 4, the new method of selectively training HMMs using multiple speech patterns jointly is proposed. Section 5 gives the experimental evaluations for the proposed algorithms, followed by conclusions in section 6.

2. Multi Pattern Dynamic Time Warping (MPDTW)

The Dynamic Time Warping (DTW) [Rabiner & Juang, 1993, Myers et al., 1980, Sakoe & Chiba, 1978] is a formulation to find a warping function that provides the least distortion between any two given patterns; the optimum solution is determined through the dynamic programming methodology. DTW can be viewed as a pattern dissimilarity measure with embedded time normalization and alignment. We extend this formulation to multiple patterns greater than two, resulting in the multi pattern dynamic time warping (MPDTW) algorithm [Nair & Sreenivas, 2007, Nair & Sreenivas, 2008 b]. The algorithm determines the optimum path in the multi-dimensional discrete space to optimally warp all the K number of patterns jointly, leading to the minimum distortion path, referred to as MPDTW path. As in standard DTW, all K patterns are warped with respect to each other. The MPDTW algorithm finds the least distortion between the K patterns and the MPDTW algorithm reduces to the DTW algorithm for K = 2. To find the MPDTW path, we need to traverse through the Kdimensional grid along the K time axes. Let the K patterns be $\mathbf{O}^1_{1:T_1}$, $\mathbf{O}^2_{1:T_2}$,. . ., $\mathbf{O}^K_{1:T_K}$,, of lengths T_1, T_2,. . ., T_K, respectively, where $\mathbf{O}^i_{1:T_i} = (O^i_1, O^i_2, \ldots, O^i_{T_i})$ is the observation vector sequence of the i^{th} pattern and $O^i_{t_i}$ is the feature vector at time frame t_i.

Fig. 1 shows an example MPDTW path for K = 2; it is the least distortion path between the two patterns. We define a path P of grid traversing as a sequence of steps in the grid, each specified by a set of coordinate *increments* [Rabiner & Juang, 1993], i.e.,

$$P \rightarrow (p^1_1, p^2_1, \ldots, p^K_1), (p^1_2, p^2_2, \ldots, p^K_2), \ldots, (p^1_T, p^2_T, \ldots, p^K_T) \tag{2}$$

where p^i_t is the increment at step t along utterance i (i^{th} dimension).

Let the t = 1 step correspond to (1, 1,. . ., 1), is the staring point in the grid where all the K patterns begin. Let us set $p^1_1 = p^2_1 = \ldots = p^K_1 = 1$. Let $t = T$ correspond to (T_1, T_2, . . . , T_K), which is the ending point of the multi-dimensional grid. $\phi_1(t)$, $\phi_2(t)$,. . ., $\phi_K(t)$ are K warping functions such that $\phi_i(t) = t_i$ for the i^{th} pattern. Let us define:

$$\phi_l(t) = \sum_{i=1}^{t} p^l_i, \qquad l = 1, 2, \ldots, K \tag{3}$$

The coordinate *increments* satisfy the constraints:

$$\sum_{t=1}^{T} p^l_t = T_l, \qquad l = 1, 2, \ldots, K \tag{4}$$

Endpoint constraints are:

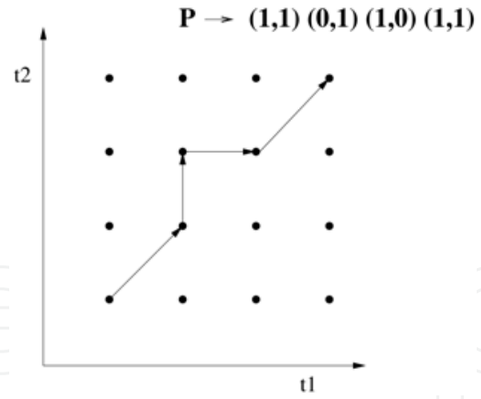

Fig. 1. An example MPDTW path for K = 2.

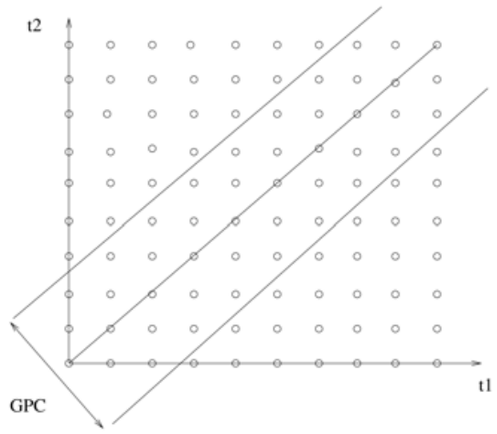

Fig. 2. An example Global Path Constraint for MPDTW for K = 2.

$$\phi_1(1) = 1, \ldots, \phi_K(1) = 1 \tag{5}$$

$$\phi_1(T) = T_1, \ldots, \phi_K(T) = T_K \tag{6}$$

Relaxed end pointing can also be introduced as in standard DTW. Various types of Local Continuity Constraints (LCC) and Global Path Constraints (GPC) as defined for DTW [Rabiner & Juang, 1993], can be extended to the K dimensional space. The LCC we used is similar to the simplest Type-1 LCC used in DTW, except that it has K dimensions. The point (t_1, t_2, . . . , t_K) can be reached from any one of the points ($t_1 - i_1$, $t_2 - i_2$, . . . , $t_K - i_K$) where i_k = 0, 1 for k = 1, 2, . . . ,K. This leads to ($2^K - 1$) predecessor paths, excluding the all-zero combination. One type of GPC for MPDTW when K = 2 is shown in Fig. 2. It can be extended for any K. For e.g., if K = 3 the GPC will look like a square cylinder around the diagonal.

The important issue in MPDTW is the distortion measure between patterns being compared. Since the goal is to minimize an accumulated distortion along the warping path, we define a positive distortion metric at each end of the node of the grid traversing. We define a joint distance measure $d(t_1, t_2, \ldots, t_K)$ between the K vectors $O^1_{t_1}, O^2_{t_2}, \ldots, O^K_{t_K}$, as follows:

$$d(t_1, \ldots, t_K) = \sum_{i=1}^{K} d(O^i_{t_i}, C_{\phi(t)}), \tag{7}$$

where $C_{\phi(t)}$ is the centroid of the K vectors $O^1_{t_1}, O^2_{t_2}, \ldots, O^K_{t_K}$ and $d(O^i_{t_i}, C_{\phi(t)})$ is the Euclidean distance between $O^i_{t_i}$ and $C_{\phi(t)}$ and $\phi(t) = (t_1, t_2, \ldots, t_K) = (\phi_1(t), \phi_2(t), \ldots, \phi_K(t))$. This measure is generalizable to any perceptually important measures such as the Itakura-Saito measure [Itakura & Saito, 1968] and a generalized centroid as defined in Vector Quantization [Gersho & Gray, 1992]. In Fig. 3, three vectors $O1$, $O2$, $O3$ and their Euclidean centroid C is shown. The joint distance measure between $O1$, $O2$, $O3$ is $d(O1,C) + d(O2,C) + d(O3,C)$, where $d(O,C)$ is the Euclidean distance between vector O and vector C.
To discourage the optimum path close to the diagonal, the slope weighting function $m(t)$ is utilized. The final accumulated distortion is normalized using

$$M_\phi = \sum_{t=1}^{T} m(t) \tag{8}$$

Thus the total distortion to be minimized is

$$D_{total} = \sum_{t=1}^{T} m(t) \sum_{j=1}^{K} d(O^j_{t_j}, C_{\phi(t)})/M_\phi \tag{9}$$

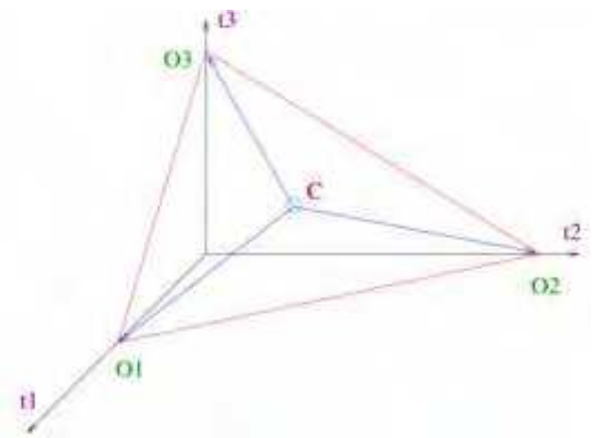

Fig. 3. 3 vectors $O1$, $O2$, $O3$ and their centroid C.

2.1 MPDTW Algorithm

Let $D_{total} = D(t_1, t_2, \ldots, t_K)$ be the accumulated cost function, which is to be minimized.

a) Initialization

$$D(1, \ldots, 1) = d(1, \ldots, 1)m(1) \tag{10}$$

b) Recursion

For $1 \le t_1 \le T_1$, $1 \le t_2 \le T_2, \ldots, 1 \le t_K \le T_K$, such that $t_1, t_2, \ldots, t_K$ lie within the allowable grid.

$$\begin{aligned} D(t_1, \ldots, t_K) \quad = \quad & \min_{(t'_1, \ldots, t'_K)} [D(t'_1, \ldots, t_K{}') + \\ & \zeta((t'_1, \ldots, t'_K)(t_1, \ldots, t_K))] \end{aligned} \tag{11}$$

where $(t'_1, \ldots, t'_K)$ are the candidate values as given by the LCC and

$$\zeta((t_1^{'},\ldots,t_K^{'})(t_1,\ldots,t_K)) \quad = \quad \sum_{l=0}^{L_s} d(\phi_1(T^{'}-l),\ldots,\phi_K(T^{'}-l))m(T^{'}-l) \tag{12}$$

$\phi_1(T^{'}) = t_1,\ldots,\phi_K(T^{'}) = t_K$ and $\phi_1(T^{'}-L_s) = t_1^{'},\ldots,\phi_K(T^{'}-L_s) = t_K^{'}$ where L_s being the number of moves in the path from (t'_1, . . . , t'_K) to (t_1, . . . , t_K) according to ϕ_1,. . ., ϕ_K. A backtracking pointer I is defined to remember the best local choice in equation 11, which will be used for the global path backtracking.

$$I(t_1,\ldots,t_K) \quad = \quad \arg \min_{(t_1^{'},\ldots,t_K^{'})} [D(t_1^{'},\ldots,t_K^{'}) + \zeta((t_1^{'},\ldots,t_K^{'})(t_1,\ldots,t_K))] \tag{13}$$

c) Termination

$$\mathbf{d}(\mathbf{O}_{1:T_1}^1,\ldots,\mathbf{O}_{1:T_K}^K) = D(T_1,\ldots,T_K)/M_\phi \tag{14}$$

where $\mathbf{d}(\mathbf{O}_{1:T_1}^1,\ldots,\mathbf{O}_{1:T_K}^K)$ is the total distortion between the K patterns $\mathbf{O}_{1:T_1}^1,\ldots,\mathbf{O}_{1:T_K}^K$; this is the best distortion measure under the constraints of the MPDTW algorithm.

d) Path Backtracking

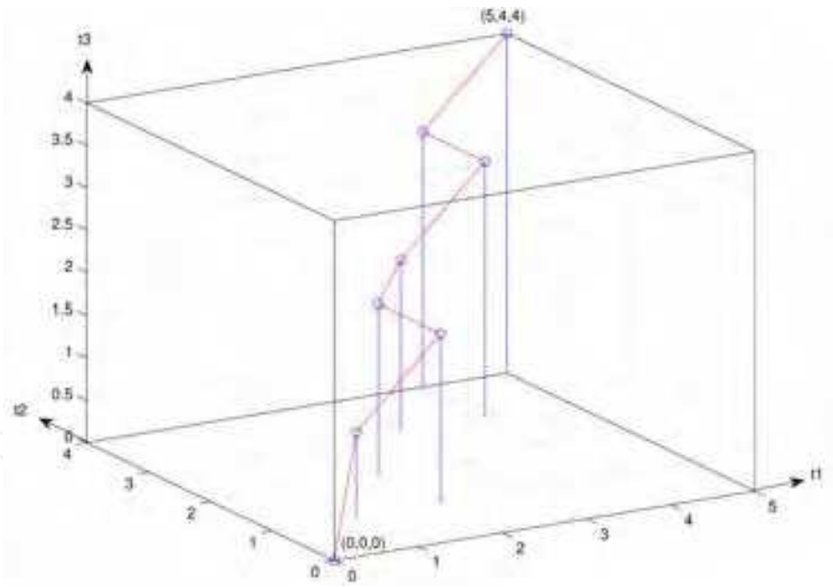

Fig. 4. An example MPDTW path for K = 3 patterns.

The optimum warping between the K patterns is obtained through backtracking, starting from the end point and "hopping" through the back-pointer grid, i.e.,

$$(t_1^*,\ldots,t_K^*) = I(t_1,\ldots,t_K) \tag{15}$$

$$(t_1,\ldots,t_K) = (t_1^*,\ldots,t_K^*) \tag{16}$$

where (t_1, . . . , t_K) = (T_1, . . . , T_K), . . . , (1, . . . , 1).

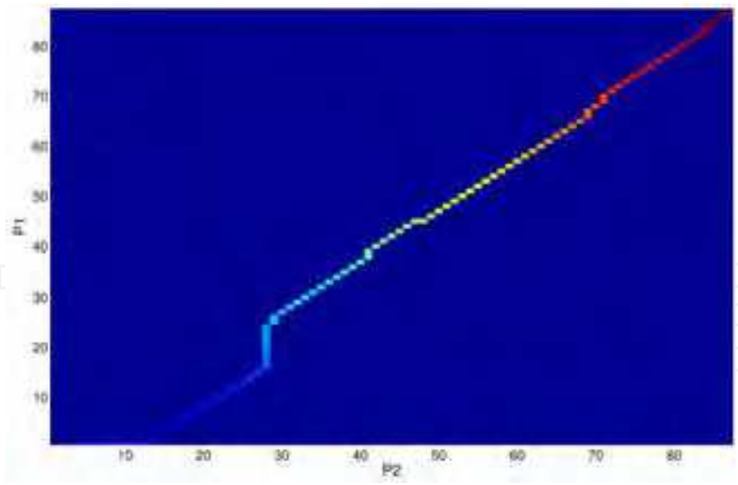

Fig. 5. MPDTW path for 3 patterns P1, P2, P3 projected on P1-P2 plane. The first 30% of the frames (frame number 1 to 27) in P2 is noisy at -15 dB SNR. P1 and P3 are clean.

The least distortion path, referred to as MPDTW path, gives us the most similar non linear time warping path between them. Let ϕ be the MPDTW path for K patterns. $\phi(t) = (t_1, \ldots, t_K)$, where $(t_1, \ldots, t_K)$ is a point on the MPDTW path. $\phi(1) = (1, \ldots, 1)$ and $\phi(T) = (T_1, \ldots, T_K)$. So $\phi = (\phi(1), \phi(2), \ldots, \phi(t), \ldots, \phi(T))$.

An example MPDTW path for $K = 3$ patterns is shown in Fig. 4. A projection of an example MPDTW path for 3 speech patterns (P1, P2, P3) on the P1-P2 plane is shown in Fig. 5, where burst noise at -15 dB Signal to Noise Ratio (SNR) is added to the first 30% in the speech pattern P2. All the three patterns belong to the word "Voice Dialer" by one female speaker. The feature vector used to represent speech was Mel Frequency Cepstral Coefficients (MFCC), Δ MFCC, Δ^2 MFCC without the energy component (36 dimension vector). Notice that the initial portion of the MPDTW path has a deviation from the diagonal path but it comes back to it. Fig. 6 shows the MPDTW path when burst noise at -5 dB SNR is added to 10% frames in the beginning portion of pattern P2. We don't see too much of a deviation from the diagonal path. This tells us that the MPDTW algorithm is relatively robust to burst noise using only 3 patterns (K= 3). This clearly shows that we can use the MPDTWalgorithm to align K patterns coming from same class even if they are affected by burst/transient noise. We will use this property to our advantage later.

2.2 MPDTW for IWR

Since MPDTW is a generalization of the DTW, it opens new alternatives to the basic problem of IWR. In a basic IWR, the test pattern and reference pattern are compared, resulting in an optimum warping path in 2-Dimension. But we have a solution in K dimensions. Hence, we can choose a variable number of test and reference patterns [Nair & Sreenivas, 2008 b]. Hence, let there be r reference patterns per class (word) and $K - r$ test patterns. We can compare the minimum distortion of the test patterns with respect to the reference patterns of different words using the MPDTW algorithm. Whichever word reference set gives the lowest distortion, it can be selected as the matched word. It should be noted that the recognition performance will be different from the 2D-DTW and likely more robust, because multiple patterns are involved both for testing and as reference templates.

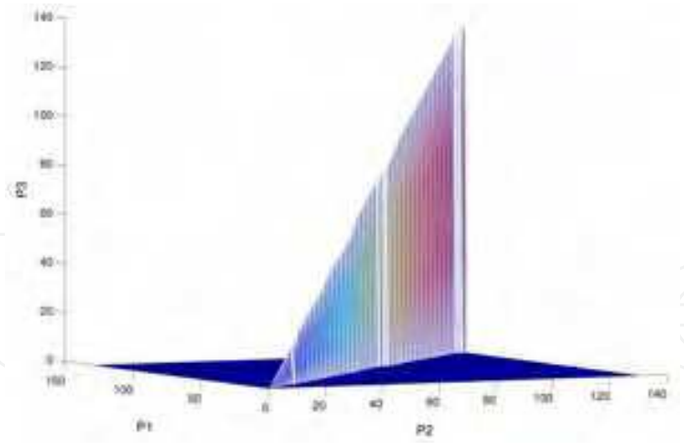

Fig. 6. MPDTW path for 3 patterns of the word "Voice Dialer". The first 10% of pattern P2 is noisy at -5 dB SNR. Patterns P1 and P3 are clean.

Case 1: Multiple Test Patterns and One Reference Pattern

We have $r = 1$ and $K - r > 1$. When the number of test patterns is more than 1, they together produce a "reinforcing" effect as there is more information. So when the $K - r$ test patterns are compared with the correct reference pattern, the distortion between the K patterns would be less, and when they are compared with the wrong reference pattern, the distortion is likely to be much higher. The discriminability between the distortions using the correct and wrong reference patterns and its robustness is likely to increase as we are using more than 1 test pattern. Therefore the recognition accuracy is likely to increase as the number of test patterns ($K - r$) increases.

Case 2: One Test Pattern and Multiple Reference Patterns

In this case we have multiple reference patterns but only one test pattern; i.e., $r > 1$ and $K - r = 1$. This MPDTWalgorithm will be repeated for different vocabulary to recognize the word. For the sake of simplicity consider an example that has only 2 reference patterns (R_1 and R_2) and one test pattern (P_1). We find the MPDTWpath (least distortion path) in the multi dimensional grid between these 3 patterns using the MPDTW algorithm. Project this MPDTW path on any of the planes containing 2 patterns, say P_1 and R_1. (We know that the optimum least distortion path between P_1 and R_1 is given by the DTW algorithm.) The projected MPDTW path on the plane containing P_1 and R_1 need not be same as least distortion path given by the DTW algorithm. Hence it is a suboptimal path and the distortion obtained is also not optimal. So taking the distortion between P_1 and R_1 (or P_1 and R_2) using the DTW algorithm, leads to lower total distortion between the 2 patterns than using a projection of the MPDTW algorithm. This sub optimality is likely to widen with increasing r, the number of reference patterns. This property holds good for incorrectly matched reference patterns also. However, since the distortion is high, the additional increase due to joint pattern matching may not be significant. So it is likely that the MPDTW algorithm will give a poorer discriminative performance than the 2-D DTW algorithm, as the number of reference patterns (r) per class increase. The use of multiple templates is common in speaker dependent ASR applications, to model the pronunciation variability. When the templates of the same class (vocabulary word) are significantly different, the joint recognition is likely to worsen the performance much more than otherwise.

Case 3: Multiple Test and Reference Patterns

This is the most general case of $r > 1$ and $K - r > 1$. From the discussions mentioned in cases 1, 2, and 3, we know that the recognition accuracy is likely to increase as $K - r$ increases and decrease when r increases. When both reference patterns and test patterns are likely to be noisy or distorted, $r > 1$ and $K - r > 1$ will likely to lead to more robust performance.

3. Joint likelihood of multiple speech patterns

Considering left to right stochastic models of speech patterns, we now propose a new method to recognize K patterns jointly by finding their joint multi pattern likelihood, i.e., $P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}/\lambda)$. We assume that the stochastic model is good, but some or all of the test patterns may be distorted due to burst/transient noise or even badly pronounced. We would like to jointly recognize them in an "intelligent" way such that the noisy or unreliable portions of speech are neglected and more weightage is given to the reliable portions of the patterns. Such a solution would be better than single pattern separate recognition.

As before, we denote the K number of observed speech sequences (patterns) $\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}$ of lengths $T_1, T_2, \ldots, T_K$, respectively, where $\mathbf{O}^i_{1:T_i} = (O^i_1, O^i_2, \ldots, O^i_{T_i})$ and $O^i_{t_i}$ is the feature vector of the ith pattern at time frame ti. Let each of these K sequences belong to the same pattern class (spoken word). They are repeated patterns of the same word. In the standard HMM decoding, we have a trellis structure in $K + 1$ dimensions, where K dimensions belong to the K patterns and one dimension belongs to the HMM states. Let $\mathbf{q}$ be any HMM state sequence jointly decoding the K patterns. N is the total number of HMM states.

$$P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}/\lambda) = \sum_{\forall \mathbf{q}} P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}, \mathbf{q}/\lambda) \tag{17}$$

We can calculate the joint multi pattern likelihood only over the optimum HMM state sequence q^* for K patterns as shown:

$$P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}, \mathbf{q}^*/\lambda) = \max_{\forall \mathbf{q}} P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}, \mathbf{q}/\lambda) \tag{18}$$

We can find the optimum HMM state sequence q^* as follows:

$$\begin{aligned} \mathbf{q}^* &= \arg\max_{\mathbf{q}} log\ P(\mathbf{q}/\mathbf{O}^1_{1:T_1}, \ldots, \mathbf{O}^K_{1:T_K}, \lambda) \\ &= \arg\max_{\mathbf{q}} log\ P(\mathbf{q}, \mathbf{O}^1_{1:T_1}, \ldots, \mathbf{O}^K_{1:T_K}/\lambda) \end{aligned} \tag{19}$$

Fig. 7 shows a schematic of two patterns $O^1_{1:T_1}$ and $O^2_{1:T_2}$ and the time alignment of the two patterns along the optimum HMM state sequence $\mathbf{q}^*$ is shown. This is opposite to the case we see in [Lleida & Rose, 2000]. In [Lleida & Rose, 2000], a 3D HMM search space and a Viterbi-like decoding algorithm was proposed for Utterance Verification. In that paper, the two axes in the trellis belonged to HMM states and one axis belongs to the observational sequence. However, here (equation 19) we have K observational sequences as the K axis, and one axis for the HMM states. We would like to estimate one state sequence by jointly decoding the K patterns since we know that the K patterns come from the same class. They are conditionally independent, that is, they are independent given that they come from the

same class. But, there is a strong correlation between the *K* patterns because they belong to the same class. The states in a Left to Right HMM roughly correspond to the stationary phonemes of the pattern and hence use of the same sequence is well justified. The advantage is this is that we can do discriminant operations like frame based voting, etc., as we will be shown later. This formulation is more complicated for Left to Right HMMs with skips or even more general ergodic HMMs.

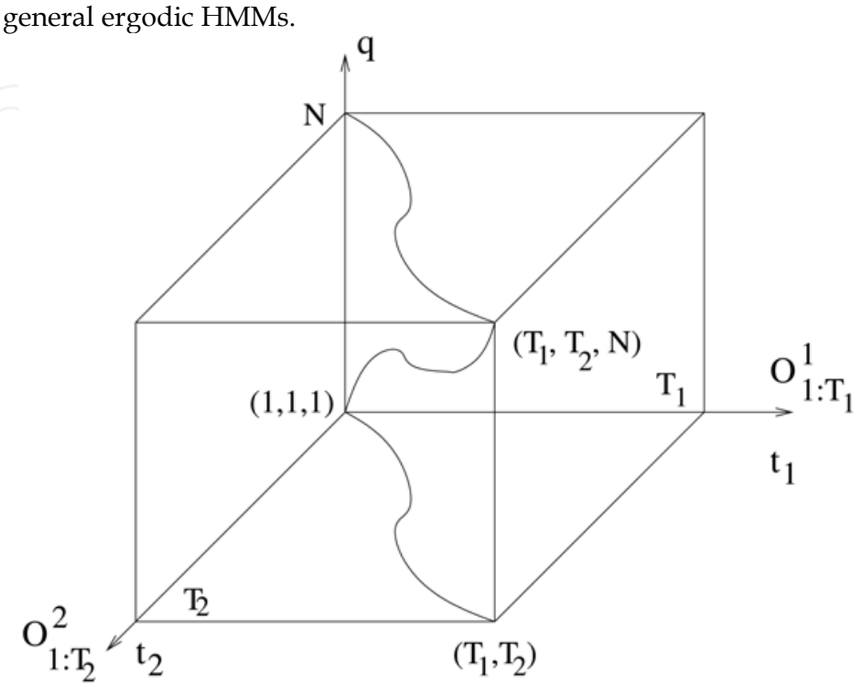

Fig. 7. A multi dimensional grid with patterns $O^1_{1:T_1}$, $O^2_{1:T_2}$ forming a trellis along the *q*-axis, and the optimum state sequence **q***. If *N* = 3 states, then an example **q***could be [1 1 1 2 2 3 3 3].

To find the total probability of *K* patterns given λ - $P(\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}/\lambda)$- we have to traverse through a trellis of *K*+1 dimensions. This leads to a high-dimensional Viterbi search in which both the state transition probabilities as well as local warping constraints of multiple patterns have to be accommodated. We found this to be somewhat difficult and it did not yield consistent results. Hence, the problem is simplified by recasting it as a two stage approach of joint recognition, given the optimum alignment between the various patterns. This alignment between the *K* patterns can be found using the Multi Pattern Dynamic Time Warping (MPDTW) algorithm. This is followed by one of the Multi Pattern Joint Likelihood (MPJL) algorithms to determine the joint multi pattern likelihood and the best state sequence for the *K* patterns jointly [Nair & Sreenivas, 2007, Nair & Sreenivas, 2008 a]. The twostage approach can also be viewed as a hybrid of both non-parametric ASR and parametric (stochastic) ASR, because we use both the non-parametric MPDTW and parametric HMM for speech recognition (Fig. 8). There is also a reduction in the computational complexity and search path from *K* + 1 dimensions to *K* dimensions, because of this two stage approach. We experimented with the new algorithms for both clean speech and speech with burst and other transient noises for IWR and show it to be advantageous. We note that similar formulations are possible for connected word recognition and continuous speech recognition tasks also. We thus come up with solutions to address the first two problems of HMMs using joint evaluation techniques.

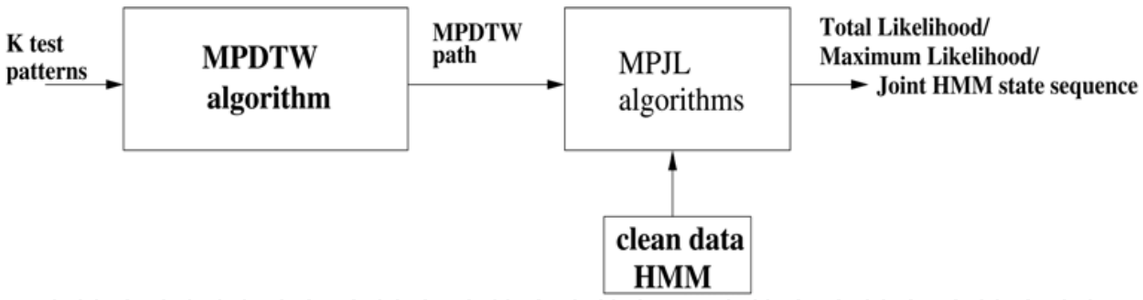

Fig. 8. Joint Likelihood of K patterns.

The MPDTW algorithm finds the least distortion mapping between the K patterns, referred to as the MPDTW path. MPDTW path gives us a handle to evaluate similar/dissimilar components of the K patterns visa-vis the HMM. Note that here all the K patterns are test patterns and there is no reference pattern (the three cases of section 2.2 does not apply). Let ϕ be the MPDTW mapping between K patterns and $\phi(t) = (t_1, \ldots, t_K)$ where $(t_1, \ldots, t_K)$ is a K tuple coordinate sequence along the MPDTW optimum path, in the K dimensional rectangular grid. Endpoint errors are not considered here since the MPDTW algorithm can accommodate relaxed end point constraints separately. Since the MPDTW path is the least distortion path between K patterns of the same class, the path groups similar sounding portions from the K patterns. Each point on the MPDTW path represents K feature vectors, each coming from one pattern.

Since our goal is to determine the joint likelihood of all the K patterns given the HMM, we can consider the MPDTW path $\phi(t)$, $1 \le t \le T$ as a 1-D evolution of the multiple patterns. Let T be the total number of distinct points in the MPDTW path and $T \ge \max_k\{T_k\}$. Thus, we can use $\phi(t)$ as a virtual path for the evolution of multiple patterns and construct the HMM trellis; the trellis comprises of N states × T sequence of K-vector sets. The joint likelihood of K vector sequences as seen along $\phi(t)$ is determined by using one of the MPJL algorithms. The MPJL algorithms are divided into two versions of determining either the total probability or the ML state sequence probability; Constrained Multi Pattern Forward Algorithm (CMPFA), Constrained Multi Pattern Backward Algorithm (CMPBA) determine total probability using either the forward or backward algorithm, and Constrained Multi Pattern Viterbi Algorithm (CMPVA) for the Viterbi score (ML state sequence). These algorithms are called "constrained" because their time path is fixed by the MPDTW algorithm.

The main feature of the MPJL algorithms is to calculate the total probability in an "intelligent" manner such that we are making use of the "best" information available and avoiding (giving less weight) to the noisy or unreliable information among multiple patterns.

3.1 Constrained Multi Pattern Forward and Backward algorithms (CMPFA and CMPBA)

The CMPFA and CMPBA are used to calculate the total joint probability of the K patterns through all possible HMM state sequences. Following the terminology of a standard HMM [Rabiner & Juang, 1993] for the forward algorithm, we define the forward variable $\alpha_{\phi(t)}(j)$ along the path $\phi(t)$; i.e.,

$$\alpha_{\phi(t)}(j) = P(\mathbf{O}^1_{1:t_1}, \mathbf{O}^2_{1:t_2}, \ldots, \mathbf{O}^K_{1:t_K}, q_{\phi(t)} = j/\lambda) \tag{20}$$

where $q_{\phi(t)}$ is the HMM state at $t \rightarrow \phi(t)$, λ is the HMM model with states $j \in 1 : N$. As in the forward algorithm, we can determine $\alpha_{\phi(t)}(j)$ recursively leading to the total probability.

3.1.1 CMPFA-1

Let us define the MPDTW path transition vector, $\Delta\phi(t) = \phi(t) - \phi(t-1) = (\Delta t^1_1, \Delta t^2_2, \ldots, \Delta t^K_K)$. Depending on the local constraints chosen in the MPDTW algorithm, $\Delta\phi(t)$ can be a K dimensional vector of only 0's and 1's; e.g., $\Delta\phi(t)$ = [0, 1, 1, 0, . . . , 1]. $\Delta\phi(t)$ will comprise of at least one non-zero value and a maximum of K non-zero values. (The [0,1] values are due to the non-skipping type of local constraints in MPDTW. The skipping-type can introduce higher range such as [0,1,2] or [0,1,2,3].) Let $S_{\phi(t)} = \{O^i_{t_i} | \Delta t^i_i \neq 0, i = 1, 2, \ldots, K\}$ be the set of vectors that have been mapped together by the MPDTW at $\phi(t)$. Let $\{O_{\phi(t)}\} = (O^m_{t_m}, \ldots, O^n_{t_n})$ such that $(O^m_{t_m}, \ldots, O^n_{t_n})$ are all the feature vectors in the set $S_{\phi(t)}$. $\{O_{\phi(t)}\}$ is a subset of the vectors $(O^1_{t_1}, O^2_{t_2}, \ldots, O^K_{t_K})$, retaining only those $O^k_{t_k}$ whose Δt^k_k are non-zero. The set $S_{\phi(t)}$ and $\{O_{\phi(t)}\}$ can have a minimum of one feature vector and a maximum of K feature vectors. Let $\phi(t)^* = \{t_i | \Delta t^i_i \neq 0, i = 1, 2, \ldots, K\}$.The recursive equation for the evaluation of $\alpha_{\phi(t)}(j)$ along all the grid points of the trellis is given by (derivation given in appendix $A1$):

$$\alpha_{\phi(t)}(j) = \sum_{i=1}^{N} \left[\alpha_{\phi(t-1)}(i).a_{ij} \right] .b_j \left(\{O_{\phi(t)}\} \right), \tag{21}$$

t = 2, 3, . . . , T , j = 1, 2, . . . , N. a_{ij} is the state transition probability from state i to state j (as in standard HMM), $b_j(\{O_{\phi(t)}\})$ is the joint likelihood of $\{O_{\phi(t)}\}$ being emitted by state j. It is the same as joint likelihood of all the vectors $(O^m_{t_m}, \ldots, O^n_{t_n})$ emitted by state j, where $(O^m_{t_m}, \ldots, O^n_{t_n})$ consist of all the feature vectors in a set $S_{\phi(t)}$. Thus, a HMM state-j can emit a variable number of vectors from the K patterns, corresponding to the number of non-zero values in the $\Delta\phi(t)$ vector. Thus, the number of feature vectors each state emits ranges from 1 to K and it is a function of the MPDTW path. But, when the recursive computation of $\alpha_{\phi(t)}$ reaches $\alpha_{\phi(T)}$, each state j would have emitted the exact number of multi-pattern feature vectors = ($T_1 + T_2 + \ldots + T_K$), irrespective of the MPDTW path.

We examine a few alternate methods to calculate the joint likelihood $b_j(\{O_{\phi(t)}\})$, shown in section 3.3. We know from HMM theory, that a HMM state can emit one feature vector at any given time. However in our case, each HMM state emits 1 upto K feature vectors at each time coordinate $\phi(t)$ and a given state j. However, we calculate the joint likelihood $b_j(\{O_{\phi(t)}\})$ by normalizing it (shown in section 3.3) such that it is comparable to a likelihood of a single vector emitted by a HMM state. This can be interpreted as inherently performing recognition for one virtual pattern created from the K similar patterns. Thus, we can consider the values of transition probability a_{ij} and state initial probability π_i as the same as that used in single pattern recognition task.

An example of MPDTW path, when there are only two patterns (K = 2) is shown in Fig. 9. The t1 time axis is for pattern $O^1_{1:T_1}$ and t2 time axis is for pattern $O^2_{1:T_2}$. We fit a layer of HMM states (of a class) on this path. For simplicity, let us consider that there is only one state j = 1. Now we traverse along this MPDTW path. In the trellis, for CMPFA-1, at time instant (1,1) feature vectors (O^1_1, O^2_1) are emitted by state j. At time instant (2,2) state j emits

vectors O_2^1 and O_2^2. At time (3,2) state *j* emits only one vector O_3^1 and not O_2^2, as O_2^2 was already emitted at time (2,2). So we are exactly recognizing all the *K* patterns such that there is no reuse of feature vectors. The total number of feature vectors emitted at the end of the MPDTW path by each state in this example will be exactly equal to $T_1 + T_2$.

3.1.2 CMPFA-2

CMPFA-2 is an approximate solution for calculating the total probability of the *K* patterns given HMM λ but it has some advantages over CMPFA-1. In CMPFA-2, a recursive solution (equation (22)) is proposed to calculate the value of $\alpha_{\phi(t)}(j)$. This solution is based on the principle that each HMM state *j* can emit a fixed (*K*) number of feature vectors for each transition. So, it is possible that some of the feature vectors from some patterns are reused based on the MPDTWpath. This corresponds to stretching each individual pattern to a fixed length *T* (which is ≥ $\max_k\{T_k\}$) and then determining the joint likelihood for the given HMM. Thus, we are creating a new virtual pattern which is a combination of all the *K* patterns (with repetitions of feature vectors possible) of length equal to that of the MPDTWpath. The HMM is used to determine the likelihood of this virtual pattern. The total number of feature vectors emitted in this case is *K.T* . Considering the forward recursion as before,

$$\alpha_{\phi(t)}(j) = \sum_{i=1}^{N} [\, \alpha_{\phi(t-1)}(i).a_{ij}].b_j(O_{t_1}^1, \ldots, O_{t_K}^K) \tag{22}$$

t = 2, 3, . . . , T , j = 1, 2, . . . , N, where N is the number of states in the HMM. a_{ij} is the state transition probability from state i to state j, π_j is the state initial probability at state j, $b_j(O_{t_1}^1, \ldots, O_{t_K}^K)$ is the joint likelihood of the observations $O_{t_1}^1, \ldots, O_{t_K}^K$ generated by state j. The various methods to calculate this joint likelihood $b_j(O_{t_1}^1, \ldots, O_{t_K}^K)$ is shown in section 3.3.

Let us consider again the example of Fig. 9. In CMPFA-2, each HMM state emits a fixed number (*K*) of feature vectors. In CMPFA-2, at time instant (1,1), feature vectors (O_1^1, O_1^2) are emitted by state *j*. At time instant (2,2), state *j* emits vectors O_2^1 and O_2^2. At time instant (3,2), state *j* emits vectors O_3^1 and O_2^2. At time instant (4,2), state *j* emits vectors O_4^1 and O_2^2. Here we see that vector O_2^2 is emitted 3 times. So some feature vectors are repeated in CMPFA-2 based on the MPDTW path. So by using CMPFA-2, the HMMs are not emitting the *K* patterns in an exact manner.

The initialization for both CMPFA-1 and CMPFA-2 is the same, i.e.,

$$\alpha_{\phi(1)}(j) = \pi_j.b_j(O_1^1, \ldots, O_1^K) \tag{23}$$

where j = 1, 2, . . . ,N, π_j is the state initial probability at state j (and it is assumed to be same as the state initial probability given by the HMM), $b_j(O_{t_1}^1, \ldots, O_{t_K}^K)$ is the joint likelihood of the observations $O_{t_1}^1, \ldots, O_{t_K}^K$ generated by state j.

The termination of CMPFA-1 and CMPFA-2 is also same:

$$P^* = \sum_{i=1}^{N} \alpha_{\phi(T)}(i), \tag{24}$$

where *P** is the total joint likelihood of *K* patterns.

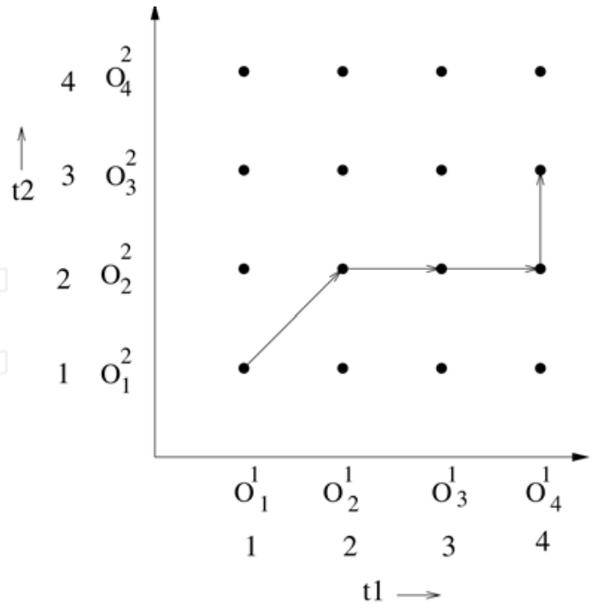

Fig. 9. An example path.

3.1.3 CMPBA-1 and CMPBA-2

We can easily construct backward recursion algorithms similar to the forward recursion of the previous section, which leads to CMPBA- 1 and CMPBA-2. We define $\beta_{\phi(t)}(j)$ as in equation (25).

$$\beta_{\phi(t-1)}(j) = P(O^1_{t_1:T_1}, \ldots, O^K_{t_K:T_K} / q_{\phi(t-1)} = j, \lambda) \tag{25}$$

For CMPBA-1 can write similar recursive equation as in CMPFA-1, by using the backward cumulative probability.

$$\beta_{\phi(t-1)}(i) = \sum_{j=1}^{N} a_{ij}.\beta_{\phi(t)}(j).b_j(\{O_{\phi(t)}\}) \tag{26}$$

t = T, . . . , 2, i = 1, 2, . . . ,N, where N is the number of states in the HMM, and the rest of the terms are as defined in section 3.1.1. Again, CMPBA-2 is an approximation to calculate $\beta_{\phi(t)}(j)$ and is similar to CMPFA-2. We define the recursive solution for calculating $\beta_{\phi(t)}(j)$ as follows.

$$\beta_{\phi(t-1)}(i) = \sum_{j=1}^{N} a_{ij}.\beta_{\phi(t)}(j).b_j(O^1_{t_1}, \ldots, O^K_{t_K}) \tag{27}$$

t = *T*, . . . , 2, *i* = 1, 2, . . . , *N*, where *N* is the number of states in the HMM.

3.2 Constrained Multi Pattern Viterbi Algorithms (CMPVA-1 and CMPVA-2)

Unlike the CMPFAs and CMPBAs which consider all state sequences through the HMM trellis, the CMPVA is evaluated for the probability along the maximum likelihood (ML) state sequence: $\mathbf{q}^* = q^*_{\phi(1)}, q^*_{\phi(2)}, \ldots, q^*_{\phi(T)}$, where $q^*_{\phi(t)}$ is the optimum state at time $\phi(t)$. We

define $\delta_{\phi(t)}(j)$ as the log likelihood of the first $\phi(t)$ observations of the K patterns through the best partial state sequence up to the position $\phi(t-1)$ and $q_{\phi(t)} = j$ along the path $\phi(t)$; i.e.,

$$\delta_{\phi(t)}(j) = \max_{q_{\phi(1):\phi(t-1)}} log\ P(\mathbf{O}^1_{1:t_1}, \ldots, \mathbf{O}^K_{1:t_K}, q_{\phi(1):\phi(t-1)}, q_{\phi(t)} = j/\lambda) \tag{28}$$

where $q_{\phi(1):\phi(T)} = q_{\phi(1)}, q_{\phi(2)}, \ldots, q_{\phi(T)}$. The recursive equation for CMPVA-1 (similar to CMPFA-1) is:

$$\delta_{\phi(t)}(j) = \max_i [\ \delta_{\phi(t-1)}(i) + log\ a_{ij}] + log\ b_j(\{O_{\phi(t)}\}) \tag{29}$$

with $t = 2, 3, \ldots, T$ and $j = 1, 2, \ldots, N$, N is the number of HMM states. The terminology is similar to that used in equation (21). The recursive solution for CMPVA-2 (similar to CMPFA-2) is:

$$\delta_{\phi(t)}(j) = \max_i [\ \delta_{\phi(t-1)}(i) + log\ a_{ij}] + log\ b_j(O^1_{t_1}, \ldots, O^K_{t_K}) \tag{30}$$

$t = (2, 3, \ldots, T)$, $j = 1, 2, \ldots, N$
Initialization for both CMPVA-1 and CMPVA-2 is done as follows:

$$\delta_{\phi(1)}(j) = log\ \pi_j + log\ b_j(O^1_1, \ldots, O^K_1), \qquad j = 1, 2, \ldots, N \tag{31}$$

The path backtracking pointer $\psi_{\phi(t)}(j)$ for CMPVA-1 and CMPVA-2 is:

$$\psi_{\phi(t)}(j) = \arg\max_i [\ \delta_{\phi(t-1)}(i) + log\ a_{ij}] \tag{32}$$

where $t = 2, 3, \ldots, T$ and $j = 1, 2, \ldots, N$.
The ML joint likelihood for both CMPVA-1 and CMPVA-2 is determined by:

$$P^* = \max_i \{\delta_{\phi(T)}(i)\} \tag{33}$$

Path Backtracking is done to find the optimum state sequence.

$$q^*_{\phi(t)} = \psi_{\phi(t+1)}(q^*_{\phi(t+1)}), \qquad t = T-1, \ldots, 1 \tag{34}$$

An example of a HMM state sequence along the MPDTW path is shown in Fig. 10.
For robust IWR, we use either CMPFA or CMPBA or CMPVA to calculate the probability P^* of the optimal sequence. For simplicity let us group CMPFA-1, CMPBA-1, CMPVA-1 as CMP?A-1 set of algorithms; and CMPFA-2, CMPBA-2, CMPVA-2 as CMP?A-2 set of algorithms.

3.3 Feature vector weighting

In missing feature theory [Cooke et al., 1994, Cooke et al., 2001], we can identify the unreliable (noisy) feature vectors, and either ignore them in subsequent processing, or they can be filled in by the optimal estimate of their putative value [Raj & Stern, 2005]. Similar to this approach we determine the joint likelihoods for any of the six algorithms discussed earlier, by differentiating as to which portions of the speech patterns are unreliable. We can

give a lesser or zero weighting to the unreliable (noisy) feature vectors and a higher weighting to the corresponding reliable ones from the other patterns. Fig. 11 shows an example of two speech patterns affected by burst noise. The strategy is to give a lesser weight to the regions of speech contaminated by burst noise and the corresponding clean speech in the other pattern should be given a higher weight. This can be interpreted as a form of voting technique which is embedded into HMM decoding. We have considered alternate criteria for weighting the feature vectors, to achieve robustness to transient, bursty noise in the test patterns.

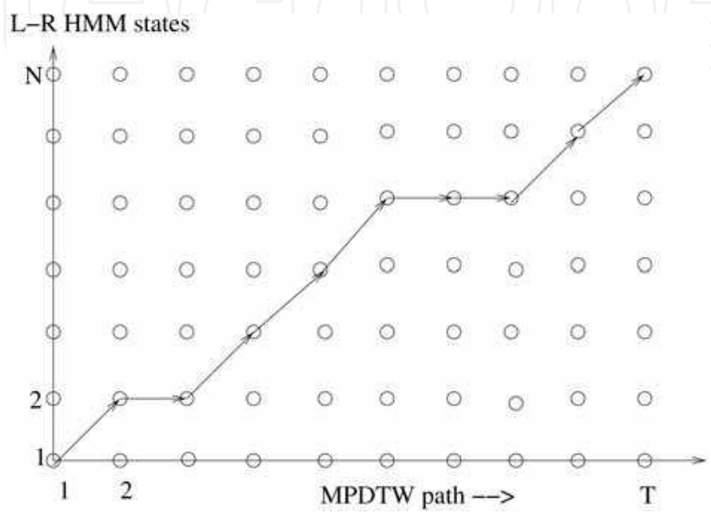

Fig. 10. Example of a HMM state sequence along MPDTW path for Left-Right HMM.

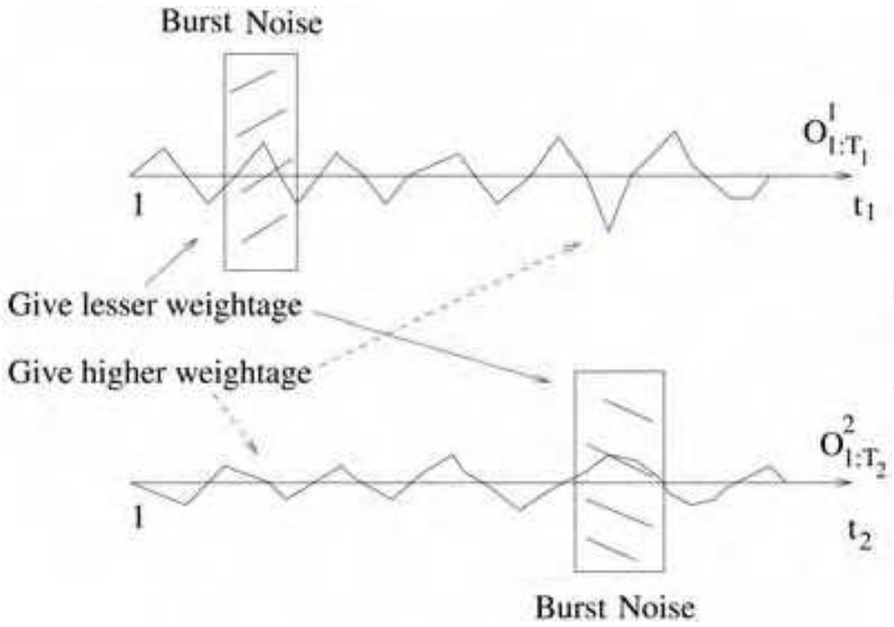

Fig. 11. Two speech patterns $O^1_{1:T_1}$ and $O^2_{1:T_2}$ affected by burst noise.

To determine the joint likelihood of emitting the set of feature vectors in a given state, we can resort to alternate formulations. Since the path traversed along the K-dimensional time

axes is already optimized, how we choose $b_j(\{O_{\phi(t)}\})$ (or $b_j(O^1_{t_1}, O^2_{t_2},\ldots, O^K_{t_K})$) affects only the total/ML joint likelihood P^* (of equations 24 and 32) and the ML state sequence (of equation 33). We define various discriminant criteria for calculating $b_j(\{O_{\phi(t)}\})$ (in equations 21, 26, 29) and $b_j(O^1_{t_1}, O^2_{t_2},\ldots, O^K_{t_K})$ (in equations 22, 27, 30) as follows:

3.3.1 Discriminant Criteria - average (DC-avg)

We know that $\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2},\ldots, \mathbf{O}^K_{1:T_K}$ are different speech patterns which come from the same class (word) and uttered by the same speaker. Given that they come from the same class, and uttered independently, we don't use any discriminating criteria, and simplify use of the vectors as they are. Even though the feature vectors $O^1_{t_1}, O^2_{t_2},\ldots, O^K_{t_K}$ come from the same class, we can assume that they are independent if it is given that they occur from the same state j, so as to compute the joint likelihood of the vectors being emitted from the HMM. So, we can express the joint probability term in CMP?A-1

$$b_j(\{O_{\phi(t)}\}) = [b_j(O^m_{t_m})\ldots b_j(O^n_{t_n})]^{1/r} \tag{35}$$

where $(O^m_{t_m},\ldots,O^n_{t_n})$are all the feature vectors in the set $S_{\phi(t)}$ as mentioned in section 3.1.1 and $b_j(O^i_{t_i})$ is the state-j emission probability for the HMM (probability of vector $O^i_{t_i}$ emitted by state j given the HMM) and r is the cardinality of the set $S_{\phi(t)}$.
Similarly, CMP?A-2, since all the patterns are used at each time, the following equation is proposed.

$$b_j(O^1_{t_1},\ldots,O^K_{t_K}) = [b_j(O^1_{t_1}).b_j(O^2_{t_2})\ldots b_j(O^K_{t_K})]^{1/K} \tag{36}$$

The independence assumption is justified because successive vectors in a pattern are only linked through the underlying Markov model and the emission densities act only one symbol at a time. The geometric mean using power of $1/r$ or $1/K$ normalizes the use of r or K vectors emitted by a HMM state, comparable to a single vector likelihood. Therefore we can use a_{ij}'s and π_i's that are defined as in single-pattern HMM. If $O^i_{t_i}$ is emitted from its actual state j from the correct HMM model λ, we can expect that $b_j(O^i_{t_i})$ to have a higher value than that if $O^i_{t_i}$ is emitted from state j of the wrong model. And taking the product of all the $b_j(O^i_{t_i})$ brings in a kind of "reinforcing effect". Therefore, while doing IWR, the values of joint likelihood P^* using the correct model and the P^* using the mismatched models, is likely to widen. Therefore we can expect better speech recognition accuracy to improve. Even if some of the K vectors are noisy, the reinforcing effect will improve speech recognition because the rest of the vectors are clean.

3.3.2 Discriminant Criteria - maximum (DC-max)

In the first discriminant criteria DC-avg, we estimate the output probability by considering the vectors emitted at each state, in an average sense; i.e., geometric mean of the independent vector probabilities is considered. Instead of the average, we consider the maximum of the independent probabilities from the set $S_{\phi(t)}$. Of course, $S_{\phi(t)}$ itself would be different for the two classes of algorithms for grid traversing (CMP?A-1 and CMP?A-2). Thus, we can express

$$b_j(\{O_{\phi(t)}\}) = \max(b_j(O^m_{t_m}),\ldots,b_j(O^n_{t_n})) \tag{37}$$

$$b_j(O_{t_1}^1, \ldots, O_{t_K}^K) = \max(b_j(O_{t_1}^1), b_j(O_{t_2}^2), \ldots, b_j(O_{t_K}^K)) \tag{38}$$

Because of the maximum (max) operation, we can expect the likelihoods in DC-max to be higher than the respective ones using DC-avg. In terms of the virtual pattern interpretation, the sequence of the T length patterns are composed of the most probable vectors corresponding to the HMM-λ. If the speech patterns are affected by noise, we expect DC-max to give better discrimination than DCavg. However, for the case of clean speech, it is possible that DC-max will reduce speech recognition accuracy than DC-avg because the max operation will also increase the likelihood P^* for the mismatched model and bring it closer to the P^* of the correct model. Also, this reinforcing effect will be absent in DC-max. So we would prefer to use DC-max when the speech patterns are noisy or badly spoken, and for the clean speech case we would prefer DC-avg.

3.3.3 Discriminant Criteria - threshold 1 (DC-thr1)

Instead of using SNR, to determine which feature vectors are reliable or unreliable as in missing feature theory, we propose a novel distortion measure called the joint distance measure. For CMP?A-1 set of algorithms, the joint distance measure is defined as, $d(\phi(t)^*) = \sum_{O_{t_i}^i \in S_{\phi(t)}} d(O_{t_i}^i, C_{\phi(t)^*})$, where $C_{\phi(t)^*}$ is the centroid of all the vectors in $S_{\phi(t)}$ as in section 3.1, and $d(O_{t_i}^i, C_{\phi(t)^*})$ is the Euclidean distance between $O_{t_i}^i$ and $C_{\phi(t)^*}$. We define $b_j(\{O_{\phi(t)}\})$ as:

$$b_j(\{O_{\phi(t)}\}) = \begin{cases} \left[\prod_{O_{t_i}^i \in S_{\phi(t)}} b_j(O_{t_i}^i)\right]^{\frac{1}{r}} & \text{if } d(\phi(t)^*) < \gamma, r = |S_{\phi(t)}| \\ \max_{O_{t_i}^i \in S_{\phi(t)}} b_j(O_{t_i}^i) & \text{if } d(\phi(t)^*) \geq \gamma \end{cases} \tag{39}$$

where γ is a threshold, r is the cardinality of the set $S_{\phi(t)}$.

In equation (39), if we choose $\gamma = \infty$, then $b_j(\{O_{\phi(t)}\})$ is always equal to $\prod_{O_{t_i}^i \in S_{\phi(t)}} b_j(O_{t_i}^i)$ (product operation), and when $\gamma < 0$, then it is always equal to $\max_{O_{t_i}^i \in S_{\phi(t)}} \{b_j(O_{t_i}^i)\}$ (max operation). The first option of $d(\phi(t)^*) < \gamma$ is provided to take care of the statistical variation among the patterns, even without noise. If the distortion is low ($< \gamma$) it implies no noise among the patterns; then we consider all the vectors to be reliable data and set $S_{\phi(t)}$ come from the same class, we can assume that $b_j(\{O_{\phi(t)}\})$ is determined as in DC-avg. When the distortion is high ($> \gamma$), it could be due to misalignment as well as distortion in the patterns. So, we choose only one vector out of all the possible r vectors, which gives the maximum probability in state j. The rest of the $r-1$ vectors are not considered for joint likelihood. This is expressed as the max operation.

For CMP?A-2 set of algorithms, the set $S_{\phi(t)}$ includes all the vectors from the K patterns at $\phi(t)$ and accordingly, the joint distance measure is defined as, $d(t_1, \ldots, t_K) = \sum_{i=1}^{K} d(O_{t_i}^i, C_{\phi(t)^*})$, where $C_{\phi(t)^*}$ is the centroid of the K vectors $(O_{t_1}^1, O_{t_2}^2, \ldots, O_{t_K}^K)$, and $d(O_{t_i}^i, C_{\phi(t)^*})$ is the Euclidean distance between $O_{t_i}^i$ and $C_{\phi(t)^*}$. Rest are similar to equation 39.

If the speech patterns are affected by noise, we would expect that the max operation to give better recognition results and for the case of well articulated clean speech, we expect the

product operation to give better results. We need to select the threshold such that it is optimum for a particular noisy environment.

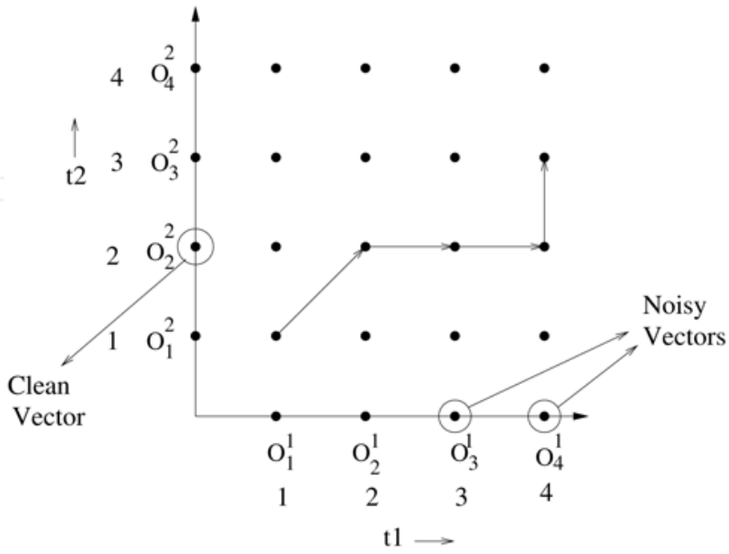

Fig. 12. MPDTW path for K = 2; vector O_2^2 is clean and vectors O_3^1 and O_4^1 are noisy

3.3.4 Discriminant Criteria - threshold 2 (DC-thr2)

In DC-thr1, while doing the max operation, we are taking only the best pattern. In practice a variable number of patterns could be noisy and we would like to use the max operation only to omit the noisy patterns and use the product operation for the rest of the patterns. So we choose only pairwise distortion between two vectors at a time and define a new criteria for the joint likelihood.

Let $1 \le a, b \le K$ be the indices of vectors belonging to the K patterns. For the CMP?A-1 set, we define the clean (undistorted) set of vectors as Z, such that $a, b \in Z$ iff $d(O_{t_a}^a, O_{t_b}^b) < \gamma$, where $d(O_{t_a}^a, O_{t_b}^b)$ is the Euclidean distance between $O_{t_a}^a$ and $O_{t_b}^b$. Let Z be the set of remaining vector indices, such that $Z \cup \overline{Z} = \{m | O_{t_m}^m \in S_{\phi(t)}\}$. Let R is the cardinality of the set $S_{\phi(t)}$ and we can search all pairs of vectors among R exhaustively, i.e., $R(R-1)/2$ combinations, since R is usually small ($R \sim 2, 3$).

$$b_j(O_{t_m}^m, \ldots, O_{t_n}^n) = \begin{cases} \prod_{\{i|i \in Z\}} \left[b_j(O_{t_i}^i)\right]^{\frac{1}{r}} \\ \text{if } Z \neq 0 \\ \max_{\{i|i \in Z \cup \overline{Z}\}} \left(b_j(O_{t_k}^k)\right) \\ \text{if } Z = 0 \end{cases} \tag{40}$$

where r is the cardinality of set Z, and 0 stands for null set.

For the CMP?A-2 set of algorithms $S_{\phi(t)} = (O_{t_1}^1, O_{t_2}^2, \ldots, O_{t_K}^K)$, be the full set of K-pattern vectors and all the steps to compute equation 40 is followed exactly.

Note that DC-thr2 becomes same as DC-thr1 if number of patterns K is equal to 2. There is one disadvantage in this criteria. Consider three vectors $O_{t_1}^1$, $O_{t_2}^2$, $O_{t_3}^3$. Let $O_{t_1}^1$ and $O_{t_2}^2$ be affected by similar kind of noise and let $O_{t_3}^3$ be clean. Then it is possible that the noisy

vectors $O_{t_1}^1$ and $O_{t_2}^2$ are pooled together in set Z and the clean vector $O_{t_3}^3$ is ejected. This can affect speech recognition performance in a negative way. Thus the clean vectors may be removed from probability computation. However this would be a very rare possibility.

3.3.5 Discriminant Criteria - weighted (DC-wtd)

The previous two criteria DC-thr1 and DC-thr2 have one difficulty of choosing a proper threshold γ. We consider a new weighting criteria without the need for the threshold. For the CMP?A-1 set of algorithms, the joint probability is defined as follows:

$$b_j(\{O_{\phi(t)}\}) = \prod_{O_{t_i}^i \in S_{\phi(t)}} b_j(O_{t_i}^i)^{\frac{b_j(O_{t_i}^i)}{\sum_{O_{t_i}^i \in S_{\phi(t)}} (b_j(O_{t_i}^i))}} \tag{41}$$

For CMP?A-2 set of algorithms, $b_j(O_{t_1}^1, \ldots, O_{t_K}^K)$ is defined as follows:

$$b_j(O_{t_1}^1, \ldots, O_{t_K}^K) = \prod_{i=1}^{K} b_j(O_{t_i}^i)^{\frac{b_j(O_{t_i}^i)}{\sum_{i=1}^{K} (b_j(O_{t_i}^i))}} \tag{42}$$

In this discriminant weighting, the individual probabilities are weighted according to the proportion of their probability value compared with that of all the vectors in $S_{\phi(t)}$. Thus distinctly higher probability values are magnified and distinctly lower probabilities are penalized. The above equations are basically a weighted geometric mean of the $b_j(O_{t_i}^i)$'s. Thus, when the values of $b_j(O_{t_i}^i)$'s are close to each other, then DC-wtd becomes close to the product operation of DC-thr1 and if the various values $b_j(O_{t_i}^i)$'s are very different, then DC-wtd becomes close to max operation of DC-thr1. DC-wtd behaves somewhat similar to DC-thr1 (when DC-thr1 is set with optimum threshold). The main advantage of using DC-wtd is that we don't need to set any threshold. We expect this type of soft-thresholding may be useful in some applications.

3.4. Analysis of CMP?A-1 versus CMP?A-2

Now we analyze which of these two set of algorithms, CMP?A-1 and CMP?A-2, is better and under what conditions. An example of MPDTW path, when there are only two patterns (K = 2) is shown in Fig. 12. The t1 time axis is for pattern $O_{1:T_1}^1$ and t2 time axis is for pattern $O_{1:T_2}^2$. We fit a layer of HMMstates (of a class) on this path. For simplicity, let us consider that there is only one state j = 1. Now we traverse along this MPDTWpath. In the example shown in Fig. 12, we assume that the vector O_2^2 is clean and the vectors O_3^1 and O_4^1 are noisy or badly articulated. Let us use DC-thr1 to calculate joint probability and choose a low value for the threshold, so that the max operation dominates. Using CMP?A-2, since O_2^2 is re-emitted (by state j) at time instants (3,2) and (4,2), the max operation will be most likely used as the joint distortion measure will most likely be above the threshold. So only the clean O_2^2 vector will be used to calculate joint probability. However in the case of CMP?A-1, since O_2^2 is emitted only once at time instant (2,2) and not emitted at time instants (3,2) and (4,2), only the noisy vectors O_3^1 and O_4^1, contribute to the calculation of joint probability. This affects P^*. So in this case the recognition accuracy is likely to decrease if we use CMP?A-1 when compared to CMP?A-2.

Now we consider an other case (Fig. 13) when we are using DC-thr1 and the value of the threshold is very high, so that the product operation dominates. Let vector *O*22 be noisy or badly articulated and vectors O_3^1 and O_4^1 be clean. Since the product operation will mostly be used, using CMP?A-2, the noisy vector O_2^2 will affect the calculation of the joint probability at time instants (3,2) and (4,2) as it is re-emitted. Now using CMP?A-1, as vector O_2^2 is not re-emitted, only the clean vectors O_3^1 and O_4^1 contribute to the calculation of joint probability. So CMP?A-1 is expected to give better speech recognition accuracy than CMP?A-2.

For the case of clean, well articulated speech, CMP?A-1 is expected to perform better than CMP?A-2 as it does not reuse any feature vector. This is true when we use DC-thr1 at lower values of threshold. At higher (sub optimal) values of threshold, CMP?A-2 could be better. If DC-wtd is used, we expect that using CMP?A-1 would give better recognition accuracy than CMP?A-2 for well articulated clean speech and worse values for speech with burst/transient noise or speech with bad articulation. This is because DC-wtd behaves similar to DC-thr1 when the threshold of DC-thr1 is optimum. Finally we conclude that if we look at the best performances of CMP?A-1 and CMP?A-2, CMP?A-2 is better than CMP?A-1 for noisy speech (burst/transient noise), and CMP?A-1 is better than CMP?A-2 for clean speech.

The recognition accuracy is expected to increase with the increase in the number of test patterns *K*.

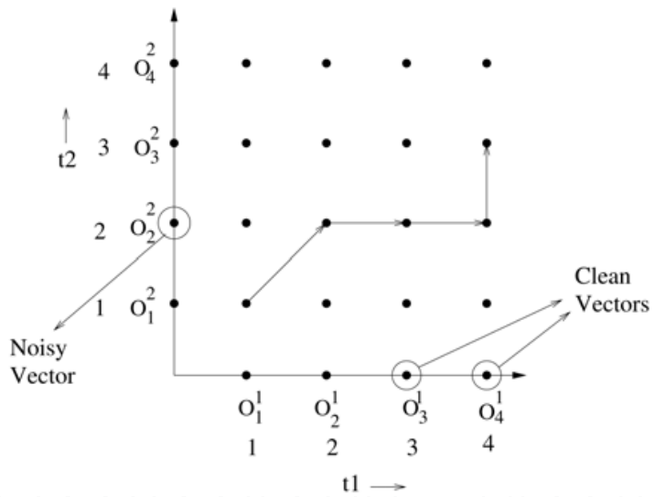

Fig. 13. MPDTW path for *K* = 2; vector O_2^2 is noisy and vectors O_3^1 and O_4^1 are clean

Fig. 14. Standard HMM Training.

4. Selective HMM training (SHT)

This is the next part of the joint multi-pattern formulation for robust ASR. We have first addressed evaluation and decoding tasks of HMM for multiple patterns. Now we consider the benefits of joint multi-pattern likelihood in HMM training. Thus, we would have

addressed all the three main tasks of HMM, to utilize the availability of multiple patterns belonging to the same class. In the usual HMM training, all the training data is utilized to arrive at a best possible parametric model. But, it is possible that the training data is not all genuine and therefore have labeling errors, noise corruptions, or plain outlier examples. In fact, the outliers are addressed explicitly in selective HMM training papers. We believe that the multi-pattern formulation of this chapter can provide some advantages in HMM training also.

Typically, in HMM training the Baum-Welch algorithm [Baum & Petrie, 1966, Baum & Egon, 1967, Baum & Sell, 1968, Baum et al., 1970, Baum, 1972] is used (Fig. 14). We would like to extend it to use the concepts of joint multi-pattern likelihood. Let us refer to this as selective training, in which the goal is to utilize the best portions of patterns for training, omitting any outliers. In selective training, we would like to avoid the influence of corrupted portions of the training patterns, in determining the optimum model parameters. Towards this, virtual training patterns are created to aid the selective training process. The selective training is formulated as an additional iteration loop around the HMM Baum-Welch iterations. Here the virtual patterns are actually created. The virtual patterns can be viewed as input training patterns that have been subjected to "filtering" to deemphasize distorted portions of the input patterns. The filtering process requires two algorithms that we have proposed earlier, viz., MPDTW and CMPVA. The CMPVA uses the MPDTW path as a constraint to derive the joint Viterbi likelihood of a set of patterns, given the HMM λ. CMPVA is an extension of the Viterbi algorithm [Viterbi, 1967] for simultaneously decoding multiple patterns, given the time alignment. It has been shown in [Nair & Sreenivas, 2007, Nair & Sreenivas, 2008 a] that these two algorithms provide significant improvement to speech recognition performance in noise.

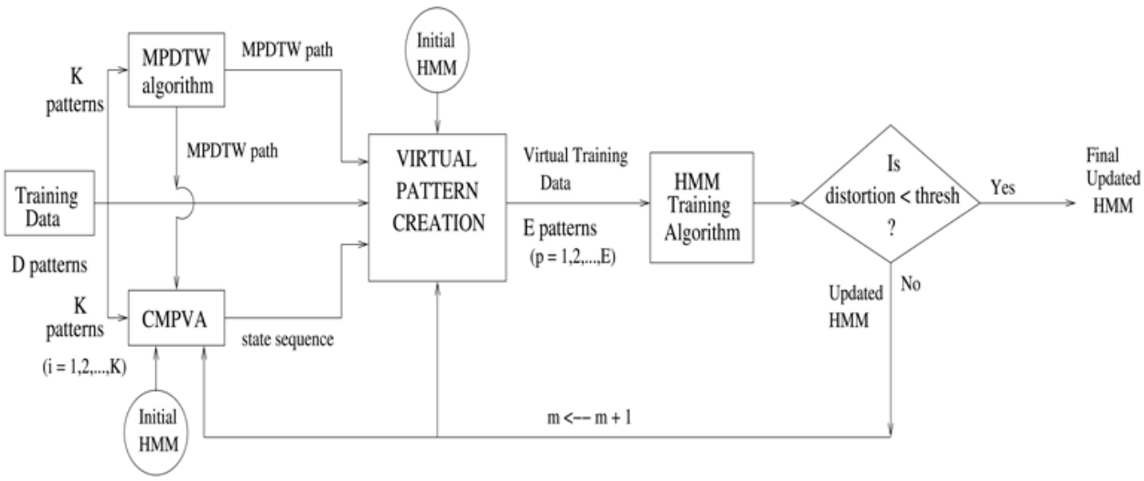

Fig. 15. Selective HMM Training.

A block diagram of the new selective HMM training scheme using virtual patterns is shown in Fig. 15. Let there be D number of training patterns (tokens) in the training data, all of them labeled as belonging to the same class (e.g. words). Let us choose K subset of patterns from these D patterns (here $2 \leq K \leq D$). We know that the K patterns come from the same class and there is strong correlation between them, although they are uttered independently. The K patterns are used to create one virtual pattern by taking advantage of the fact that they are correlated. Then the next K patterns we choose from the pool of D patterns are used

to create another virtual pattern. Let the total maximum number of virtual patterns we can create be equal to *J*. These virtual patterns are now considered independent with respect to every other virtual pattern. All these virtual patterns together constitute the training pattern set which is used for HMM training instead of the original patterns directly. The maximum number of virtual patterns (*J*) given *D* training patterns is equal to $\sum_{2\leq K\leq D} C_K^D$ (the number of training combinations with at least two in the set), where $C_K^D = \frac{D!}{K!\,(D-K)!}$ and *K*! stands for *K* factorial ($(K! = \prod_{i=1}^{K} i)$.). However since this value can be very high for a large database, we can choose a subset of *E* patterns from *J* patterns, in an intelligent way. These *E* patterns form the virtual training data. The higher the value of *E* the better would be the statistical stability of the estimation and also the various speech variabilities is likely to be modeled better.

Let $\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}$ be the *K* patterns selected from the *D* pattern training data. We apply the MPDTW algorithm to find the MPDTW path. The MPDTW path now acts as an alignment to compare similar sounds. Using this MPDTW path we find the HMM state sequence using the CMPVA. It may be noted that even the virtual patterns have different lengths, depending on the MPDTW path.

4.1 Virtual pattern creation

The MPDTW path and the CMPVA decoded HMM state sequence are used as inputs to create a virtual pattern. An initial HMM is found out using the HMM training algorithm on the original training data and the HMM parameters are used to start the iterations. The virtual pattern is created by aligning the *K* patterns and weighting the feature vectors of the different patterns such that the more likely vectors are given a higher weighting and the less reliable vectors are given a lower weighting. The MPDTW path is a global alignment of the input patterns, which does not change through the selective iterations. Let ϕ be the MPDTW mapping between one set of *K* patterns and $\phi(t) = (t_1, \ldots, t_K) = (\phi_1(t), \phi_2(t), \ldots, \phi_K(t))$, where $(t_1, \ldots, t_K)$ is a *K* tuple coordinate sequence belonging to the *K* patterns along the MPDTW optimum path, and $\phi_i(t) = t_i$, such that $\phi_i(t)$ is the projection of the MPDTW path $\phi(t)$ onto the t_i axis. $\phi = (\phi(1), \phi(2), \ldots, \phi(T_p))$, where T_p is the number of points in the MPDTW path such that $T_p \geq \max\{T_1, T_2, \ldots, T_K\}$. Since the MPDTW path provides the least distortion alignment between the *K* patterns we are able to compare all the similar sounds from the multiple patterns and weight them appropriately to create a virtual pattern. The length of the virtual pattern is equal to the length of the MPDTW path. The HMM state sequence emitting *K* patterns jointly is given by the CMPVA. Let $\mathbf{q} = q_{\phi(1)}, q_{\phi(2)}, \ldots, q_{\phi(T_p)}$ be the jointly decoded HMM state sequence, where $q_{\phi(t)}$ is the HMM state at time ϕ (t) which is jointly emitting the feature vectors $(O^1_{t_1}, O^2_{t_2}, \ldots, O^K_{t_K})$. We define a virtual pattern to be created as below:

$$\mathbf{V}^p_{\phi(1):\phi(T_p)} = f\{\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}; \phi(1) : \phi(T_p)\} \tag{43}$$

where $\mathbf{V}^p_{\phi(1):\phi(T_p)}$ is the p^{th} virtual pattern of length T_p, $1 \leq p \leq E$; $f(.)$ is a function which maps the *K* patterns to one virtual pattern. $\mathbf{V}^p_{\phi(1):\phi(T_p)} = (V^p_{\phi(1)}, V^p_{\phi(2)}, \ldots, V^p_{\phi(T_p)})$ and $V^p_{\phi(t)}$ is the feature vector of the p^{th} virtual pattern at time $\phi(t)$. Each feature vector in the virtual pattern

is defined as a weighted combination of the feature vectors from the K patterns (of the subset of training data), determined through the K tuple of the MPDTW path, i.e.

$$V^p_{\phi(t)} = \sum_{i=1}^{K} w_i(\phi(t)).O^i_{\phi_i(t)} \tag{44}$$

where $\phi(t) = (t_1, \ldots, t_K) = (\phi_1(t), \phi_2(t), \ldots, \phi_K(t))$, $\phi_i(t) = t_i$, and $w_i(\phi(t))$ is the weight for $O^i_{\phi_i(t)}$. $O^i_{\phi_i(t)}$ is the feature vector of the i^{th} pattern which lies on the time frame t_i ($O^i_{\phi_i(t)}$ is same as $O^i_{t_i}$ defined before). $w_i(\phi(t))$ is itself defined as:

$$w_i(\phi(t)) = \frac{b_{q_{\phi(t)}}(O^i_{\phi_i(t)})}{\sum_{i=1}^{K} b_{q_{\phi(t)}}(O^i_{\phi_i(t)})} \tag{45}$$

where $b_{q_{\phi(t)}}(O^i_{\phi_i(t)})$ is the likelihood of feature vector $O^i_{\phi_i(t)}$ emitted from state $q_{\phi(t)}$ of the current HMM λ. The above weighting factor is similar to the geometric weighting proposed in DC-wtd, but used differently in equation 44.

Similarly, we consider the next K subset of training patterns from the database and create another virtual pattern, and so on till E virtual patterns are created. All the E virtual patterns together are used to train the HMMparameters, using the Baum-Welch algorithm. After Baum-Welch convergence, the updated model is used iteratively for the re-creation of the virtual patterns as shown in Fig. 15. For each SHT iteration, a new set of virtual training patterns are created because the weights in equation 45 get modified because of the updated HMM parameters. However, the warping path is $\phi(t)$ for each virtual pattern is not a function of HMM parameters and hence does not vary with SHT iterations. We define a distortion measure to stop the SHT iterations. The number of features in the p^{th} virtual pattern path is fixed by the MPDTWpath and does not vary with iterations. Therefore, we can define convergence of the virtual patterns themselves as a measure of convergence. The change in the virtual pattern vector $V^p_{\phi(t)}$ of the p^{th} virtual pattern, $d(V^{p,m}_{\phi(t)}, V^{p,m-1}_{\phi(t)})$, is defined as the Euclidean distance between $V^p_{\phi(t)}$ at iteration number m and $V^p_{\phi(t)}$ at iteration number $m-1$. The total change at iteration m for the p^{th} virtual pattern sequence is

$$D^m(\mathbf{V}^p_{\phi(1):\phi(T_p)}) = \frac{1}{T_p}\sum_{t=1}^{T_p} d(V^{p,m}_{\phi(t)}, V^{p,m-1}_{\phi(t)}) \tag{46}$$

where $p = 1, 2, \ldots, E$. The average distortion D^m_{avg} for all the E virtual patterns at the m^{th} iteration is calculated and if it is below a threshold the SHT iterations are stopped.

The virtual pattern has the property that it is "cleaner" (less distorted) compared to the original patterns. Let us take an example when one (or all) of the original training patterns have been distorted by burst noise. The virtual pattern is created by giving less weighting to the unreliable portions of the original data through DC-wtd and the weight parameters $w_i(\phi(t))$ of equation 45. When most of the training patterns are not outliers, the initial HMM is relatively noise free. So $b_{q_{\phi(t)}}(O^i_{\phi_i(t)})$ of equation 45 can be expected to be higher for reliable values of $O^i_{\phi_i(t)}$. Hence, $w_i(\phi(t))$ has a higher value for reliable feature vectors and lower for the unreliable ones. With each SHT iteration, the virtual patterns become more

and more noise free leading to a better HMM. In the standard HMM training, the model converges optimally to the data. In the SHT, the model and the data are both converging. Since the data is moving towards what is more likely, it is possible that the variance parameter of the Gaussian mixture model (GMM) in each HMM state gets reduced after each iteration. This could deteriorate the generalizability of HMM and hence its speech recognition performance for the unseen data, as the new HMMs might not be able to capture the variability in the test patterns. So we have chosen to clamp the variance after the initial HMM training and allow only the rest of the HMM parameters to adapt. Also we have considered some variants to the formulation of the virtual pattern given in equation 44. Through the experiments, we found that there may be significant variation of the weight parameter $w_i(\phi(t))$ for each $\phi(t)$, and also across iterations. Therefore we propose below two methods of smoothing the $w_i(\phi(t))$ variation, which leads to better convergence of HMM parameters.

A weighted averaging in time for the weights $w_i(\phi(t))$s is done by placing a window in time as shown below:

$$V^p_{\phi(t)} = \sum_{i=1}^{K} \left[\sum_{j=-P}^{P} l_i(\phi(t+j)).w_i(\phi(t+j)) \right] .O^i_{\phi_i(t)} \tag{47}$$

where $2P + 1$ is the length of the window placed over time $\phi(t - P)$ to $\phi(t + P)$; $l_i(\phi(t + j))$ is the weighting given to the weight $w_i(\phi(t + j))$ at time $\phi(t + j)$, such that $\sum_{j=-P}^{P} l_i(\phi(t+j)) = 1$. Smoothing of the weights $w_i(\phi(t))$ allows the reduction of some sudden peaks and also uses the knowledge of the neighboring vectors. This improved ASR accuracy. Weighted averaging can also be done for the weights over successive iterations:

$$V^p_{\phi(t)} = \sum_{i=1}^{K} \left[\sum_{j=0}^{P} l_i^{m-j}(\phi(t)).w_i^{m-j}(\phi(t)) \right] .O^i_{\phi_i(t)} \tag{48}$$

where m is the iteration number, and $P + 1$ is the window length over iterations for the weights. $w_i^m(\phi(t))$ is the value of weight $w_i(\phi(t))$ at iteration m. $l_i^{m-j}(\phi(t))$ is the weighting given to the weight $w_i^{m-j}(\phi(t))$ at iteration $m - j$, such that $\sum_{j=0}^{P} l_i^{m-j} = 1$.

5. Experimental evaluation

5.1 MPDTW experiments

We carried out the experiments (based on the formulation in section 2) using the IISc-BPL database[1] which comprises a 75 word vocabulary, and 36 female and 34 male adult

[1] IISc-BPL database is an Indian accented English database used for Voice Dialer application. This database consists of English isolated words, English TIMIT sentences, Native language (different for different speakers) sentences, spoken by 36 female and 34 male adult speakers recorded in a laboratory environment using 5 different recording channels: PSTN-telephone (8 kHz sampling), Cordless local phone (16 kHz sampling), Direct microphone (16 kHz sampling), Ericsson (GSM) mobile phone (8 kHz sampling), Reverberant room telephone (Sony) (8 kHz sampling).

speakers, with three repetitions for each word by the same speaker, digitized at 8 kHz sampling rate. The vocabulary consists of a good number of phonetically confusing words used in Voice Dialer application. MFCCs, Δ MFCCs, and Δ² MFCCs is used without their energy components (36 dimensions) as feature vector. The experiment is carried out for speaker dependent IWR for 20 speakers and 75 word vocabulary.

The slope weighting function $m(t)$ is set equal to 1. The global normalization factor M_ϕ (equation 8) that we used is $M_\phi = T_1 + T_2 + \ldots + T_K$. (In this section 5.1, K stands for the sum of the number of test and template patterns.) Through the experiments we found that normalization using only the sum of the total frames of reference patterns in each class gives worse recognition than the normalization we used. In an experiment where each speaker utters 2 test patterns of the same class (of lengths T_1 and T_2 frames) and 1 reference pattern (of length T_3 frames) per class per speaker for 20 speakers, the percentage ASR accuracy using $M_\phi = T_1 + T_2 + T_3$ is 96.47%. If $M_\phi = T_3$ then the percentage accuracy reduces to 89.07%.

Table 1 summarizes the results based on the formulation of section 2. In the table, the experiment-1 DTW-1 test-1 templ, corresponds to standard DTW algorithm applied when there is 1 test pattern spoken by each speaker for each word and it's distortion is compared with the reference patterns (1 reference pattern per speaker per word for 20 speakers) of each word in the vocabulary. In experiment-2 DTW-2 test-1 templ each speaker utters 2 patterns of a word. Each one of them is compared separately with the reference patterns (1 template per speaker per word). In experiment-3 DTW-2 test-1 templ (minimum of two), the minimum of the two distortions of the two test patterns (of the same word by a speaker) with the reference patterns, is considered to calculate recognition accuracy. In experiment-4 MPDTW-2 test-1 templ, each speaker utters 2 test patterns. The MPDTW algorithm is applied on the 2 test patterns and 1 reference pattern at a time (1 reference pattern per speaker per word) to find the distortion between them. In experiment-5 DTW-1 test-2 templ, 1 test pattern, each speaker speaks 1 test pattern and 2 reference patterns. The test pattern is now compared with the reference patterns (2 reference patterns per speaker per word for 20 speakers). In experiment-6 MPDTW-1 test-2 templ, the MPDTW algorithm is applied on 1 test pattern and 2 reference patterns (2 reference patterns per speaker per word) and then IWR is done. In this experiment K is equal to the sum of the number of test and template patterns.

Experiment	Clean	Noisy
1. DTW-1 test-1 templ	94.67	91.13
2. DTW-2 test-1 templ	95.17	92.47
3. DTW-2 test-1 templ (minimum of two)	95.40	89.73
4. MPDTW-2 test-1 templ	96.47	94.47
5. DTW-1 test-2 templ	96.67	94.07
6. MPDTW-1 test-2 templ	90.80	85.60

Table 1. Comparison of ASR percentage accuracy for clean and noisy test pattern. For noisy speech, burst noise is added for 10% of test pattern frames at -5 dB SNR (local). Reference patterns (templ) are always clean. IWR is done for 20 speakers.

We see from the results that when there are 2 test patterns uttered by a speaker and 1 reference pattern (case 2) and the MPDTW algorithm (for K = 3) is used, the speech recognition word error rate reduced by 33.78% for clean speech and 37.66% for noisy test speech (10% burst noise added randomly with uniform distribution at -5 dB SNR (local) in both the test patterns), compared to the DTW algorithm (same as MPDTW algorithm when

K = 2) for only 1 test pattern. Even when using the minimum distortion among two test patterns (experiment-2 DTW-2 test-1 templ (minimum of two)), we see that the MPDTW algorithm works better. However, when we use only 1 test pattern and 2 reference patterns and when the MPDTWalgorithm (for K = 3) is used, the percentage accuracy reduces as predicted in section 2.2. Hence we see that use of multiple test repetitions of a word can significantly improve the ASR accuracy whereas using multiple reference patterns can reduces the performance.

5.2 MPJL experiments

Based on the formulations in section 3, we conducted experiments - A1M, A1P, A2, A3 for speaker independent IWR along with the base line system of standard Forward Algorithm (FA) or Viterbi Algorithm (VA) for a single pattern, for the cases of both clean and noisy speech. Since the normal FA/VA uses one pattern to make a recognition decision and the proposed algorithms use K patterns to make a decision, the comparison of results may not be fair. For a fairer comparison we formulated the experiment A1M, which also uses K patterns using the standard FA/VA and the best (max) likelihood of the K patterns is chosen. Experiment A1P uses the product of the K likelihoods of the K patterns. So we compare the new algorithms (experiment A2 and A3) with the experiments A1M, A1P also. (In this section 5.2, K stands for the total number of test patterns of a word spoken by a speaker.)

The experiment A1M is as described. Given $\mathbf{O}^1_{1:T_1}, \mathbf{O}^2_{1:T_2}, \ldots, \mathbf{O}^K_{1:T_K}$ as the individual patterns belonging to the same class, we can obtain the joint likelihood score as $\theta_j = \max_{1 \le i \le K} P(\mathbf{O}^i_{1:T_i}/\lambda_j)$, where λ_j are the clean word models and the FA/VA is used to calculate $P(\mathbf{O}^i_{1:T_i}/\lambda_j)$. We select the pattern as j^* = argmax$_j$ θ_j . We are actually doing a voting, where the pattern which has the highest likelihood is chosen. Experiment A1P when the joint likelihood score $\theta_j = \prod_{1 \le i \le K} P(\mathbf{O}^i_{1:T_i}/\lambda_j)$. For the experiments, we have restricted to two or three patterns per test speaker. When K = 2, for each word of a test speaker, A1M and A1P are done for pattern 1 and pattern 2, pattern 2 and 3, pattern 3 and 1. When K = 3, all the 3 patterns are considered. Experiment A2 is the MPDTW algorithm followed by CMP?A-2. Experiment A3 is the MPDTW algorithm followed by CMP?A-1. In all joint recognition experiments, we have restricted to two or three pattern joint recognition and compared the performance with respect to single pattern recognition. When K = 2, for each word of a test speaker, pattern 1 is jointly recognized with pattern 2, pattern 2 with 3, pattern 3 with 1. When K = 3 all the three patterns are jointly recognized. Please note that in the noisy case, all the three patterns are noisy. As the number of test patterns K = 2, for the new experiments we chose the Local Continuity Constraints for MPDTW as (1,0) or (0,1) or (1,1) and the slope weighting function $m(t)$ = 1. Similar extensions are done for K = 3.

The IISc-BPL database was used for experimentation. Left to Right HMMs are trained for clean speech using the Segmental K Means (SKM) algorithm [Rabiner et al., 1986, Juang & Rabiner, 1990]. 25 male and 25 female speakers are used for training, with three repetitions of each word by each speaker. We tested the algorithm for 20 unseen speakers (11 female and 9 male) in both clean and noisy cases. Test words are three patterns for each word by each speaker, at each SNR.

We first run the experiment for speech affected by burst noise. Burst noise was added to some percentage of the frames of each word at -5 dB, 0 dB, 5 dB SNRs (local) to all the three patterns. (The remaining frames are clean; the range of -5 dB to +5 dB indicates severe to

mild degradation of the noise affected frames.) The burst noise can occur randomly anywhere in the spoken word with uniform probability distribution. MFCCs, Δ MFCC, and Δ^2 MFCC is used without their energy components (36 dimensions). Energy components are neglected and Cepstral Mean Subtraction was done. Variable number of states are used for each word model; i.e. proportional to the average duration of the training patterns, for each second of speech, 8 HMM states were assigned, with 3 Gaussian mixtures per state.
We experimented for various values of the threshold γ in DC-thr1 and found that there is indeed an optimum value of γ where the performance is maximum. When K = 2, for the noisy patterns with burst noise added to 10% of the frames at -5 dB SNR, γ = 0.5 is found to be optimum. It is also clear that γ < 0 provides closer to optimum performance than $\gamma = \infty$, indicating that the max operation is more robust than the product operation. Using DC-wtd was shown to have similar results to using DC-thr1 with optimum threshold.

Algorithm	ASRA(Clean)	ASRA(Clean)	-5 dB ASRA	-5 dB ASRA	0 dB ASRA	0 dB ASRA	5 dB ASRA	5 dB ASRA
FA	89.61	89.61	56.87	56.87	61.24	61.24	67.13	67.13
A1M, K=2	89.81	89.81	60.07	60.07	64.00	64.00	69.40	69.40
A1P, K=2	92.29	92.29	64.44	64.44	68.84	68.84	74.44	74.44
DC-thr1, $\gamma = \infty$, K=2	91.87, A2	91.80, A3	61.31, A2	61.53, A3	66.07, A2	66.11, A3	72.42, A2	72.22, A3
DC-thr1, $\gamma = 2$, K=2	91.87, A2	91.78, A3	73.91, A2	73.18, A3	77.02, A2	76.02, A3	80.58, A2	80.11, A3
DC-thr1, $\gamma = 1$, K=2	91.80, A2	92.13, A3	78.09, A2	76.09, A3	80.82, A2	78.76, A3	83.80, A2	82.24, A3
DC-thr1, $\gamma = 0.75$, K=2	91.73, A2	92.18, A3	79.13, A2	76.98, A3	81.69, A2	79.76, A3	84.51, A2	83.05, A3
DC-thr1, $\gamma = 0.5$, K=2	91.49, A2	92.02, A3	79.47, A2	77.18, A3	82.00, A2	79.84, A3	84.76, A2	83.44, A3
DC-thr1, $\gamma = 0.25$, K=2	91.49, A2	91.98, A3	79.44, A2	77.09, A3	82.00, A2	79.91, A3	84.73, A2	83.38, A3
DC-thr1, $\gamma < 0$, K=2	91.49, A2	91.98, A3	79.44, A2	77.09, A3	82.00, A2	79.91, A3	84.73, A2	83.38, A3
DC-wtd, K=2	91.38, A2	92.02, A3	79.35, A2	77.11, A3	82.05, A2	79.96, A3	84.78, A2	83.38, A3
DC-wtd, K=3	92.73, A2	93.13, A3	88.07, A2	86.73, A3	89.20, A2	87.73, A3	90.07, A2	89.20, A3

Table 2. Comparison of ASR percentage accuracy (ASRA) for clean and noisy speech (10% burst noise) for FA, A1M, A1P, A2, and A3. FA - Forward Algorithm, Experiment A1M, K = 2 - best (max of likelihoods) of two patterns using FA, Experiment A1P, K = 2 - product of the likelihoods (using FA) of two patterns, Experiment A2 - MPDTW algorithm + CMPFA-2, Experiment A3 – MPDTW algorithm + CMPFA-1. K is the number of test patterns used.

The results for clean and noisy speech for FA and CMPFA is given in Table 2. We have not shown the results of CMPBA as it is similar to CMPFA. In the table, ASRA (Clean) stands for ASR accuracy for clean speech. In the tables, for experiment A2, in the ASRA column, the ASR percentage accuracy is written. Note that DC-thr1 is equivalent to DC-avg when $\gamma = \infty$ (product operation) and it is equivalent to DC-max when γ < 0 (max operation). Also, DC-thr1 is same as DC-thr2 when K = 2. When K = 2, two patterns are recognized at a time, while K = 3 stands for 3 patterns being recognized at a time. In the table, -5 dB ASRA stands for ASR accuracy for noisy speech which has 10% burst noise at SNR -5 dB. It can be seen that the baseline performance of FA for clean speech is close to 90%. For example, for noisy case at -5 dB SNR, for speech with 10% burst noise, it decreases to ≈ 57%. Interestingly, the experiment A1M (for K = 2 patterns) provides a mild improvement of 0.2% and 3.2% for clean and noisy speech (at -5 dB SNR burst noise) respectively, over the FA benchmark. This shows that use of multiple patterns is indeed beneficial, but just maximization of likelihoods is weak. Experiment A1P (for K = 2 patterns) works better than A1M clearly indicating that taking the product of the two likelihoods is better than taking their max. The proposed new algorithms (experiment A2 and A3) for joint recognition provides dramatic improvement for the noisy case, w.r.t. the FA performance. For example at -5 dB SNR 10% burst noise, for K = 2 patterns, the proposed algorithms (experiments A2 and A3) using DC-thr1 at

threshold γ = 0.5, gave an improvement of 22.60% speech recognition accuracy (using CMPFA-2) and 20.31% speech recognition accuracy (using CMPFA-1) compared to FA performance. For K = 3 patterns, the recognition accuracy increases by 31.20% (using CMPFA-2) and 29.86% (using CMPFA-1). So there was almost a 10% improvement in speech recognition accuracy from K = 2 to K = 3. We also see that as the SNR improves, the gap in the speech recognition accuracy between performance of DC-thr1 at threshold $\gamma = \infty$ and $\gamma < 0$ reduces. In fact as SNR approaches to that of clean speech, $\gamma = \infty$ is better than $\gamma < 0$.

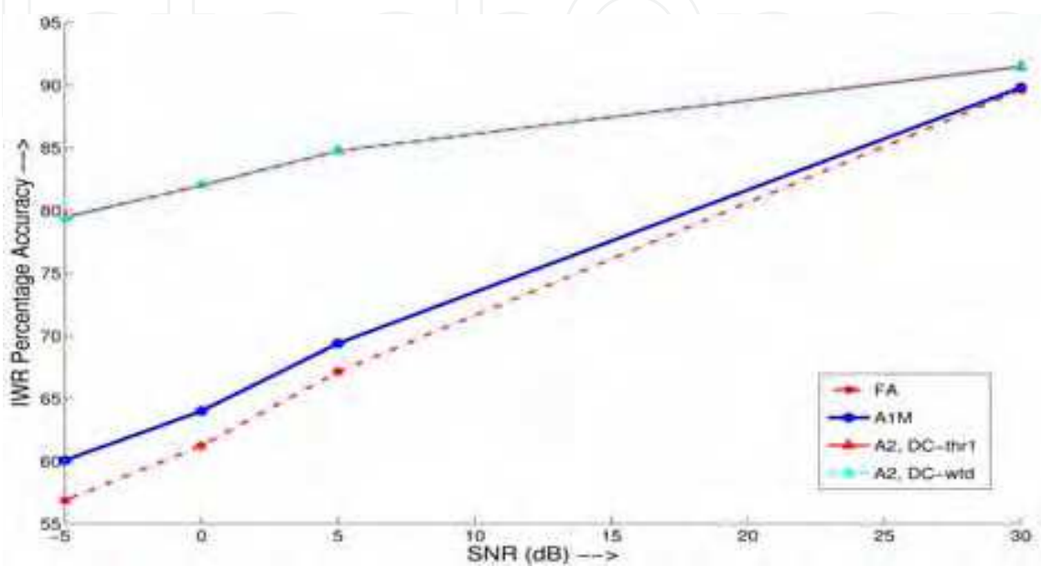

Fig. 16. Percentage accuracies for experiments FA, A1M, A2 for different levels of burst noises. FA - Forward Algorithm, A1M - best of two patterns using FA, A2 - MPDTW algorithm + CMPFA-2 algorithm. Results for A2 using DC-thr1 (at threshold γ = 0.5) and DC-wtd are shown.

For clean speech, the speech recognition accuracy when K = 2, improved by 2.26% using CMPFA-2 and 2.57% using CMPFA-1 (DC-thr1) over that of FA. This improvement could be because some mispronunciation of some words were be taken care of. It is also better than experiment A1M. We also see CMPFA-1 is better than CMPFA-2 for clean speech. However, experiment A1P works the best for clean speech (when K = 2). This could be because in the proposed methods, the K patterns are forced to traverse through the same state sequence. Doing individual recognition on the two patterns and then multiplying the likelihoods has no such restrictions. And in clean speech there is no need for selectively weighting the feature vectors. So experiment A1P works slightly better than experiments A2 and A3 for clean speech.

We also see that as per our analysis in section 3.4 and the results shown in Table 2, using DC-thr1 for CMPFA-1 algorithm (experiment A3) for clean speech at lower thresholds gives better recognition results than using it for CMPFA-2 algorithm (experiment A2). At higher thresholds CMPFA-2 algorithm is better. For noisy speech (speech with burst noise) it is better to use CMPFA-2 than CMPFA-1.

From the table, it is seen that as K increases 2 to 3, there is an improvement in recognition accuracy. We also see that for K = 3, the performance does vary much, whether the burst noise is at -5 dB SNR or +5 dB SNR. This is because the noise corrupted regions of speech is

almost completely neglected during recognition (whether noise is at -5 dB SNR or +5 dB SNR) and the clean portion of the other patterns are given a higher weight.
A graph showing variation of IWR percentage accuracy versus the burst noise at some SNR is shown in Fig. 16. In this figure the experiment A2 using DC-thr1 is plotted at threshold γ = 0.5, where DC-thr1 works very well for speech with burst noise. We see that using DC-wtd gives us optimum or near optimum values. The performance of DC-wtd is equal to to the optimum performance of DC-thr1, as we predicted in section 3.4. And the advantage of DC-wtd is that we don't need any threshold. We also see that as per our analysis in section 3.4 and the results shown in Table 2, using DC-thr1 for CMPFA-1 algorithm (experiment A3) for clean speech at lower thresholds gives better recognition results than using it for CMPFA-2 algorithm (experiment A2). At higher thresholds CMPFA-2 algorithm is better. For noisy speech (speech with burst noise) it is better to use CMPFA-2 than CMPFA-1.
The results for clean and noisy speech for VA and CMPVA is given in Table 3. It is seen that CMPVA-1 performs similarly to CMPFA-1 and CMPVA-2 performs similar to CMPFA-2.

Algorithm	ASRA (Clean)	ASRA (Clean)	-5 dB ASRA	-5 dB ASRA	0 dB ASRA	0 dB ASRA	5 dB ASRA	5 dB ASRA
VA	89.70	89.70	57.13	57.13	61.49	61.49	67.38	67.38
A1M, K=2	89.87	89.87	60.33	60.33	64.29	64.29	69.49	69.49
A1P, K=2	92.38	92.38	65.00	65.00	69.36	69.36	74.84	74.84
DC-thr1, $\gamma = \infty$, K=2	91.91, A2	91.78, A3	61.44, A2	61.73, A3	66.36, A2	66.18, A3	72.62, A2	72.47, A3
DC-thr1, $\gamma = 2$, K=2	91.87, A2	91.76, A3	74.11, A2	73.20, A3	77.18, A2	76.18, A3	80.69, A2	80.09, A3
DC-thr1, $\gamma = 1$, K=2	91.78, A2	92.20, A3	78.11, A2	76.36, A3	80.84, A2	78.73, A3	83.80, A2	82.27, A3
DC-thr1, $\gamma = 0.75$, K=2	91.76, A2	92.13, A3	79.11, A2	77.04, A3	81.76, A2	79.84, A3	84.60, A2	83.07, A3
DC-thr1, $\gamma = 0.5$, K=2	91.51, A2	92.02, A3	79.47, A2	77.24, A3	82.13, A2	79.96, A3	84.80, A2	83.42, A3
DC-thr1, $\gamma = 0.25$, K=2	91.53, A2	92.00, A3	79.47, A2	77.18, A3	82.16, A2	80.02, A3	84.80, A2	83.40, A3
DC-thr1, $\gamma < 0$, K=2	91.53, A2	92.00, A3	79.47, A2	77.18, A3	82.16, A2	80.02, A3	84.80, A2	83.40, A3
DC-wtd, K=2	91.44, A2	91.98, A3	79.42, A2	77.22, A3	82.07, A2	79.98, A3	84.76, A2	83.42, A3
DC-wtd, K=3	92.73, A2	93.07, A3	88.20, A2	86.80, A3	89.13, A2	87.67, A3	90.07, A2	89.27, A3

Table 3. Comparison of ASR percentage accuracy (ASRA) for clean and noisy speech (10% noise) for VA, A1M, A1P, A2, and A3. VA - Viterbi Algorithm, Experiment A1M, K = 2 - best of two patterns using VA, Experiment A1P, K = 2 - product of the likelihoods (using VA) of two patterns, Experiment A2 - MPDTW algorithm + CMPVA-2, Experiment A3 - MPDTW algorithm + CMPVA-1.

Percentage noise	-5dB FA	-5 dB CMPFA-1	0 dB FA	0 dB CMPFA-1	5 dB FA	5 dB CMPFA-1
10%, burst	56.87	77.11	61.24	79.96	67.13	83.38
20%, burst	44.55	64.93	49.47	69.38	56.71	75.36
30%, burst	33.67	50.98	38.67	56.89	46.49	64.02
50%, burst	11.38	11.91	16.09	16.42	25.20	29.09
AWGN	3.62	2.89	8.89	7.47	22.84	19.44

Table 4. Comparison of ASR percentage accuracy (ASRA) for noisy speech (burst) for FA and CMPFA-1 (for K = 2 patterns). DC-wtd is used. Different percentages of burst noise (10%, 20%, 30%, 50%) noises added to the speech pattern form the test cases. Additive White Gaussian Noise (AWGN) is added to 100% of the speech pattern is also a test case.

So far we have shown the results of burst noise added to only 10% of the speech patterns. Now different percentage of burst noise is added randomly to the patterns. The results for FA and CMPFA-1 (for K = 2 patterns) is shown in Table 4. DC-wtd is used. We see that speech is affected with 10%, 20%, 30% burst noise, the ASR performance using CMPFA-1 (for K = 2 patterns) is much better than using just using FA. However when noise is added to 50% of the frames, there is only a marginal increase in performance. This is because many

regions of both the patterns will be noisy and the CMPFA-1 does not have a clean portion of speech to give a higher weighting. When 100% of the speech are affected by additive white Gaussian noise (AWGN), then just using FA is better than CMPFA-1. Similar results are given in Table 5 for VA and CMPVA-1.

Percentage noise	-5dB VA	-5 dB CMPVA-1	0 dB VA	0 dB CMPVA-1	5 dB VA	5 dB CMPVA-1
10%, burst	57.13	77.22	61.49	79.98	67.38	83.42
20%, burst	44.67	65.02	49.64	69.56	57.27	75.40
30%, burst	34.33	51.20	39.31	56.91	47.00	64.18
50%, burst	11.89	12.33	16.89	16.89	26.29	29.78
AWGN	3.73	3.20	8.96	7.44	22.89	19.40

Table 5. Comparison of ASR percentage accuracy for noisy speech (burst) for VA and CMPVA-1 (for K = 2 patterns). DC-wtd is used. Different percentages of burst noise (10%, 20%, 30%, 50%) noises added to the speech pattern form the test cases. Additive White Gaussian Noise (AWGN) is added to 100% of the speech pattern is also a test case.

Algorithm	5 dB	10 dB
FA (babble noise)	44.18	59.64
CMPFA-1 (babble noise), K=2	47.80	64.36
CMPFA-1 (babble noise), K=3	47.40	65.93
VA (babble noise), K=2	44.27	59.73
CMPVA-1 (babble noise), K=2	47.84	64.33
CMPVA-1 (babble noise), K=3	47.40	65.87
FA (machine gun noise)	66.53	71.36
CMPFA-1 (machine gun noise), K=2	76.76	81.33
CMPFA-1 (machine gun noise), K=3	81.53	84.20
VA (machine gun noise)	66.71	71.47
CMPVA-1 (machine gun noise), K=2	76.89	81.16
CMPVA-1 (machine gun noise), K=3	81.60	84.13

Table 6. Comparison of ASR percentage accuracy for noisy speech (babble noise or machine gun noise) for FA, VA, CMPFA-1 and CMPVA-1 (for K = 2 patterns). SNR of the noisy speech is 5 dB or 10 dB. DC-wtd is used.

Now we compare the results of the proposed algorithms with other kinds of transient noises like machine gun noise and babble noise. Machine gun and babble noise from NOISEX 92 was added to the entire speech pattern at 5 dB or 10 dB SNR. The results are given in Table 6. The HMMs are trained using clean speech in the same way as done before. We see that for speech with babble noise at 10 dB SNR, the percentage accuracy using FA is 59.64%. It increases to 64.36% when CMPFA-1 (when K = 2 patterns) is used, which is rise of nearly 5%. When K = 3, the accuracy is further improved. When machine gun noise at 10 dB SNR is used the FA gives an accuracy of 71.36%, while the CMPFA-1 when K = 2 patterns gives an accuracy of 81.33% and when K = 3 the accuracy is 84.20%. We see an increase of nearly 10% when K = 2 and 13% when K = 3. We see from the results that the more transient or bursty the noise is, the better the proposed algorithms work. Since machine gun noise has a more transient (bursty) nature compared to babble noise it works better. In the case of machine gun noise, if there is some portion of speech affected by noise in $O_{1:T_1}^1$, there could be a

corresponding clean speech portion in $O^2_{1:T_2}$. This clean portion will be given a higher weight by the proposed algorithms during recognition. However this is not possible if the entire speech is affected by white Gaussian noise or babble noise. When both the patterns are affected by similar noise at the same portion, then there is no relatively clean portion of speech for the proposed algorithms to choose. Hence they work worse. We see that as K increases from 2 to 3 patterns, the ASR accuracy improves significantly for machine gun noise, while for babble noise, the effect is small (in fact at 5 dB the ASR accuracy slightly reduces from K = 2 to K = 3).

5.3 Selective HMM training experiment

Here again, we carried out speaker independent IWR experiments (based on the formulation in section 4), using the IISc-BPL database. Left to Right HMMs are trained using the Baum-Welch training algorithm. MFCC, Δ MFCC, and Δ² MFCC are used without the energy components (total 36 dimension vector). Cepstral mean subtraction is done. 25 male and 25 female speakers are used for training, with three repetitions of each word by each speaker. So the total number of training patterns (*D*) for each word is 150. 5 HMM states per word is used with 3 Gaussian mixtures per state. Both clean speech and speech with a burst noise of 10% at -5 dB SNR (local) was used for HMM training. The burst noise can occur randomly anywhere in the spoken word with uniform probability distribution. Note that in the noisy case all the training patterns have burst noise in them. We used CMPVA-2 for our experiments. (In this section 5.3, *K* stands for the number of training patterns used to create one virtual pattern.)

We first consider an example to gain insight into the working of SHT. Four clean patterns of the word "Hello", O1, O2, O3, O4, by a speaker are considered. Noise at -5 dB SNR is added to the first 10% of the frames in pattern O3. For this example, the initial HMM was found using the Baum-Welch algorithm using only these 4 patterns. One virtual pattern is created (using equation 43) from the 4 patterns (*K* = 4). Fig. 17 shows the decrease in the distortion measure with each SHT iteration showing clear convergence. However, for all the patterns this decrease is not always monotonic in nature and there may be small fluctuations at higher iterations.

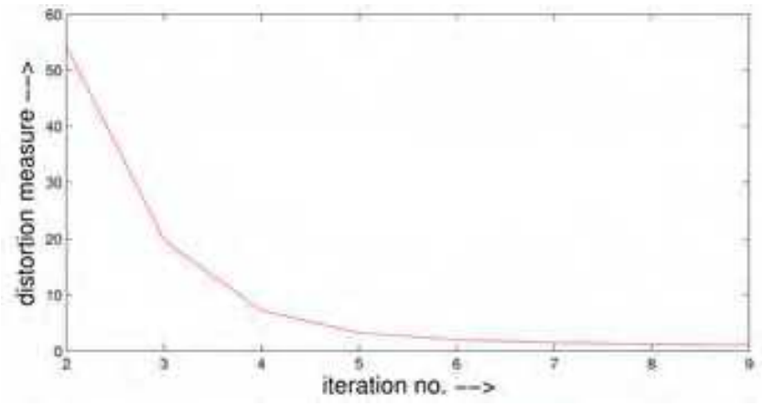

Fig. 17. Distortion with each iteration.

Fig. 18 shows the weights $w_3(\phi(t))$ given to each frame of the virtual pattern for pattern O3 for the first and ninth iteration. We can see the amount of contribution of O3, i.e., to the virtual pattern: i.e., the all the initial frames are given a low weight as O3 is noisy in this

region. And this weighting decreases with every iteration as the virtual pattern converges. Similarly Fig. 19 gives the weights of O2. We see that the initial few frames do not have less weighting, contrasting with O3.

Fig. 20 shows the difference in likelihood of O1, O2, O4 with O3 given the HMMλ are shown. $P(O1/\lambda)-P(O3/\lambda)$, $P(O2/\lambda)- P(O3/\lambda)$, $P(O4/\lambda) - P(O3/\lambda)$ are the three curves shown in that figure. These probabilities are computed using the Forward algorithm. In Fig. 20, at iteration 0, the HMMs is the Baum-Welch algorithm run on the original training data. $P(O2/\lambda)$ and $P(O4/\lambda)$ are greater than $P(O3/\lambda)$. After each SHT iteration the HMM is updated and the differences of $P(O2/\lambda) -P(O3/\lambda)$ and $P(O4/\lambda) - P(O3/\lambda)$ increases. This happens because the HMM is updated by giving less weightage to the noisy portion of O3. We also see that although some portion of O3 is noisy, $P(O1/\lambda)$ is less than $P(O3/\lambda)$. This is because the HMM is trained using only 4 patterns out of which one HMM pattern (O3) is partially noisy. So the initial HMM using the Baum-Welch algorithm is not very good. We see that after the iterations the difference $P(O1/\lambda) - P(O3/\lambda)$ reduces. This indicated that after each iteration the HMM parameters are updated such that the unreliable portions of O3 is getting a lesser weight.

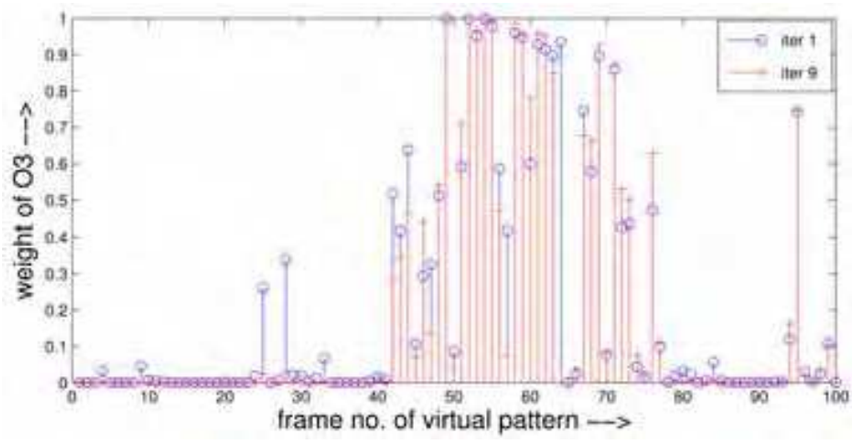

Fig. 18. Weight of O3 for creating of virtual pattern at iterations numbers 1 and 9. Noise at -5 dB SNR is added to the first 10% of the frames in pattern O3.

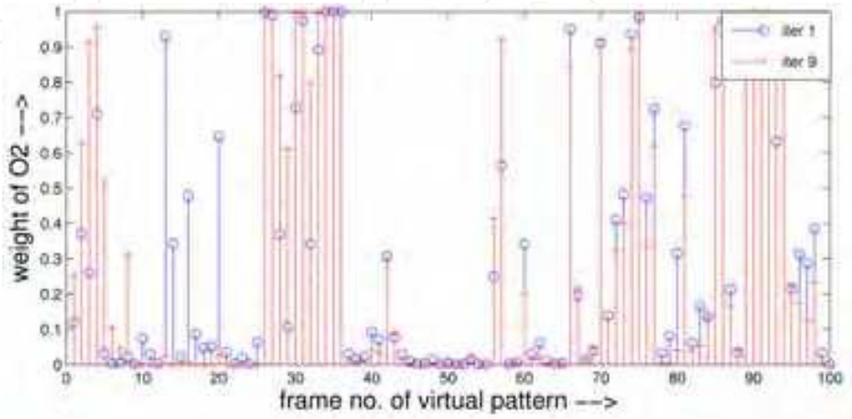

Fig. 19. Weight of O3 for creating of virtual pattern at iterations numbers 1 and 9. Noise at -5 dB SNR is added to the first 10% of the frames in pattern O3.

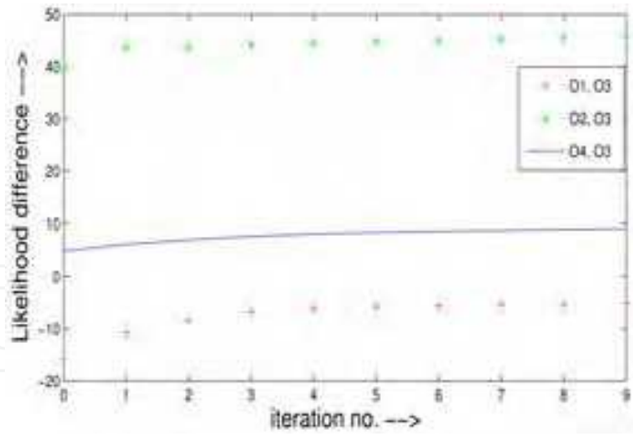

Fig. 20. Difference between Likelihood of patterns O1, O2, O4 with O3 given HMM λ.

While the above results are promising in terms of the functioning of the SHT algorithm, the level of improvement in HMM performance for test data could be limited, depending on the nature and size of the test data. Using the small size pilot experiment, the HMM performance is tested using Forward algorithm (FA) for 20 unseen speakers (11 female and 9 male) using clean speech. There are 3 test patterns per speaker. The experimental setup used for training was done as mentioned in the first paragraph of this sub-section. Each of the 150 training patterns (3 training patterns per speaker, for 50 speakers) are affected by 10% burst noise at -5 dB SNR. The covariance matrix for the Gaussian mixtures for each HMM state was fixed to that of the initial Baum-Welch algorithm run on the original training data. The number of patterns to create one virtual pattern (K) is 3. The 3 patterns spoken by the same speaker are considered for creating one virtual pattern as the MPDTW path alignment may be better for the same speaker. So we have a total of 50 virtual patterns ($E = D/K$) per word since the total number of training patterns (D) per word is 150 (number of speakers is 50). The virtual patterns are created using equation 44. We used CMPVA-2 for the experiments. When Baum-Welch (BW) algorithm is used to train speech with 10% burst noise at -5 dB then for the clean test data the ASR percentage accuracy is 88.36%. It increases to 88.76% using the new SHT (using equation 44 to create virtual patterns) when the covariance matrix of each HMM state is kept constant (*covar − const*). Let this experiment be called *SHT* − 2. If the covariance matrix is allowed to adapt (*covar − adapt*), the covariance decreases (determinant value) after each iteration as the virtual patterns are converging to what is more likely and this may reduce the ability of the recognizer to capture the variabilities of the test patterns. (Let such an experiment be called experiment *SHT* − 1.) When the covariance matrix is not kept constant the percentage accuracy reduces to 86.04%. So we keep the covariance constant.

We now experiment by keeping averaging the weights over time and iterations (see equations 47, 48). When averaging the weights over time (equation 47) keeping the covariance matrix constant, the percentage accuracy increases to 88.84%. Let this be experiment be called *SHT* − 3. In equation 47, we set $P = 1$, $l_i(\phi(t)) = 0.5$, $l_i(\phi\ (t-1)) = l_i(\phi\ (t+1)) = 0.25$. This shows that smoothing the weights $w_i(\phi(t))$'s improves ASR accuracy. For averaging the weights over iterations (equation 48), $l_i^m(\phi(t)) = 0.5, l_i^{m-1}(\phi(t)) = 0.3, l_i^{m-2}(\phi(t)) = 0.2$. However, the ASR accuracy reduces to 73.38%. Let this experiment be called *SHT* − 4.

Now we increase the number of virtual patterns used for training. In the database used, the number of patterns of a word spoken by a speaker is 3. Let the patterns be O1, O2, O3. We can create 1 virtual pattern using O1, O2, O3 (K = 3). We can also create 3 other virtual patterns using O1-O2, O2-O3, O3-O1 (K = 2 for each). Thus we have 4 virtual patterns created using training patterns O1, O2, O3. We do this for very speaker and every word. So the total number of virtual patterns per word is 200 (since the number of training patterns per word is 150). HMMs are trained on these virtual patterns. The covariance matrix is kept constant and equation 44 is used for calculating the virtual patterns. Let this be called experiment *SHT* − 5. The ASR accuracy using FA increases to 89.07%. The more number of virtual patterns we use to train the HMMs, the better the test data variability is captured. We see from the experiments that as the number of virtual patterns per word (using equation 44) increases from 50 to 200, the percentage accuracy also increases from 88.76% to 89.07%, clearly indicating that it helps using more virtual patterns as training data. However in this case, by averaging over time (called experiment *SHT* − 6), using equation 47, the accuracy remained at 89.07%. The results are summarized in Table 7. Thus it was shown that the word error rate decreased by about 6.1% using the proposed SHT training method over the baseline Baum-Welch method.

Experiment	Percentage ASR accuracy
BW	88.36
SHT-1-covar-adapt	86.04
SHT-2-covar-const	88.76
SHT-3-covar-const	88.84
SHT-4-covar-const	73.38
SHT-5-covar-const	89.07
SHT-6-covar-const	89.07

Table 7. Comparison of Percentage ASR percentage accuracy for different algorithms. The training patterns have 10% of their frames affected by burst noise at -5 dB SNR. Testing was done on clean speech using FA. BW - Baum-Welch algorithm. *SHT* −1 - SHT using 1 virtual patterns per word per speaker and adapting covariance. *SHT* − 2 - SHT using 1 virtual patterns per word per speaker and constant covariance. *SHT* − 3 - SHT using 1 virtual patterns per word per speaker and constant covariance; averaging of weights across time is done. *SHT* − 4 - SHT using 1 virtual patterns per word per speaker and constant covariance; averaging of weights across iterations is done. *SHT* − 5 - SHT using 4 virtual patterns per word per speaker and constant covariance. *SHT* − 6 – SHT using 4 virtual patterns per word per speaker and constant covariance; averaging of weights across time is done.

HMMs were also trained using clean speech. Testing is also done on clean speech on unseen speakers using the Forward algorithm. Using Baum-Welch algorithm, we get an ASR accuracy of 91.18%. Using the proposed SHT algorithm (experiment *SHT* − 5), we get an accuracy of 91.16%. When averaging over time is done (experiment *SHT* −6), the ASR accuracy remains at 91.16%. Thus we see that using the proposed SHT training method does not reduce the ASR performance for clean speech.

HMMs were also trained using training patterns corrupted with machine gun noise (from NOISEX 92 database) at 10 dB SNR. The experimental setup is same as used before. Totally there are 150 training patterns (3 per speaker for 50 speakers). Testing was done on unseen clean speech using the FA. Using normal Baum-Welch training the ASR accuracy was

85.56%. However it reduced to 85.20% using 200 virtual training virtual patterns (experiment *SHT* − 5). Using averaged weights over time (experiment *SHT* − 6), the percentage accuracy increased to 85.29%. However it is still lower compared to the Baum-Welch training algorithm. The performance could have been better if there were more virtual patterns created from the training data set of 150 patterns.

6. Conclusions

We have formulated new algorithms for joint evaluation of the likelihood of multiple speech patterns, using the standard HMM framework. This was possible through the judicious use of the basic DTW algorithm extended to multiple patterns. We also showed that this joint formulation is useful in selective training of HMMs, in the context of burst noise or mispronunciation among training patterns.

Although these algorithms are evaluated in the context of IWR under burst noise conditions, the formulation and algorithm can be useful in different contexts, such as connected word recognition (CWR) or continuous speech recognition (CSR). In spoken dialog systems, if the confidence level of the test speech is low, the system can ask the user to repeat the pattern. However, in the continuous speech recognition case, a user cannot be expected to repeat a sentence/s exactly. But still the proposed methods can be used. Here is one scenario. For booking a railway ticket, the user says, "I want a ticket from Bangalore to Aluva". The recognition system asks the user, "Could you please repeat from which station would you like to start?". The user repeats the word "Bangalore". So this word "Bangalore" can be jointly recognized with the word "Bangalore" from the first sentence to improve speech recognition performance.

One of the limitations of the new formulation is when the whole pattern is noisy, i.e., when the noise is continuous not bursty; the proposed algorithms don't work well. Also, for the present, we have not addressed the issue of computational complexity, which is high in the present implementations. Efficient variations of these algorithms have to be explained for real-time or large scale CSR applications.

Finally we conclude that jointly evaluating multiple speech patterns is very useful for speech training and recognition and it would greatly aid in solving the automatic speech recognition problem. We hope that our work will show a new direction of research in this area.

Appendix A1 - Proof for the recursive equation in CMPFA-1

We shall derive the recursive equation for CMPFA-1 (equation 21) using the example shown in Fig. 9. Consider two patterns $O^1_{1:T_1}$ and $O^2_{1:T_2}$. The MPDTW algorithm gives the time alignment (MPDTW path) between these two patterns (as shown in Fig. 9). We now fit a layer of HMM states on this MPDTW path.

$$\alpha_{\phi(t)}(j) = P(\mathbf{O}^1_{1:t_1}, \mathbf{O}^2_{1:t_2}, \ldots, \mathbf{O}^K_{1:t_K}, q_{\phi(t)} = j/\lambda) \tag{49}$$

where $q_{\phi(t)}$ is the HMM state at $\phi(t)$ and λ is the HMM model with state $j \in 1:N$, where N is the total number of states in the HMM. In the given example $\phi(1) = (1, 1)$, $\phi(2) = (2, 2)$, $\phi(3) = (3, 2)$. Each state j can emit a variable number of feature vectors varying from 1 to K.

$$\begin{aligned} \alpha_{\phi(1)}(i) &= P(O_1^1, O_1^2, q_{\phi(1)} = i/\lambda) \\ &= P(O_1^1, O_1^2/q_{\phi(1)} = i, \lambda)\ P(q_{\phi(1)} = i/\lambda) \\ &= \pi_i\ b_i(O_1^1, O_1^2) \end{aligned} \tag{50}$$

where $b_i(O_{t_1}^1, O_{t_2}^2)$ is the probability of feature vectors $O_{t_1}^1$ and $O_{t_2}^2$ emitted given state i, $\pi_i = P(q_{\phi(1)} = i/\lambda)$ which is the state initial probability. It is assumed to be same as the state initial probability given by the HMM. This can be done because the value of $b_j(O_{t_1}^1, O_{t_2}^2)$ is normalized as shown in section 3.3, such that the probability of K (here $K = 2$) vectors being emitted by a state is comparable to the probability of a single vector being emitted by that state. So we are inherently recognizing one virtual pattern from K test patterns.

$$\begin{aligned} \alpha_{\phi(2)}(j) &= P(O_1^1, O_1^2, O_2^1, O_2^2, q_{\phi(2)} = j/\lambda) \\ &= \sum_{q_{\phi(1)}=i} P(O_1^1, O_1^2, O_2^1, O_2^2, q_{\phi(1)} = i, q_{\phi(2)} = j/\lambda) \\ &= \sum_{q_{\phi(1)}=i} P(O_1^1, O_1^2/q_{\phi(1)} = i)\ P(O_2^1, O_2^2/q_{\phi(2)} = j)\ P(q_{\phi(1)} = i)\ P(q_{\phi(2)} = j/q_{\phi(1)} = i) \\ &= \left[\sum_i \alpha_{\phi(1)}(i)\ a_{ij}\right]\ b_j(O_2^1, O_2^2) \end{aligned} \tag{51}$$

where $a_{ij} = P(\phi(t) = j/\phi(t-1) = i)$. It is the transition probability of moving from state i to state j and is assumed to be same as that given by the HMM.

$$\begin{aligned} \alpha_{\phi(3)}(j) &= P(O_1^1, O_1^2, O_2^1, O_2^2, O_3^1, q_{\phi(3)} = j/\lambda) \\ &= \sum_{q_{\phi(2)}=i} \sum_{q_{\phi(1)}=k} P(O_1^1, O_1^2, O_2^1, O_2^2, O_3^1, q_{\phi(1)} = k, q_{\phi(2)} = i, q_{\phi(3)} = j/\lambda) \\ &= \sum_{q_{\phi(1)}=i} \sum_{q_{\phi(2)}=k} P(O_1^1, O_1^2/q_{\phi(1)} = k) P(O_2^1, O_2^2/q_{\phi(2)} = i) \\ &\quad P(O_3^1/q_{\phi(3)} = j) P(q_{\phi(1)} = k) P(q_{\phi(2)} = i/q_{\phi(1)} = k) P(q_{\phi(3)} = j/q_{\phi(2)} = i) \\ &= \left[\sum_i \alpha_{\phi(2)}(i)\ a_{ij}\right]\ b_j(O_3^1) \end{aligned} \tag{52}$$

We assume a first order process. Here state j at $\phi(3)$ emits only O_3^1 and not O_2^2, as O_2^2 was already emitted at $\phi(2)$ by the HMM state. So we don't reuse vectors.

What was done in this example can be generalized to K patterns given the MPDTWpath ϕ between them and we get the recursive equation in equation 21. Since CMPVA-1 is similar to CMPFA-1, almost the same derivation (with minor changes) can be used to derive the recursive relation of CMPVA-1.

7. References

[Arslan & Hansen, 1996] Arslan, L.M. & Hansen, J.H.L. (1996). "Improved HMM training and scoring strategies with application to accent classification," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*.

[Arslan & Hansen, 1999] Arslan, L.M. & Hansen, J.H.L. (1999). "Selective Training for Hidden Markov Models with Applications to Speech Classication," *IEEE Trans. on Speech and Audio Proc.*, vol. 7, no. 1.

[Bahl et al., 1983] Bahl, L.R.; Jelinek, F. & Mercer, R.L. (1983). "A maximum likelihood approach to continuous speech recognition", *IEEE Trans. PAMI*, PAMI-5 (2), pp. 179-190.

[Baum & Petrie, 1966] Baum, L.E. & Petrie, T. (1966). "Statistical inference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, 37: pp. 1554-1563.

[Baum & Egon, 1967] Baum, L.E.& Egon, J.A. (1967). "An inequality with applications to statistical estimation for probabilistic functions of aMarkov process and to a model for ecology," *Bull. Amer. Meteorol. Soc.*, 73: pp. 360-363.

[Baum & Sell, 1968] Baum, L.E. & Sell, G.R. (1968). "Growth functions for transformations on manifolds," *Pac. J. Math.*,, 27 (2): pp. 211-227.

[Baum et al., 1970] Baum, L.E., Petrie, T., Soules, G. & Weiss, N. (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Stat.*, 41 (1): pp. 164-171.

[Baum, 1972] L.E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, 3: pp. 1-8, 1972.

[Cincarek et al., 2005] Cincarek, T.; Toda, T.; Saruwatari, H. & Shikano, K. (2005). "Selective EM Training of Acoustic Models Based on Sufficient Statistics of Single Utterances," *IEEE Workshop Automatic Speech Recognition and Understanding*..

[Cooke et al., 1994] Cooke, M.P.; Green, P.G. & Crawford, M.D. (1994). "Handling missing data in speech recognition," *Proc. Int. Conf. Spoken Lang. Process.*, pp. 1555-1558.

[Cooke et al., 2001] Cooke, M.; Green, P.; Josifovski, L. & Vizinho, A. (2001). "Robust automatic speech recognition with missing and unreliable acoustic data,"*Speech Commun.* 34(3), pp. 267-285.

[Fiscus, 1997] Fiscus, J.G. (1997). "A Post-Processing System to Yield Reduced Word Error Rates: Recognizer Output Voting Error Reduction (ROVER)", *Proc. IEEE ASRU Workshop*, Santa Barbara.

[Gersho & Gray, 1992] Gersho, A. & Gray, R.M. (1992). *Vector Quantization and Signal Compression*, Kluwer Academic Publishers.

[Haeb-Umbach et al., 1995] Haeb-Umbach, R.; Beyerlein, P. & Thelen, E. (1995). "Automatic Transcription of Unknown Words in A Speech Recognition System," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pp. 840-843.

[Holter et al., 1998] Holter, T. & Svendsen, T. (1998). "Maximum Likelihood Modeling of Pronunciation Variation," *Proc, of ESCA Workshop on Modeling Pronunciation Variation for ASR*, pp. 63-66.

[Huang et. al, 2004] Huang, C.; Chen, T. & Chang, E. (2004). "Transformation and Combination of Hidden Markov Models for Speaker Selection Training," *Proc. Int. Conf. on Spoken Lang. Process.*, pp. 10011004.

[Itakura & Saito, 1968] Itakura, F. & Saito, S. (1968). "An Analysis-Synthesis Telephony Based on Maximum Likelihood Method," *Proc. Int'l Cong. Acoust.*, C-5-5.

[Juang & Rabiner, 1990] Juang, B.-H. & Rabiner, L.R. (1990). "The segmental K-means algorithm for estimating parametersof hidden Markov models," *IEEE Trans. Audio, Speech, and Signal Process.*, vol. 38, issue 9, pp. 1639-1641.

[Lleida & Rose, 2000] Lleida, E. & Rose, R.C. (2000). "Utterance verification in continuous speech recognition: decoding and training procedures", *IEEE Trans. on Speech and Audio Proc.*, vol. 8, issue: 2, pp. 126-139.

[Myers et al., 1980] Myers, C., Rabiner, L.R. & Rosenburg, A.E. (1980). "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-28(6): 623-635.

[Nair & Sreenivas, 2007] Nair, N.U. & Sreenivas, T.V. (2007). "Joint Decoding of Multiple Speech Patterns For Robust Speech Recognition," *IEEE Workshop Automatic Speech Recognition and Understanding*, pp. 93-98, 9-13 Dec. 2007.

[Nair & Sreenivas, 2008 a] Nair, N.U. & Sreenivas, T.V. (2008). "Forward/Backward Algorithms For Joint Multi Pattern Speech Recognition," *Proceeding of* 16*th European Signal Processing Conference (EUSIPCO-2008)*.

[Nair & Sreenivas, 2008 b] Nair, N.U. & Sreenivas, T.V. (2008). "Multi Pattern Dynamic Time Warping for Automatic Speech Recognition," *IEEE TENCON 2008*.

[Nilsson, 1971] Nilsson, N. (1971). *Problem-Solving Methods in Artificial Intelligence*, NY, NY, McGraw Hill.

[Nishimura et al., 2003] Nishimura, R.; Nishihara, Y.; Tsurumi, R.; Lee, A.; Saruwatari, H. & Shikano, K. (2003). "Takemaru-kun: Speech-oriented Information System for RealWorld Research Platform," *International Workshop on Language Understanding and Agents for RealWorld Interaction*, pp. 7078.

[Rabiner et al., 1986] Rabiner, L.R.; Wilpon, J.G. & Juang, B.H. (1986). "A segmental K-means training procedure for connected word recognition," *AT & T Tech. J.*, vol. 64. no. 3. pp. 21-40.

[Rabiner, 1989] Rabiner, L.R. (1989). "A tutorial to Hidden Markov Models and selected applications in speech recognition", *Proceedings of IEEE*, vol. 77, no. 2, pp. 257-285.

[Rabiner & Juang, 1993] Rabiner, L.R. & Juang, B.H. (1993). *Fundamentals of Speech Recognition.*, Pearson Education Inc.

[Raj & Stern, 2005] Raj B. & Stern, R.M. (2005). "Missing-feature approaches in speech recognition," *IEEE Signal Proc. Magazine.*, vol. 2, pp. 101-116.

[Sakoe & Chiba, 1978] Sakoe, H. & Chiba, S. (1978). "Dynamic programming optimization for spoken word recognition," *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-26 (1): 43-49.

[Schwartz & Chow, 1990] Schwartz, R. & Chow, Y.-L. (1990). "The N-best algorithms: an efficient and exact procedure for finding the N most likely sentence hypotheses", *Proc. IEEE ICASSP*, vol.1, pp. 81-84.

[Shannon, 1948] Shannon, C.E. (1948). "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656.

[Singh et al., 2002] Singh, R.; Raj, B. & Stern, R.M. (2002). "Automatic Generation of Subword Units for Speech Recognition Systems," *IEEE Transactions on Speech and Audio Processing*, vol. 10(2), 89-99.

[Svendson, 2004] Svendsen, T. (2004). "Pronunciation modeling for speech technology," *Proc. Intl. Conf. on Signal Processing and Communication (SPCOM04)*.

[Soong & Hung, 1991] Soong, F.K.& Hung, E.-F. (1991). "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition," *Proc. IEEE ICASSP 91*, vol 1, pp. 705-708.

[Viterbi, 1967] Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Trans. Inf.Theory*, vol. IT-13, no.2, pp. 260-269.

[Wu & Gupta, 1999] Wu, J. & Gupta, V. (1999). "Application of simultaneous decoding algorithms to automatic transcription of known and unknown words", *Proc. IEEE ICASSP*, vol. 2, pp. 589-592.

[Yoshizawa et al., 2001] Yoshizawa, S.; Baba, A.; Matsunami, K.; Mera, Y.; Yamada, M.; Lee, A. & Shikano, K. (2001). "Evaluation on unsupervised speaker adaptation based on sufficient HMM statistics of selected speakers," *Proc. European Conference on Speech Communication and Technology*, pp. 1219-1222.

Speech Recognition

Edited by France Mihelic and Janez Zibert

ISBN 978-953-7619-29-9

Hard cover, 550 pages

Publisher InTech

Published online 01, November, 2008

Published in print edition November, 2008

Chapters in the first part of the book cover all the essential speech processing techniques for building robust, automatic speech recognition systems: the representation for speech signals and the methods for speech-features extraction, acoustic and language modeling, efficient algorithms for searching the hypothesis space, and multimodal approaches to speech recognition. The last part of the book is devoted to other speech processing applications that can use the information from automatic speech recognition for speaker identification and tracking, for prosody modeling in emotion-detection systems and in other speech processing applications that are able to operate in real-world environments, like mobile communication services and smart homes.

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen