Language identification using ergodic Markov patterns

Amogh Rajanna
ECE Dept, SJCE, Mysore, India
amoevol@yahoo.co.in

Abstract:
In this paper, a spoken language recognition system using ergodic Hidden Markov Models as reference patterns is studied. The HMM's considered are continuous valued and have multivariate Gaussian Mixture Distribution. The system is tested against 3 language utterances for its robustness.

I **Introduction**:

Spoken language recognition is an interesting research problem and has been around for the last decade with many optimistic solutions [5], [4]. The earliest works in language recognition by House and Neuberg [3] was based on the hidden Markov model (HMM) where the potential of the discrete ergodic HMM to model sequential characteristics of broad phonetic labels derived from texts of different languages is exploited. Following this, there have been a few more ideas to use HMMs for language recognition [2], [1].

II **Linear Prediction Front End Analysis**:

The feature vectors corresponding to the input speech necessary for Language Recognition are derived from LPC Front end analysis. The LPC order and the cepstrum order used for the front end analysis are 10 and 12 respectively. The block diagram in the next page shows how the LPC Front End Analysis is implemented for obtaining feature vectors corresponding to the input speech. Detailed description of each block is given below

2.1 Premphasis:

The input speech signal s[n] is passed through a premphasis block to account for the attenuation of high frequency components in the vocal tract. The premphasis filter used is a digital FIR High Pass Filter with transfer function H (z) = 1 - $0.95z^{-1}$.

$$sp[n] = s[n] - 0.95s[n-1]$$

2.2 Frame Blocking:

The premphasised speech signals sp[n] is segmented into frames. A frame contains 300 samples. Adjacent frames are distinct by 100 samples.

$$SF[i, n] = sp[(i-1)*100+n]; \quad 1<=n<=300$$

2.3 Hamming Windowing:

Each speech frame is windowed by a Hamming window of length 300.Each speech frame is windowed to taper the speech frame at the edges to zero.
Hamming Window

$$w[n] = 0.54 - 0.46\cos(2\pi n/299); \quad 0<=n<=299$$

$$SF[i,n] = w[n-1]SF[i,n]; \quad 1<=n<=300$$

2.4 Autocorrelation Analysis:

The autocorrelation analysis leads to an autocorrelation vector given by

$$AC[m] = \sum_{1}^{300} F[n]F[n+m]; \quad 0<= m <=p$$

where F[n] is the nth sample in the speech frame, p is the LPC order.

2.5 LPC Coefficients:

LPC Coefficients are derived recursively from autocorrelation coefficients using the standard Levinson-Durbin Algorithm.

$$k[i] = (AC[i] - \sum_{1}^{i-1} AL[i-1, j]\, AC[i-j]) / E[i-1]$$

$$AL[i, i] = k[i]$$

$$AL[i, j] = AL[i-1,j] - k[i]AL[i-1,i-j]; \quad 1<= j <= (i-1)$$

$$E[i] = (1 - k[i]^2)E[i-1]$$

The algorithm is repeated for the range 1<= i<= p and then, the LPC coefficients are

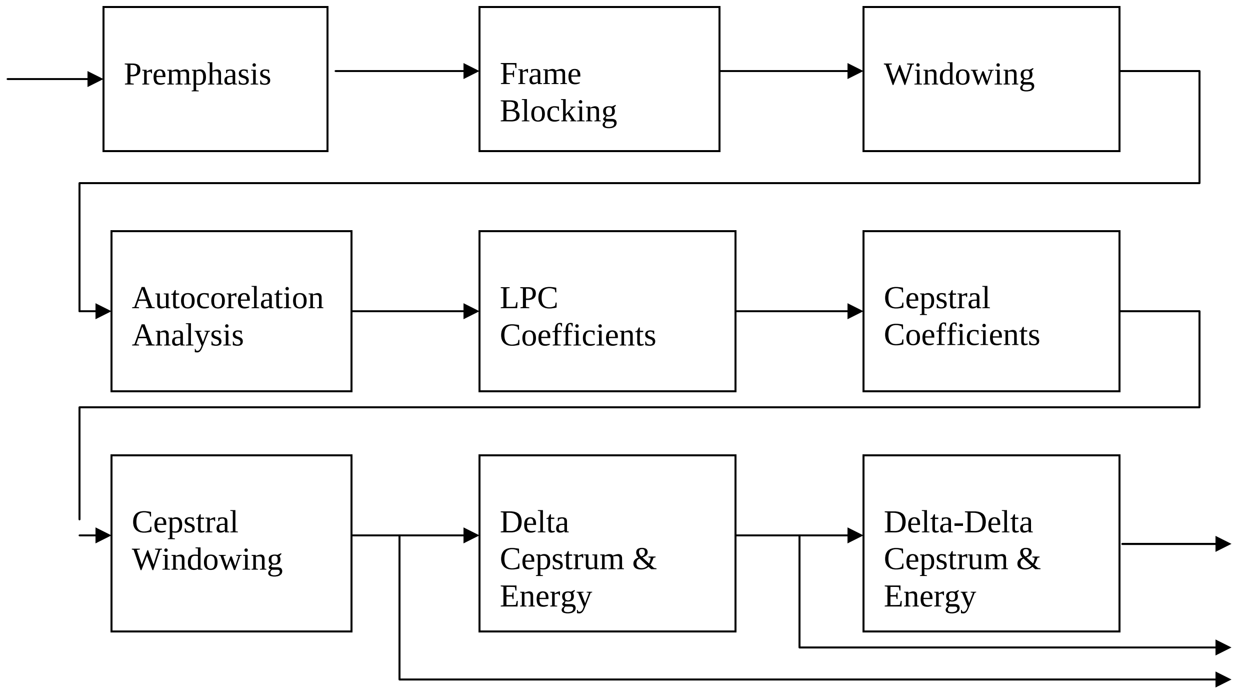

Diagram 1: LPC Front end Analysis

AL[i , j] for 1<= j <=p.

2.6 Cepstral Coefficients:
The cepstral coefficients are derived from the LPC Coefficients using the following relations
c[0] = log(G) where G is the gain term of the LPC model.

$$c[m] = LPC[m] + \sum_{1}^{m-1} kc[k]LPC[m-k]/m;$$

1<= m <=p.

$$c[m] = \sum_{1}^{m-1} kc[k]LPC[m-k]/m; \; m>p$$

2.7 Cepstral Windowing:
The weighted cepstral coefficients are obtained from the cepstral coefficients using a cepstral Window of length Q.
Cepstral Window
$w[n] = 1+ Q \sin(\pi n/Q)/2$
$WC[m] = c[m]w[m]$; 1<= m <=Q

2.8 Delta Cepstrum & Energy:
The delta cepstrum & delta energy for each frame are derived using the following relations

$$DC[t, m] = G1 \sum_{-2}^{2} kc[t+k, m]; \; 1 <= m <= Q$$

where G1 = 0.375

$$DE[t] = G2 \sum_{-2}^{2} kE[t+k]; \; G2=0.0375$$

2.9 Delta-Delta Cepstrum & Energy:
The delta-delta cepstrum & delta-delta energy of each frame are derived using the following relations.

DDC[t , m] = G (DC[t+1, m] - DC[t-1, m])
1<= m <=Q & G = 0.375

DDE[t] = G (DE[t+1] - DE[t-1])

Therefore the feature vector of a speech frame = Weighted cepstrum + Delta cepstrum + Delta-Delta cepstrum + Delta energy + Delta-Delta energy.
The parameters defining the LPC Front end analysis are
Sampling rate F_s: 10000 Hz
Number of samples in a speech frame
N: 300
Number of distinct samples in a speech frame M:100

LPC Order p:10
Cepstrum Order Q:12
G1:0.375
G2:0.0375
G:0.375
If the speech utterance contained T frames, the observation sequence contains T frames & is given by $O = \{O_1,O_2,O_3......O_T\}$ where O_i is the observation frame corresponding to the i^{th} speech frame.

III **Language Recognition**:

Hidden Markov Model is a statistical model in which the state sequence is hidden & output sequence is observable. Each language to be recognized is statistically modeled by a Hidden Markov Model. HMM is a widely used statistical model used in speech recognition, Language recognition, bioinformatics, weather forecasting & ISI Channel modeling in wireless communication etc.
The HMM's used for language modeling were ergodic. Ergodic HMM is a HMM in which transition from any state ' i ' to other N-1 states is possible. The output of the HMM's used were continuous valued vectors with multivariate Gaussian mixture densities. A HMM is characterized by 3 parameters namely

1.) Transition Probability Matrix A:
A is a square matrix of order N, where N is the number of states in the HMM. The elements are A[i , j] with 1<=i , j <=N .The element gives the probability of HMM transiting from state ' i ' to state ' j '.
2.) Initial Probability Vector PI:
PI is a vector with N elements, where PI[i] represents the probability of HMM initially being in state ' i '.
3.) State Observation Density B:
Discrete HMM: B is a matrix of dimension N*M, where M is the number of discrete observation symbols. An element B[i , j] represents the probability of symbol S[j] being produced by state ' i '.
Continuous HMM: B is a vector with N elements, where B[i] represents the state ' i ' observation probability.

In continuous HMM case, the observation vector O_i produced by state ' i ' is modeled by a Multivariate Gaussian Mixture Density. So each state has Gaussian mixture components.

$$b[j , \mathbf{o}] = W[j , k] * N(\mathbf{o}, M[j , k], V[j , k])$$

where
j = State of Continuous HMM
o = Observation frame.
W[j , k] = Weight of state j & mixture k
M[j , k] = Mean vector of state j & mixture k
V[j , k] = Covariance matrix of state j & mixture k
L = Length of multivariate Observation vector.
M_G = Number of mixture components in a HMM state.
and

$$N(\mathbf{o}, M[j , k], V[j , k]) = \left((2\pi)^L |V[j , k]| \exp((\mathbf{o} - M[j , k])(V[j ,k])^{-1}(\mathbf{o} - M[j , k])^H)\right)^{-0.5}$$

In our implementation, each language is modeled by a 5 state HMM & each state having 5 Gaussian mixture components.

Training the HMM's:
Segmental K-Means algorithm training was used to train the HMM's for specific languages. From a random initial HMM for each language, the optimum HMM is obtained using Segmental K-Means training.
The block diagram of Segmental K-Means training is shown in the next page.

1.) Viterbi Segmentation:
Given an observation sequence & HMM [A, PI, B]; the Viterbi segmentation is done as follows.

Define the maximum joint probability variable

$$d(t,i) = \max_{q_1 q_2 ... q_{t-1}} P(q_1\, q_2 ....q_{t-1}, q_t= i,O_1O_2 .....O_t)$$

Initialization: $d(1,i) = PI[i]\, b[i,O_1]$;$1<= i <=N$
Recursion:

$$d(t, j) = \max_{1<=i<=N} \{ d(t-1, i) A[i, j] \} * b[j, O_t] ;$$

$2<=t<=T$ & $1<=j<=N$

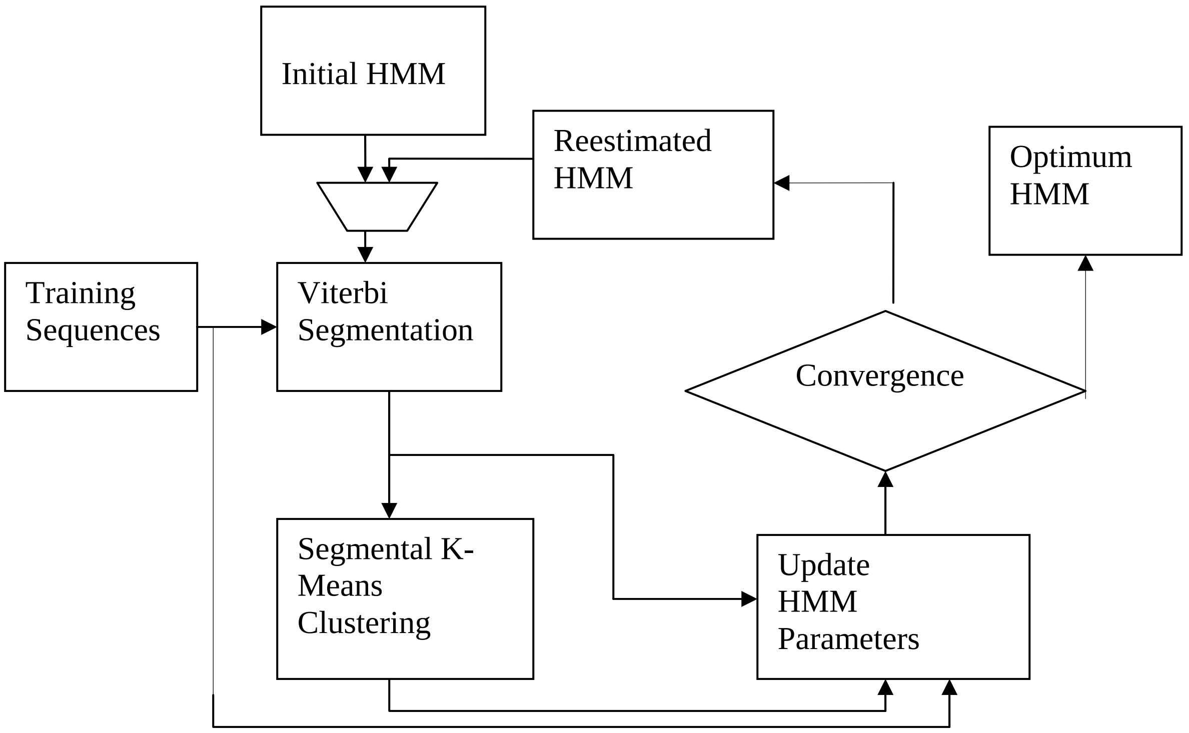

Diagram 2: Segmental K-Means Reestimation

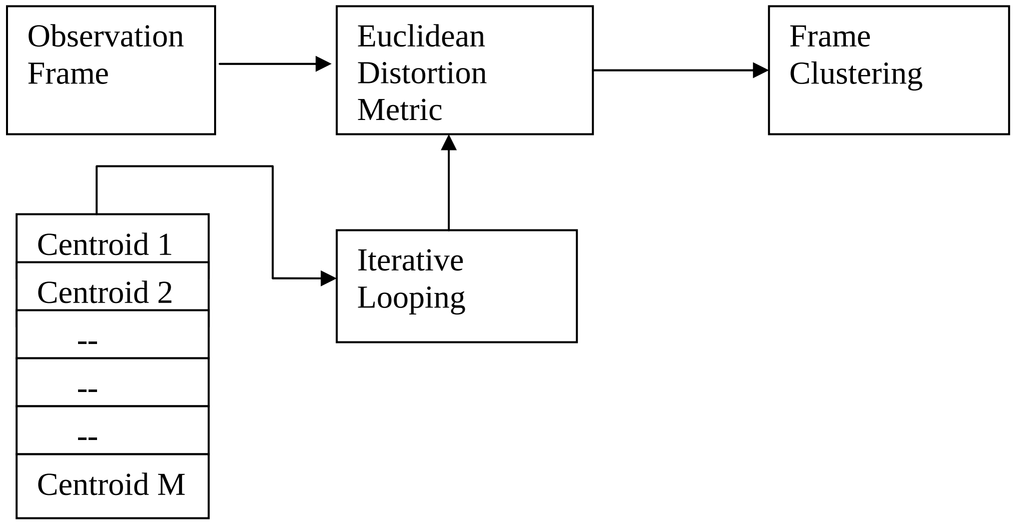

Diagram 3: Segmental K-Means Clustering

$$PH(t, j) = \arg\max_{1<=i<=N} \{ d(t-1, i) A[i, j] \}$$

Path Backtracking:

$$Q[T] = \arg\max_{1<=i<=N} \{ d(T,i) \}$$

$Q[t] = PH(t+1, Q[t+1]); t = [T-1,1]$

2.) Segmental k-means clustering: Observation frames belonging to each state are clustered into M clusters, each cluster represents a Gaussian mixture component of that state. Observation frames in a state are clustered into M clusters using M centroids corresponding to M mixtures & Euclidean Distortion Metric.

3.) Update HMM:

$A[i, j] = N_{ij} / N_i$
N_{ij} = number of times HMM transists from state ' i ' to state ' j '.
N_i = number of times HMM transists from state ' i '.

$W[j, k] = H_{jk} / N_j$
H_{jk} = number of observation frames in state ' j ' & mixture ' k '.

V[j , k] = sample covariance matrix of observation frames in state j & mixture k .

M[j , k] = sample mean of observation frames in state ' j ' & mixture ' k '.

4.) Convergence:
Viterbi Likelihoods of HMM's with respect to the observation sequence are used to test for convergence. If V(r) & V(r-1) represent the Viterbi likelihoods of HMM's at the iterations ' r ' & ' r-1 ' respectively, then if $V(r) - V(r-1) < 10^{-4}$,convergence is achieved.

Algorithm for Language Recognition:
The diagram in the next page shows the algorithm used for language recognition. For a given test utterance, the language corresponding to the HMM which produces the maximum Viterbi likelihood of observation is hypothesized as the spoken language.

Viterbi Likelihood:
For a given HMM [A, PI, B] and Observation sequence $\{O_1,O_2,......O_T\}$, the Viterbi Likelihood V is calculated as follows
$V[i] = P_{LM} * P_A$
where P_{LM} = Language Model Likelihood and P_A = Acoustic Likelihood
$P_{LM} = \mathbf{P(}Q_{i1}Q_{i2}......Q_{iT}\mathbf{)}$
and
$P_A = \mathbf{P(}O_1O_2.....O_T/Q_{i1}Q_{i2} .....Q_{iT}\mathbf{)}$
where the state sequence used is the Maximum Likelihood Viterbi state sequence obtained by Viterbi segmentation.

IV **Simulation Results**:
The system was tested against speech utterances across 3 languages (English, German & Hindi) and the performance was satisfactory. Improvements to the current system can be done by employing Baum Welch/Forward Backward algorithm for HMM generation and training.

V **References**:
[1]. M. Savic, E. Acosta, and S. K. Gupta. An automatic language identification system. In *Proc. ICASSP*, pages 817–820, 1991.
[2] M. A. Zissman. Automatic language identification using Gaussian mixture and hidden Markov models. In *Proc. ICASSP*, pages 399–402, Apr 1993.
[3] A. S. House and E. P. Neuberg. Toward automatic identification of the language of an utterance. *Journal of Acoustic Society of America*, 62(3):708–713, Sep 1977.
[4] M. A. Zissman and K. M. Berkling. Automatic language identification. *Speech Commun*, 35(1-2):115–124, Aug 2001.
[5] Y. K. Muthusamy, E. Barnard, and R. A. Cole. Reviewing automatic language identification. *IEEE Signal Processing Magazine*, 11(4):33–41, Oct 1994.

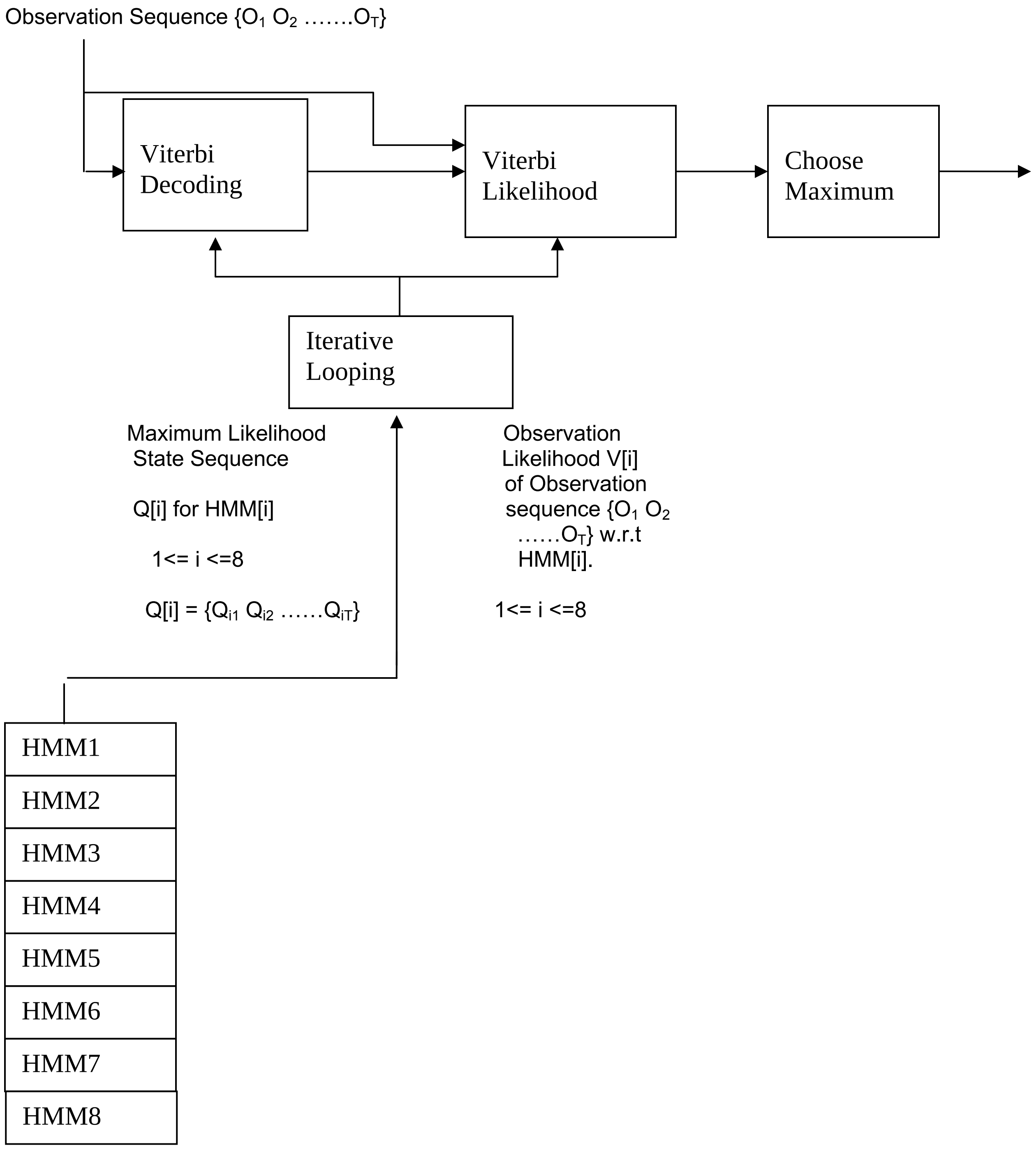

Diagram 4: Algorithm for language recognition.