ASR using GMM-HMM and Language model

SamudraVijaya K
Adjunct Faculty, IIT Dharwad; Visiting Faculty, NMIMS Mumbai
20-JUL-2021

samudravijaya@gmail.com

Outline

- Sequential pattern recognition
- Signal processing
- Recognition of static patterns
    a. Statistical and probability model
    b. Gaussian Mixture Model (GMM)
- Recognition of sequential patterns
- Hidden Markov Model (HMM)
    a. Motivation
    b. Training and testing HMM
    c. Kaldi: mono, tri1, tri2, tri3 models
- Language Models

Speech and speaker recognition

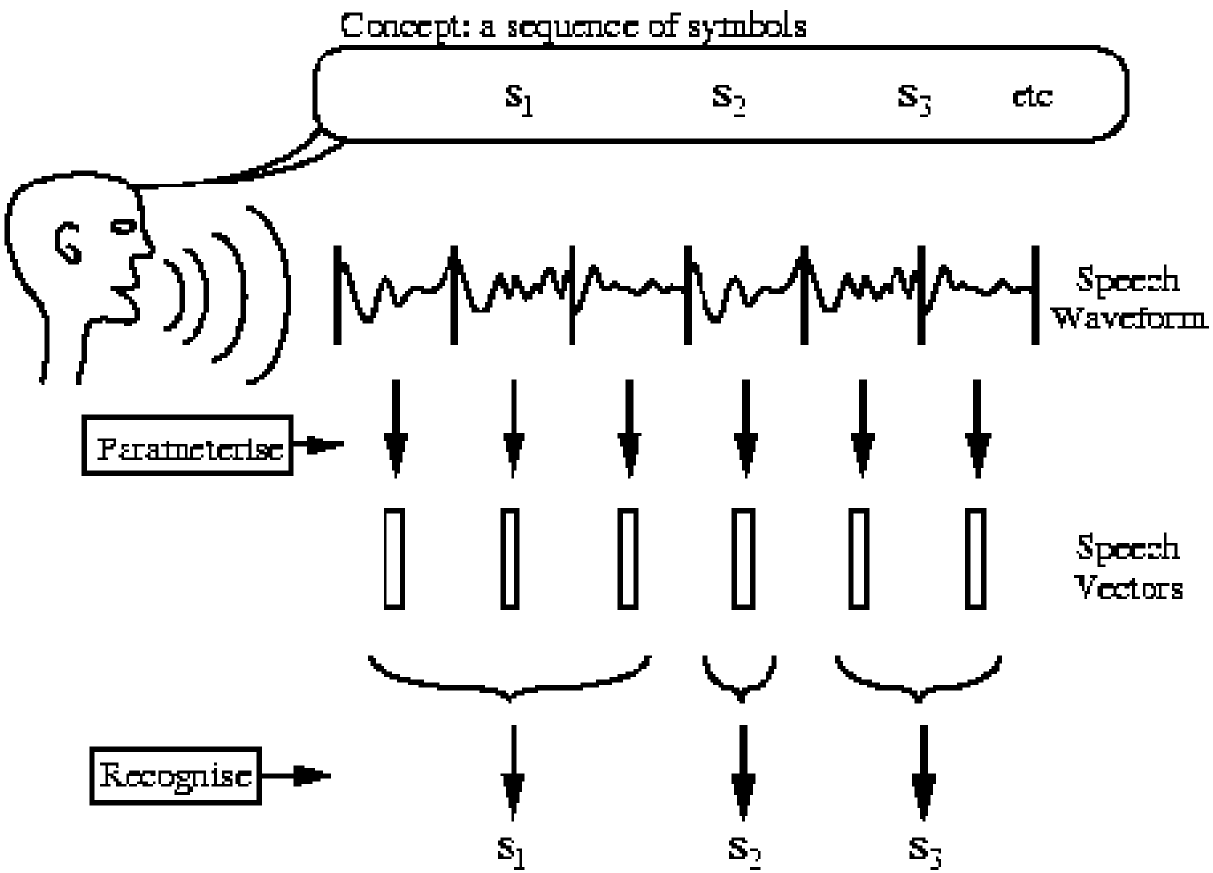
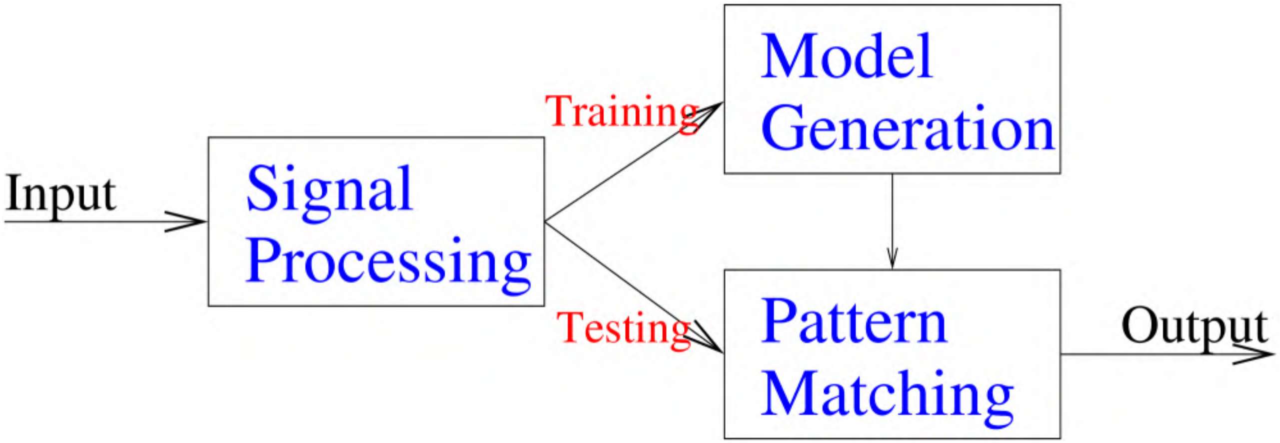


Fig. 1.1 Message Encoding/Decoding

Source: HTKbook

samudravijaya@gmail.com

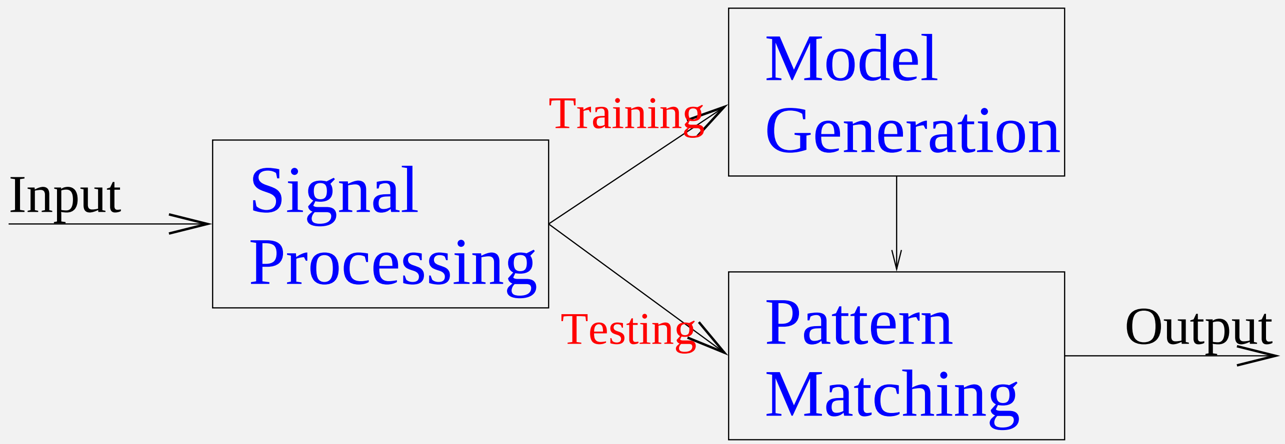
Speech recognition is recognition of sequential patterns



Goal: Recognise sequential pattern from reference templates / models

Two phases: Training (learning) and Testing (recognition)

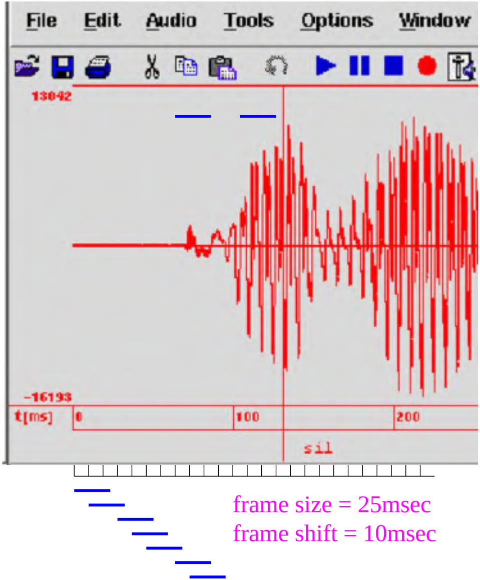
Pattern Recognition



GMM: static patterns

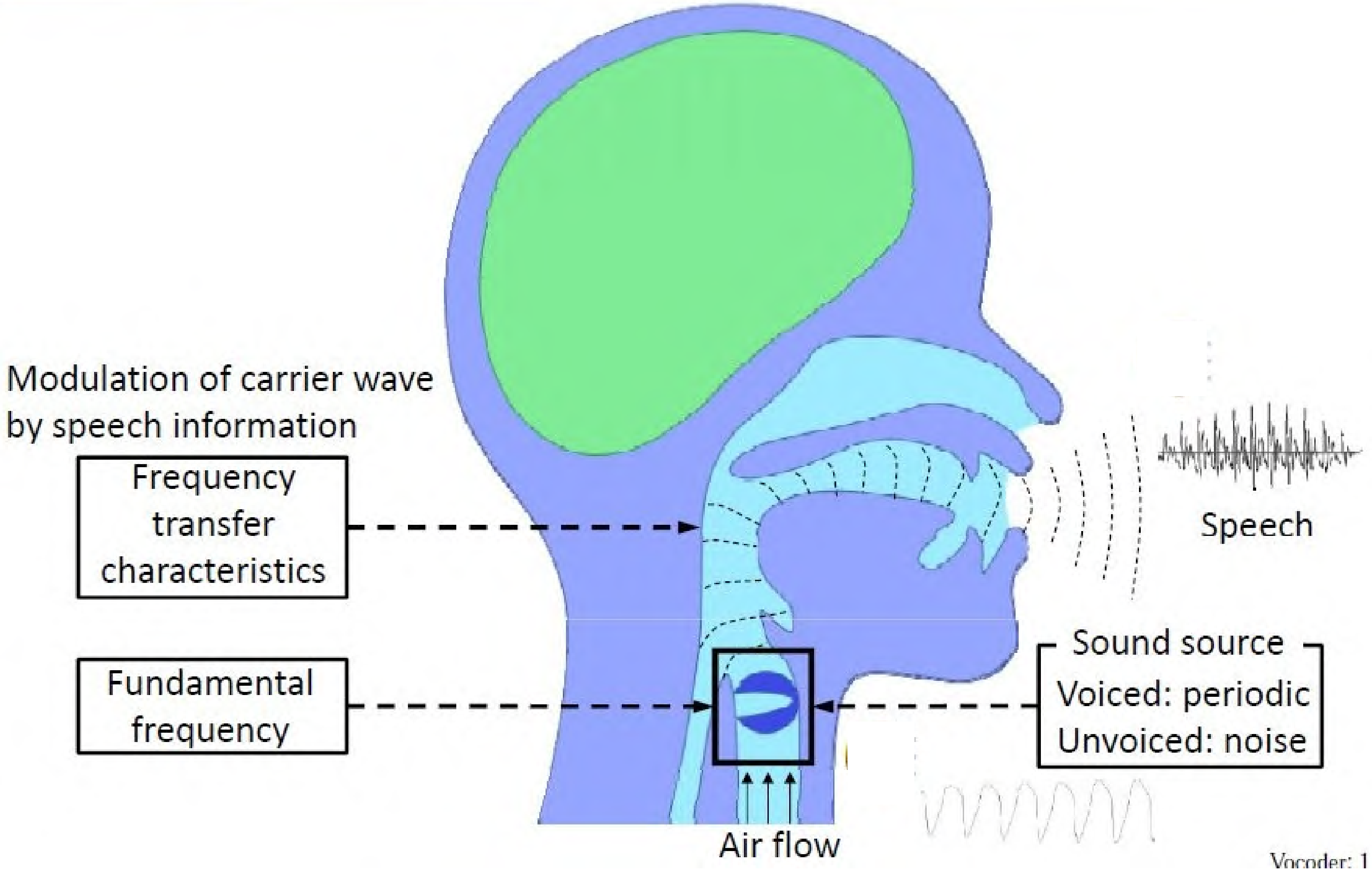
HMM: sequential patterns (quasi-stationary)

Short time speech processing



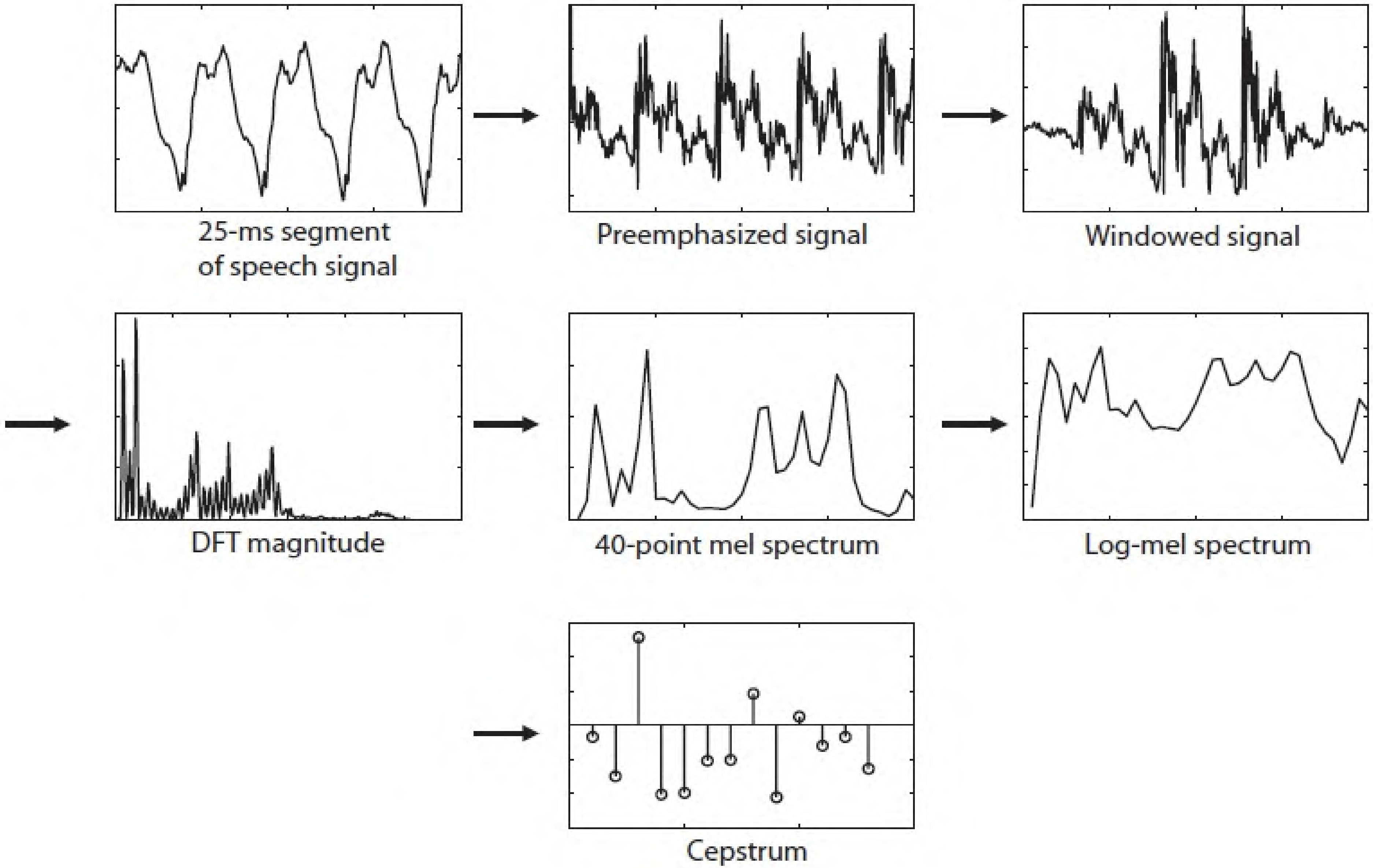
frame size = 25msec
frame shift = 10msec

Speech Production Mechanism



Source: Tomoki Toda; WiSSAP 2013

Wave → MFCC



source: e-Book: "Techniques for Noise Robust Automatic Speech Recognition"

samudravijaya@gmail.com

Speech Signal Processing (Feature Extraction)

- Digitisation of analog speech signal
- Blocking signal into frames
- FFT $\rightarrow$ mel filter $\rightarrow$ log $\rightarrow$ IFFT $\Rightarrow$ MFCC
- Slope and curvature
- Sequence of feature vectors $: x_1, x_2, \ldots x_T$

$$: o_1, o_2, \ldots o_T$$

Phone: A sound generated by human vocal apparatus and used for human communication in a language.

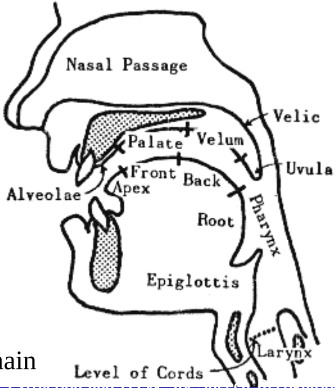
Phoneme: Smallest meaningful contrastive unit in the phonology of a language.

Allophones: "p" and "ph" are allophones of one phoneme /p/ in English,

are two distinct phonemes in Hindi

Minimal pair:

Place and Manner of articulation



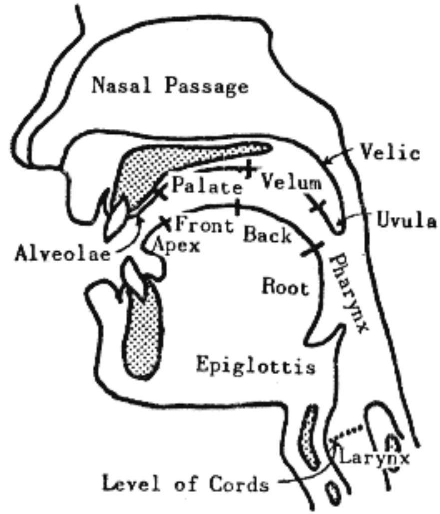
source: public domain

source: public domain

Place and Manner of Articulation

अ	आ, ◌ा	इ, ◌ि	ई, ◌ी	उ, ◌ु	ऊ , ◌ू	ए , ◌े	ऐ , ◌ै	ओ , ◌ो	औ , ◌ौ
a	aa	i	ii	u	uu	e	ee	o	oo

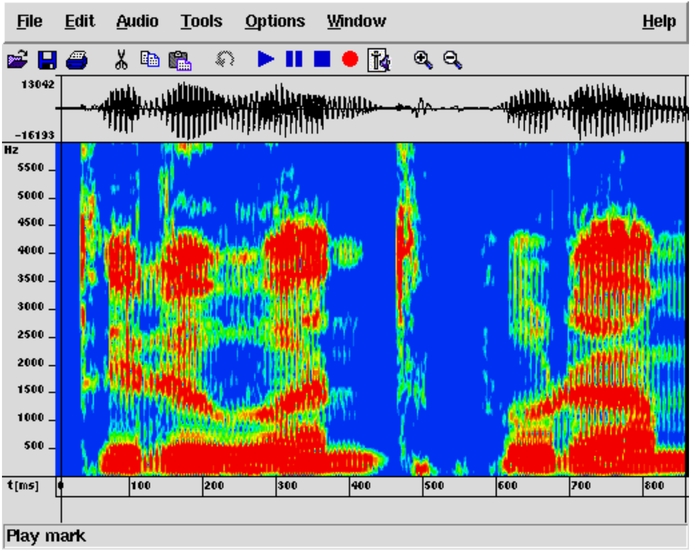
क k	ख kh	ग g	घ gh	ङ ng
च c	छ ch	ज j	झ jh	ञ nj
ट tx	ठ txh	ड dx	ढ dxh	ण nx
त t	थ th	द d	ध dh	न n
प p	फ ph	ब b	भ bh	म m



source: public domain

य	र	ल	व	श	ष	स	ह
y	r	l	w	sh	sx	s	h

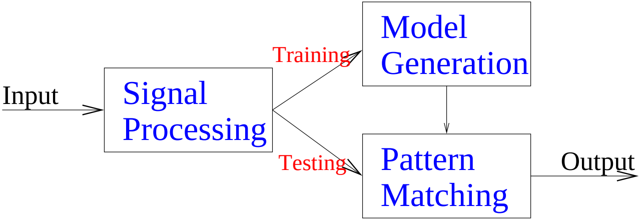
Speech: a dynamic signal



Formant: frequency of resonance: F1, F2, F3, ...

Slope and curvature of trajectory

Recognition of (static) patterns

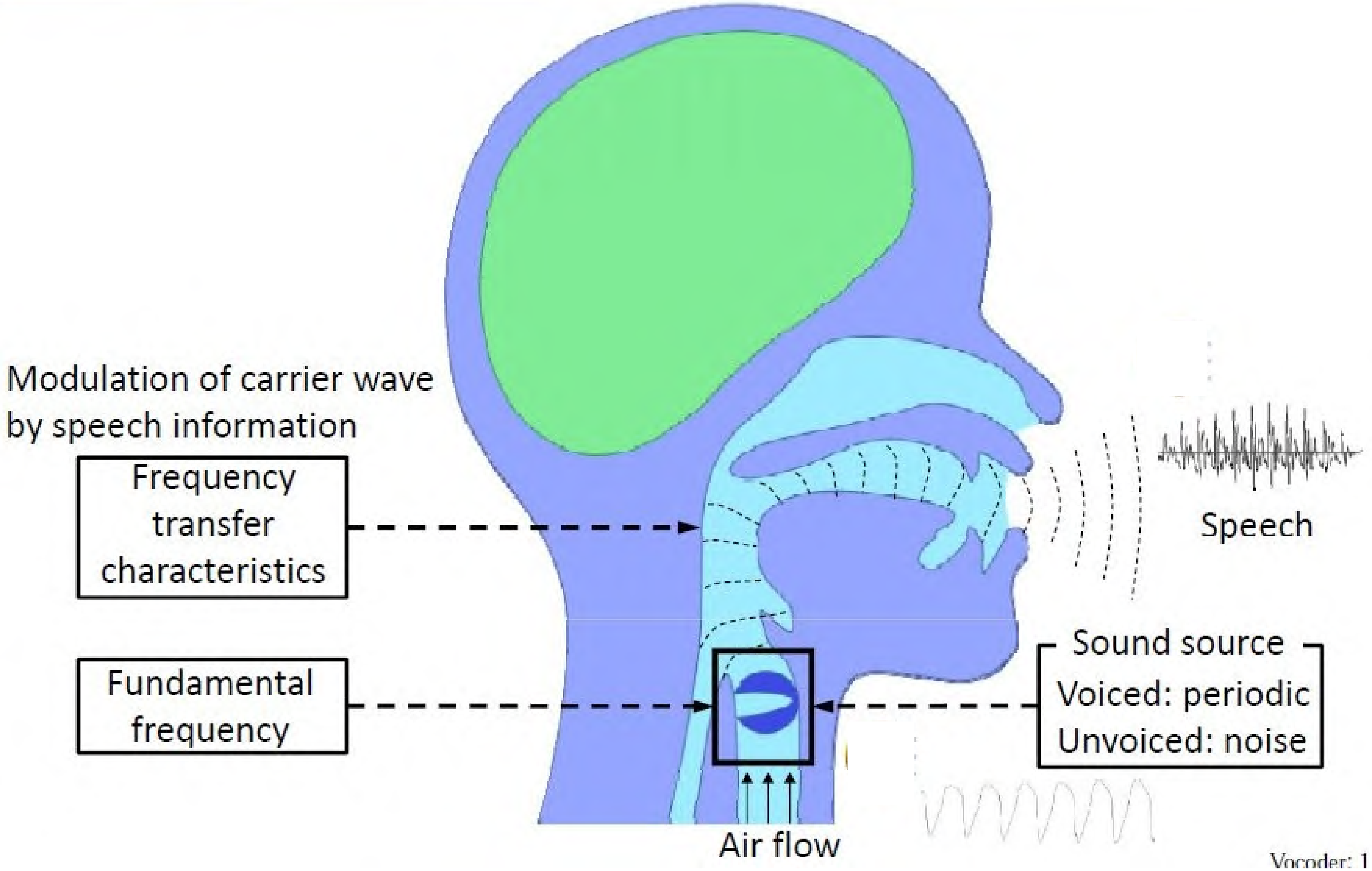


Signal Processing ⇒ Sequence of feature vectors

Pattern Recognition

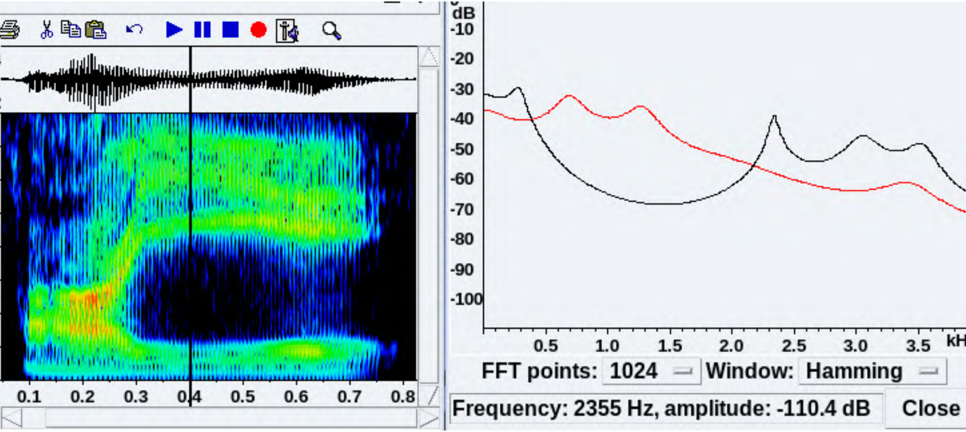
Illustration: Vowel recognition with the first 2 Formant frequencies as features

Speech Production Mechanism



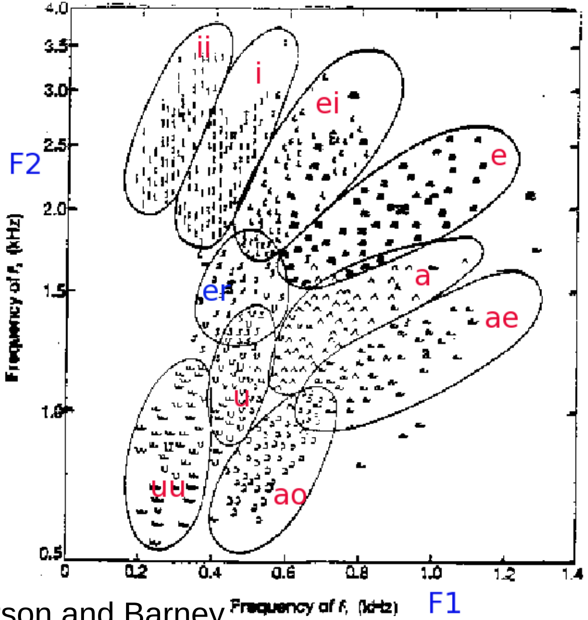
Source: Tomoki Toda; WiSSAP 2013

Measurement of Formant frequencies



Vowel: formant frequencies (Hz) (Signatures)
/aa/: F1=700; F2=1300
/i/ : F1=300; F2=2300
/e/ : F1=350; F2=2100

Formant space of vowels

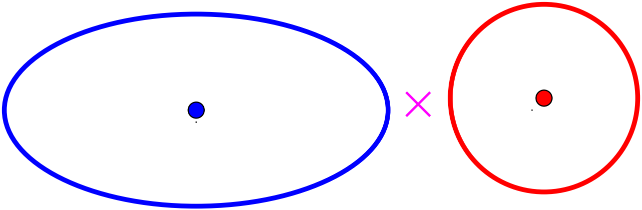


source: Peterson and Barney

Classification criterion

* *E*uclidean Distance

$$x \in C_k \quad \text{if } (x-\mu_k)^2 \;\leq\; (x-\mu_j)^2 \;\; \forall j$$



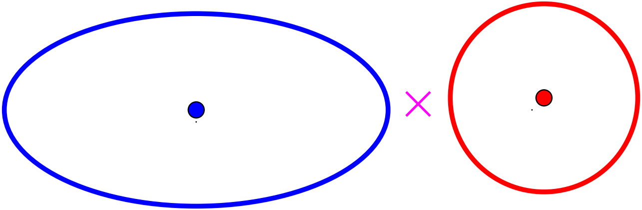
* Weighted Euclidean distance

$$d^k = \sqrt{\left(\frac{\mathbf{x}-\mu^{\mathbf{k}}}{\sigma^k}\right)^2}$$

Classification criterion

* *E*uclidean Distance

$$x \in C_k \quad \text{if } (x-\mu_k)^2 \;\leq\; (x-\mu_j)^2 \;\; \forall j$$



* Weighted Euclidean distance

$$d^k = \sqrt{\left(\frac{\mathbf{x}-\mu^{\mathbf{k}}}{\sigma^k}\right)^2}$$

* Extension to multiple features

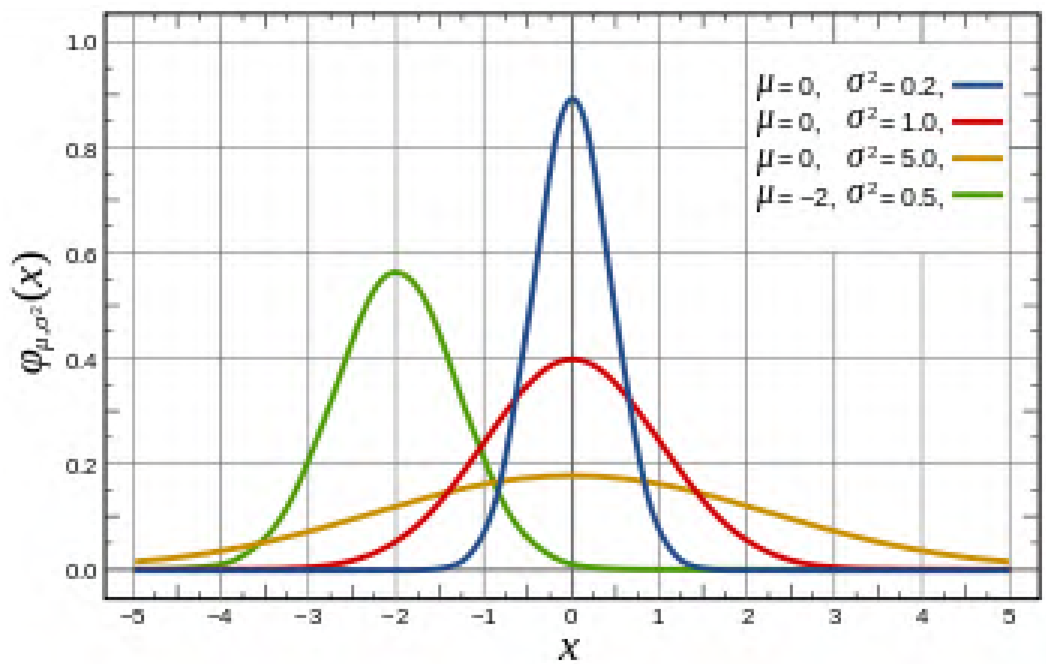
$$d^k = \sqrt{\sum_i \left(\frac{\mathbf{x_i}-\mu_{\mathbf{i}}^{\mathbf{k}}}{\sigma_i^k}\right)^2}$$

$d(\overline{\mathbf{x}}, \overline{\mu_{\mathbf{k}}})$

• Univariate Gaussian Distribution

- Normal distribution: $\mathbf{N}(\mu;\sigma)$ $\mathbf{p(x)} = \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi\sigma^{\mathbf{2}}}}\mathbf{exp}\{-\mathbf{0.5}(\frac{\mathbf{x}-\mu}{\sigma})^{\mathbf{2}}\}$

- Parameters:
  - Mean (**μ**)
  - Variance (**σ²**)



source: public domain

Estimation of parameters

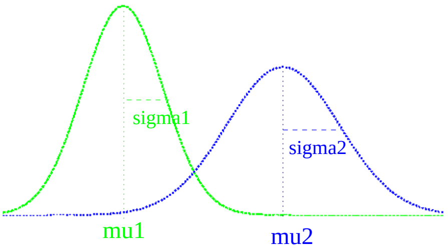
Probability Vs Likelihood (conditional probability)

- samudravijaya@gmail.com

Two class problem

Normal Distribution: $N(\mu;\sigma)$

$$p(x) \;=\; \frac{1}{\sqrt{2\pi\sigma^2}} exp\left\{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right\}$$



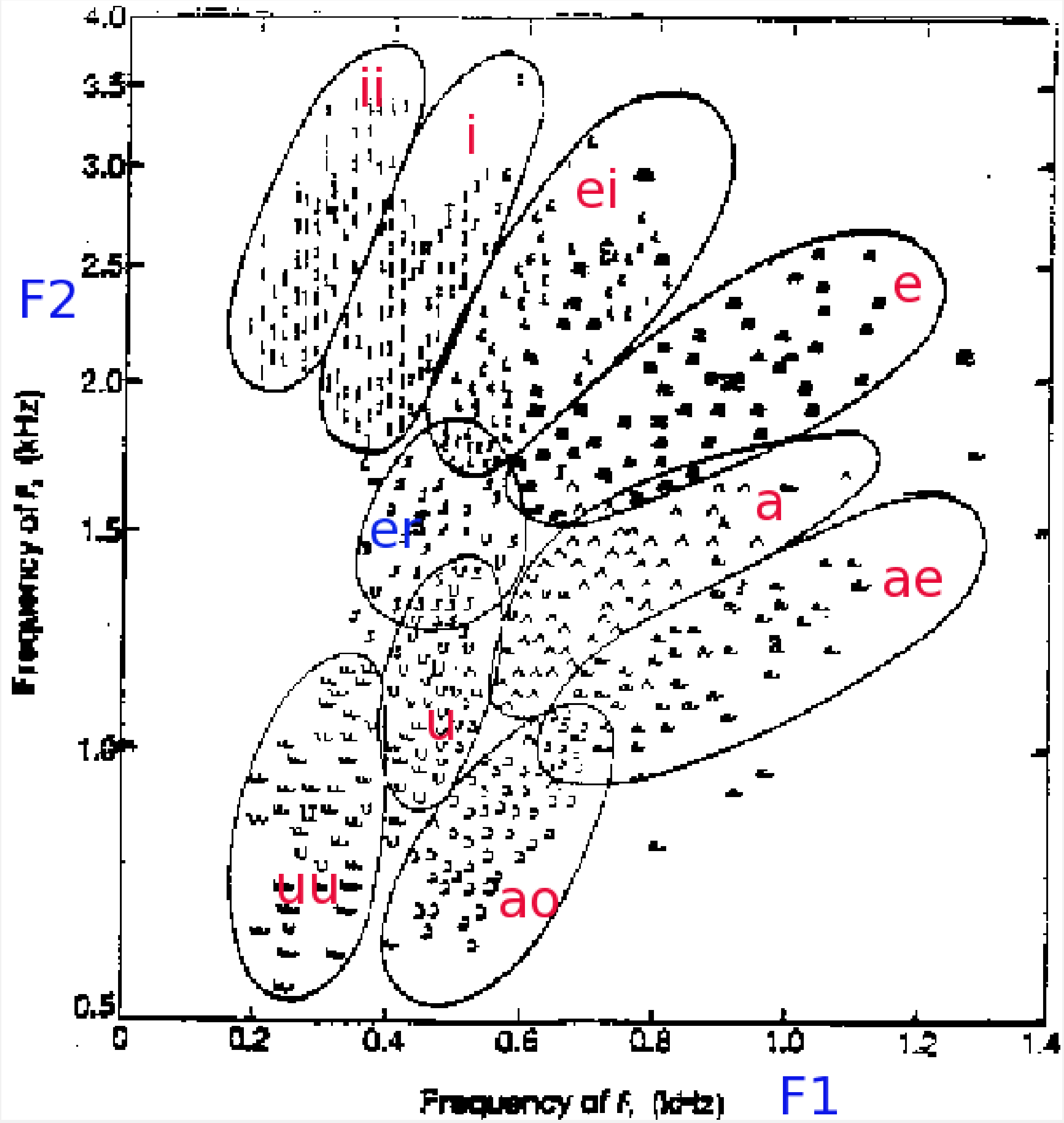
Maximum Likelihood classification criterion:

$$x \in C_k \quad \text{if } p(x|N(\mu_k;\sigma_k)) \;\geq\; p(x|N(\mu_j;\sigma_j)) \quad \forall j$$

Refer to vowel F1-F2 diagram

Need for GMM : vowels in F1-F2 space

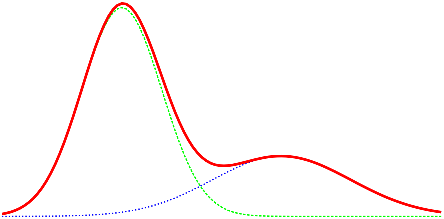
fnlp/lect



(1952)

source: Peterson and Barney

Gaussian Mixture Model(GMM)



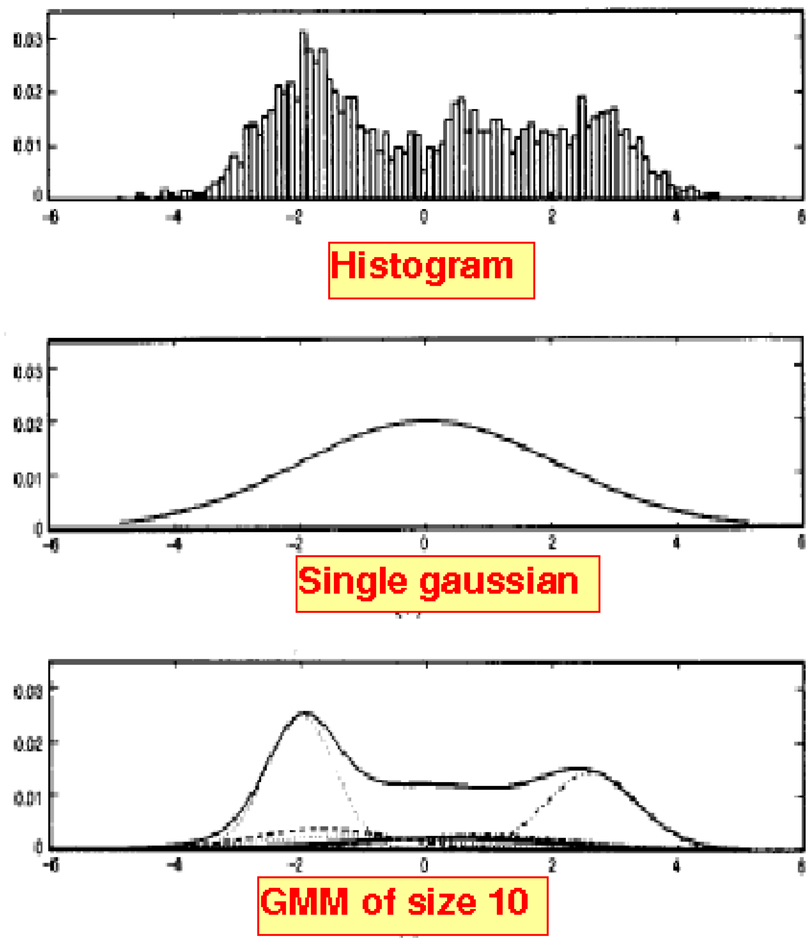
$$p(x|GMM(k)) \; = \alpha p(x : N[\mu_1; \sigma_1]) \quad + (1-\alpha)\; p(x : N[\mu_2; \sigma_2])$$

Maximum Likelihood classification criterion for GMM case:

$$x \in C_k \quad \text{if } p(x|GMM(k)) \;\geq\; p(x|GMM(j)) \quad \forall j$$

Extension to Multi-dimensional space

Multi-modal Distributions

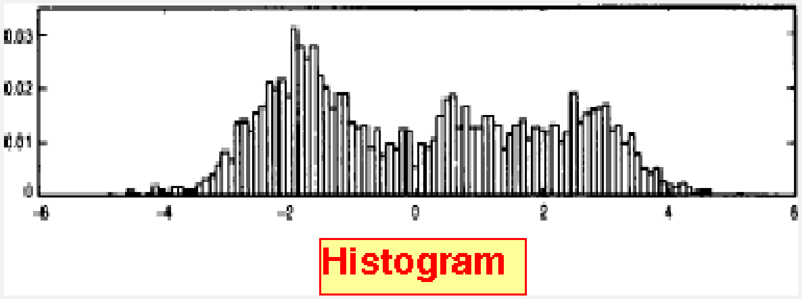


source: public domain

- Distribution of cepstral coefficient of a phone

- samudravijaya@gmail.com

Distribution of a Cepstral Coefficient



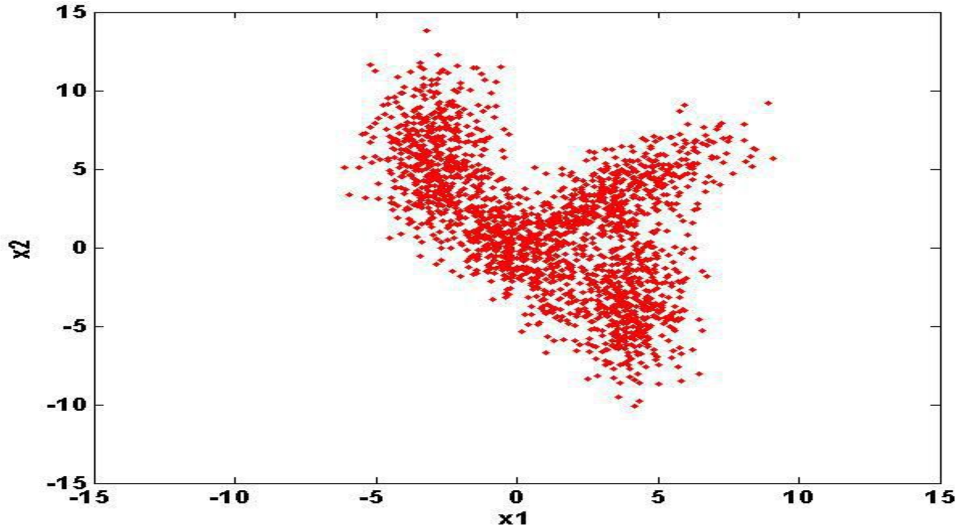
source: public domain

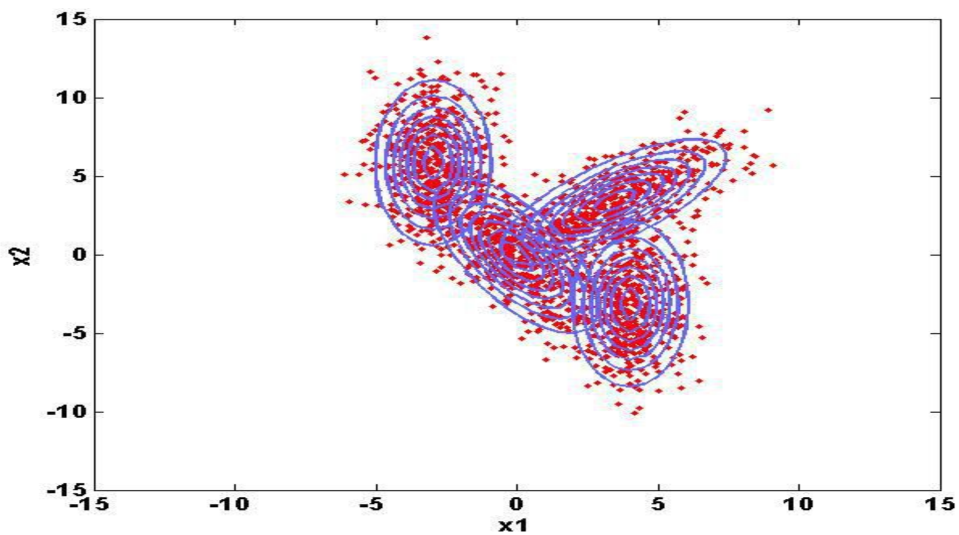
This can be modeled by a GMM of 3 mixtures.

$$p(x) = w_1 N(x;\mu_1,\sigma_1) + w_2 N(x;\mu_2,\sigma_2) + w_3 N(x;\mu_3,\sigma_3) \quad \text{and} \sum w_i = 1$$

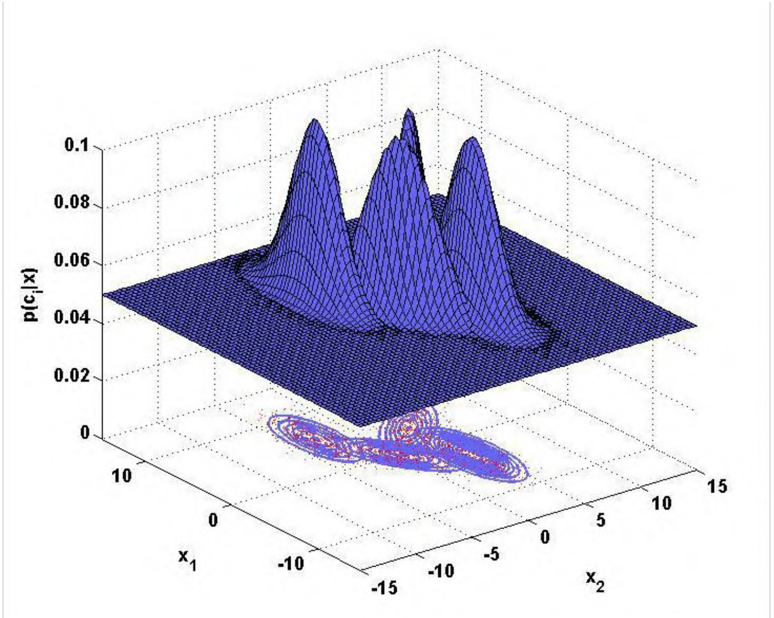
Multimodal Distribution

For a class whose data is considered to have multiple clusters, the probability distribution is multimodal





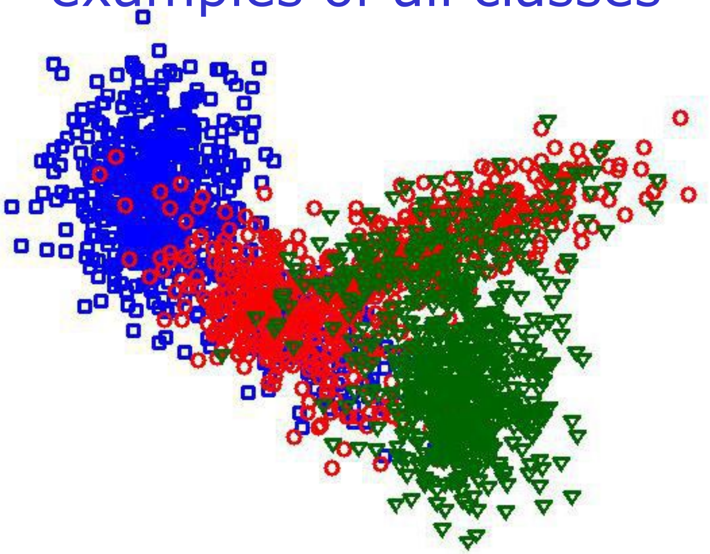
Bivariate multimodal distribution



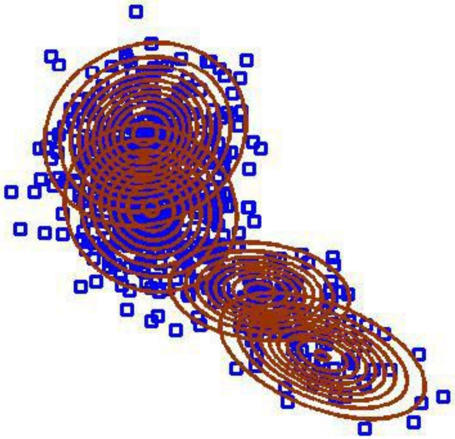
Source: C. ChandraSekhar, IITM

GMMs for Different Classes

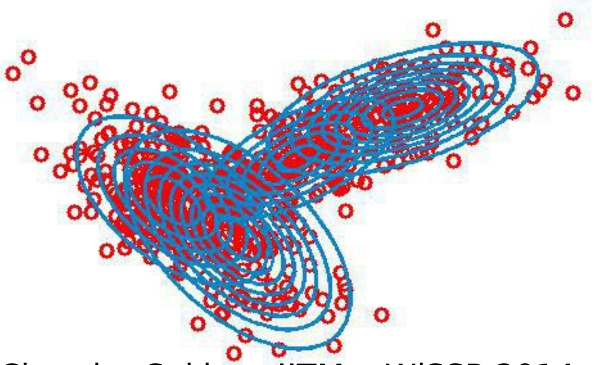
Feature vectors from examples of all classes



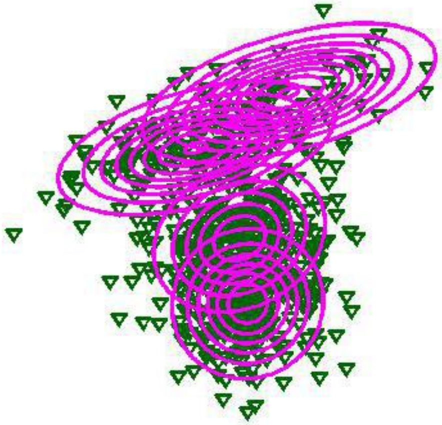
GMM for class 1, λ_1



GMM for class 2, λ_2



GMM for class 3, λ_3



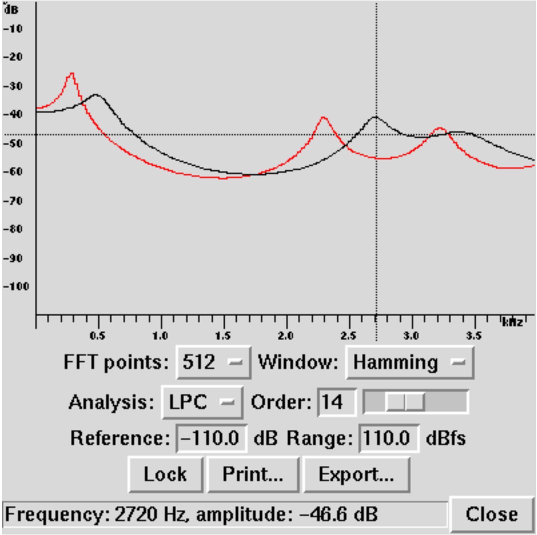
Credit: C. Chandra Sekhar, IITM; WiSSP 2014

Why speech recognition is difficult?

Sources of variabilities

- Speaker specific: physiological, emotional, cultural
- Continuous signal: no well defined boundaries between linguistic units
- Ambience: noise, Lombard effect, room acoustics
- Channel: additive/convolutional noise, compression
- Transducer: omni/uni-directional, carbon/electret mic
- Phonetic context

Spectra of the vowel ‘i’ in word “pin” spoken by male and female speakers



No well defined boundaries between linguistic units

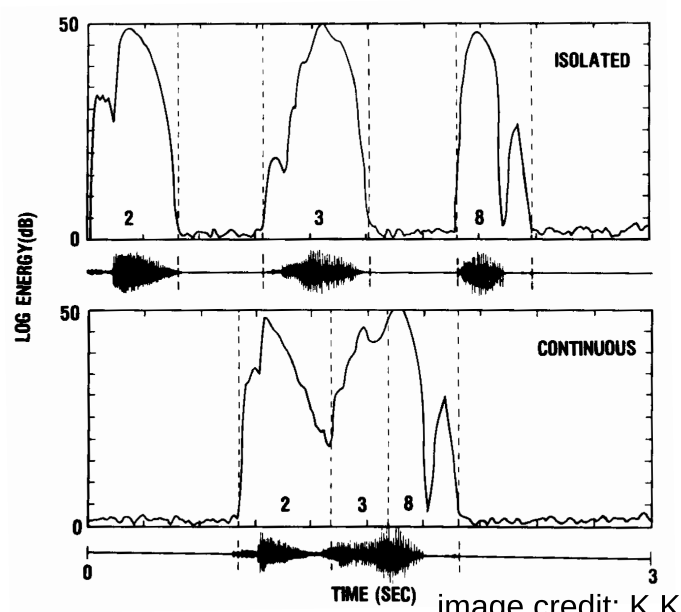


image credit: K.K.Paliwal

Diversity of transduction characteristics of microphones

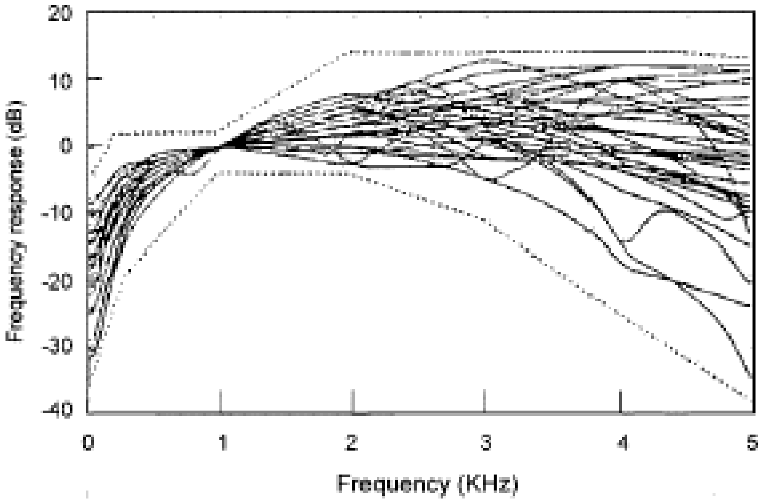
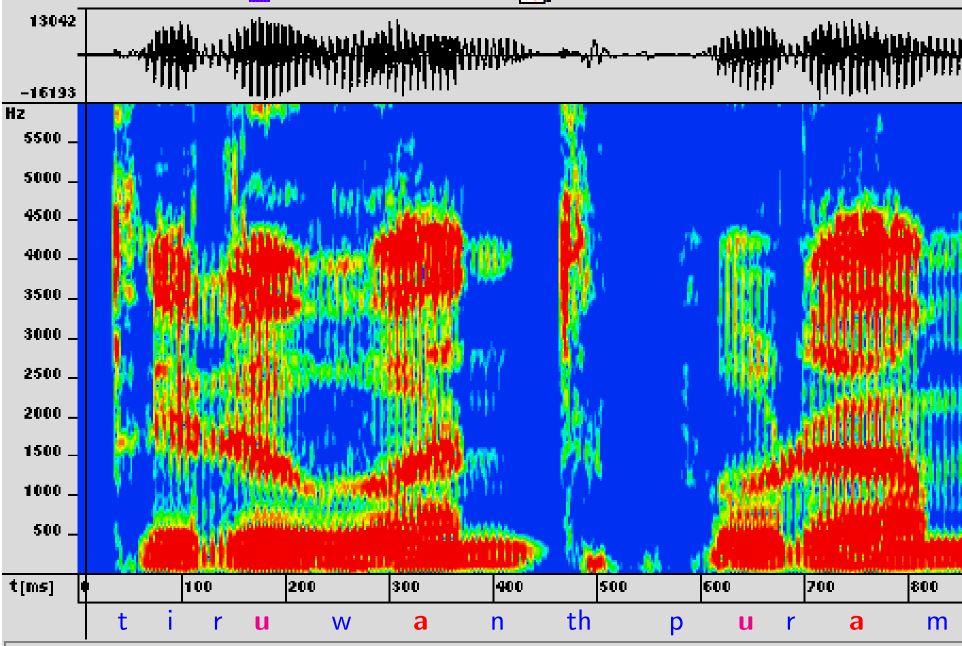


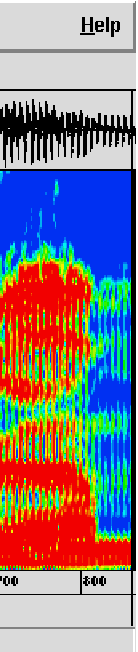
Fig. 6. Diversity of transducer characteristics in telephone set [25].

source: public domain

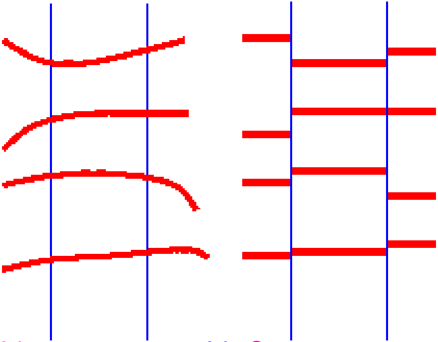


Spectrogram of thiruvananthapuram

Formant trajectories → states

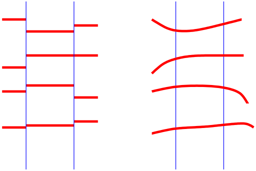


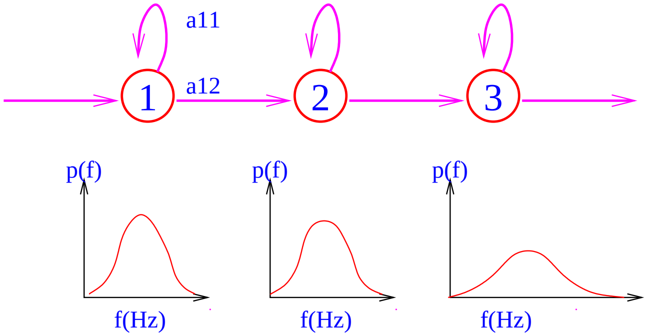
Instead of representing temporal variation of a phoneme as a sequence of feature vectors (deterministic model), represent it as a sequence smaller number of states (probabilistic model: mean and Variance of vectors)



Vector sequence Vs State sequence

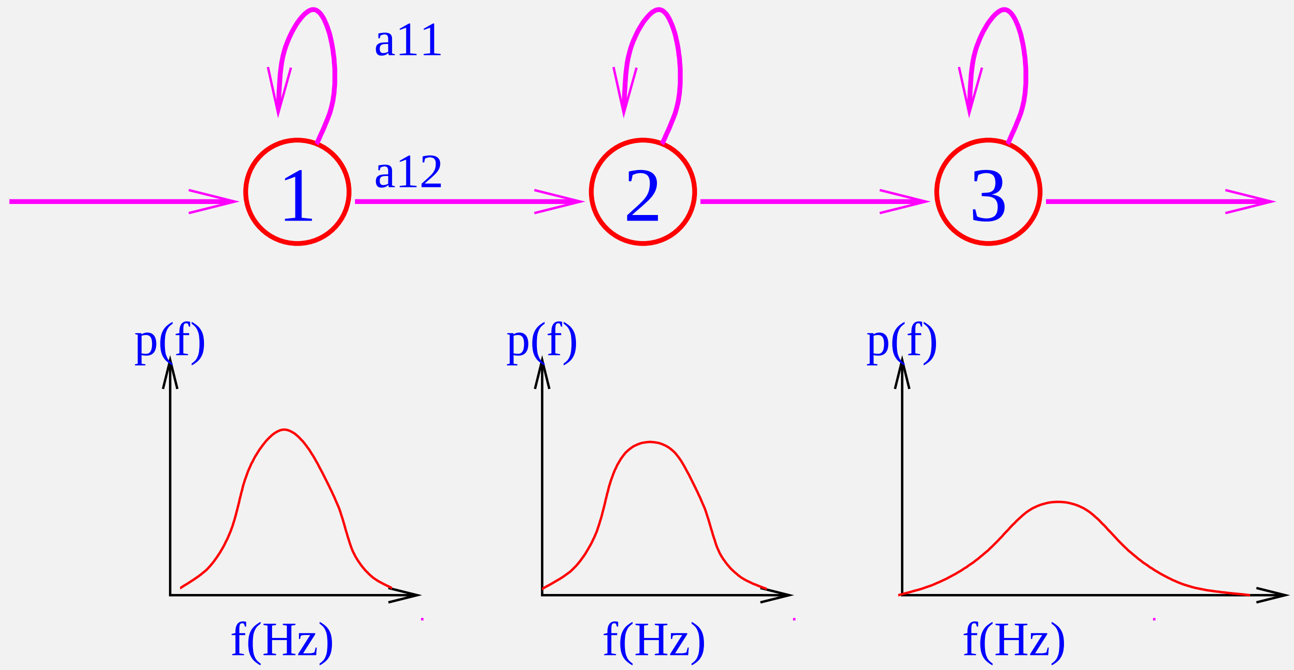
hidden Markov model (HMM)

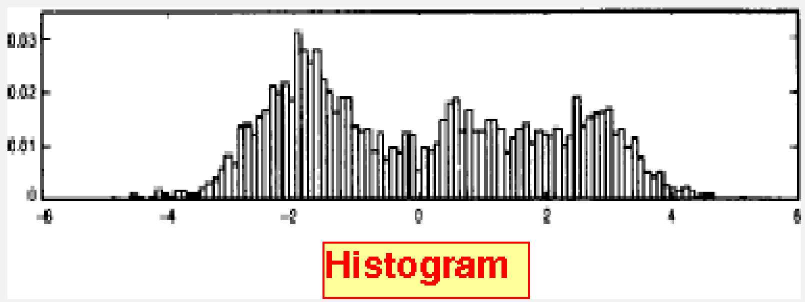




Parameters of a HMM: A, B, π A,B model duration and features of phoneme; π: skipping initial part)

GMM and HMM





GMM-HMM

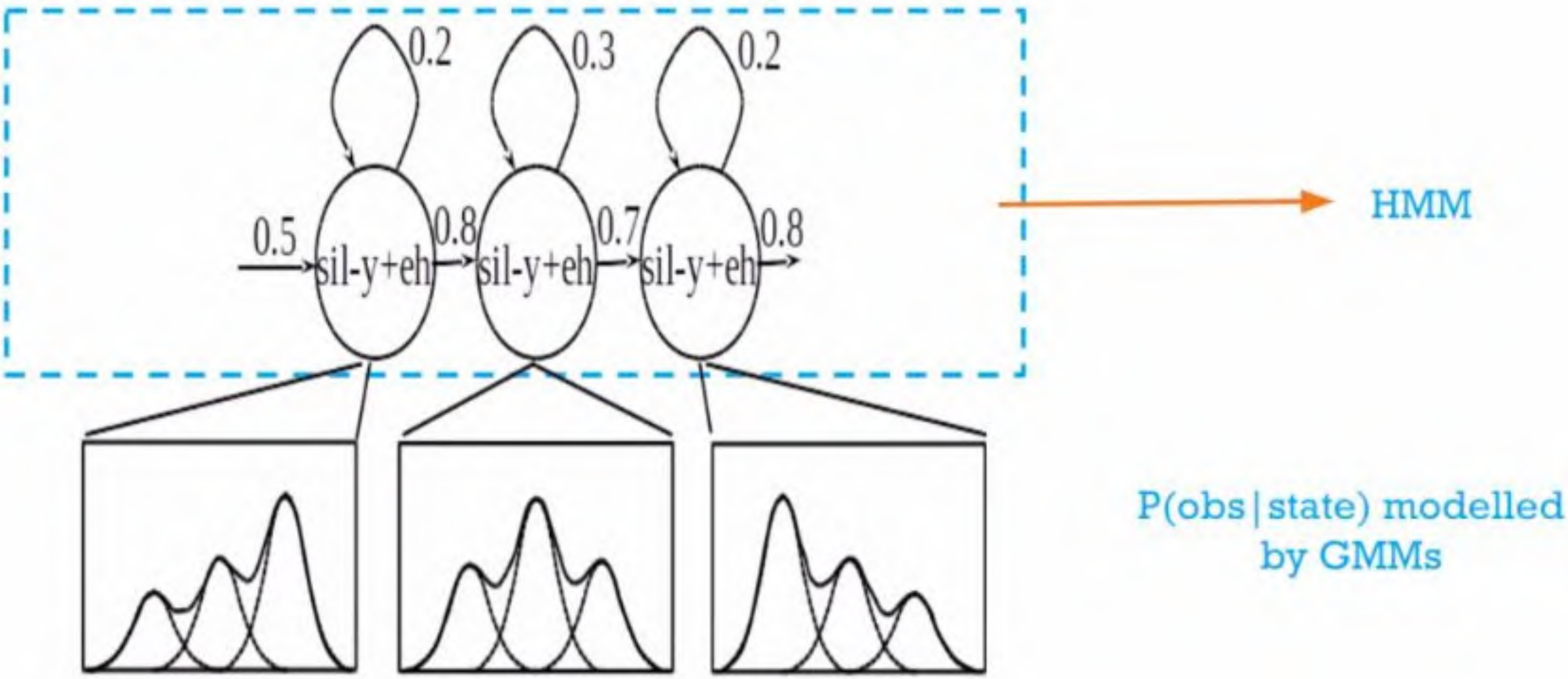


Image Source: John Paul Hosom's Slides

Prof. Umesh's slide

Basic Probability

Joint and Conditional probability (Definitions)

$$p(A, B) = p(A|B)\ p(B) = p(B|A)\ p(A)$$

Bayes' rule

$$p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}$$

Basic Probability

Joint and Conditional probability (Definitions)

$$p(A, B) = p(A|B)\ p(B) = p(B|A)\ p(A)$$

Bayes' rule

$$p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}$$

If A_is are mutually exclusive events,

$p(B)$
$= p(B|A_1)p(A_1) + p(B|A_2)p(A_2) + p(B|A_3)p(A_3) + ...$
$= \sum_i p(B|A_i)\ p(A_i)$

Basic Probability

Joint and Conditional probability (Definitions)

$$p(A,B) = p(A|B)\ p(B) = p(B|A)\ p(A)$$

Bayes' rule

$$p(A|B) = \frac{p(B|A)\ p(A)}{p(B)}$$

If A_is are mutually exclusive events,

$p(B)$
$= p(B|A_1)p(A_1) + p(B|A_2)p(A_2) + p(B|A_3)p(A_3) + \ldots$
$= \sum_i p(B|A_i)\ p(A_i)$

$$p(A|B) = \frac{p(B|A)\ p(A)}{\sum_i p(B|A_i)\ p(A_i)}$$

Chain rule

$$P(A_1, A_2, A_3, \ldots A_n)$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1}) \, P(A_1, A_2, A_3, \ldots A_{n-1})$$

= probability of nth event occuring after the initial n-1 events

X joint probability of the initial n-1 events

samudravijaya@gmail.com

Chain rule

$$P(A_1, A_2, A_3, \ldots A_n)$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1}) \, P(A_1, A_2, A_3, \ldots A_{n-1})$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1})$$

$$P(A_{n-1} \mid A_1, A_2, A_3, \ldots A_{n-2}) \, P(A_1, A_2, A_3, \ldots A_{n-2})$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1}) \ \ldots P(A_2 \mid A_1) \, P(A_1)$$

samudravijaya@gmail.com

Chain rule

$$P(A_1, A_2, A_3, \ldots A_n)$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1})\, P(A_1, A_2, A_3, \ldots A_{n-1})$$

$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1})$$

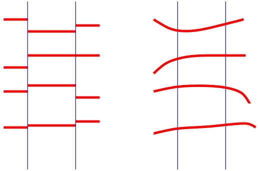
$$P(A_{n-1} \mid A_1, A_2, A_3, \ldots A_{n-2})\, P(A_1, A_2, A_3, \ldots A_{n-2})$$

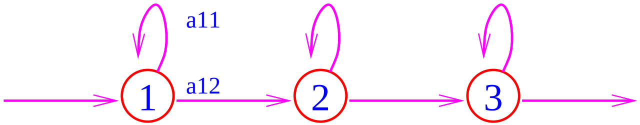
$$= P(A_n \mid A_1, A_2, A_3, \ldots A_{n-1}) \ldots P(A_2 \mid A_1)\, P(A_1)$$

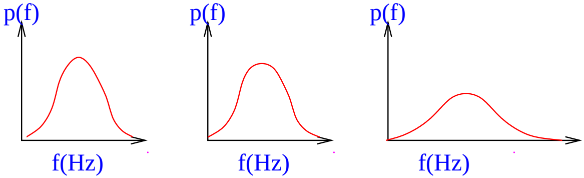
$= P(A_n)\ P(A_{n-1})\ P(A_{n-2}) \ldots P(A_3)\ P(A_2)\ P(A_1)$ if all A_i are independent

samudravijaya@gmail.com

hidden Markov model (HMM)







Parameters of a HMM: A, B, π A,B model duration and features of phoneme; π: skipping initial part)

HIDDEN MARKOV MODEL (HMM)

Markov Model: Useful to model a sequence of states/events
E.g. Rainy, Rainy, Cloudy, Sunny, Sunny, Cloudy

Hidden MM: Find underlying hidden Low or High Pressure from Observation Rainy, Rainy etc.

- Written language — Alphabets (a,x,s, அ, ஆ, ஐ, ற)
- Spoken language — Phonemes (ih, ay, aa)

Example:

Sentence: It's fun to recognize speech

Phonemes: ih t s f ah n t uw r eh k ah g n ay z s p iy ch

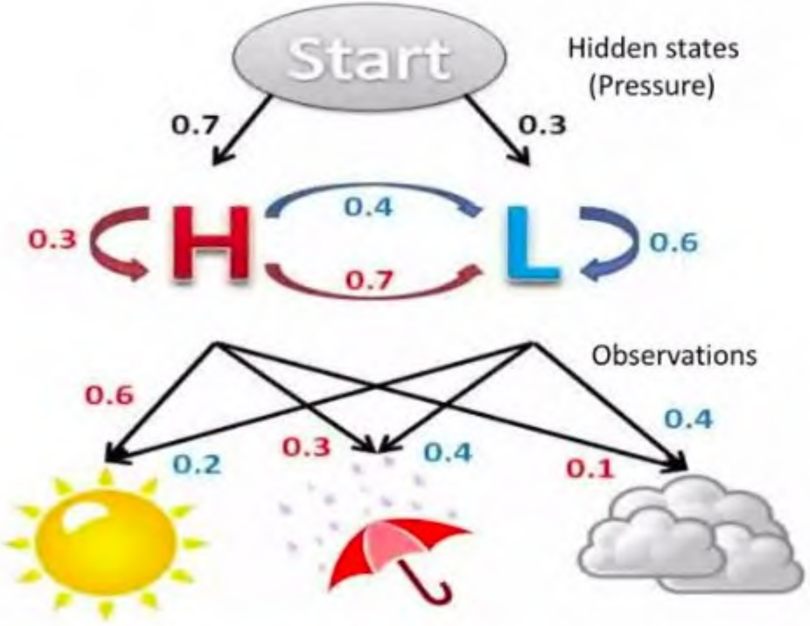


Image Source: http://guizzetti.ca/blogs/lenny/2012/04/predicting-the-weather-with-hidden-markov-models/

Goal: To find the most likely sequence of phonemes for a given observation of speech.

Source: https://sites.google.com/site/jphosomcslu/cs552

Prof. Umesh's slide

Elements of HMM

N: number of hidden states

Q: set of states: $Q = \{q_1, q_2, q_3, ..., q_N\}$

B: observation probability distribution: $B = \{b_j\} \quad 1 \leq j \leq N$

A: state transition probability matrix: $A = \{a_{ij}\}$

$$a_{ij} = P(q_{t+1} = j | q_t = i), \quad 1 \leq i, j, \leq N$$

π: initial state distribution:

$$\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$$

λ: the entire model: $\lambda = (A, B, \pi)$

HMM: assumptions

- **First order Markov assumption** (finite history):

$$P(q_t = j \mid q_{t-1} = i,\ q_{t-2} = k, \ldots) \quad = \quad P(q_t = j \mid q_{t-1} = i)$$

- **Stationarity (parameters do not change with time):**

  a_{ij} does not change with time

  ⇒ the probability of the next phone starting now does not depend on the duration of the current phone.

  ⇒ exponential duration distribution

- **Output independence assumption:**

$$P(o_t \mid q_1, q_2, \ldots q_t, \ldots q_n,\ o_1, o_2, \ldots o_t, \ldots o_n) \quad = \quad P(o_t \mid q_t)$$

3 problems in HMM

1. Matching: Given an observation sequence $O = o_1, o_2, o_3, ..., o_T$, and a trained model $\lambda = (A, B, \pi)$, how to efficiently compute the likelihood, $P(O|\lambda)$ (likelihood of the model λ generating the observation sequence) O?
   Solution: forward algorithm (use recursion for computational efficiency)
   Use: Given two models λ_1 and λ_2, choose λ_1 if $P(O|\lambda_1) > P(O|\lambda_2)$

3 problems in HMM

1. Matching: Given an observation sequence $O = o_1, o_2, o_3, ..., o_T$, and a trained model $\lambda = (A, B, \pi)$, how to efficiently compute the likelihood, $P(O|\lambda)$ (likelihood of the model λ generating the observation sequence) O?
   Solution: forward algorithm (use recursion for computational efficiency)
   Use: Given two models λ_1 and λ_2, choose λ_1 if $P(O|\lambda_1) > P(O|\lambda_2)$

2. Optimal path: Given O and λ, how to find the optimal state sequence $(Q = q_1, q_2, q_3, ..., q_T)$?
   Solution: Viterbi algorithm (similar to DTW)
   Use: Derive word/phone sequence

3 problems in HMM

1. Matching: Given an observation sequence $O = o_1, o_2, o_3, ..., o_T$, and a trained model $\lambda = (A, B, \pi)$, how to efficiently compute the likelihood, $P(O|\lambda)$ (likelihood of the model λ generating the observation sequence) O?
   Solution: forward algorithm (use recursion for computational efficiency)
   Use: Given two models λ_1 and λ_2, choose λ_1 if $P(O|\lambda_1) > P(O|\lambda_2)$

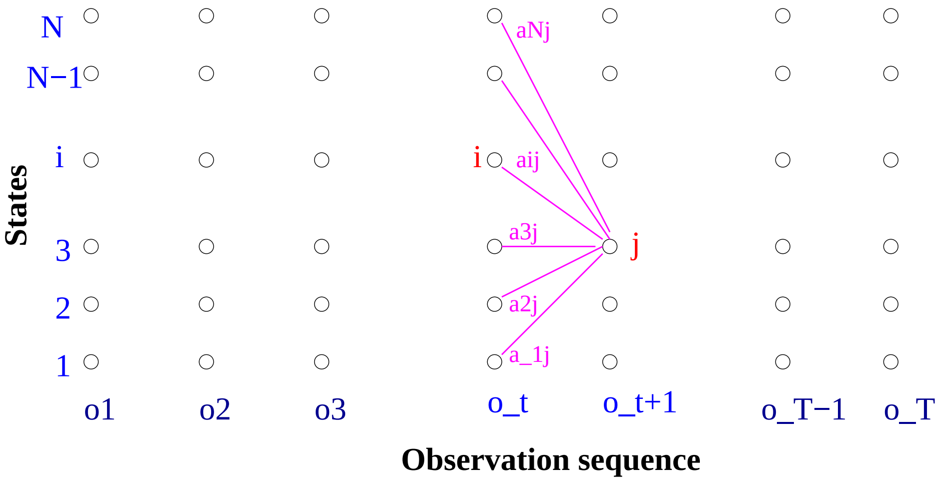
2. Optimal path: Given O and λ, how to find the optimal state sequence ($Q = q_1, q_2, q_3, ..., q_T$)?
   Solution: Viterbi algorithm (similar to DTW)
   Use: Derive word/phone sequence

3. Training: How to estimate the parameters of the model: $\lambda = (A, B, \pi)$ that maximise $P(O|\lambda)$?
   Solution: Forward-backward algorithm.

Youtube videos

Youtube videos on ASR using HMM-GMM

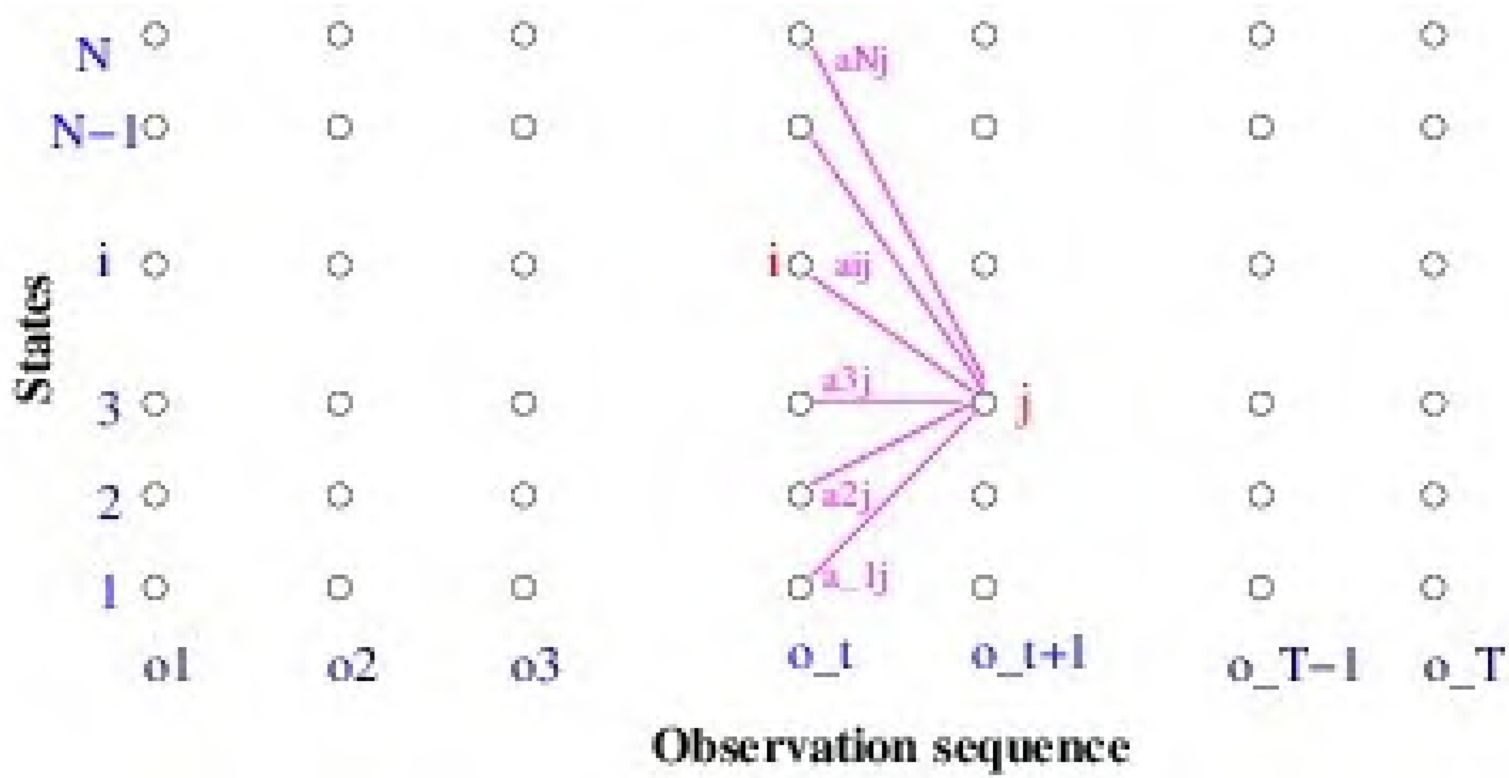
https://www.youtube.com/watch?v=mCtVraO2Xzo


N
N−1
i
3
2
1
States
aNj
aij
a3j
a2j
a_1j
i
j
o1
o2
o3
o_t
o_t+1
o_T−1
o_T
Observation sequence

Match observation (speech vector) sequence with a model

Goal: To compute $P(o_1, o_2, o_3, ..., o_T|\lambda)$

Steps: There are many state sequences (paths). Consider one state sequence $q = q_1, q_2, q_3, ..., q_T$



Match observation (speech vector) sequence with a model

Goal: To compute $P(o_1, o_2, o_3, ..., o_T|\lambda)$

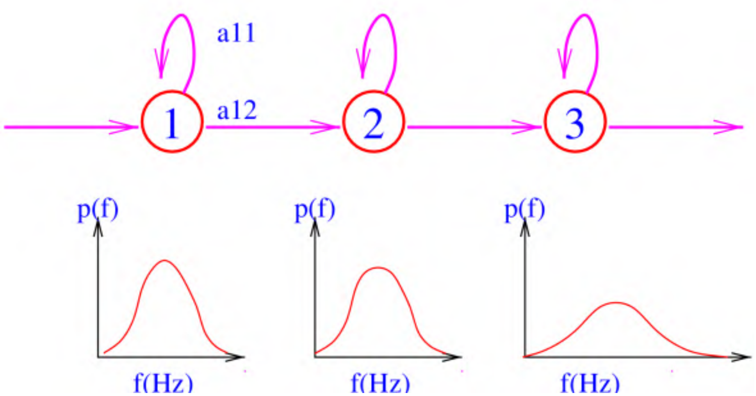
Steps: There are many state sequences (paths). Consider one state sequence $q = q_1, q_2, q_3, ..., q_T$

If we assume that observations are independent,

$$P(O|q,\lambda) = \prod_{i=1}^{T} P(o_t|q_t,\lambda) = b_{q1}(o_1)b_{q2}(o_2)\dots b_{qT}(o_T)$$

Probability of a particular state sequence is:

$$P(q|\lambda) = \pi_{q1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$



Match observation (speech vector) sequence with a model

Goal: To compute $P(o_1, o_2, o_3, ..., o_T | \lambda)$

Steps: There are many state sequences (paths). Consider one state sequence $q = q_1, q_2, q_3, ..., q_T$

If we assume that observations are independent,

$$P(O|q,\lambda) = \prod_{i=1}^{T} P(o_t|q_t,\lambda) = b_{q1}(o_1) b_{q2}(o_2) \dots b_{qT}(o_T)$$

Probability of a particular state sequence is:

$$P(q|\lambda) = \pi_{q1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T}$$

$P(O, q | \lambda) = P(O | q, \lambda) P(q | \lambda)$ because $P(A,B) = P(A|B) P(B)$

Match observation (speech vector) sequence with a model

Goal: To compute $P(o_1, o_2, o_3, ..., o_T|\lambda)$

Steps: There are many state sequences (paths). Consider one state sequence $q = q_1, q_2, q_3, ..., q_T$

If we assume that observations are independent,

$$P(O|q,\lambda) = \prod_{i=1}^{T} P(o_t|q_t,\lambda) = b_{q1}(o_1)b_{q2}(o_2)\dots b_{qT}(o_T)$$

Probability of a particular state sequence is:

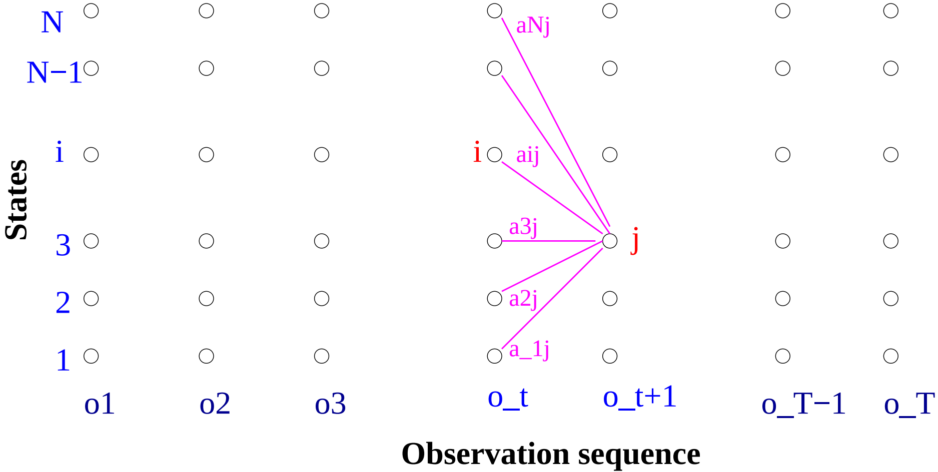
$$P(q|\lambda) = \pi_{q1}a_{q_1q_2}a_{q_2q_3}\dots a_{q_{T-1}q_T}$$

Enumerate paths and sum probabilities:

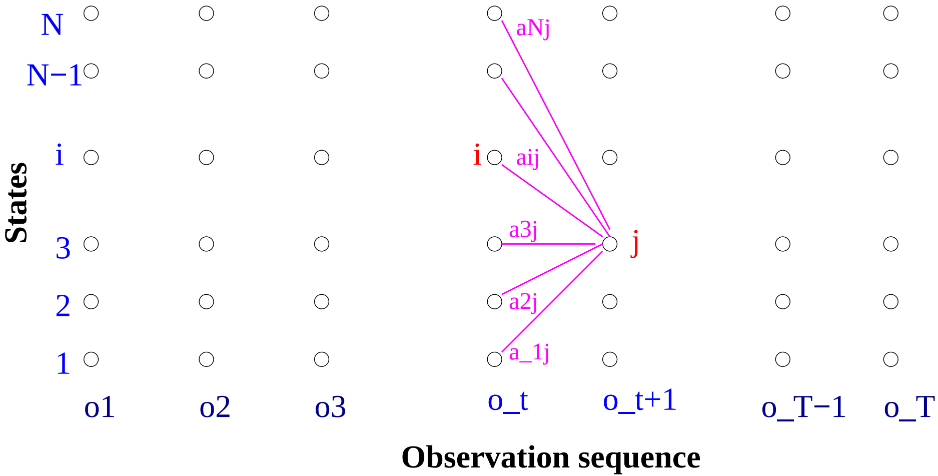
$$P(O|\lambda) = \sum qP(O|q,\lambda)P(q|\lambda)$$

$\Rightarrow N^T$ state sequences and O(T) calculations

$\Rightarrow N^T O(TN^T)$ computational complexity: exponential in length!


States
N
N−1
i
3
2
1
aNj
aij
a3j
a2j
a_1j
i
j
o1
o2
o3
o_t
o_t+1
o_T−1
o_T
Observation sequence

Forward Algorithm: Intution



Let $\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda)$. Then

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1})$$

Forward Algorithm

Define a forward variable $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda)$$

$\alpha_t(i)$ is the probability of observing the partial sequence ($o_1, o_2, \ldots, o_t$) and o_t being generated by i^{th} state (i.e., $q_t = i$).

Induction:

Initialization:

$$\alpha_1(i) = \pi i b_i(o_1)$$

Recursion:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] \;\; b_j(o_{t+1})$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Forward Algorithm

Define a forward variable $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = i|\lambda)$$

$\alpha_t(i)$ is the probability of observing the partial sequence ($o_1, o_2, \ldots, o_t$) and o_t being generated by i^{th} state (i.e., $q_t = i$).

Induction:

Initialization:

$$\alpha_1(i) = \pi i b_i(o_1)$$

Recursion:

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] \;\; b_j(o_{t+1})$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Computational complexity: $O(N^2 T)$

Use: Match a test speech feature vector sequence with all models. Choose λ_i if $P(O|\lambda_i) > P(O|\lambda_j)\forall j$

Viterbi Algorithm: Intution

Problem 2: Given O and λ, how to find the optimal state sequence ($Q = q_1, q_2, q_3, ..., q_T$) (Optimal path)?

Viterbi Algorithm: Intution

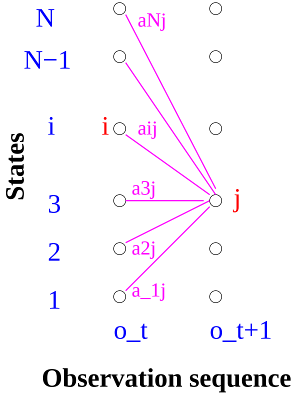
Problem 2: Given O and λ, how to find the optimal state sequence $(Q = q_1, q_2, q_3, ..., q_T)$ (Optimal path)?

Define $\delta_t(i)$ (the highest probability path ending at state i at time t) as:

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \cdots, q_t = i, o_1, o_2, \ldots, o_t | \lambda)$$

Viterbi recursion:

$$\delta_{t+1}(j) = \max_i \ \delta_t(i) a_{ij} b_j(o_{t+1})$$



Viterbi Algorithm: Intution

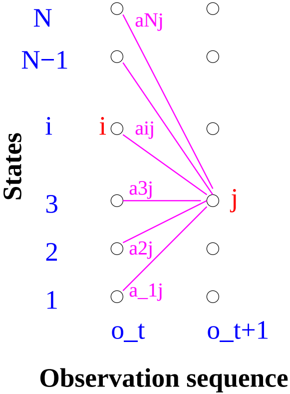
Problem 2: Given O and λ, how to find the optimal state sequence $(Q = q_1, q_2, q_3, ..., q_T)$ (Optimal path)?

Define $\delta_t(i)$ (the highest probability path ending at state i at time t) as:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1, q_2, \cdots, q_t = i, o_1, o_2, \ldots, o_t|\lambda)$$

Viterbi recursion:

$$\delta_{t+1}(j) = \max_i \ \delta_t(i) a_{ij} b_j(o_{t+1})$$



Contrast the above with the recursion in Forward algorithm:

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1})$$

Viterbi Algorithm

Initialization:

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$$

$$\psi_1(i) = 0$$

Recursion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \; b_j(o_t)$$

$$\psi_t(j) = \underset{1 \leq i \leq N}{argmax} [\delta_{t-1}(i) a_{ij}] \quad 2 \leq t \leq T, \; 1 \leq j \leq N$$

Termination:

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)]$$

$$q_T^* = \underset{1 \leq i \leq N}{argmax} [\delta_T(i)]$$

Path (optimal state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \cdots, 2, 1.$$

Training

Problem 3: Given training data and its transcription, how to estimate the parameters of the model, $\lambda = (A, B, \pi)$, that maximises the probability of representation of training data by the model, $P(O|\lambda)$?
There is no analytic solution because of its complexity. So, we employ Expectation-Maximisation (an iterative) algorithm.

Expectation Maximization algorithm to train a HMM

1. Start with an initial (approximate) model, λ_0
2. **E-step**: Using the current model, compute likelihood of the training data: $P(O|\lambda)$.
   In addition, compute the expected number of time instances (probability of)
   the system 'occupying' **i**th state at time **t** (i,e., the **t**th observation is emitted by **i**th state).
3. **M-step**: Reestimate the parameters **A,B,π** following maximum likelihood approach (maximise **P(O|λ')**) where **λ'** is the revised model whose parameters are the reestimated values of **A,B,π**
4. Repeat steps 2 and 3 if

$$P(O|\lambda') \quad > \quad P(O|\lambda) * \Delta$$

5. Stop otherwise.

The algorithm, as applied to training of HMM is known as Baum-Welch algorithm.
It is also known as Forward-Backward algorithm.

Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t: Sum of probabilities of all paths such that partial sequence $o_{t+1}, \ldots, o_T$ are observed

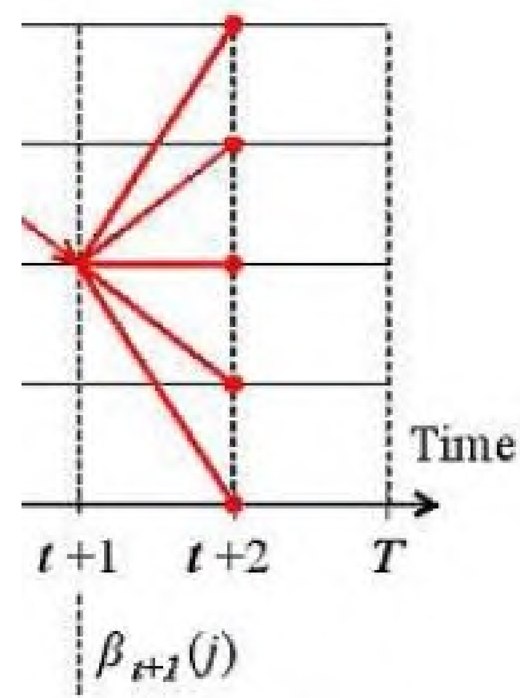
Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \dots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t:
Sum of probabilities of all paths such that
partial sequence $o_{t+1}, \dots, o_T$ are observed

Starting with the initial condition at the last speech vector $(t = T)$:

$$\beta_T(i) = 1.0, \quad 1 \leq i \leq N,$$



samudravijaya@gmail.com

Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t:
Sum of probabilities of all paths such that
partial sequence $o_{t+1}, \ldots, o_T$ are observed

Starting with the initial condition at the last speech vector ($t = T$):

$$\beta_T(i) = 1.0, \quad 1 \leq i \leq N,$$

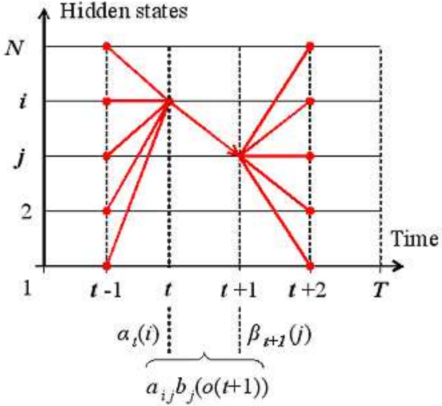
we can recursively compute $\beta_t(i)$ for every state $i = 1, 2, \ldots, N$ backwards in time (t = T-1, T-2, ..., 2, 1) as follows:

$$\beta_t(i) = \underbrace{\sum_{j=1}^{N}[a_{ij} b_j(o_{t+1})]}_{\text{Going to each node from } i^{th} \text{ node}} \quad \underbrace{\beta_j(t+1)}_{\text{Prob. of observation } o_{t+2} \ldots o_T \text{ given now we are in } j^{th} \text{ node at } t+1}$$

Joint event: state i at time t AND state j at t+1

Define $\xi_t(i,j)$ as the probability of system being in state i at time t and in state j at time t+1:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$



Source: http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html

Re-estimation Formulae: $\widehat{\pi}_i$ and $\widehat{a}_{ij}$

The revised estimate of initial probability, π_i, is the expected frequency in state i at time (t=1):

$$\widehat{\pi}_i^{new} = \sum_{j=1}^{N} \xi_1(i, j)$$

Estimating Transition Probability

Trans. Prob. from state i to j $= \dfrac{\text{No. of times transition was made from } i \text{ to } j}{\text{Total number of times we made transition from } i}$

$\xi_t(i,j) \Rightarrow$ prob. of being in "state=i at time=t" and "state=j at time=t+1"

If we average $\xi_t(i,j)$ over all time-instants, we get the number of times the system was in i^{th} state and made a transition to j^{th} state. So, a revised estimation of transition probability is

$$\widehat{a}_{ij}^{new} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T} \Big(\underbrace{\sum_{j=1}^{N} \xi_t(\mathbf{i},\mathbf{j})}_{\text{all transitions out of i at time=t}} \Big)}$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Parameters of State Probability Density Function

Let us assume that the state output distribution function is Gaussian. If there was just one state j, the maximum likelihood estimation of parameters would be

$$\widehat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} o_t$$

$$\widehat{\Sigma}_j = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_j)(o_t - \mu_j)'$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Parameters of State Probability Density Function

Let us assume that the state output distribution function is Gaussian. If there was just one state j, the maximum likelihood estimation of parameters would be

$$\widehat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} o_t$$

$$\widehat{\Sigma}_j = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_j)(o_t - \mu_j)'$$

* Difficulty: Speech HMMs have many states.
* Speech vector $\leftrightarrow$ state mapping is unknown because the state sequence itself is unknown.
* Solution: Assign each speech vector to every state in proportion to the likelihood of system being in that state when the speech vector was observed.

Re-estimation Formulae: $\widehat{b}_j(t)$

Let $L_j(t)$ denote the probability of being in state j at time t.

$$\begin{aligned} L_j(t) &= p(q_t = j|\mathbf{O}, \lambda) \\ &= \frac{p(q_t = j, \mathbf{O}|\lambda)}{p(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(j)}{\sum_i \alpha_T(i)} \end{aligned}$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Let $L_j(t)$ denote the probability of being in state j at time t.

$$\begin{aligned} L_j(t) &= p(q_t = j|\mathbf{O}, \lambda) \\ &= \frac{p(q_t = j, \mathbf{O}|\lambda)}{p(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(j)}{\sum_i \alpha_T(i)} \end{aligned}$$

Revised estimates of the state *pdf* parameters are

$$\widehat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)}$$

$$\widehat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)}$$

The expected values (estimations) are weighted averages, weights being the probability of being in state j at time t.

Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \dots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t:
Sum of probabilities of all paths such that partial sequence $o_{t+1}, \dots, o_T$ are observed

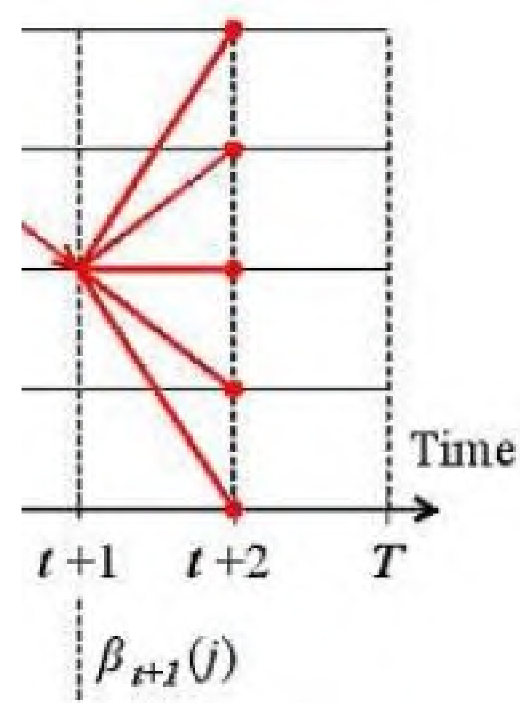
Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t:
Sum of probabilities of all paths such that
partial sequence $o_{t+1}, \ldots, o_T$ are observed

Starting with the initial condition at the last speech vector $(t = T)$:

$$\beta_T(i) = 1.0, \quad 1 \leq i \leq N,$$



samudravijaya@gmail.com

Forward-Backward Algorithm: $\beta_t(i)$

Define a backward variable $\beta_t(i) = p(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$

$\beta_t(i) \Rightarrow$ Given that we are at node i at time t:
Sum of probabilities of all paths such that
partial sequence $o_{t+1}, \ldots, o_T$ are observed

Starting with the initial condition at the last speech vector $(t = T)$:

$$\beta_T(i) = 1.0, \quad 1 \le i \le N,$$

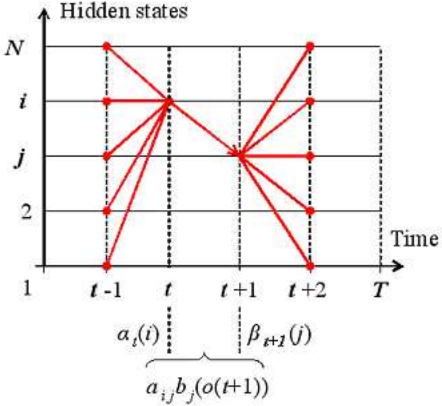
we can recursively compute $\beta_t(i)$ for every state $i = 1, 2, \ldots, N$ backwards in time (t = T-1, T-2, ..., 2, 1) as follows:

$$\beta_t(i) = \underbrace{\sum_{j=1}^{N} [a_{ij} b_j(o_{t+1})]}_{\text{Going to each node from } i^{th} \text{ node}} \quad \underbrace{\beta_j(t+1)}_{\text{Prob. of observation } o_{t+2} \ldots o_T \text{ given now we are in } j^{th} \text{ node at } t+1}$$

Joint event: state i at time t AND state j at t+1

Define $\xi_t(i,j)$ as the probability of system being in state i at time t and in state j at time t+1:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$



Source: http://www.shokhirev.com/nikolai/abc/alg/hmm/hmm.html

Re-estimation Formulae: $\widehat{\pi}_i$ and $\widehat{a}_{ij}$

The revised estimate of initial probability, π_i, is the expected frequency in state i at time (t=1):

$$\widehat{\pi}_i^{new} = \sum_{j=1}^{N} \xi_1(i,j)$$

Estimating Transition Probability

$$\text{Trans. Prob. from state } i \text{ to } j = \frac{\text{No. of times transition was made from } i \text{ to } j}{\text{Total number of times we made transition from } i}$$

$\xi_t(i,j) \Rightarrow$ prob. of being in "state=i at time=t" and "state=j at time=t+1"

If we average $\xi_t(i,j)$ over all time-instants, we get the number of times the system was in i^{th} state and made a transition to j^{th} state. So, a revised estimation of transition probability is

$$\widehat{a}_{ij}^{new} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T} (\underbrace{\sum_{j=1}^{N} \xi_{\mathbf{t}}(\mathbf{i},\mathbf{j})}_{\text{all transitions out of i at time=t}})}$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Parameters of State Probability Density Function

Let us assume that the state output distribution function is Gaussian. If there was just one state j, the maximum likelihood estimation of parameters would be

$$\widehat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} o_t$$

$$\widehat{\Sigma}_j = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_j)(o_t - \mu_j)'$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Parameters of State Probability Density Function

Let us assume that the state output distribution function is Gaussian. If there was just one state j, the maximum likelihood estimation of parameters would be

$$\widehat{\mu}_j = \frac{1}{T}\sum_{t=1}^{T} o_t$$

$$\widehat{\Sigma}_j = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_j)(o_t - \mu_j)'$$

* Difficulty: Speech HMMs have many states.
* Speech vector $\leftrightarrow$ state mapping is unknown because the state sequence itself is unknown.
* Solution: Assign each speech vector to every state in proportion to the likelihood of system being in that state when the speech vector was observed.

Re-estimation Formulae: $\widehat{b}_j(t)$

Let $L_j(t)$ denote the probability of being in state j at time t.

$$\begin{aligned}
L_j(t) &= p(q_t = j | \mathbf{O}, \lambda) \\
&= \frac{p(q_t = j, \mathbf{O} | \lambda)}{p(\mathbf{O}|\lambda)} \\
&= \frac{\alpha_t(i)\beta_t(j)}{\sum_i \alpha_T(i)}
\end{aligned}$$

Re-estimation Formulae: $\widehat{b}_j(t)$

Let $L_j(t)$ denote the probability of being in state j at time t.

$$
\begin{aligned}
L_j(t) &= p(q_t = j|\mathbf{O}, \lambda) \\
&= \frac{p(q_t = j, \mathbf{O}|\lambda)}{p(\mathbf{O}|\lambda)} \\
&= \frac{\alpha_t(i)\beta_t(j)}{\sum_i \alpha_T(i)}
\end{aligned}
$$

Revised estimates of the state *pdf* parameters are

$$
\widehat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)}
$$

$$
\widehat{\Sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)}
$$

The expected values (estimations) are weighted averages, weights being the probability of being in state j at time t.

Expectation Maximization algorithm to train a HMM

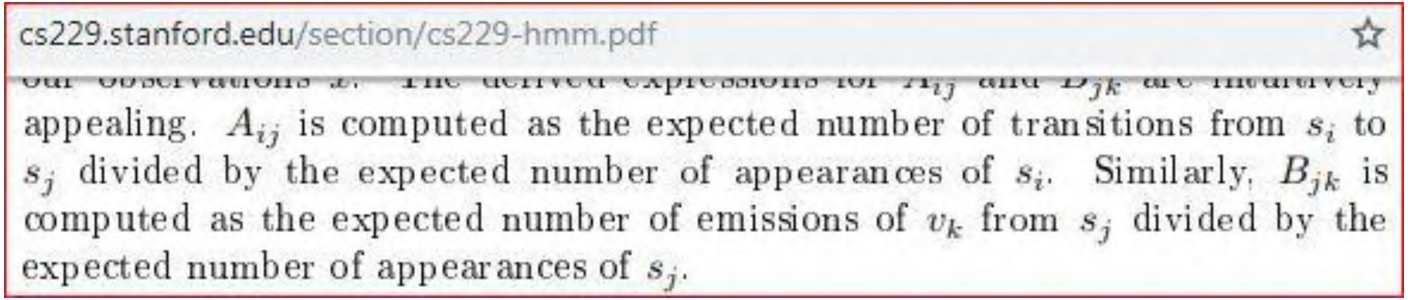
- **E-step**:

  Run the forward and the backward algorithms to compute $\alpha_t(i)$ and $\beta_t(j)$

  Then compute the expected number of transitions

  from **i**th state to **j**th state as $\Sigma_t \quad \xi_t(i, j)$

- **M-step**: Reestimate the parameters A,B,π following maximum likelihood approach


cs229.stanford.edu/section/cs229-hmm.pdf

our observations X. The derived expressions for A_{ij} and B_{jk} are intuitively appealing. A_{ij} is computed as the expected number of transitions from s_i to s_j divided by the expected number of appearances of s_i. Similarly, B_{jk} is computed as the expected number of emissions of v_k from s_j divided by the expected number of appearances of s_j.

Some remarks

Types of HMM
* Ergodic Vs left-to-right
* Semi-Markov (state duration)
* Discriminative models

Implementational Issues
* Number of states
* Initial parameters
* Scaling, addition of logLikelihoods
* Multiple observations (tokens/repetitions)
* Discrete Vs Continuous probability functions (with GMMs)
* Concatenation of smaller HMMs $\rightarrow$ larger HMM

SphinxTrain Training sub-word HMMs

Stages of training (Reference: http://www.speech.cs.cmu.edu/sphinxman/fr4.html):

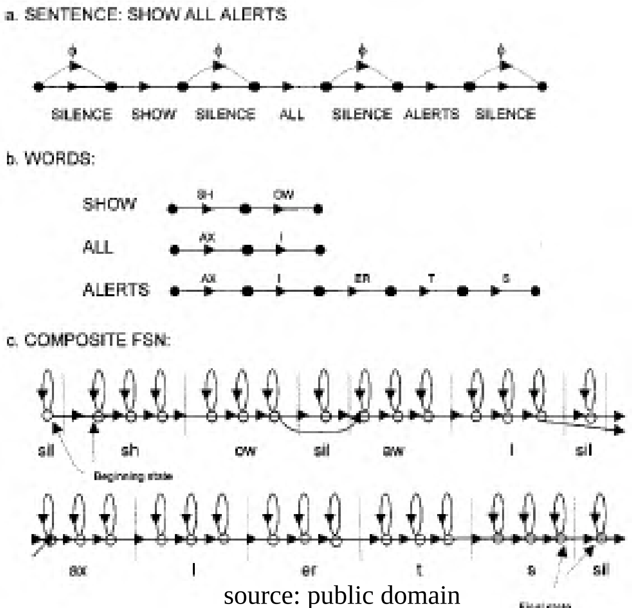
1. Training context Independent phone HMMs
2. Training context Dependent phone HMMs
3. Decision tree building
4. Training context Dependent tied phone HMMs
5. Recursive Gaussian splitting

Training Context Independent phone HMMs

2 steps: Initialization and Embedding re-estimation.

Inputs:
* Feature vector sequences
* Word-level transcriptions
* Pronunciation dictionary

Sentence HMM is composed of Phone HMMs



source: public domain

Training subword HMMs

An iterative algorithm (Baum-Welch, also known as Forward-Backward) is used. The Maximum Likelihood approach guarantees increase of the likelihood of the trained model matching with training data with each iteration. To begin with, an initial estimation of parameters of HMMs (A, B, π) is required.

Q: How to get an initial estimation of $(\lambda = \{A, B, \pi\}$?

A: We can estimate parameters if we know the boundaries of every subword HMM in training utterances.

Training subword HMMs

An iterative algorithm (Baum-Welch, also known as Forward-Backward) is used. The Maximum Likelihood approach guarantees increase of the likelihood of the trained model matching with training data with each iteration. To begin with, an initial estimation of parameters of HMMs (A, B, π) is required.

Q: How to get an initial estimation of $(\lambda = \{A, B, \pi\}$?

A: We can estimate parameters if we know the boundaries of every subword HMM in training utterances.

Practical solution: Assume that the durations of all units (phones) are equal. If there are N phones in a training utterance, divide the feature vector sequence into N equal parts. Assign each part, to a phoneme in the phoneme sequence corresponding to the transcription of the utterance. Repeat for all training utterances.

Initial estimation of HMM parameters: an illustration

Let the transcription of the 1st wave file be the following sequence of words: mera bhaarat mahaan

Let the relevant lines in the dictionary be as follows:

bhaarata	bh aa r a t
mahaana	m a h aa n
mera	m e r aa

The phonemeHMM sequence (of length 16) corresponding to this sentence is sil m e r aa bh aa r a t m a h aa n sil

Initial estimation of HMM parameters: an illustration

Let the transcription of the 1st wave file be the following sequence of words: mera bhaarat mahaan

Let the relevant lines in the dictionary be as follows:

bhaarata bh aa r a t
mahaana m a h aa n
mera m e r aa

The phonemeHMM sequence (of length 16) corresponding to this sentence is sil m e r aa bh aa r a t m a h aa n sil

If the duration of the wavefile is 1.0sec, there will 98 feature vectors (frame shift = 10msec and frame size = 25msec).

Assign the first 6 feature vectors to "sil" HMM; the next 6 (7 through 12) to "m"; the next 6 (13 through 18) to "e"; ... ; the last 8 feature vectors to "sil". If HMM has 3 states, assign 2 feature vector to each state; compute mean,SD.
Assume $a_{i,j}$=0.5 if j=i or j=i+1; else assign 0.

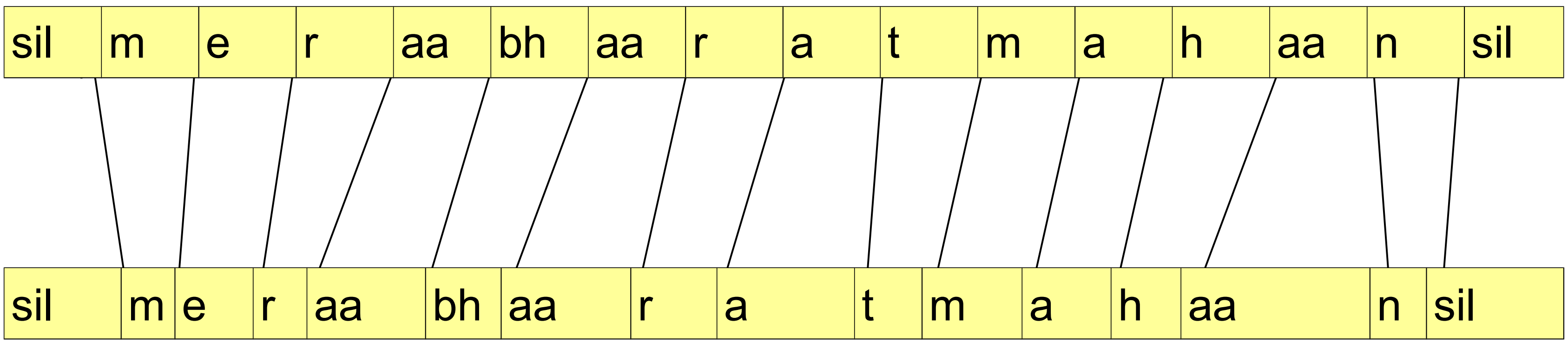
Better estimation of HMM parameters

Initial assumption: all phonemes have equal duration

==> boundaries between phonemes are equidistant

100 vectors

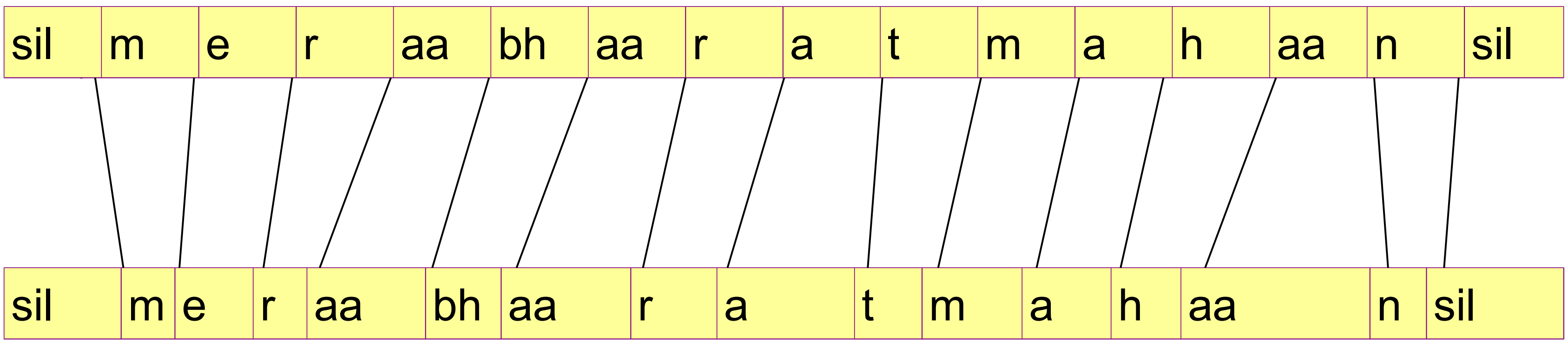
16 phones



Adjust the boundaries for better estimation of HMM parameters.

Re-estimation of HMM parameters

Adjust the boundaries for better estimation of HMM parameters.



Search for those set of phoneme boundaries such that

the HMM parameters estimated by the revised boundaries

represent the training data better.

Search for the set of phoneme/state boundaries such that the likelihood of the training data given the current model is the highest.

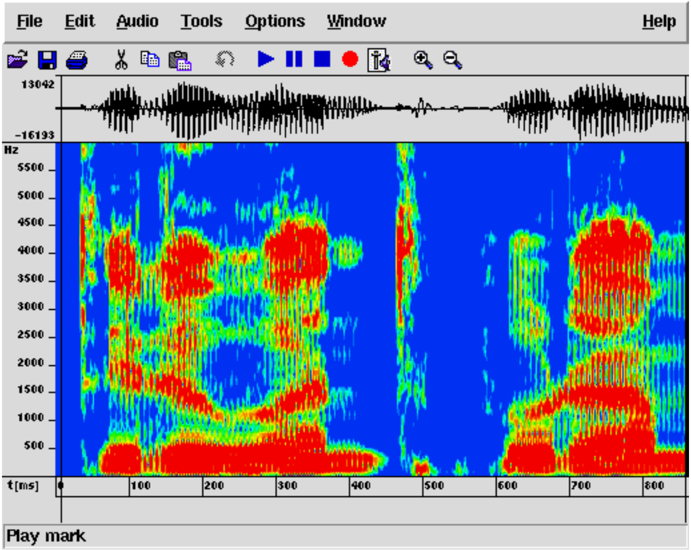
Embedded Re-estimation

(II) Embedding re-estimation:

1. For each utterance, do the following:
   - Using the phone-level transcriptions, compose a sentence HMM out of phone HMMs.
   - Forward-Backward algorithm: compute the likelihood of each feature vector being generated by each state of each phone HMM in the sentence HMM
   - Accumulate likelihoods of feature vectors being generated by each state.
2. For each state: re-estimate HMM parameters using the accumulated likelihoods.

Repeat the Embedded Re-estimation a few times.

Speech: a dynamic signal



Formant: frequency of resonance: F1, F2, F3, ...

Slope and curvature of trajectory

Training Context Dependent phone HMMs

1. Initialise N^3 triphone models, where N is the number of phones.
2. Compose sentence HMM out of triphone (CD) models instead of monophone (CI) models.
3. Carry out the Embedded Re-estimation for a few iterations.

The sequence of CI HMMs was
sil m e r aa bh aa r a t m a h aa n sil
The sequence of CD HMMs (triphones) is
sil   sil-m+e   m-e+r   e-r+aa   r-aa+bh ...

Training Context Dependent phone HMMs

1. Initialise N^3 triphone models, where N is the number of phones.
2. Compose sentence HMM out of triphone (CD) models instead of monophone (CI) models.
3. Carry out the Embedded Re-estimation for a few iterations.

The sequence of CI HMMs was
sil m e r aa bh aa r a t m a h aa n sil
The sequence of CD HMMs (triphones) is
sil   sil-m+e   m-e+r   e-r+aa   r-aa+bh ...

If $N = 50$, each HMM has 3 states, there may be upto 375,000 states. Each state is associated with one Gaussian. Huge amount of speech data is needed for robust estimation of the parameters $(\mu, \mathbf{\Sigma})$ of 375,000 Gaussians!

Training Context Dependent phone HMMs

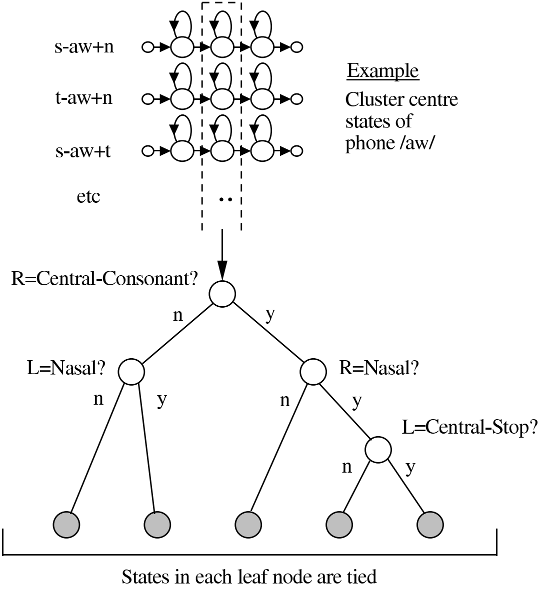
1. Initialise N^3 triphone models, where N is the number of phones.
2. Compose sentence HMM out of triphone (CD) models instead of monophone (CI) models.
3. Carry out the Embedded Re-estimation for a few iterations.

The sequence of CI HMMs was
sil m e r aa bh aa r a t m a h aa n sil
The sequence of CD HMMs (triphones) is
sil sil-m+e m-e+r e-r+aa r-aa+bh ...

If $N = 50$, each HMM has 3 states, there may be upto 375,000 states. Each state is associated with one Gaussian. Huge amount of speech data is needed for robust estimation of the parameters $(\mu, \mathbf{\Sigma})$ of 375,000 Gaussians!

Reduce the number of states by state-tying; use Decision Trees.

source: HTK Book (S.Young et al.)



Decision trees are used to decide which of the HMM states of all the triphones (seen and unseen) are similar to each other, so that data from all these states are collected together and used to train one global state, which is called a senone (also called a tied state). Example: Left states of 1st and 3rd triphones above would be similar

Training Context Dependent tied phone HMMs

1. Prune the Decision trees so that the number of senones (tied states) is commensurate with the amount of training data.

Training Context Dependent tied phone HMMs

1. Prune the Decision trees so that the number of senones (tied states) is commensurate with the amount of training data.
2. Create CD tied model definition file that has (a) all triphones which are seen during training, and (b) has the states corresponding to these triphones identified with senones from the pruned trees (state-senone mapping).
3. Carry out the Embedded Re-estimation (tied CD models) for a few iterations.

Training Context Dependent tied phone HMMs

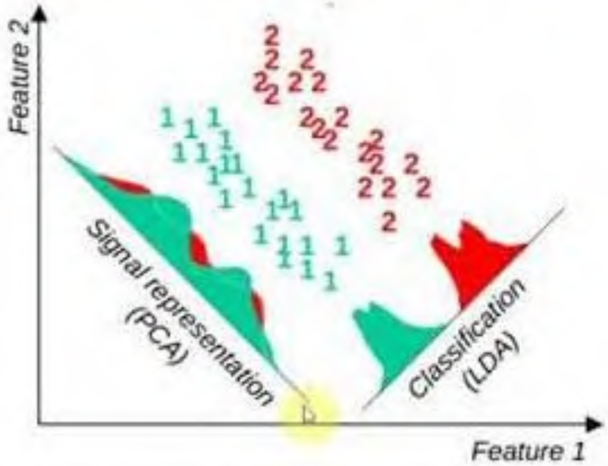
1. Prune the Decision trees so that the number of senones (tied states) is commensurate with the amount of training data.
2. Create CD tied model definition file that has (a) all triphones which are seen during training, and (b) has the states corresponding to these triphones identified with senones from the pruned trees (state-senone mapping).
3. Carry out the Embedded Re-estimation (tied CD models) for a few iterations.
4. Generate Gaussian mixtures for each senone (tied state) and re-train. Repeat this step till the desired number (say 8) of mixtures are created for each GMM (senone).

GMM-HMM based ASR using Kaldi toolki

Model name		Characteristics
➢ **Mono**	CI HMMs	13 static, 13delta-, 13delta-delta MFCCs
➢ **Tri1**	CD HMMs	-do-

Linear Discriminant Analysis

- What is the difference between LDA & PCA?



http://stackoverflow.com/questions/33576963/dimensions-reduction-in-matlab-using-pca

52

Created by : Gopal Prasad Malakar

Speech Recognition
Lecture 12: Acoustic Model Adaptation

Eugene Weinstein
Google, NYU Courant Institute
eugenew@cs.nyu.edu

MLLR

- Maximum likelihood linear regression: linear transformation of Gaussian mean vectors and/or covariances to produce speaker-adaptive model.

$$\mu_{\mathrm{MLLR}} = \mathbf{A}\mu_0 + b \qquad \Sigma_{\mathrm{MLLR}} = \mathbf{H}\Sigma_0\mathbf{H}^\top$$

- Constrained MLLR (CMLLR): $\mathbf{H} = \mathbf{A}$
- Equivalent to transforming the features (with a scaling factor of $|\mathbf{A}|$ when calculating Gaussian likelihoods): $x_i^{\mathrm{CMLLR}} = \mathbf{A}^{-1}x_i + \mathbf{A}^{-1}b$
- Transformation parameters estimated with EM to maximize adaptation data likelihood.

Speaker-Adaptive Training (SAT)

- Baseline MLLR approach: first train speaker-independent models, then adapt model parameters with MLLR using adaptation data.
- SAT idea: learn parameters of the models with MLLR transforms in place.
- With CMLLR, this amounts to transforming the source data (easier to implement).
  - Also allows for MLLR to be used in conjunction with discriminative training (MMI).
- MLLR can be combined with VTLN.

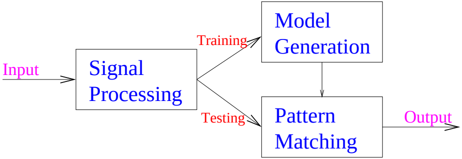
GMM-HMM based ASR using Kaldi toolki

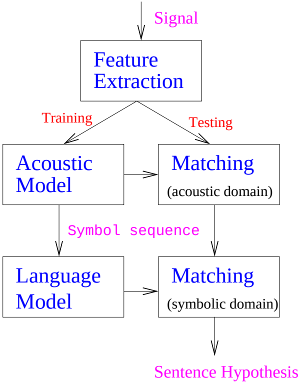
	Model name		Characteristics
➢	**Mono**	CI HMMs	13 static, 13delta-, 13delta-delta MFCCs
➢	**Tri1**	CD HMMs	-do-
➢	**Tri2**	-do-	13x9 MFCCs → **LDA** (40) → **MLLR**
➢	**Tri3**	-do-	**SAT**

A peek into the future

Assumptions/constraints on acoustic model

Model	frames are independent	pronunciation dictionary	DT to reduce senones	Markov assumption	ML training	data is split among mixtures
HMM-GMM	X	X	X	X	X	X
HMM-DNN	X	X	X	X		
RNN-CTC	X					
Encoder-decoder						


Input
Signal Processing
Training
Testing
Model Generation
Pattern Matching
Output


Signal
Feature Extraction
Training
Testing
Acoustic Model
Matching
(acoustic domain)
Symbol sequence
Language Model
Matching
(symbolic domain)
Sentence Hypothesis

Knowledge sources

Phone sequence/phone hypothesis lattice

==> Sentence hypothesis

Knowledge sources

Phone sequence/phone hypothesis lattice

==> Sentence hypothesis

Lexicon

man

mna

Syntax

Some man brought the apple.

Apple the brought man some.

Knowledge sources

Phone sequence/phone hypothesis lattice

$==>$ Sentence hypothesis

Lexicon

man

mna

Syntax

Some man brought the apple.

Apple the brought man some.

Semantics

Time flies like an arrow

Fruit flies like banana

Pragmatics

Turn left for the nearest chemist

Combining Acoustic and Language Models

Let Y : Acoustic feature sequence
W : Word sequence

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{argmax} \quad P(\mathbf{W}|\mathbf{Y})$$

Combining Acoustic and Language Models

Let Y : Acoustic feature sequence
W : Word sequence

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{argmax} \quad P(\mathbf{W}|\mathbf{Y})$$

Bayes' rule:

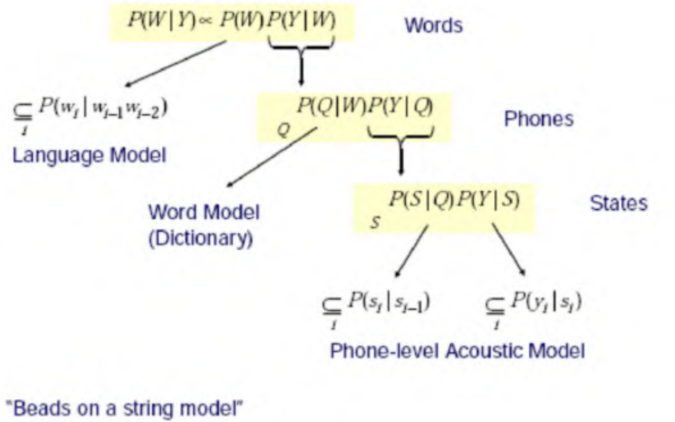
$$P(\mathbf{W}|\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{Y})}$$

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{argmax} \quad \frac{P(\mathbf{Y}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{Y})}$$

CSR: Acoustic model, Language model and Hypothesis search

Hierarchy of Units

$$\widehat{\mathbf{W}} = \underset{\mathbf{W}}{argmax} \quad \frac{P(\mathbf{Y}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{Y})}$$



Source: "State of the Art in ASR (and beyond)", Steve Young

Pronunciation dictionary

* Representing a word as a sequence of units of recognition
* Pronunciation rules can be used
* Manual verification is necessary

```
aage     aa vbg g e
aaja     aa vbj j
aba      a vbb b
abbAsa   a vbbb b aa s
abhI     a vbb bh ii
```

Pronunciation dictionary

* Representing a word as a sequence of units of recognition
* Pronunciation rules can be used
* Manual verification is necessary

```
aage     aa vbg g e
aaja     aa vbj j
aba      a vbb b
abbAsa   a vbbb b aa s
abhI     a vbb bh ii
```

Multiple pronunciations

```
vij~nAna         v i vbj j n aa n
vij~nAna         v i vbg g y aa n
```

Examples of pronunciation variability

Feature spreading in coalescence:

c ae n t −> c ae t where ae is nasalised

Assimilation causing changes in place of articulation:

n −> m before labial stop as in input, can be, grampa

Asynchronous articulation errors causing stop insertions:

warm[p]th, ten[t]th, on[t]ce, leng[k]th

Fast speech:

probably −−> probly

---

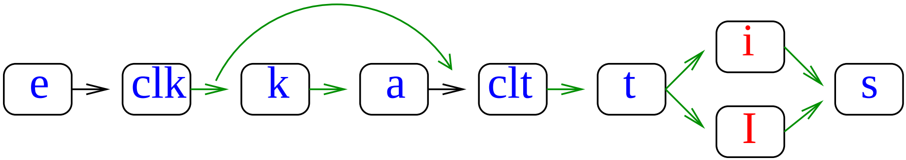
r-insertion in vowel-vowel transitions:

stir [r]up, director [r]of

Context dependent deletion:

nex[t] week

Representation of a word as a phone net



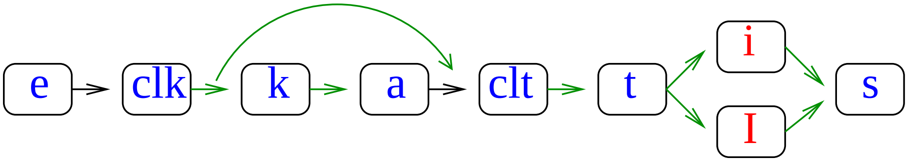
e clk k a clt t I s

e clk k a clt t i s

e clk clt t I s

e clk clt t i s

Representation of a word as a phone net



e clk k a clt t I s

e clk k a clt t i s

e clk clt t I s

e clk clt t i s

* "probabilities" of pronunciations can be estimated
* many pronunciations → higher word confusions
→ performance degradation

Generation of word hypotheses

Generation of word hypotheses can result in

* a single sequence of words,
* in a collection of the n-best word sequences,
* in a lattice of partially overlapping word hypotheses.

Generation of word hypotheses

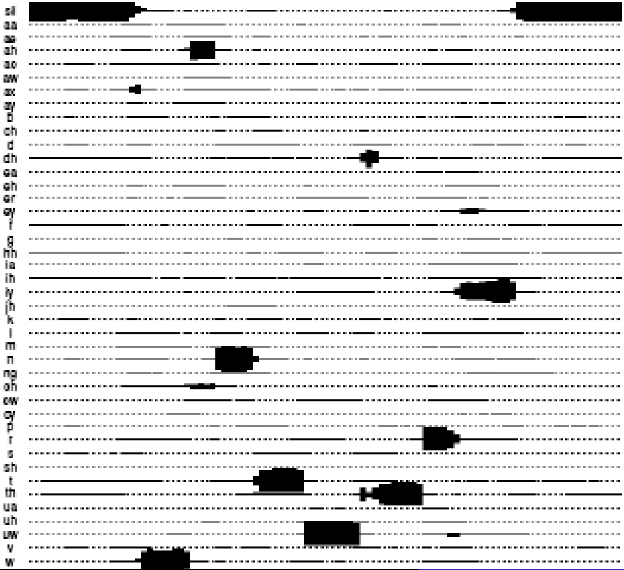
Generation of word hypotheses can result in

* a single sequence of words,
* in a collection of the n-best word sequences,
* in a lattice of partially overlapping word hypotheses.

Goal: Find the path with the least cost (most likely word sequence)

Acoustic evidence $\rightarrow$ Word lattice $--\ >$ DAG

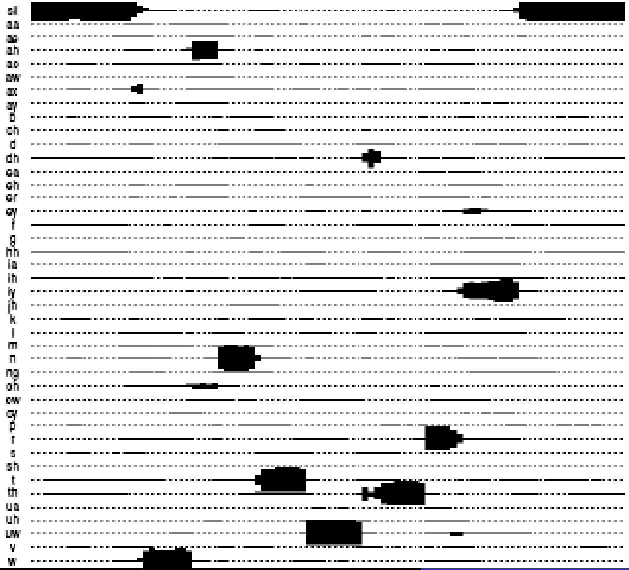
Probabilities of phones at various time instants

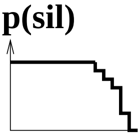


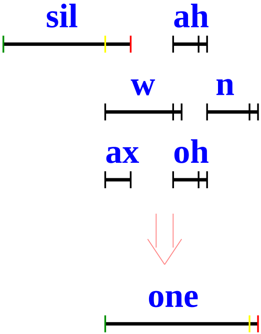
source: public domain

Probabilities of phones at various time instants

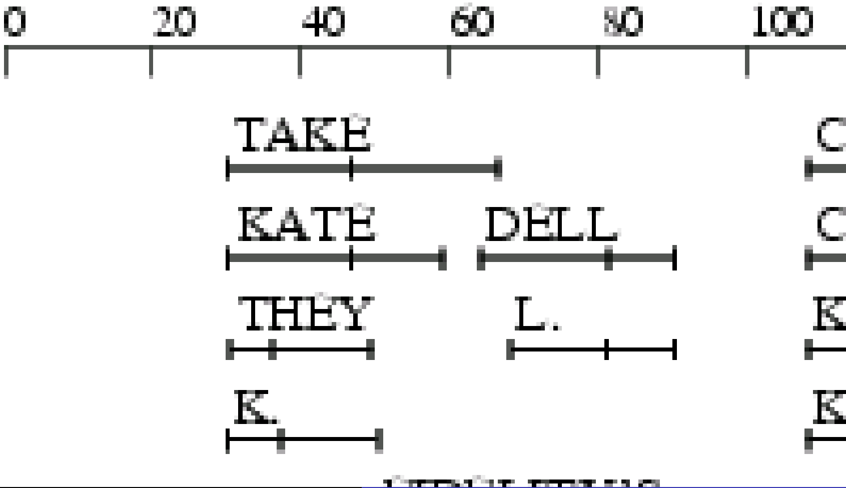
source: public domain



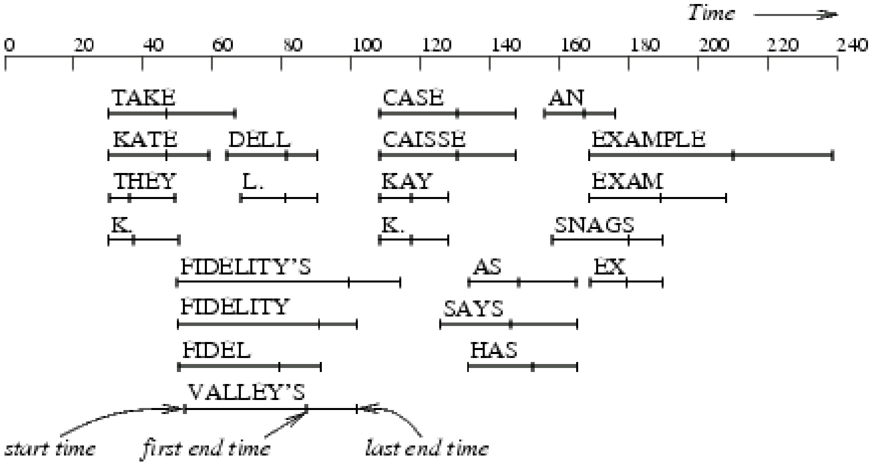




Lattice of phone hypotheses → lattice of word hypotheses



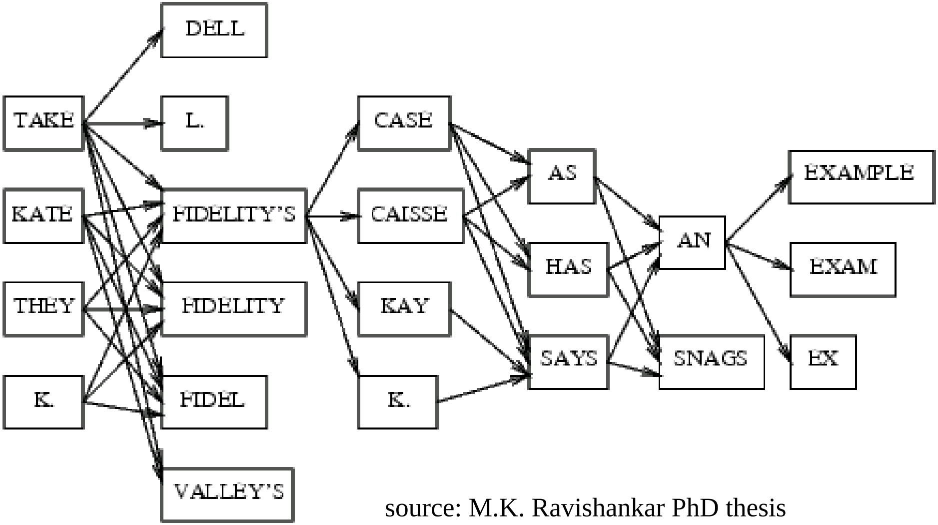
Word hypotheses at various time instants



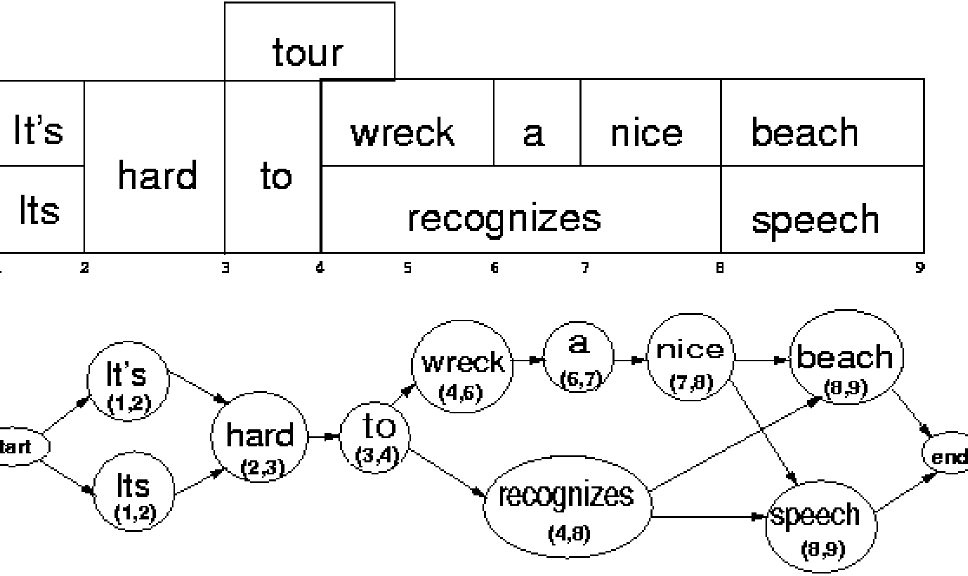
Take Fidelity's case as an example

Source: "Efficient algorithms for Speech Recognition", M.K.Ravishankar, PhD thesis: CMU-CS-96-143

Word Lattice as a Directed Acyclic Graph



source: M.K. Ravishankar PhD thesis



Source: http://www.cs.washington.edu/research/jair/volume5/helzerman96a-html/node7.html

Search for most likely utterance

Goal: Find the path with the least cost
=== most likely word sequence

Associate cost with each edge of DAG

cost = - (acoustic evidence + language evidence)

Given a graph with N nodes and E edges, the least-cost path can be found in time proportional to N+E

Context Free Grammar

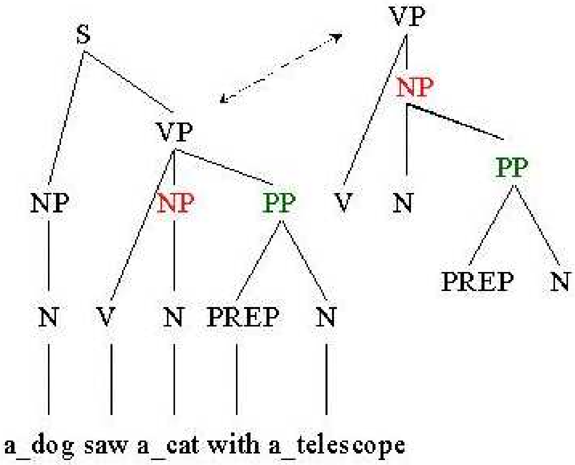
A commonly used mathematical structure for modeling constituent structure of natural languages.

* Rules : S $-->$ NP + VP
* Terminal symbols : vocabulary (words of the language)
* Non-terminal symbols : NP

Sentence Generator or parser

An example

- #1 S → NP VP
- #2 VP → V NP PP
- #3 VP → V NP
- #4 NP → N
- #5 NP → N PP
- #6 PP → PREP N
- #7 N → a_dog
- #8 N → a_cat
- #9 N → a_telescope
- #10 V → saw
- #11 PREP → with



11/22/00 JHU CS 600.465/ Intro to NLP/Jan Hajic 3

Stochastic CFG assigns probs to various transitions

Backus-Naur Form

BNF grammar is useful for ASR in a specific task domain.

An example

[क्या] Trainname (का | मे) [Digit] (रिज़र्वेशन | Class का टिकट) Aaj के लिए Milegaa [क्या]?;

Probability of a word sequence

Let **W** denote the word sequence $w_1, w_2, \cdots, w_i$.

$$p(\mathbf{W}) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \cdots \times p(w_i \mid w_{i-1}, w_{i-2}, \cdots, w_1)$$

Not practical due to 'unlimited history':
too many parameters for even a short **W**

Probability of a word sequence

Let $\mathbf{W}$ denote the word sequence $w_1, w_2, \cdots, w_i$.

$$p(\mathbf{W}) = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \cdots \times p(w_i \mid w_{i-1}, w_{i-2}, \cdots, w_1)$$

Not practical due to 'unlimited history':
too many parameters for even a short $\mathbf{W}$

Markovian assumption:

- Disregard 'too old' history
- k^{th} order Markovian approximation: remember only 'k' previous words
- Assume stationarity

Parameter Estimation

Maximum Likelihood Estimation: relative frequencies
Use counts from training data.

unigram:

$$p(w) = C(w)/|V|$$

Parameter Estimation

Maximum Likelihood Estimation: relative frequencies
Use counts from training data.

unigram:

$$p(w) = C(w)/|V|$$

bigram:

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{\sum_w C(w_{n-1} w_n)}$$

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

Parameter Estimation

Maximum Likelihood Estimation: relative frequencies
Use counts from training data.

unigram:

$$p(w) = C(w)/|V|$$

bigram:

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{\sum_w C(w_{n-1} w_n)}$$

$$p(w_n|w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}$$

n-gram:

$$p(w_n|w_1 w_2 \cdots w_{n-1}) = \frac{C(w_1, w_2, \cdots, w_{n-1}, w_n)}{C(w_1, w_2, \cdots, w_{n-1})}$$

Data sparsity

Example: 1000 word vocabulary corpus divided into training set of size 1,500,000 words and test set of size 300,000 words.

Observation: 23% of the trigrams occuring in test data never occurred in the training subset!
Similar observation with a 38 million word newspaper corpus.

Robust parameter estimation is needed

Eliminating Zero Probabilities : Smoothing

From the same training data, derive revised n-grams such that no n-gram is zero.

Discounting: Take away some counts from 'high count words' and distribute them among 'zero/low count words'.

Good-Turing Discounting

Let N_c denote the number of bigrams that occured c times in the corpus.

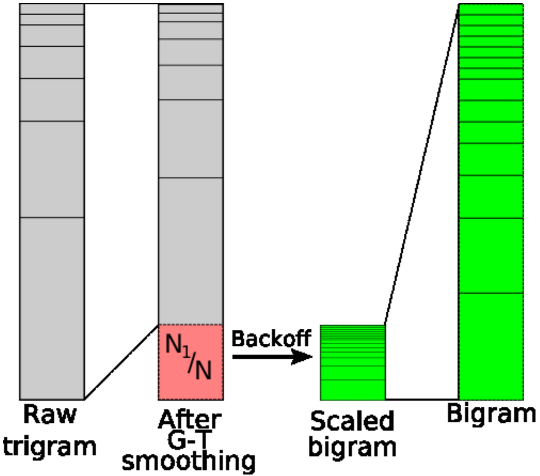
For bigrams that never occured, the revised count is

$$c^* = \frac{N_1}{N_0}$$

In general,

$$c^* = (c+1)\frac{N_{c+1}}{N_c}$$

Good-Turing Discounting: Illustration



source:www.inf.ed.ac.uk/teaching/courses/fnlp/lectures/05_slides.pdf

teaching/courses/fnlp/lectures/05_slides.pdf

Using n-gram 'hierarchy': Combining frequencies

Linear interpolation of n-grams

$$\hat{p}(w_3|w_1, w_2) = \lambda_1 p(w_3|w_1, w_2) + \lambda_2 p(w_3|w_2) + \lambda_3 p(w_3)$$

with $\lambda_i > 0; \quad \sum_i \lambda_i = 1.0$

Using n-gram 'hierarchy': Backoff if needed

Linear interpolation:

$$\hat{p}(w_3|w_1, w_2) = \lambda_1 p(w_3|w_1, w_2) + \lambda_2 p(w_3|w_2) + \lambda_3 p(w_3)$$

Backoff:
if trigram count > 0 no interpolation
Backoff to bigram otherwise

We "backoff" to a lower order n-gram only if we have zero evidence for a higher order n-gram.

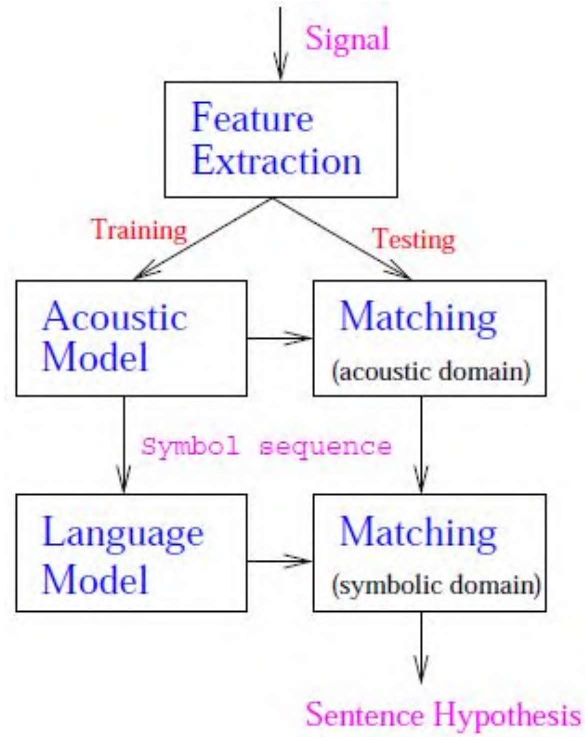
A non-linear method of combining counts.

Backoff Grammar

An algorithm for computing backoff trigram grammer is

```
if  (trigramCount) > 0) {
                              // no change in trigramProb
else if (bigramCount) > 0){
                              trigramProb = a1 * bigramProb
} else {
                              trigramProb = a2 * unigramProb
}
```

ASR using GMM-HMM and Language model



samudravijaya@gmail.com