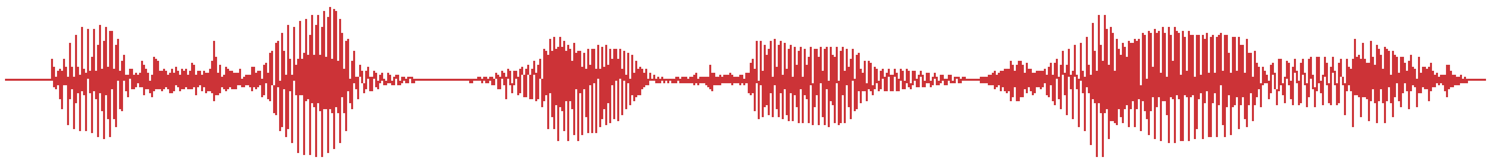
Guest Speakers Today

- Homayhoon Beigi:
  - Homayoon Beigi is a Columbia grad (Bachelors, Ms, and PhD) from the Department of Mechanical Engineering
  - Joined the Handwriting Recognition Group at IBM T.J. Watson Research Center in 1990 and the Speech Recognition Group (1991-2001)
  - **President** of **Recognition Technologies, Inc.** in Yorktown Heights, NY and the **Vice President** of **Internet Server Connections, Inc.**, White Plains, NY. In addition, he holds a position as an **Adjunct Professor** at the School of Engineering of **Columbia University** teaching courses including "Fundamentals of Speaker Recognition (COMS-E6998-005)," "Fundamentals of Speech Recognition (COMS-E6998-004)," "Digital Control Systems (EEME-E4601y)," "Applied Signal Recognition and Classification (ME-E6620y)", and "Speech and Handwriting Recognition (EE-E6820x).
  - Author of first and only textbook on speaker recognition, "Fundamentals of Speaker Recognition."

- Fadi Biadsy
  - MS thesis in handwriting recognition for Arabic
  - Columbia CS PhD in 2011 "Automatic Dialect and Accent Recognition and its Application to Speech Recognition”
  - Research in charisma, biography generation, language and dialect ID
  - Now at Google Research in NY working on ASR and many other areas of research

Speech Recognition



Homayoon Beigi

Beigi@RecoTechnologies.com

http://www.RecognitionTechnologies.com/beigi

Recognition Technologies, Inc.

South Salem, NY, U.S.A.

&

Computer Science Department and
Mechanical Engineering Department
Columbia University, NYC, NY, U.S.A.

Background

- **Recognition Technologies, Inc.** - *President* – since 2003
  Products: RecoMadeEasy® Speaker Recognition, Speech Recognition,
  Face Recognition, Access Control, Language Rating Engines

- **Internet Server Connections, Inc.** - *Vice President* – since 2001
  Products: CommerceMadeEasy® Cookieless Commerce, Portfolio Optimization
  Encyclopaedia Iranica, Specialized Hosting

- **Columbia University** – *Adjunct Professor* – since 1995
  Departments: EE, ME, CS, and Civil
  Courses: Fundamentals of Speaker Recognition, Applied Signal Recognition,
  Fundamentals of Speech Recognition, and Digital Control Systems

- **IBM T.J. Watson Research Center** – Research Staff Member – 1991-2001
  Departments: Online Handwriting Recognition
  Speech Recognition (Human Language Technologies)

- **Columbia University** –Center for Telecommunications Research - 1990-1991

- **Columbia University** – BS (1984), MS (1985) & PhD (1990)

Beigi@RecoTechnologies.com

Speech Technologies
1950s

- **Speech Recognition – Speech to Text (ASR)**
  - **1950s – Mostly Digit Recognition and Spectral methods**
    Concentrated on Digits, Monosyllabic words, and Vowels

U.S.: AT&T Bell Labs – isolated speaker-dependent digit recognition
RCA Labs – isolated speaker-dependent monosyllabic recognition
Used an analog filter bank – concentrating on vowels
MIT Lincoln Labs – isolated stop-vowl-stop (b-vowel-t)
transition syllables, speaker-independent

U.K.: University College, London – isolated 4 vowels and 9 consonants
Used primitive statistical
language model

Speech Technologies
1960s

- **Speech Recognition – Speech to Text (ASR)** – *continued*

  - **1960s – Fundamental research that would lead to practical techniques**

    Japan: Radio Research Lab – vowel recognition hardware
    Kyoto University – phoneme recognition hardware
    NEC Labs – digit recognition hardware

    U.S.: RCA Labs – Martin's Simple time-warping techniques
    Formed Threshold Technology with a product
    Bogert, et al. – 1963 – Seminal paper on Cepstrum computation
    Work related to echo analysis later became the basis of modern speech recognition feature extraction
    CMU – Reddy's dynamic phoneme tracking, enabling continuous S.R.

    U.S.S.R.: T.K. Vintsyuk – use of Dynamic Programming

Beigi@RecoTechnologies.com

Speech Technologies
1970s

- **Speech Recognition – Speech to Text (ASR) –** *continued*

  - **1970s – Isolated word recognition, Dynamic Programming,**
    Linear Predictive Coding used as features,
    Task-driven large-vocabulary and speaker-independent S.R.

    U.S.: Itakura – worked on using LPC for speech recognition
    Developed metrics for
    IBM – Princeton's Baum with ARPA contract to develop IBM HMM-based engine
    Tappert, et al. DTW, used Dynamic programming for
    continuous speech recognition
    using a segment then recognize methodology.
    Task-driven large vocabulary:
    1. Database Access, 2. Laser Patent context
    3. Tangora – office correspondence.
    AT&T – Speaker-independent recognition using clustering
    techniques
    U.S.S.R.: Velichko – 200-word recognizer

    Japan: NEC – Sakoe used dynamic programming techniques

Speech Technologies
1980s

- **Speech Recognition – Speech to Text (ASR)** **–** *continued*

  - **1980s – Continuous speech recognition – 1000 word continuous Speech recognition, data collection, DARPA funding**

U.S.:AT&T – Level building time warping continuous S.R.
IBM – HMM for continuous speech recognition later
used by Dragon systems and IDA (Institute for Defense Analyses)
and later by AT&T and other resarch labs.
Experiments with Tangora using 20,000 words (1987).
CMU – Weibel used TDNNs for speech recognition
DARPA – sponsored large-vocabulary (1000 words)
continuous speech recognition research mostly
through data collection and competitions
(IBM, CMU SHPINX, BBN BYBLOS, MIT Lincoln
Labs, SRI (Sarnoff), and AT&T)
Wall street journal corpus, and HUB corpora.

Japan: NEC – two-level dynamic programming

U.K.: JSRU (Joint Speech Research Unit) – Word-template
continuous speech recognition

Speech Technologies
1990s

- **Speech Recognition – Speech to Text (ASR) –** *continued*

  - **1990s – Continuous speech recognition and speaker-independent**
    Large vocabulary – 25,000 + words, Desktop products, Standards

U.S.:AT&T – HMM and LPC(AR) derived MFCC
IBM – HMM (MA version) MFCC based system – products out of research
Tangora -> ViaVoice Simply Speaking and VoiceType
Dragon Systems – Naturally Speaking
CMU – SPHINX
DARPA – More effort on data collection – HUB4 and NIST evaluations
LDC – Linguistic Data Consortium
Nuance – spins off from SRI
Standards – API Standards were developed by different Consortia: SAPI, SRAPI, RTC, etc.
Microsoft – Buys Entropic and does several investigative projects.

Europe: ELRA – European Language Resource Association

Speech Technologies
2000s

- **Speech Recognition – Speech to Text (ASR) –** *continued*

  - **2000s – Consolidation of products**

    U.S.: Dragon Systems – was bought by L&H (a Belgian company)
    ScanSoft – bought the rights from L&H in a bankruptcy proceeding and took over Dragon Naturally Speaking
    IBM – sold exclusive rights of ViaVoice to ScanSoft
    Nuance – A previously financially troubled company and offshoot of SRI, went public, then merged with ScanSoft and became the rights owner for Dragon and ViaVoice
    Nuance purchased 33 companies in this decade
    CMU – terminated support of SPHINX and made it completely open-source
    SRI – provided solutions for mostly government based projects
    Recognition Technologies – Started with Speaker and added Speech Reco.
    Europe: L&H – Lernout and Hauspie (Belgian company) bought Dragon Systems in 2000 and went bankrupt the same year
    Cambridge University: HMM toolkit was released for use by researchers
    COST – European Cooperation in Science and Technology projects
    Japan: Julius and Julian OpenSource speech recognition engines (only decoding)

Beigi@RecoTechnologies.com

Speech Technologies
2010s

- **Speech Recognition – Speech to Text (ASR) –** *continued*

  - **2010s – Two poles – OpenSource and further consolidation**
    Multilingual support, Cloud-based services, SmartPhone apps

Worldwide.: Nuance – licensed all of IBM's patents on speech
SIRI and its own assistant
bought Vlingo which was the basis for Samsung's S-Voice
Since 2010 Nuance has purchased another 20+ companies
Latest: Voicebox in May 2018 for $82M
HTK – HTK became widely used in academic research
KALDI – An alternative to HTK was created
Google – hired many from IBM and ATT to create its S.R.
Amazon – Introduced Alexa and Speech Services
Microsoft – Invested more in speech and created Cortana
Apple – SIRI for the iPhone
IBM – Watson project now starting to use speech recognition
It is unclear whose speech recognizer will be used!
Recognition Technologies – Large Vocabulary Embedded Engines
Multilingual Speech Recognition
Interactions – Bought AT&T's remaining speech group (Nov. 2014)

Beigi@RecoTechnologies.com

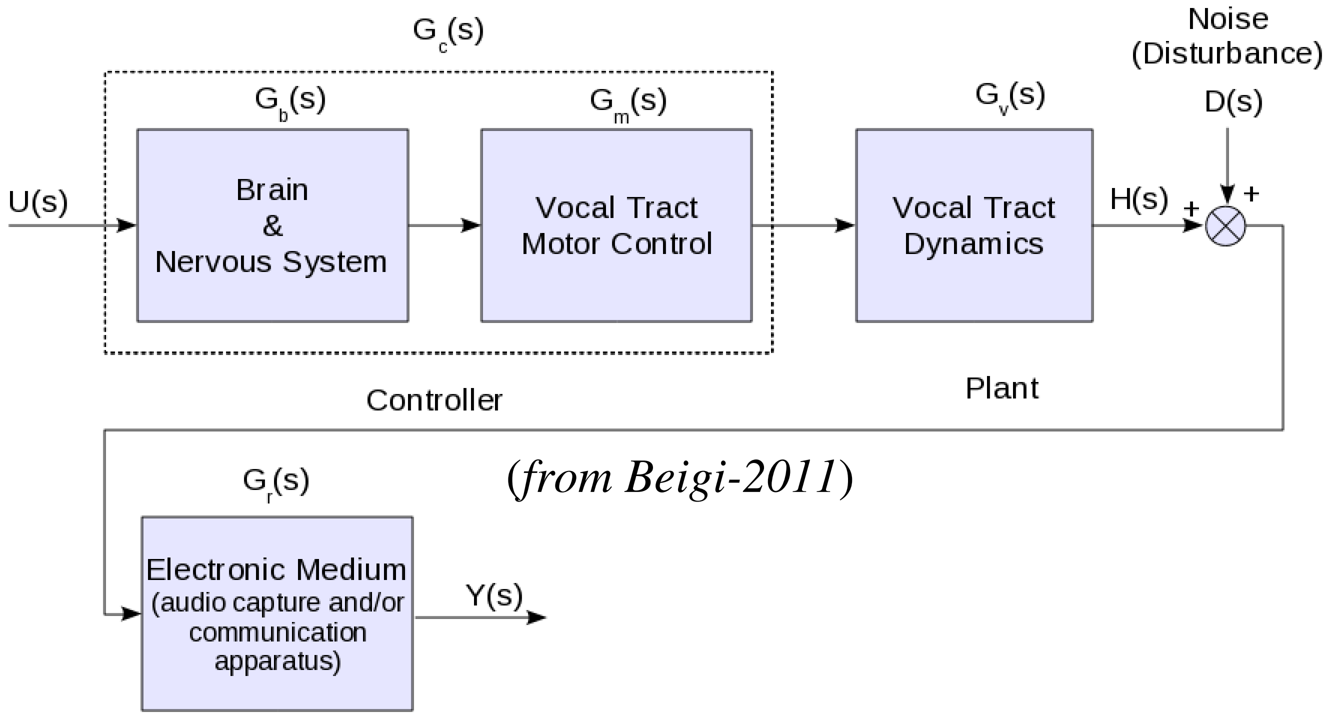
Speech Technologies
Other

- **Speech Synthesis – Text to Speech (TTS) –** AT&T Voder 1939 World Fair (H.W. Dudley 1936)
  Texas Instruments – 1978 Speak & Spell (Toy)
- **Speaker Recognition –** Biometrics – Peterson (1952), Pollack (1954)
- **Speech Understanding –** DARPA SUR (Speech Understanding Research) – 1971
  IBM – Airline automatic reservation demonstration
  Automatic financial transaction demonstration
- **Language Translation –** IBM statistical translation – English-French (1990s)

Highly Multidisciplinary

- **Anatomy of Speech and Signal Representation**
- **Signal Processing**
- **Feature Extraction**
- **Practical Issues and Solutions**
- **Phonetics and Phonology**
- **More Feature Extraction and Processing – including Suprasegmental Features**
- **Integral transforms and Cepstral Computation**
- **Probability Theory**
- **Information Theory and Metrics and Divergences**
- **Decision Theory and Parameter Estimation**
- **Hidden Markov Models**
- **Support Vector Machines, Neural Networks – including Deep Neural Networks**
- **Language Modeling and NLP**
- **Search Techniques**

Control System View of Speech Production



(from Beigi-2011)

$$H(s) = G_v(s)G_c(s)U(s)$$
$$Y(s) = G_r(s)\left[H(s) + D(s)\right]$$

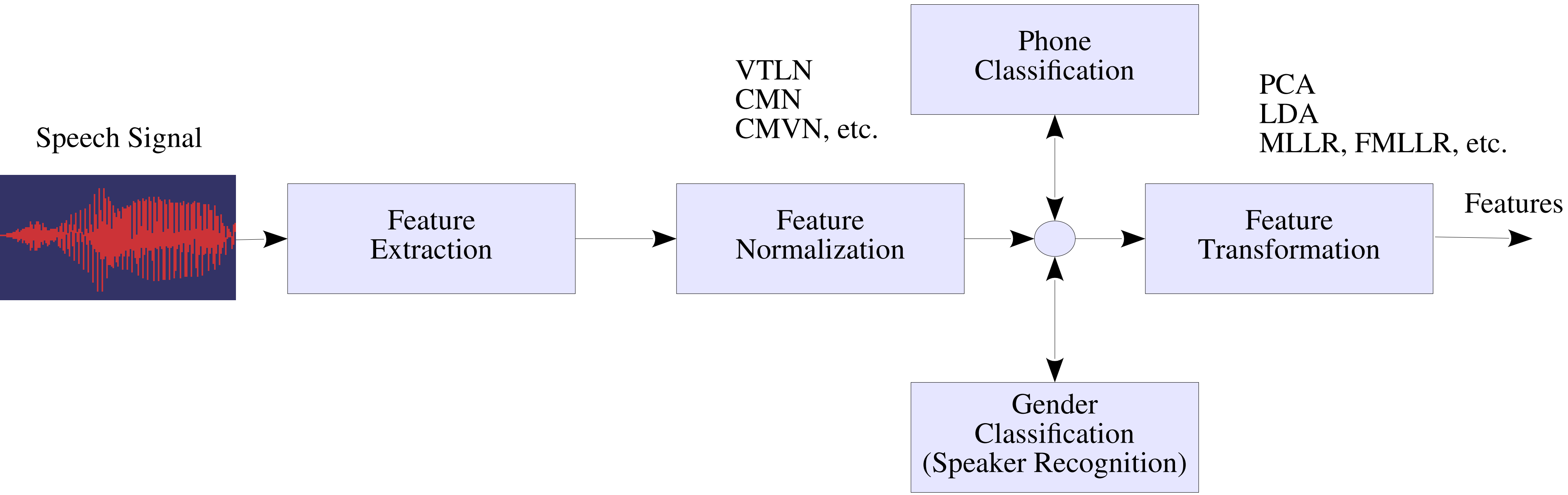
$$\longrightarrow \quad Y(s) = G_r(s)\left[G_v(s)G_c(s)U(s) + D(s)\right]$$

Voice
(Speaker Recognition)

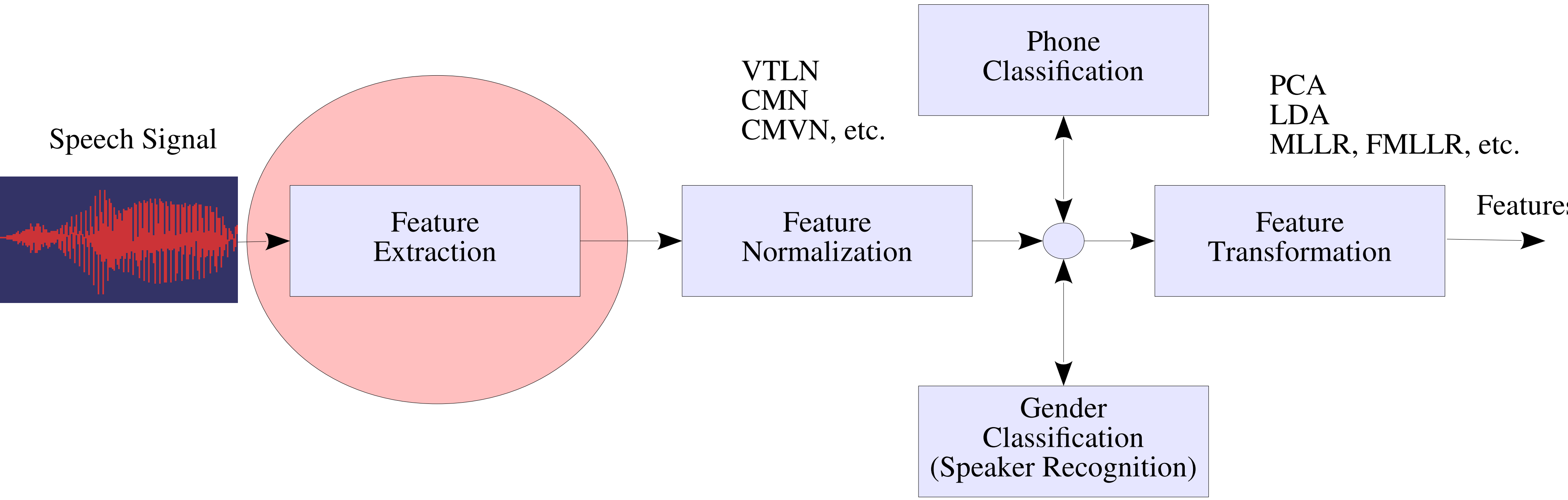
Content
(Speech Recognition)

Beigi@RecoTechnologies.com

A Generic Speech Recognition System (Front-End)



A Generic Speech Recognition System -- Recap (Front-End)



Features

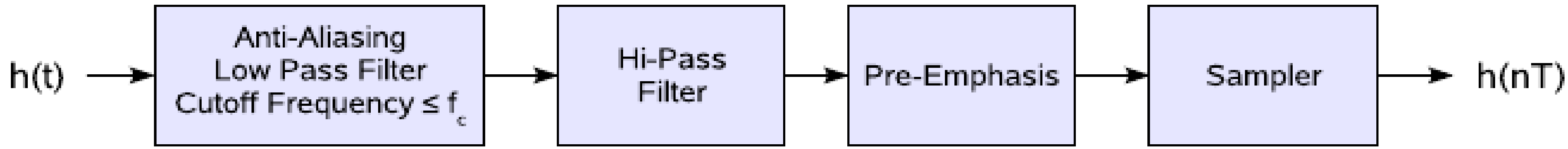
- **Linear Predictive Coding Features** – *LPC, Reflection, Log Area Ratio, …*
- **Filter Banks** – *EIH Model, Wavelet Filter Banks, Other Nonuniform Filter Banks, …*
- **Modulation Features – AM/FM, Empirical Mode Decomposition (EMD)**
- **Mel Frequency Cepstral Coefficients** (MFCC)
  - **1963 – Cepstra**: *Bogert, Healy and Tukey*
  - **1937 – Mel***ody: Stevens, Volkman and Newman*
  - **Zero Mean** – *All Coefficients but c0*
- **Suprasegmental Features**

Beigi@RecoTechnologies.com

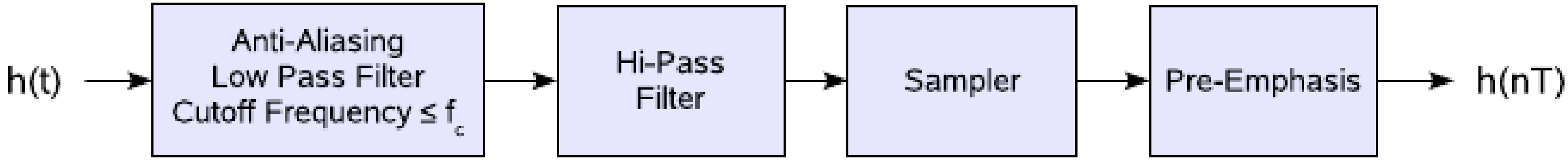
Suprasegmental Features

- **Prosodic Features**
  - **Pitch**
  - **Loudness (Sonority)**
- **Metrical Features**
  - **Stress**
  - **Rhythm**
- **Temporal Features**
- **Co-Articulation**

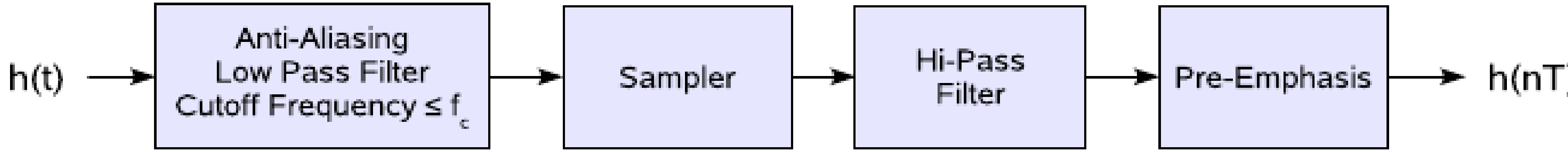
Sampling Process



Best Method



Alternative Method



Alternative Method

Beigi-2011

Beigi@RecoTechnologies.com

Feature Extraction
(Spectral Analysis – e.g. MFCC)

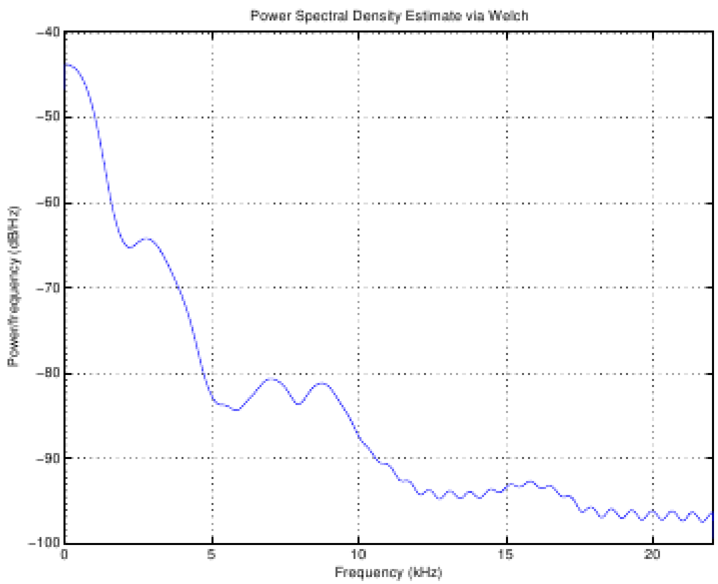
- **Direct Method** – *Moving Average (MA)*
  - **Framing**
  - **Windowing** – *Hamming, Hann(ing), Welch, Triangular, …*
  - **DFT** – *Spectral Estimation*
  - **Frequency Warping** – *e.g., Mel or Bark*
  - **Magnitude Warping**
  - **Mel Frequency Cepstral Coefficient Computation (MFCC)**
  - **Mel Cepstral Dynamics – Delta and Delta-Delta Cepstra**

Bandpass filtering and Pre-emphasis

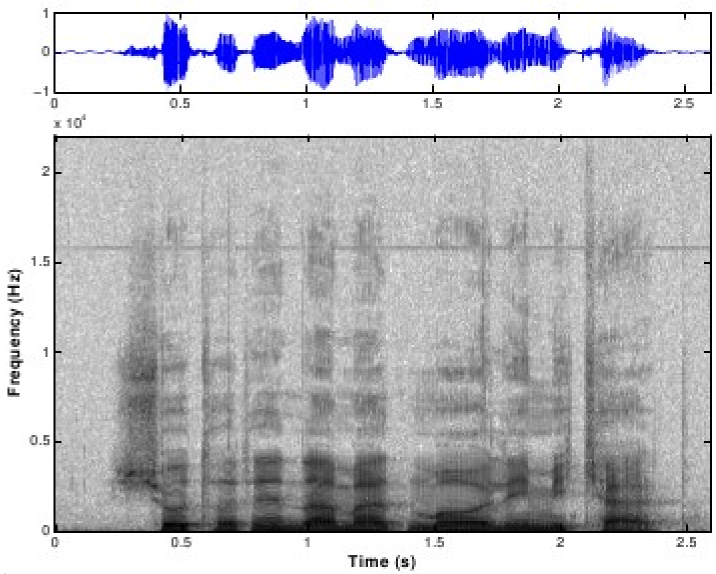
- Highpass portion of the filter – ~ *20 Hz*
- Lowpass portion of the filter – ~ $f_c - 400Hz$
- Pre-emphasis of the signal

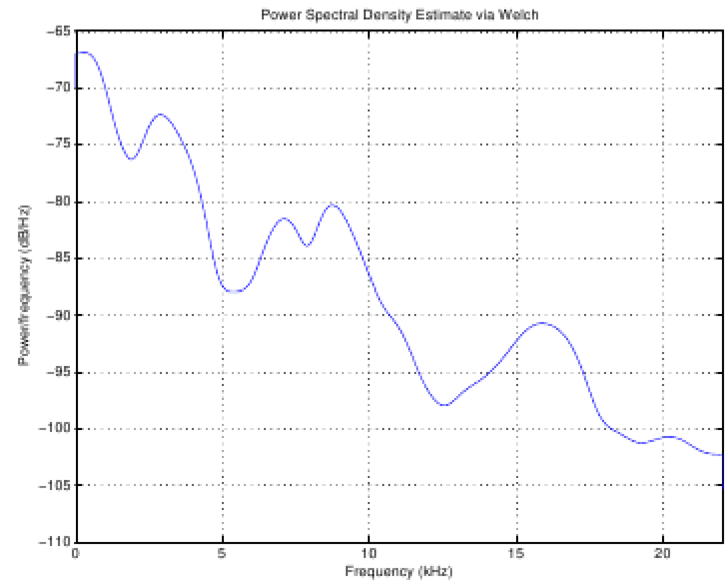
$$H_p(z) = 1 - \alpha z^{-1} \qquad \mathbf{0.95 \leq \alpha \leq 0.97}$$

Pre-Emphasis

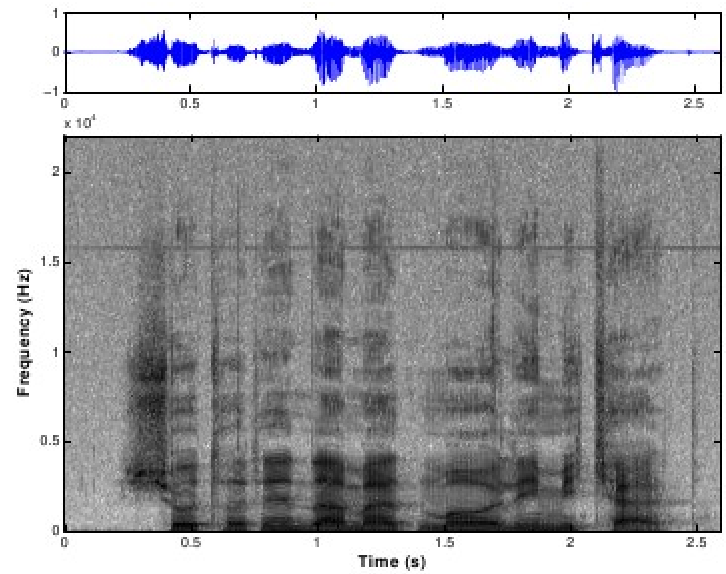


Top: Welch PSD Estimate, Bottom: Spectrogram
Original Signal: Speech Sampled at 44100 Hz





Top: Welch PSD Estimate, Bottom: Spectrogram
Pre-Emphasized Signal:



Windowing the Signal (DSTFT) (Summary)

$$H_{km} = \sum_{n=0}^{N-1} h_n w(n-m) e^{-i\frac{2\pi nk}{N}}$$

$$h_{nm} = \frac{1}{N} \sum_{k=0}^{N-1} H_{km} e^{i\frac{2\pi nk}{N}}$$

Where,

$$h_{nm} \stackrel{\Delta}{=} h_n w(n-m)$$

$$\sum_{m=0}^{N-1} w(n-m) = 1 \quad \forall \ n$$

$$H_k = \sum_{m=0}^{N-1} H_{km}$$

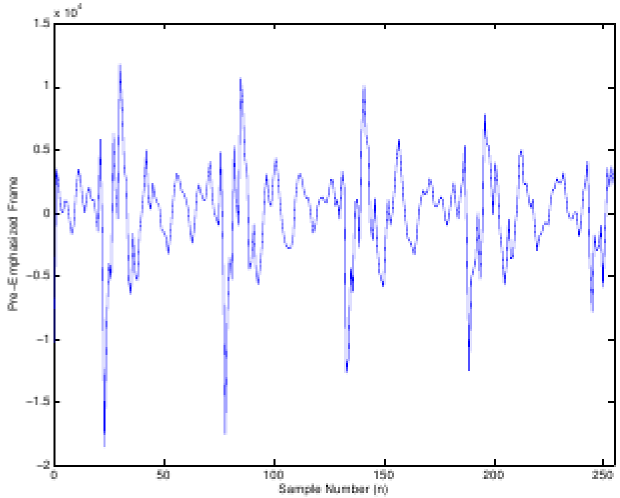
Windowing the Signal (DSTFT)

- *N point window – 20 – 30 ms window*
- *Ξ point shift – 10 ms shift*
- *l is the index of frame* $l \in \{0, 1, \cdots\}$
- *Reference to sample* ${}_l h_n$
- *Reference to sample* ${}_l H_k$
- *Windowed signal*

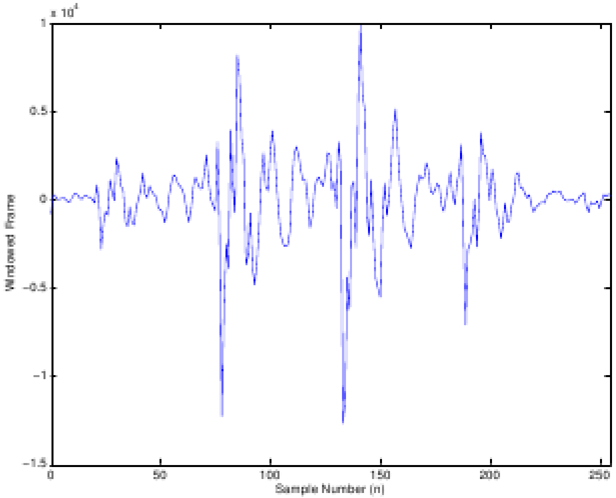
$$ {}_l\tilde{h}_n = {}_l h_n w(n) \quad for \;\; n \in \{0, 1, \cdots, N-1\} $$

- *Pad data with zeros when not available – as we do for STFT*

Windowing the Signal (DSTFT)



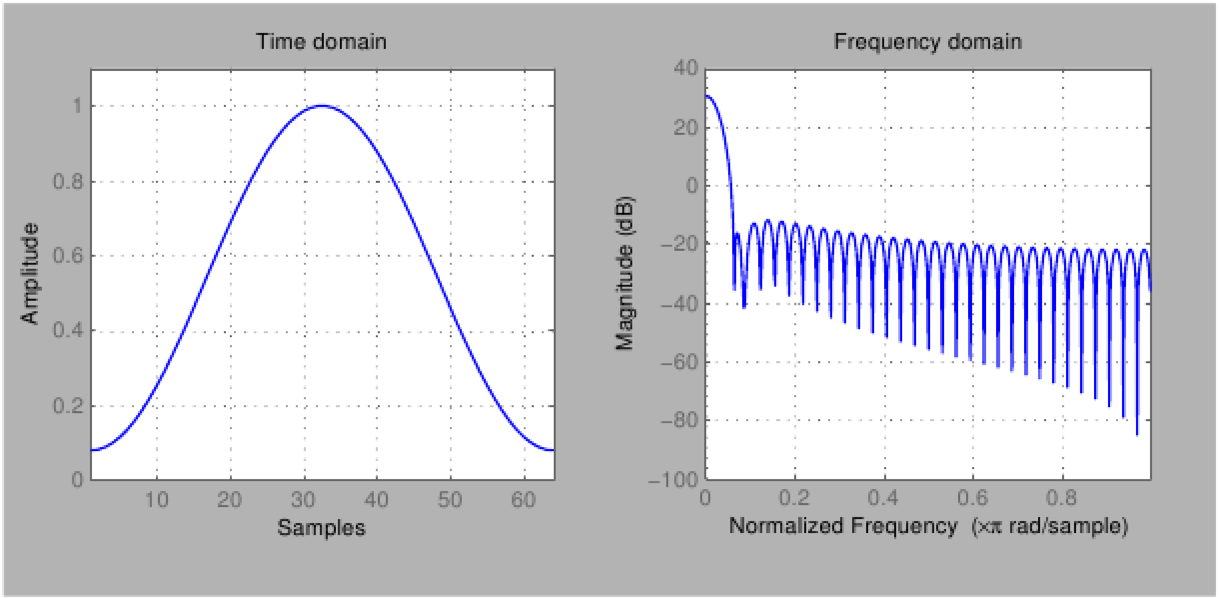
Pre-Emphasized Frame of audio N = 256



Windowed Frame N = 256

Windows
(Hamming)

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$



Beigi-2011

- *Endpoints fall off very quickly (very popular) – pro*
- *Side lobes (higher harmonics) stay flat – con*

Frequency Warping

$$\left|{}_{l}\breve{H}_{m}\right| = \mathscr{M}_{mk}\left|{}_{l}H_{k}\right|$$

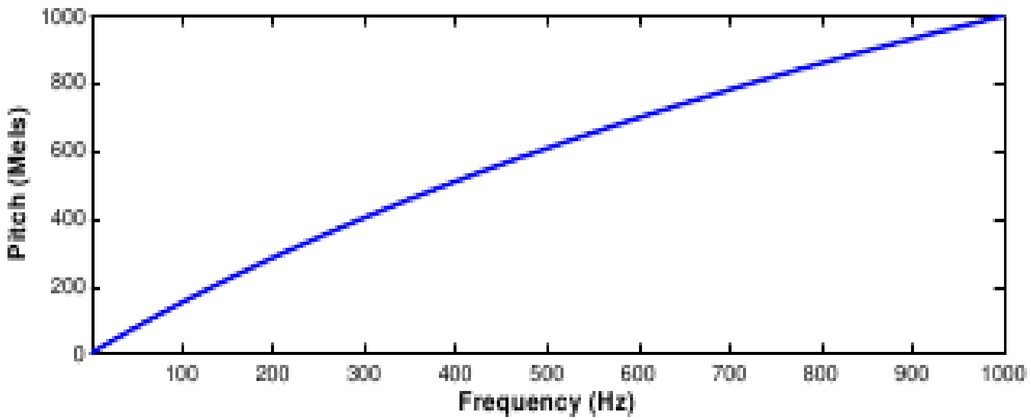
- *Matrix of elements* $\mathscr{M}_{mk}$ *is the mapping by Triangular filters from Linear to Mel*

  $\mathscr{M}, \{\mathscr{M} : \mathscr{R}^{N} \mapsto \mathscr{R}^{M}\}$
- *M = 24 according to critical bands*
- *M = 40 for high resolution MFCC computation*

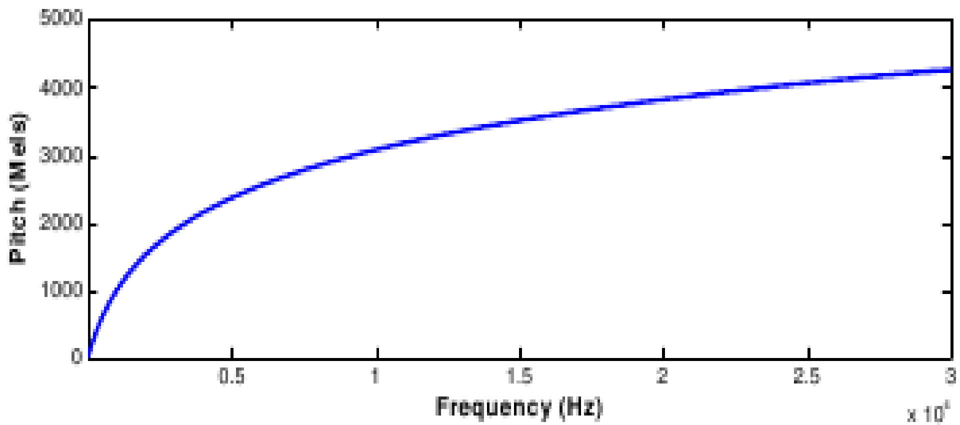
Pitch *(Mel Scale)*

$$\wp = \frac{1000}{\ln(1+\frac{1000}{700})}\ln(1+\frac{f}{700})$$

$$\wp = \frac{1000}{\log(2)}\log(1+\frac{f}{1000})$$



Up to 1 kHz

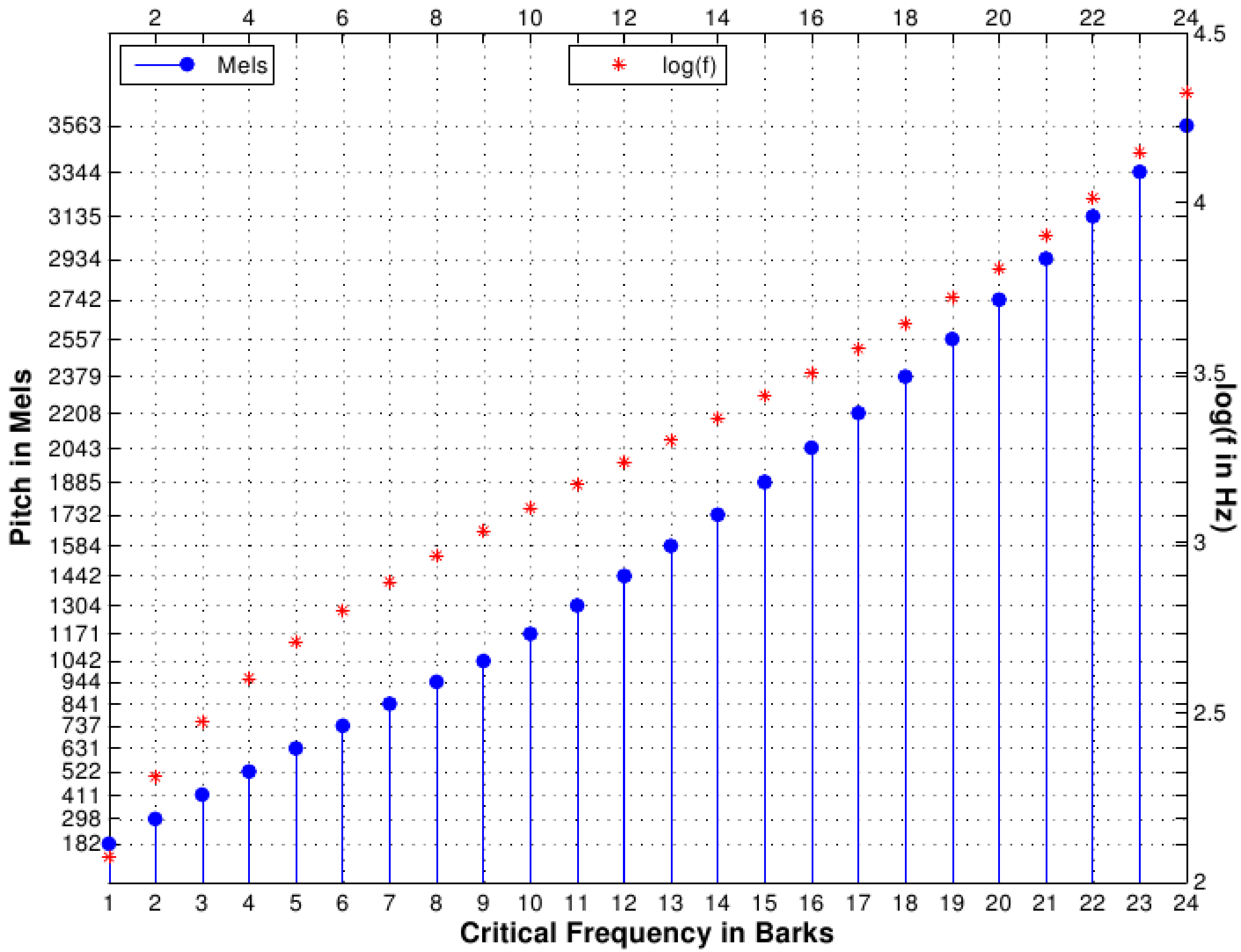


Audible Range

f (Hz)	$\wp$ (Mel)	f (Hz)	$\wp$ (Mel)	f (Hz)	$\wp$ (Mel)
40	43	867	928	4109	2314
161	257	1000	1000	5526	2600
200	300	2022	1542	6500	2771
404	514	3000	2000	7743	2914
693	771	3393	2142	12000	3228

Frequency-Pitch

Pitch-Frequency Relation

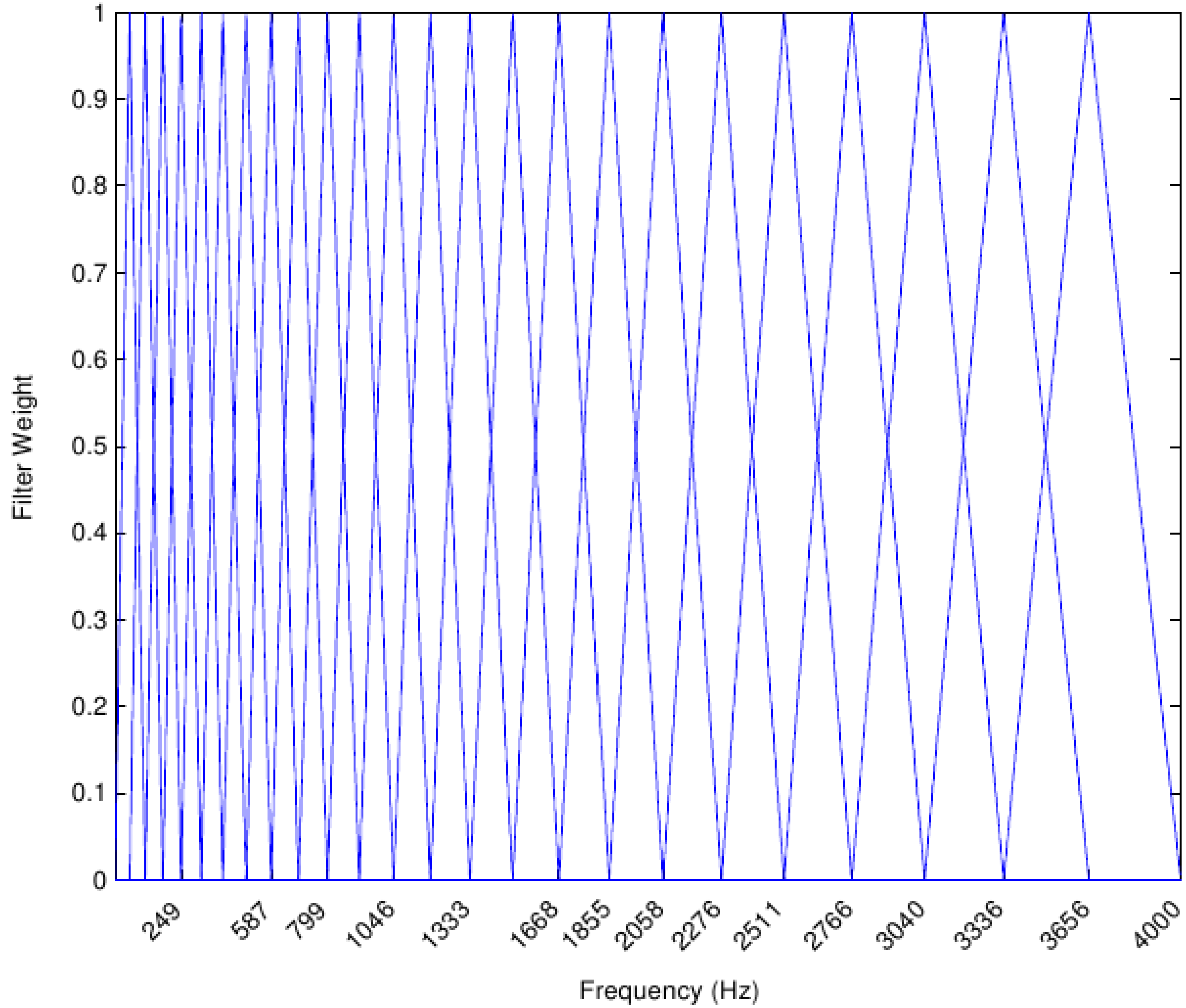


Beigi-2011

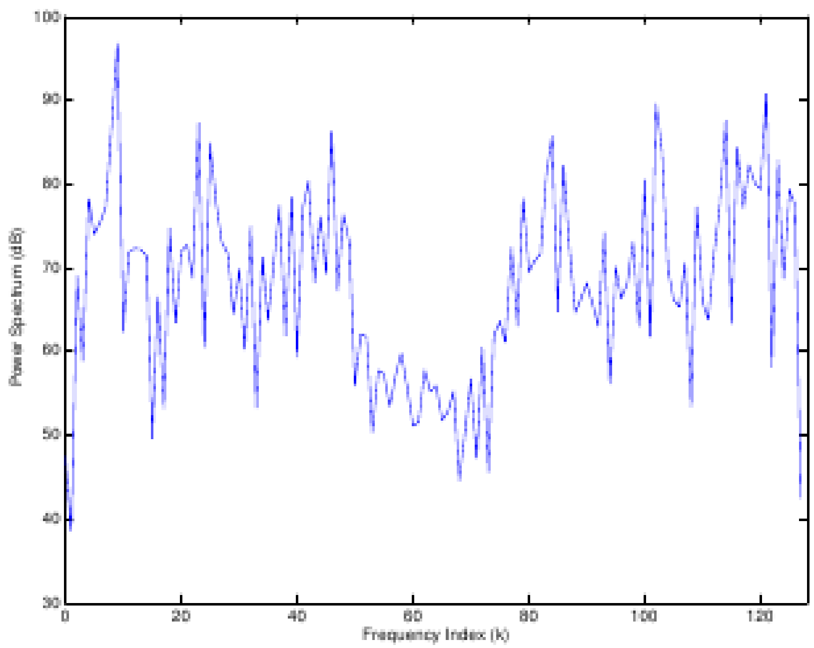
Beigi@RecoTechnologies.com

Frequency Warping

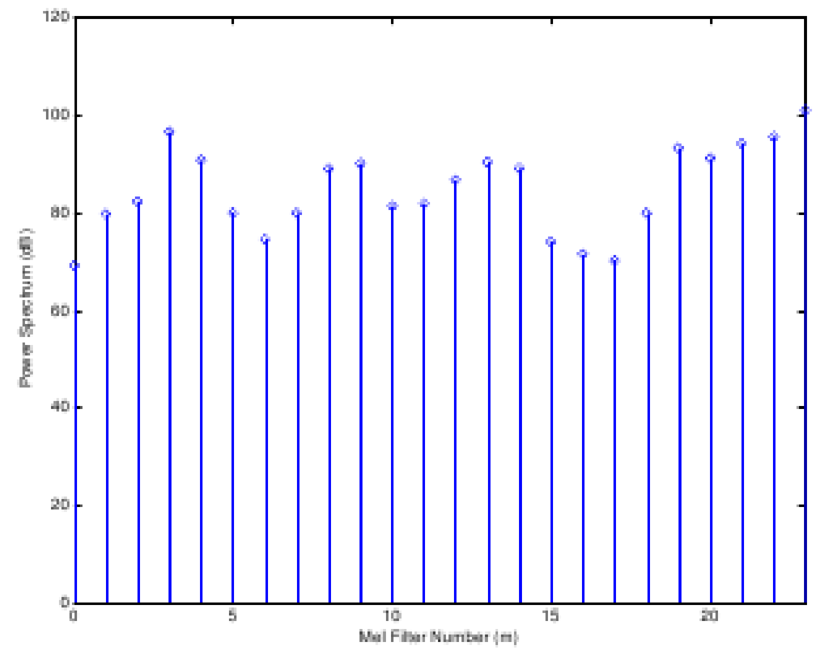
(center frequencies: 110 Mels apart)



Frequency Warping



Power Spectrum of the Frame N = 256



Power Spectrum in the Mel Frequency Domain
N = 256, M = 24

Magnitude Warping

- *Relative Intensity computation in dB*

$$ {}_l\tilde{C}_m = 10\log\left(\frac{\left|{}_l\breve{H}_m\right|^2}{I_0}\right) $$

- *Log of spectrum, missing the normalization*

$$ {}_lC_m = \log\left(\left|{}_l\breve{H}_m\right|^2\right) $$

Power Cepstrum

$$\tilde{h}_{pc} \stackrel{\Delta}{=} \left[\mathscr{Z}^{-1}\{\log\left(|H(z)|^2\right)\}\right]^2$$
$$= \left[\frac{1}{2\pi i}\oint_{\Gamma_c} \log\left(|H(z)|^2\right) z^{n-1} dz\right]^2$$

- *In contrast with the Complex Cepstrum and the Phase Cepstrum*
- *Let us limit ourselves to those z lying on the unit circle to get FFT*

$$\tilde{h}_{pc} = \left[\frac{1}{2\pi}\int_{-\pi}^{\pi} \log\left(|H(\omega)|^2\right) e^{i\omega t} d\omega\right]^2$$

Cepstrum of Convolution of Signals

Time Domain

$$h(t) = h_1(t) * h_2(t)$$ ***(Convolution)***

Fourier Transfer (Frequency or Spectral) Domain

$$H(\omega) = H_1(\omega)H_2(\omega)$$ ***(Product)***

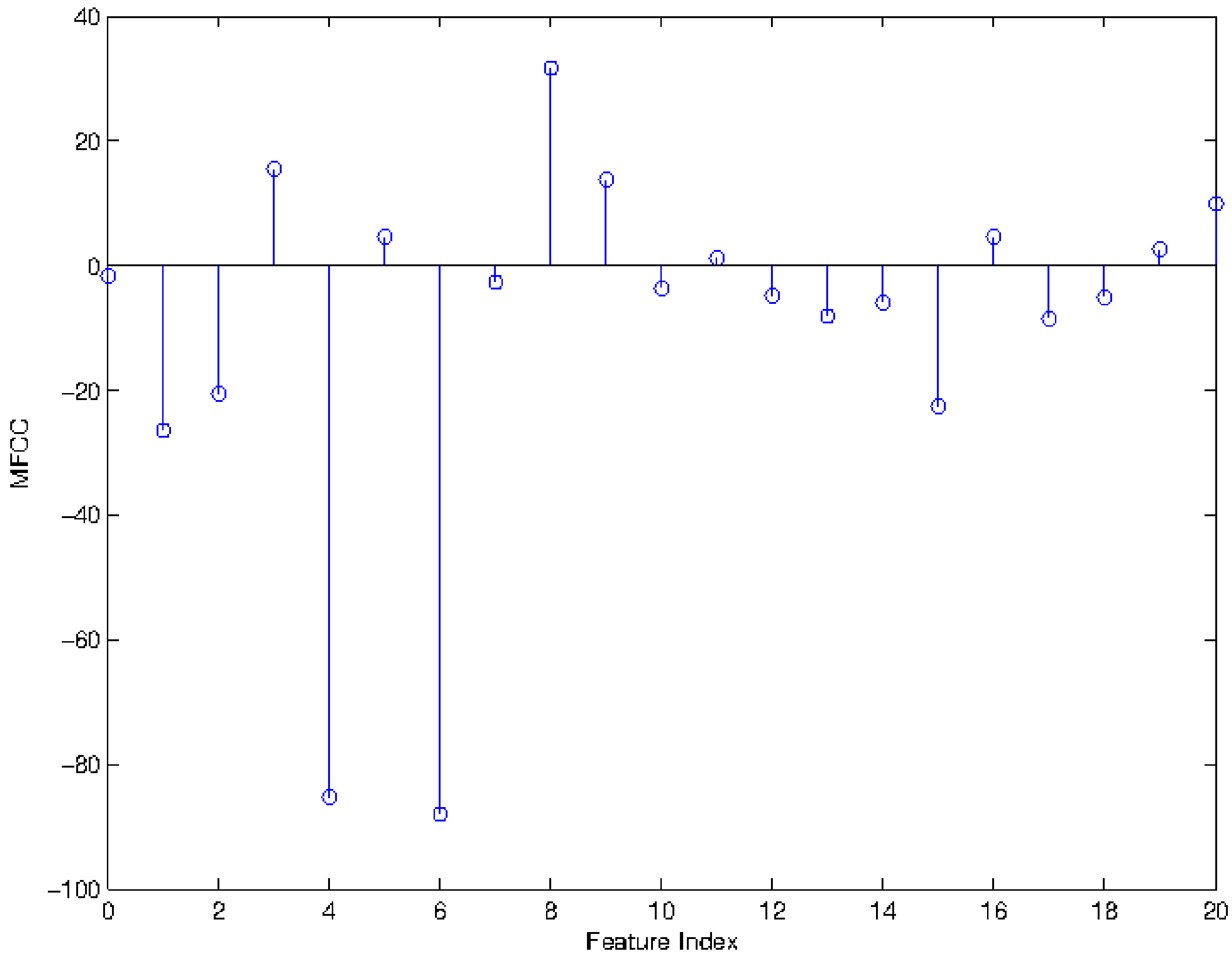
Cepstral (Quefrency) Domain

$$\hat{h}(t) = \hat{h}_1(t) + \hat{h}_2(t)$$ ***(Sum)***

- ***Cepstral mean subtraction*** *(CMS or CMN)* is a ***liftering*** technique, akin to deconvolution of noise which may be seen as a common signal having convolved with the variable part of the signal.

- ***Cepstral mean subtraction*** does not *affect additive* noise in the same way, although it is known to help with additive noise as well, when some limits are set for the signal to noise ratio.

Feature Extraction
(MFCC)



MFCC (1st component)

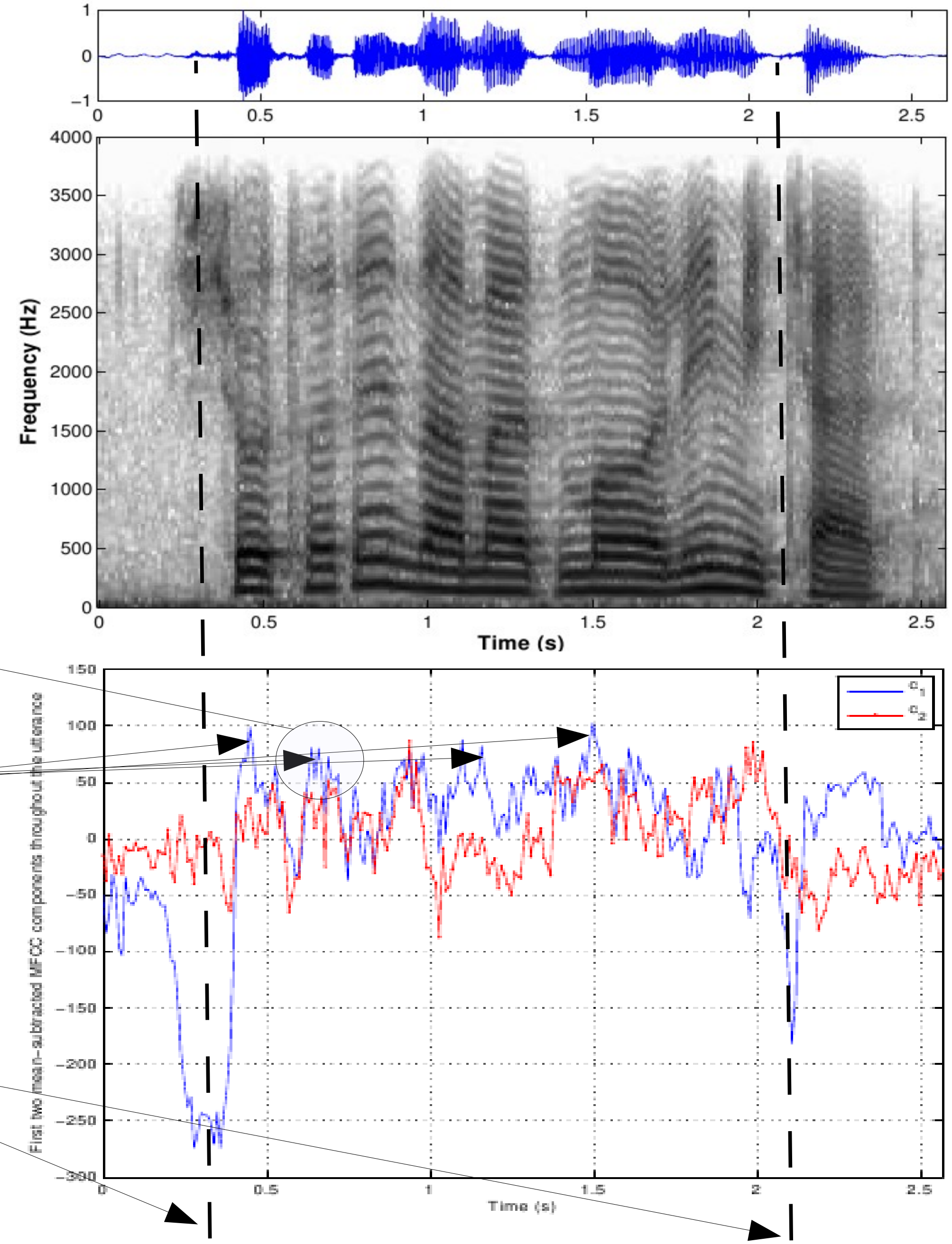
$$
{}_l c_d = \sum_{m=0}^{M-1} a_m \; {}_l C_m \cos\left(\frac{\pi(2d+1)m}{2M}\right)
$$

$$
a_m = \begin{cases} \frac{1}{M} & for\ m = 0 \\ \frac{2}{M} & \forall \quad m > 0 \end{cases}
$$

/t/

Presence of Stops
(High $\mathbf{c}_1$ *suggests more contribution from lower frequencies such as in stops)*

Presence of Fricatives
(Highly negative $\mathbf{c}_1$ *suggests more contribution from higher frequencies)*

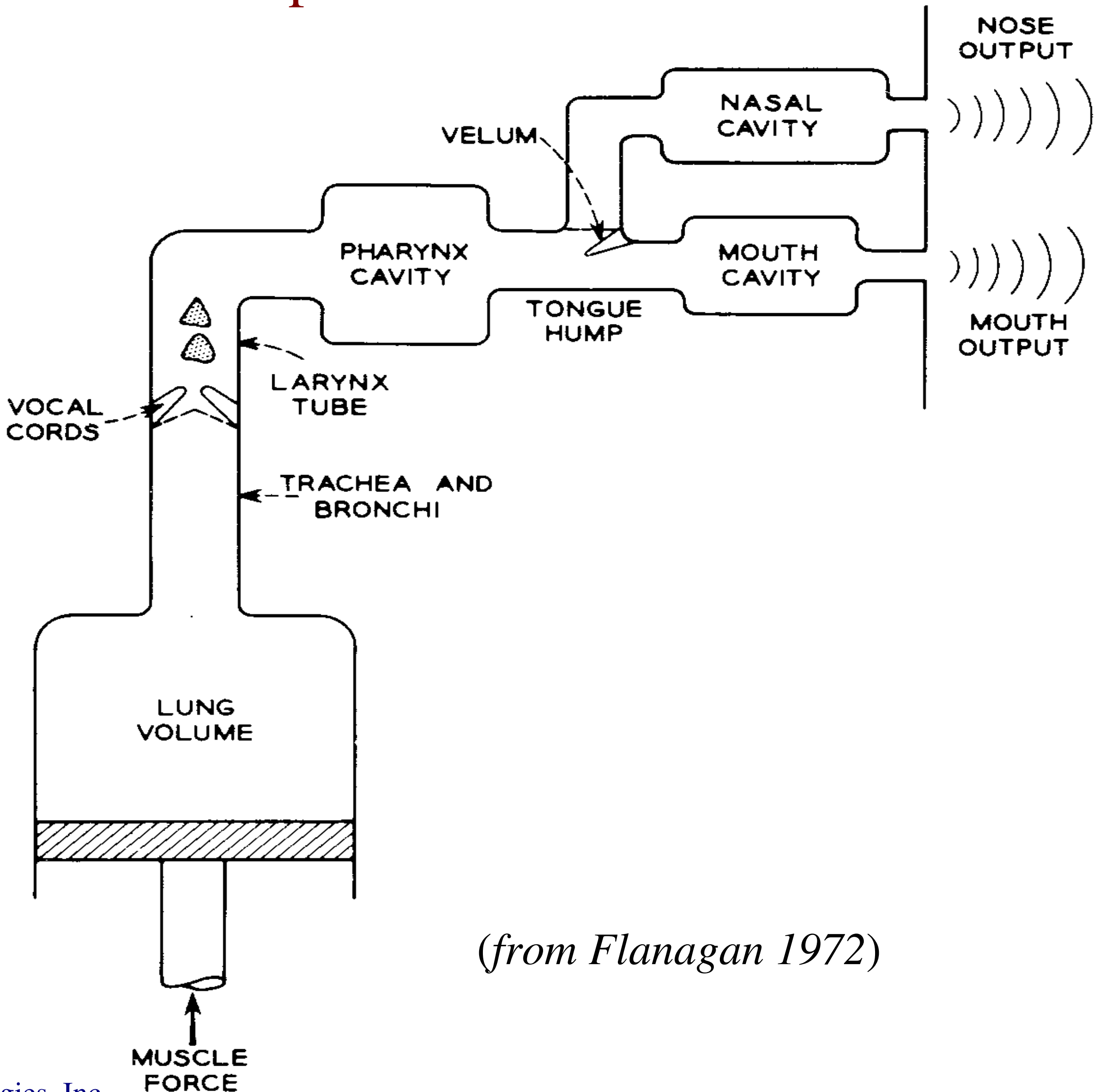


MFCC (Dynamics)

No. of significant terms

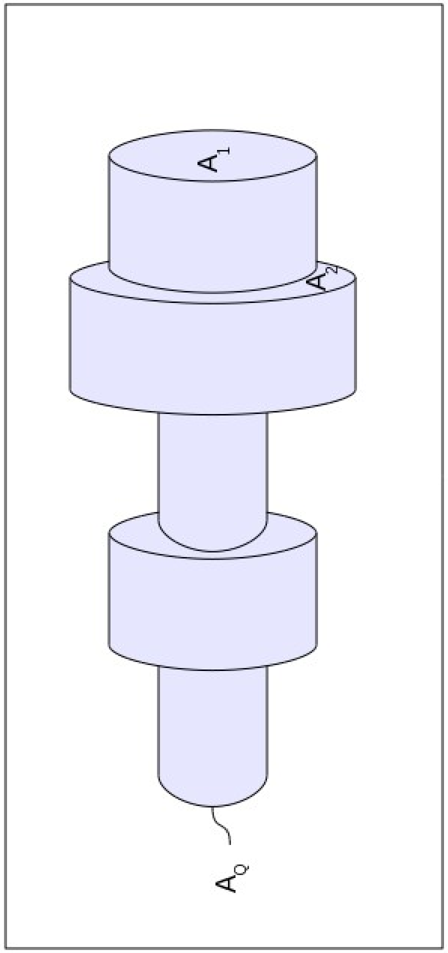
- Cepstra
- Delta Cepstra
- Delta-Delta Cepstra

Speech Production Model



(from Flanagan 1972)

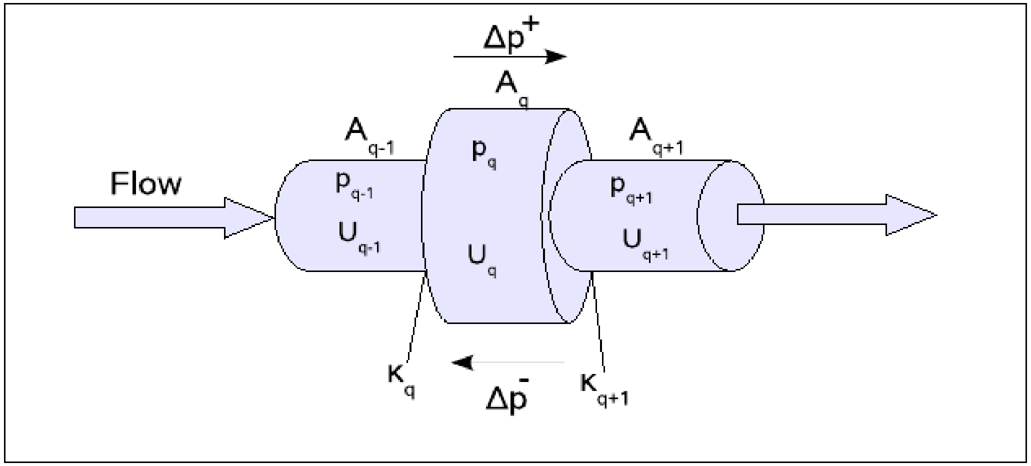
Feature Extraction
(Linear Predictive Methods – e.g. LPCC)



(*from Beigi-2010*)

- **Linear Predictive Method** – *AutoRegressive (AR)*
  - **Framing**
  - **Windowing** – *Hamming, Hann(ing), Welch, Triangular, …*
  - **AutoRegressive Estimation of the PSD**
  - **LPC Features** – *e.g., LPC Coefficients, PARCOR, Log Area Ratio (LAR), …*
  - **Frequency Warping** – *e.g., Mel or Bark*
  - **Magnitude Warping**
  - **Linear Predictive Cepstral Coefficient Computation (LPCC)**
  - **Feature Cepstral Dynamics – Delta and Delta-Delta Cepstra**

LPC Features: (PARCOR – reflection coeffs.)

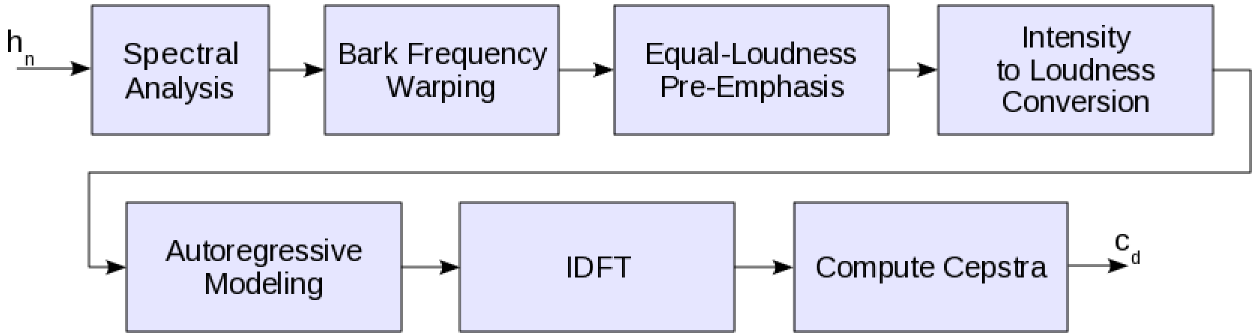


$$\kappa_q = \frac{A_{q-1} - A_q}{A_{q-1} + A_q}$$

Webster's Equation

$$\frac{\partial^2 p(x,t)}{\partial x^2} + \frac{1}{A(x)} \frac{\partial p(x,t)}{\partial x} \frac{\partial A(x)}{\partial x} = \frac{1}{c^2} \frac{\partial^2 p(x,t)}{\partial t^2}$$

Feature Extraction
(Perceptual Linear Predictive Method – PLP)



(*Beigi-2011*)

Feature Extraction (Wavelet Filterbanks)

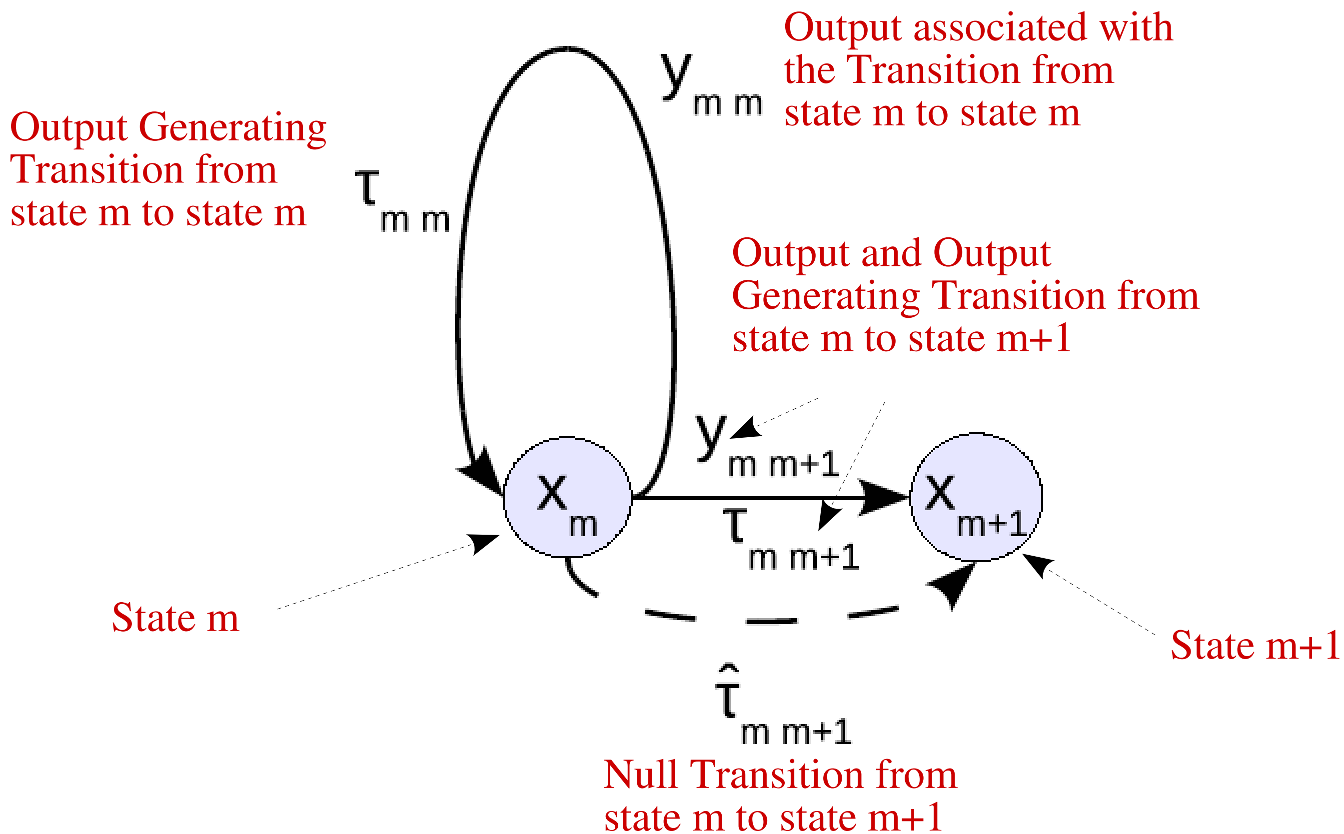
- **Mel-Frequency Discrete Wavelet Coefficients (MFDWC)**
  - **Use DWT instead of DCT to reduce the effect of frequency band spill-over**
  - **Process Sub-bands Separately**
    - **Relax the assumption that each frame contains only one phoneme**
- **Wavelet Octave Coefficients of Residues (WOCOR)**
  - **Hi-Pass Filter**
  - **Pre-Emphasis**
  - **Pitch Extraction**
  - **AutoRegressive Residue Computation**
  - **Compute Pitch-Sync Wavelet Coeffs**
  - **Expand Residual Signal**
  - **Subdivide Wavelet Coeffs**
  - **Compute WOCOR**

Feature Extraction
(Other Features)

- **Modulation Features**
  - **Amplitude Modulation**
  - **Frequency Modulation**
  - **Amplitude-Frequency Modulation**
    - **FEPSTRUM – log of Magnitude of non-overlapping narrowband filters – an AM signal**
    - **Mel-Cepstrum Modulation Spectrum (MCMS) – Modulation spectrum in cepstral domain**
- **Empirical Mode Decomposition (EMD)**

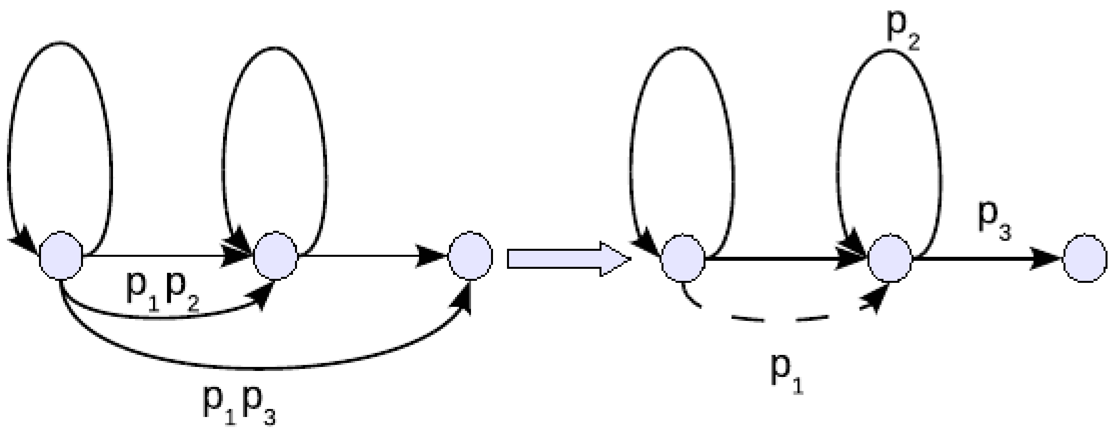
Hidden Markov Models

Basic HMM Element for Transition Output HMMS



Hidden Markov Models

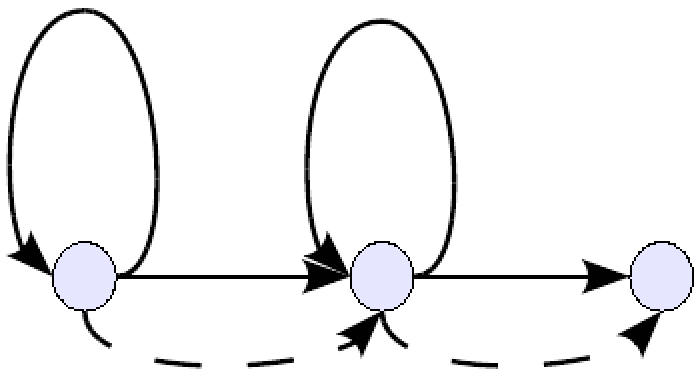
Simplification using Null Transitions



Simplification using a Null Transition

Hidden Markov Models

Average phone model



HMM of an Average Phone

The Bakis model named after a colleague and friend at IBM, ***Raimo Bakis***

Hidden Markov Models

Training, State Sequence, and Decoding

Three Problems may be formulated:

1. *Training*: Estimate parameters of the Markov source with highest likelihood of producing output sequences, $\{w\}_1^K$.

2. *State sequence*: Find the most probable state sequence that would produce the output sequence, $W_\omega = \{y_\omega\}_1^{N_\omega}$ as its output for Markov source with parameter vector, $\boldsymbol{\varphi}$. This problem may be solved using a special case of *dynamic programming* [6] called the *Viterbi algorithm* [26].

3. *Decoding*: Compute the probability of an output sequence for the Markov source with parameter vector, $\boldsymbol{\varphi}$,

$$P(W_\omega|\boldsymbol{\varphi}) = P(\{y_\omega\}_1^{N_\omega}|\boldsymbol{\varphi})$$

Hidden Markov Models
Decoding

Consider the third problem (decoding) – for the *non-unifilar* case:

$$P(W_\omega = \{y_\omega\}_1^{N_\omega} | \boldsymbol{\varphi}).$$

Computing this is the decoding problem

$$P(W_\omega|\boldsymbol{\varphi}) = \sum_{\{x_\omega\}_1^{N_\omega} : \{y_\omega\}_1^{N_\omega}} P(\{y_\omega\}_1^{N_\omega}, \{x_\omega\}_1^{N_\omega} | \boldsymbol{\varphi})$$

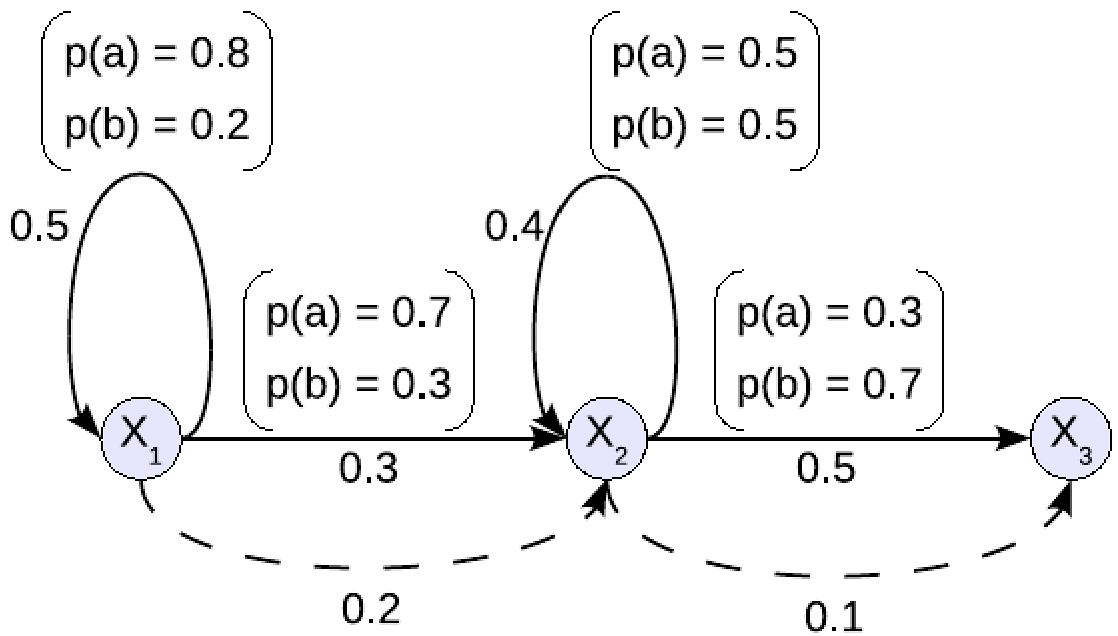
Sum of joint probability of the output sequences and the different possible state sequences.

$$\{x_\omega\}_1^{N_\omega} : \{y_\omega\}_1^{N_\omega}$$

Means all the state sequences that can produce the given output sequence

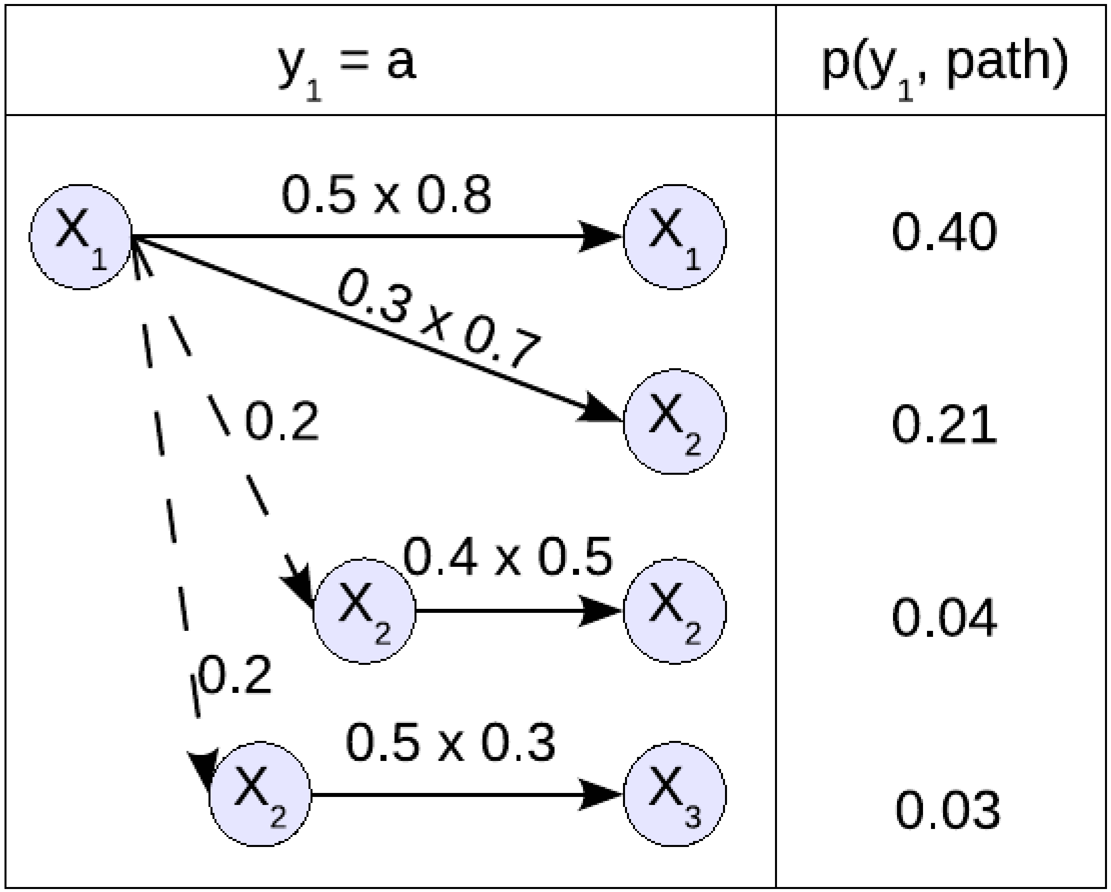
Hidden Markov Models
Decoding

Example 13.4 (Decoding the Output Sequence).
Let us consider the HMM described in the finite state diagram of Figure 13.7 and compute the output probability of the sequence, $\{y\}_1^4 = \{aabb\}$ where $Y : y \in \{a, b\}$. First, consider all the different possible transitions which may output the first output in the sequence, $y_1 = a$. Figure 13.8 shows the different paths for generating $y_1 = a$, by the model in Figure 13.7.



Hidden Markov Models

Decoding



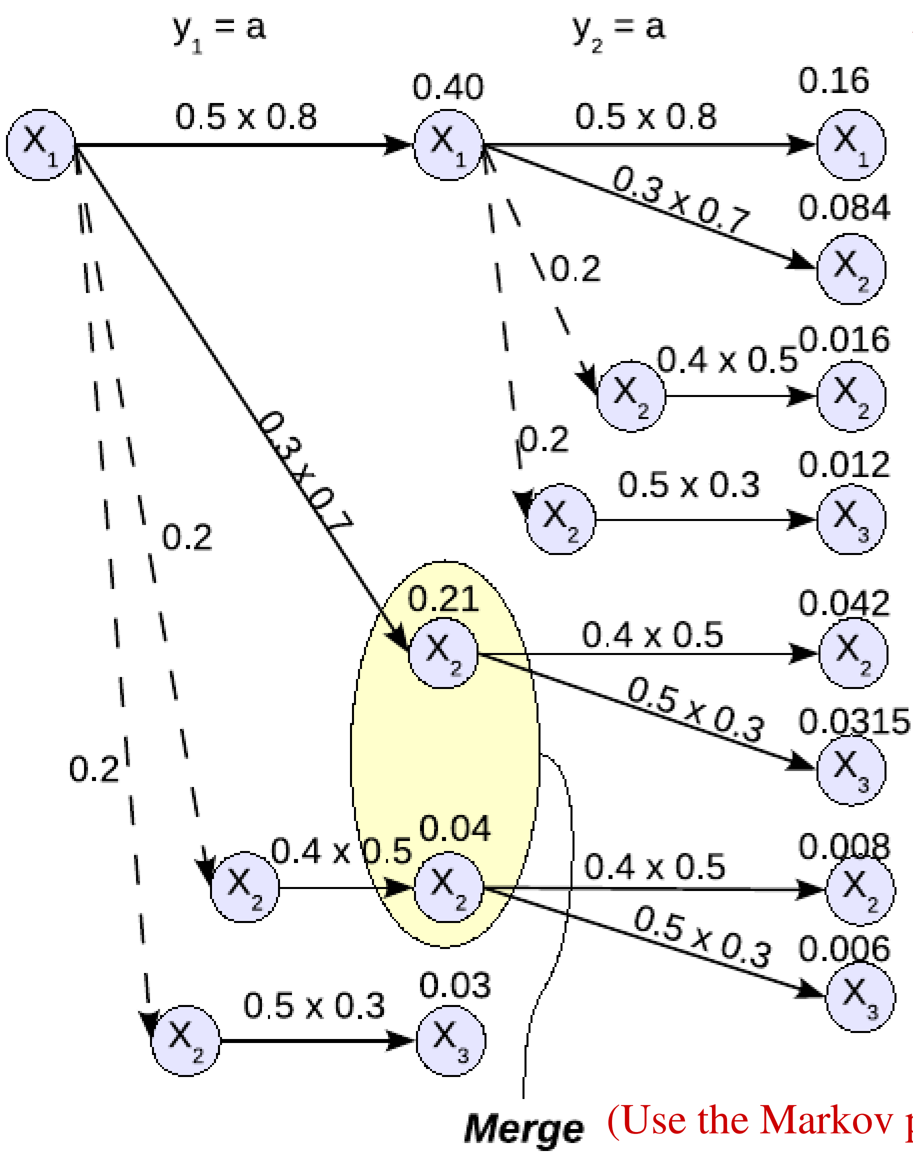
Possible Paths for generating $y_1 = a$ in Example 13.4

$$P(y_1 = a|\boldsymbol{\varphi}) = 0.40 + 0.21 + 0.04 + 0.03$$
$$= 0.68$$

Sum of all joint probabilities with the corresponding paths

Hidden Markov Models

Decoding

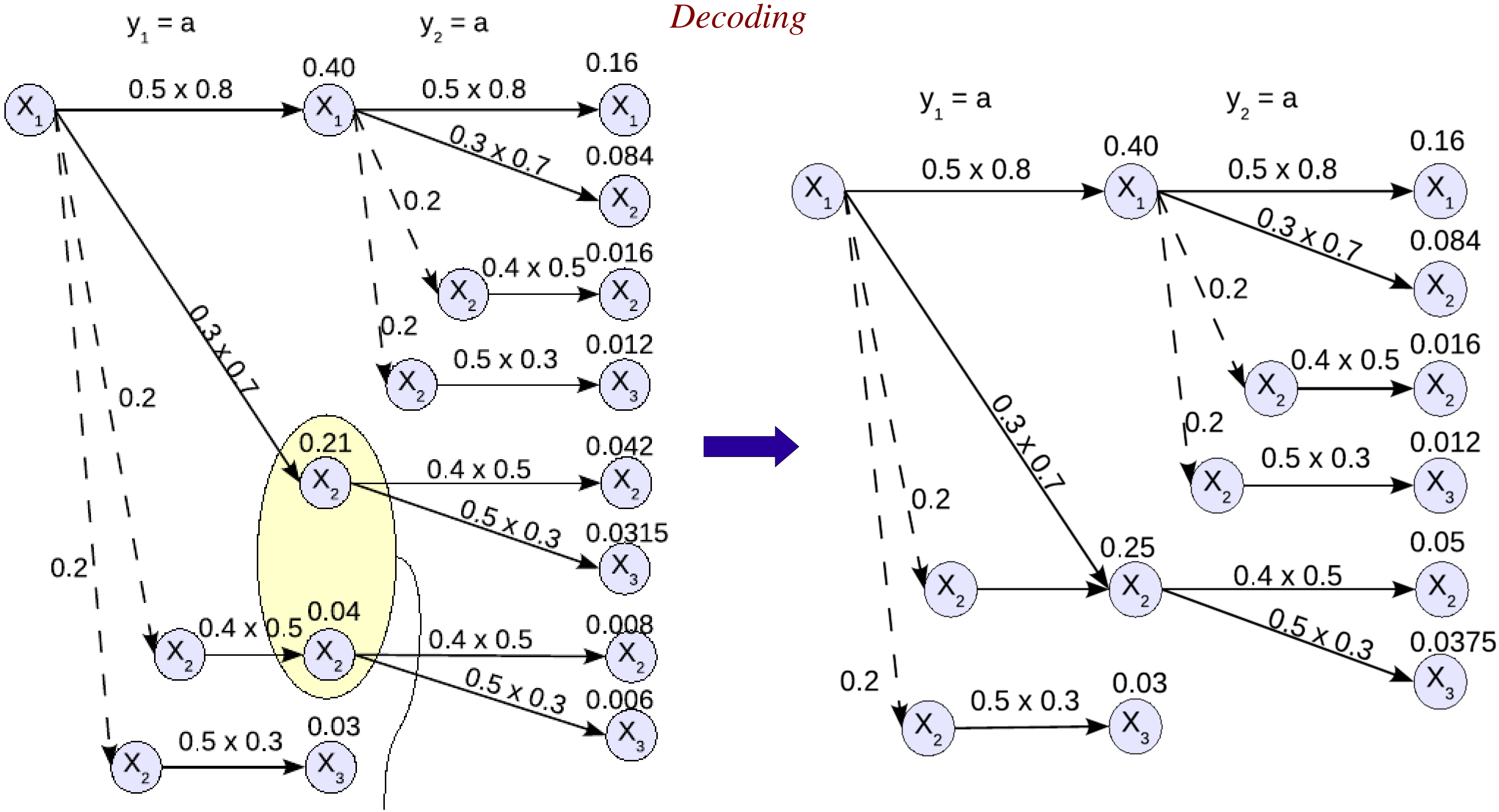


$$P(\{y\}_1^2 = \{aa\}|\boldsymbol{\varphi}) = 0.16 + 0.084 + 0.016 + 0.012 + 0.042 + 0.0315 + 0.008 + 0.006 = 0.3595$$

(Use the Markov property to merge paths)

Hidden Markov Models

Decoding

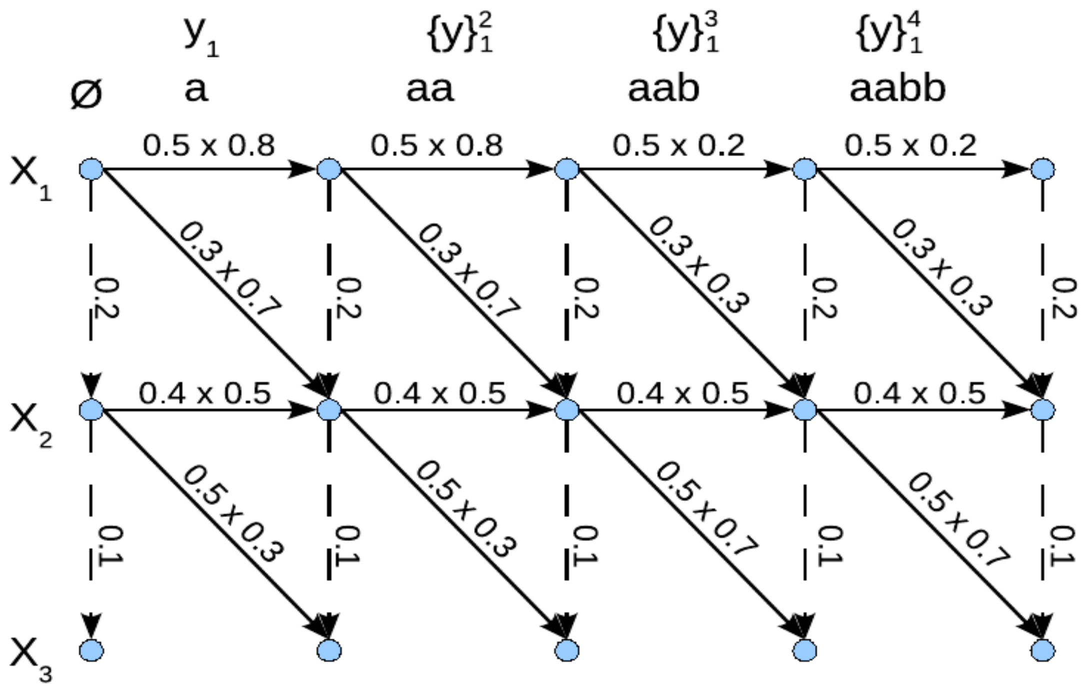


Merge (Use the Markov property to merge paths)

Hidden Markov Models

Decoding (Trellis Diagram)

1. A pure vertical descent in the diagram is equivalent to a *null transition*, since the state is changed, but no output is generated, hence no time has elapsed.
2. A pure horizontal motion corresponds to a self-transition since the state does not change, but an output is generated, elapsing one sample.
3. A diagonal motion is akin to moving along a forward transition from any state to the next state, while generating an output sample.



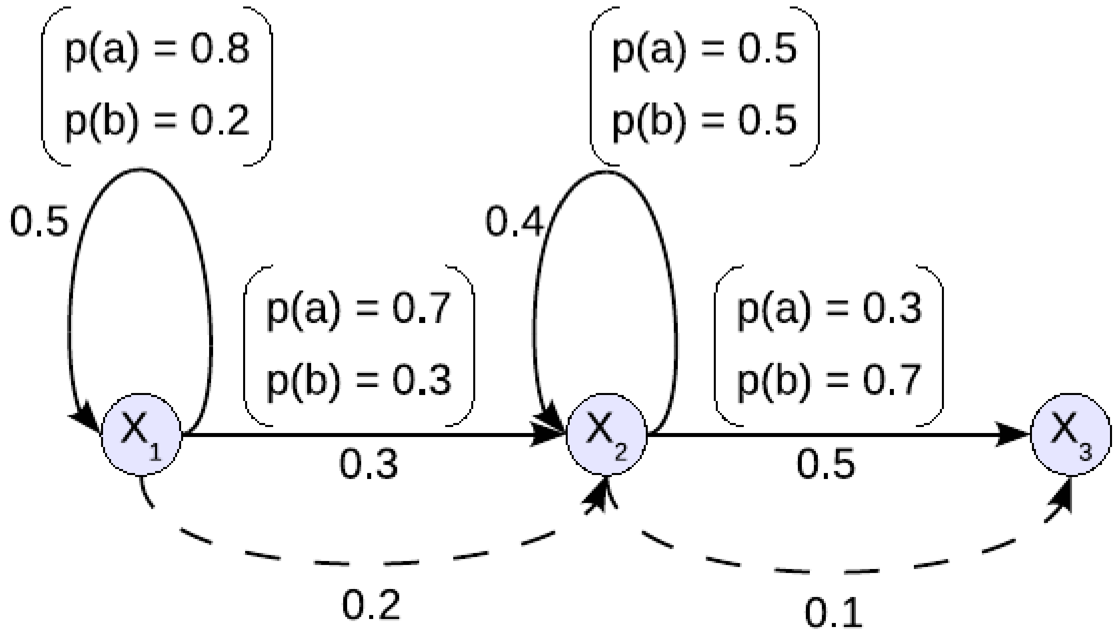
Trellis diagram for the output sequence,

$$\{y\}_1^4 = \{aabb\}$$

being generated by the HMM

Hidden Markov Models
Decoding

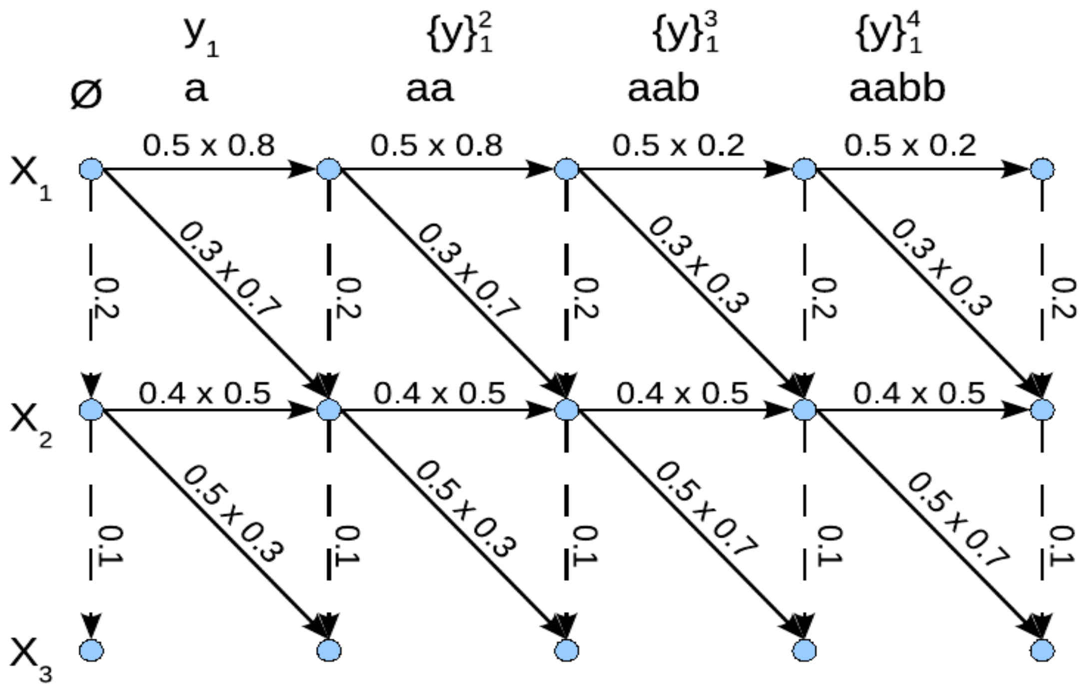
Example 13.4 (Decoding the Output Sequence).
Let us consider the HMM described in the finite state diagram of Figure 13.7 and compute the output probability of the sequence, $\{y\}_1^4 = \{aabb\}$ where $Y : y \in \{a,b\}$. First, consider all the different possible transitions which may output the first output in the sequence, $y_1 = a$. Figure 13.8 shows the different paths for generating $y_1 = a$, by the model in Figure 13.7.



Hidden Markov Models

Decoding (Trellis Diagram)

1. A pure vertical descent in the diagram is equivalent to a *null transition*, since the state is changed, but no output is generated, hence no time has elapsed.
2. A pure horizontal motion corresponds to a self-transition since the state does not change, but an output is generated, elapsing one sample.
3. A diagonal motion is akin to moving along a forward transition from any state to the next state, while generating an output sample.



Trellis diagram for the output sequence,

$$\{y\}_1^4 = \{aabb\}$$

being generated by the HMM

Hidden Markov Models

Forward Pass (match) Algorithm

$$\alpha_n(X_m|\boldsymbol{\varphi}) \stackrel{\Delta}{=} P(X_m, \{y\}_1^n|\boldsymbol{\varphi}) \;\; \forall \, X_m \in \mathscr{X}, \; 1 \leq n \leq N$$

Probability at each lattice point (node)

Simplification of the notation – it is known that it is a parametric model, so everything is conditioned upon the parameters:

$$P(X_m, \{y\}_1^n|\boldsymbol{\varphi}) \longrightarrow P(X_m, \{y\}_1^n)$$

$$\alpha_n(X_m|\boldsymbol{\varphi}) \longrightarrow \alpha_n(X_m)$$

$$P(y, \tau|\boldsymbol{\varphi}) \longrightarrow P(y, \tau)$$

Since null transitions are only allowed to go from one state to the next (left to right),

$\hat{\tau}_m \equiv \hat{\tau}_{m(m+1)}$ Null Transition

$\tau_{m(m+1)}$ Output Generating Transition

Hidden Markov Models

Forward Pass (match) Algorithm

$$\alpha_n(X_m|\boldsymbol{\varphi}) \overset{\Delta}{=} P(X_m, \{y\}_1^n|\boldsymbol{\varphi}) \;\; \forall\, X_m \in \mathscr{X},\; 1 \leq n \leq N$$

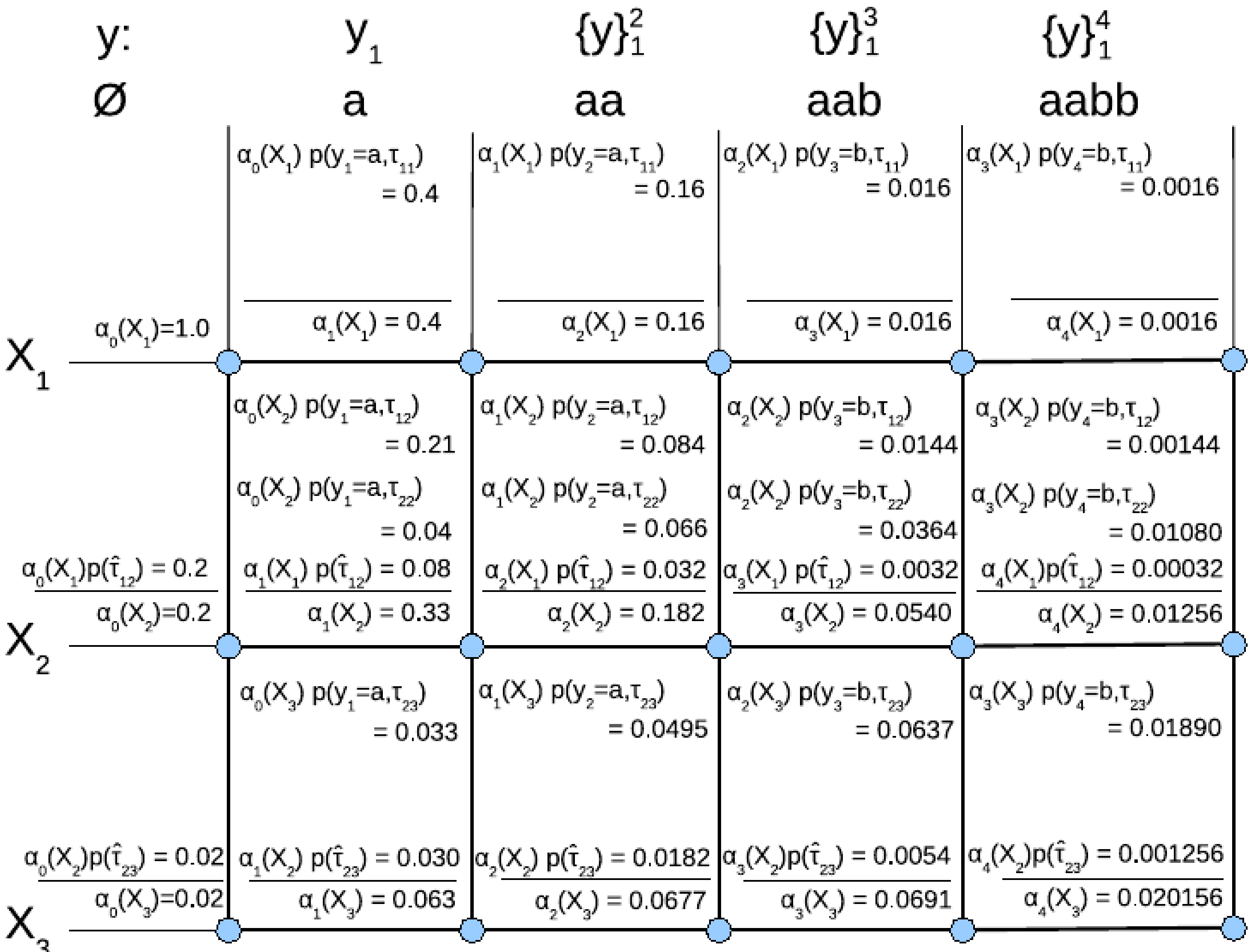
$$\begin{aligned}
\alpha_n(X_m) &= P(X_m, \{y\}_1^n) \\
&= \sum_{m' : m' \overset{\tau_{m'm}}{\longmapsto} m} P(X_{m'}, \{y\}_1^{n-1}) P(y_n, \tau_{m'm}) + \\
&\quad \sum_{m' : m' \overset{\hat{\tau}_{m'm}}{\longmapsto} m} P(X_{m'}, \{y\}_1^{n}) P(\hat{\tau}_{m'm}) \\
&= \sum_{m' : m' \overset{\tau_{m'm}}{\longmapsto} m} \alpha_{n-1}(X_{m'}) P(y_n|\tau_{m'm}) P(\tau_{m'm}) + \\
&\quad \sum_{m' : m' \overset{\hat{\tau}_{m'm}}{\longmapsto} m} \alpha_n(X_{m'}) P(\hat{\tau}_{m'm})
\end{aligned}$$

Recursive Equation with the following initial condition:

$$\alpha_0(X_1) = 1.0$$

Hidden Markov Models

Decoding – Forward Pass (Trellis Diagram)



Computation load is linearly proportional to the number of outputs versus being exponential when all possible paths are computed independently

α computation for the HMM of Example 13.4

Hidden Markov Models

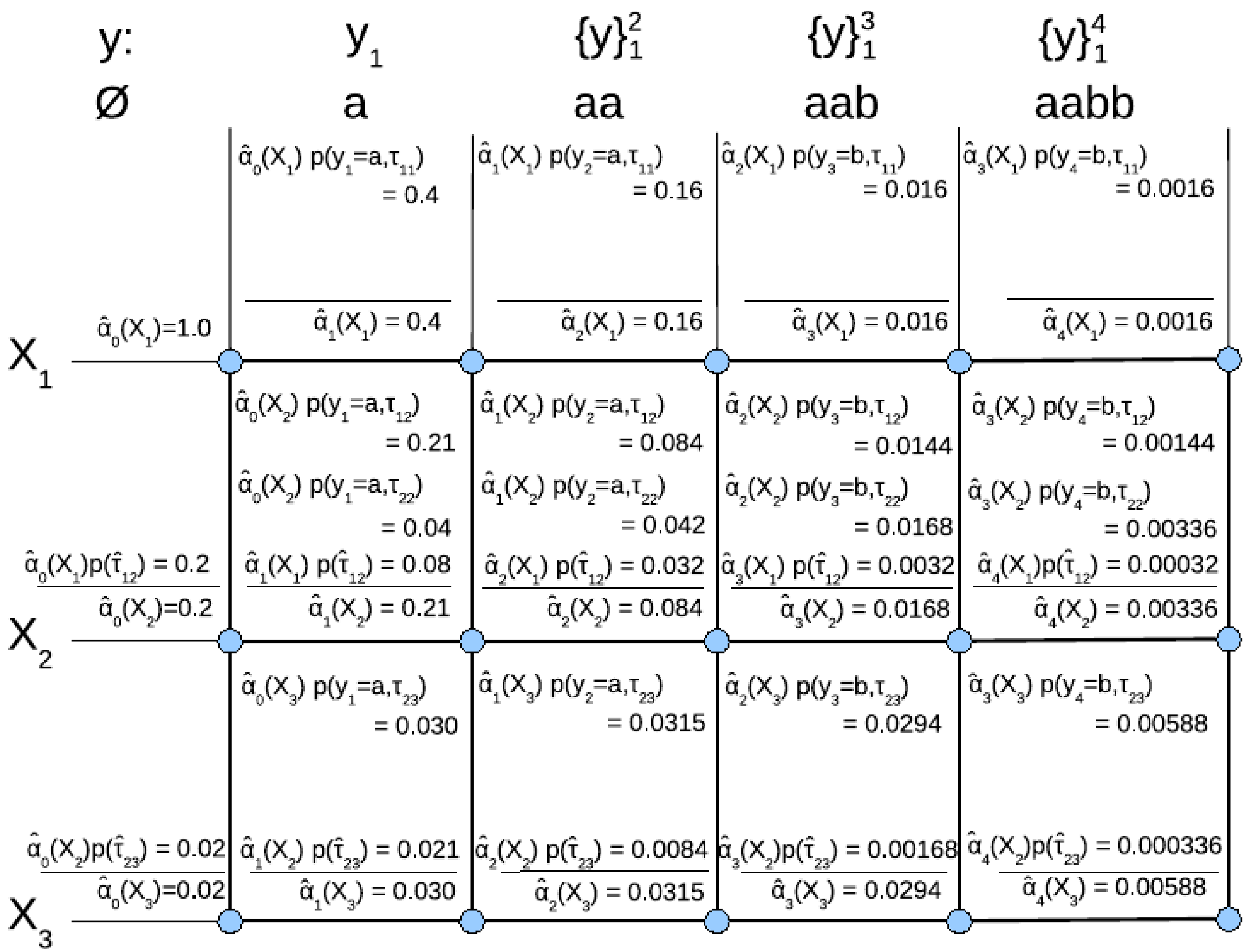
Best Path – Viterbi (Trellis Diagram)

$$
\begin{aligned}
\hat{\alpha}_n(X_m) &\stackrel{\Delta}{=} P_{max}(X_m, \{y\}_1^n) \\
&= \max\left(\max_{m' : m' \overset{\tau_{m'm}}{\longmapsto} m} \left(\hat{\alpha}_{n-1}(X_{m'}) P(y_n|\tau_{m'm}) P(\tau_{m'm})\right), \right. \\
&\qquad \left. \max_{m' : m' \overset{\hat{\tau}_{m'm}}{\longmapsto} m} \left(\hat{\alpha}_n(X_{m'}) P(\hat{\tau}_{m'm})\right) \right)
\end{aligned}
$$

Changing the Sum to Max produces the alphas for the Viterbi algorithm – finding the best path (problem 2)

Hidden Markov Models

Best Path – Viterbi Pass (Trellis Diagram)



α computation for the HMM of Example 13.4

Computation load is linearly proportional to the number of outputs versus being exponential when all possible paths are computed independently

Hidden Markov Models

Training – Baum-Welch (Forward-Backward) (Trellis Diagram)

Probability of transiting from a node in the trellis to a neighboring node while outputting the n^th output

$$P(y_n, \tau_{mm'} \mid w = \{y\}_1^N) = \underbrace{\alpha_{n-1}(X_{m'})}_{\text{up to the transition}} \quad \underbrace{P(y_n|\tau_{mm'})P(\tau_{mm'})}_{\text{the transition}} \quad \underbrace{\beta_n(X_m)}_{\text{to the end of output}}$$

$$P(\hat{\tau}_{mm'}, t = n|w = \{y\}_1^N) = \underbrace{\alpha_n(X_{m'})}_{\text{up to the transition}} \quad \underbrace{P(\tau_{mm'})}_{\text{the transition}} \quad \underbrace{\beta_n(X_m)}_{\text{to the end of output}}$$

$$\begin{aligned}\beta_n(X_m) &= P(\{y\}_{n+1}^N|X_m) \\ &= \sum_{m':m \xmapsto{\tau_{mm'}} m'} \beta_{n+1}(X_{m'})P(y_{n+1}|\tau_{mm'})P(\tau_{mm'}) + \\ &\quad \sum_{m':m \xmapsto{\hat{\tau}_{mm'}} m'} \beta_n(X_{m'})P(\hat{\tau}_{mm'})\end{aligned}$$

Hidden Markov Models

Training – Baum-Welch (Forward-Backward)

Transition Probabilities for a specific sequence w

$$P(\tau_{mm'} \mid w) = \frac{\sum_{n=1}^{N} P(y_n, \tau_{mm'}|w)}{\sum_{m':m \xrightarrow{\tau_{mm'}} m'} \sum_{n=1}^{N} P(y_n, \tau_{mm'}|w) + \sum_{m':m \xrightarrow{\hat{\tau}_{mm'}} m'} P(\hat{\tau}_{mm'}, t = n|w)}$$

$$P(\hat{\tau}_{m'm} \mid w) = \frac{\sum_{n=1}^{N} P(\hat{\tau}_{mm'}, t = n|w)}{\sum_{m':m \xrightarrow{\tau_{mm'}} m'} \sum_{n=1}^{N} P(y_n, \tau_{mm'}|w) + \sum_{m':m \xrightarrow{\hat{\tau}_{mm'}} m'} P(\hat{\tau}_{mm'}, t = n|w)}$$

In practice the summations in the numerators and denominators are accumulated into variables.

The trellis is swept from left to right at the end to compute the transition probabilities.

Hidden Markov Models

Training – Baum-Welch (Forward-Backward)
Output Probabilities for a specific sequence w

$$P(y_n = Y_q, \tau_{mm'} \mid w) = \underbrace{\alpha_{n-1}(X_{m'})}_{\text{up to the transition}} \quad \underbrace{P(y_n = Y_q | \tau_{mm'}) P(\tau_{mm'})}_{\text{the transition}} \quad \underbrace{\beta_n(X_m)}_{\text{to the end of output}}$$

Output probability written for a specific output $y = Y_q$

$$P(y = Y_q | \tau_{mm'}, w) = \frac{\sum_{n=1}^{N} P(y_n = Y_q, \tau_{mm'} | w)}{\sum_{n=1}^{N} P(y_n, \tau_{mm'} | w)}$$

Conditional Probability for the output generating transition producing the output is the sum of all the instances of this output having been generated by this transition.

Hidden Markov Models

Training – Baum-Welch (Forward-Backward)
Transition Probabilities over all training data (K Sequences)

$$P(\tau_{mm'}) = \sum_{\omega=1}^{\Omega} P(\tau_{mm'}, w = W_\omega)$$

$$= \sum_{\omega=1}^{\Omega} P(\tau_{mm'} | w = W_\omega) P(w = W_\omega)$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} P(\tau_{mm'} | w_k)$$

All sequences

Training sequences
(Law of large numbers)

$$P(\hat{\tau}_{mm'}) = \sum_{\omega=1}^{\Omega} P(\hat{\tau}_{mm'}, w = W_\omega)$$

$$= \sum_{\omega=1}^{\Omega} P(\hat{\tau}_{mm'} | w = W_\omega) P(w = W_\omega)$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} P(\hat{\tau}_{mm'} | w_k)$$

Hidden Markov Models

Training – Baum-Welch (Forward-Backward)
Output Probabilities over all training data (K Sequences)

$$P(y = Y_q | \tau_{mm'}) = \sum_{\omega=1}^{\Omega} P(y = Y_q | \tau_{mm'}, w = W_\omega) P(w = W_\omega)$$

All sequences

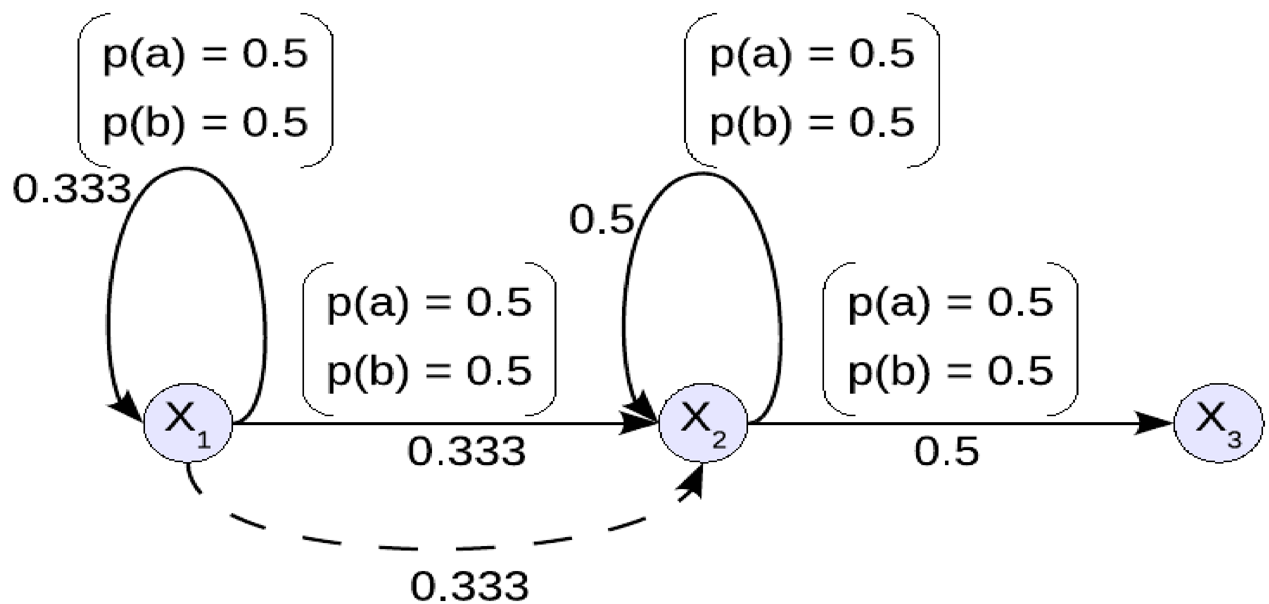
$$\approx \frac{1}{K} \sum_{k=1}^{K} P(y = Y_q | \tau_{mm'}, w_k)$$

Training sequences
(Law of large numbers)

Hidden Markov Models

Training – Baum-Welch (Example)

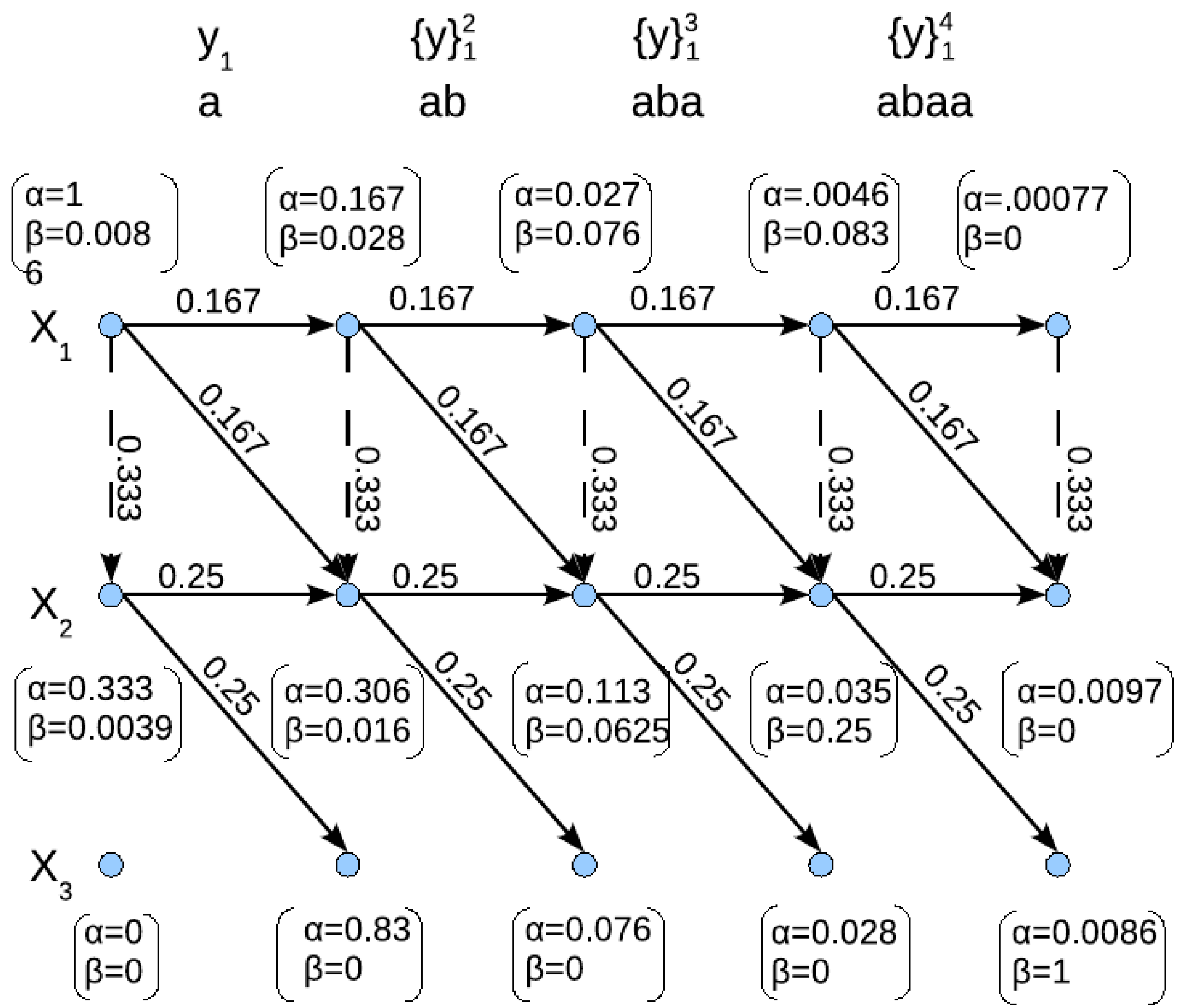
Example 13.5 (Simple Training).
To make the computations more manageable, let us start with a simplified version of the model in Example 13.4 which was used for the illustration of the decoding process. This time, we will remove the null transition from the second state to the third. Since we are still left with one null transition from the first state to the second state, we will not be impairing our coverage; we will only be simplifying the number of calculations that have to be done for the illustration of the Baum-Welch algorithm.



HMM of Example 13.5 with maximum entropy initial distributions

Hidden Markov Models

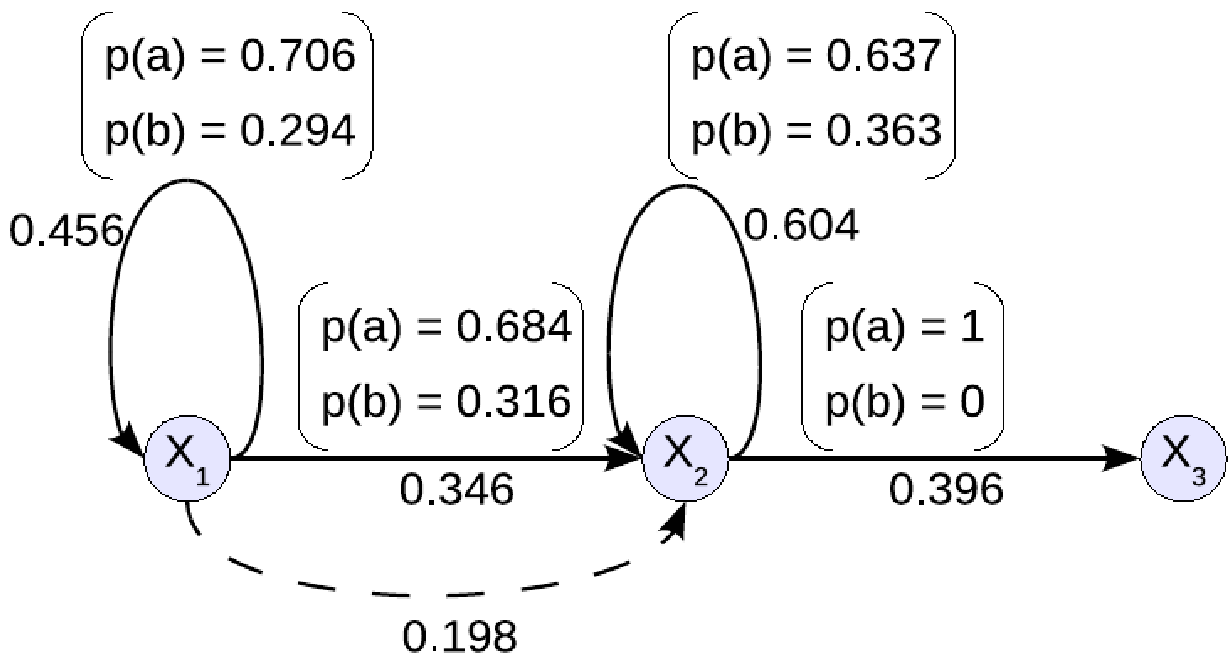
Training – Baum-Welch (Example – Trellis)



Trellis of Example 13.5 with maximum entropy initial distributions

Hidden Markov Models

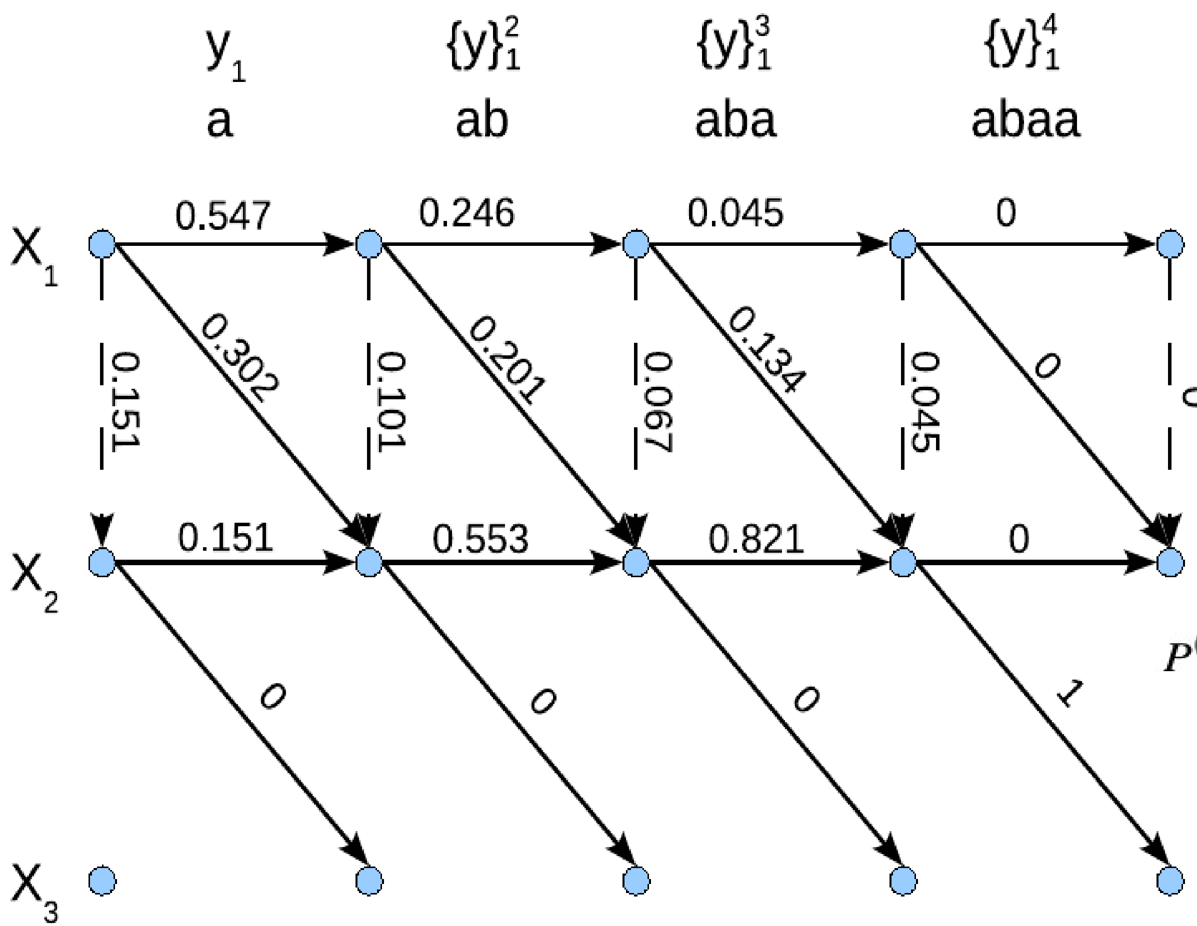
Training – Baum-Welch (Example – Trellis)



HMM of Example 13.5 with recomputed distributions and transition probabilities after one iteration of Baum-Welch

Hidden Markov Models

Training – Baum-Welch (Example – Trellis)



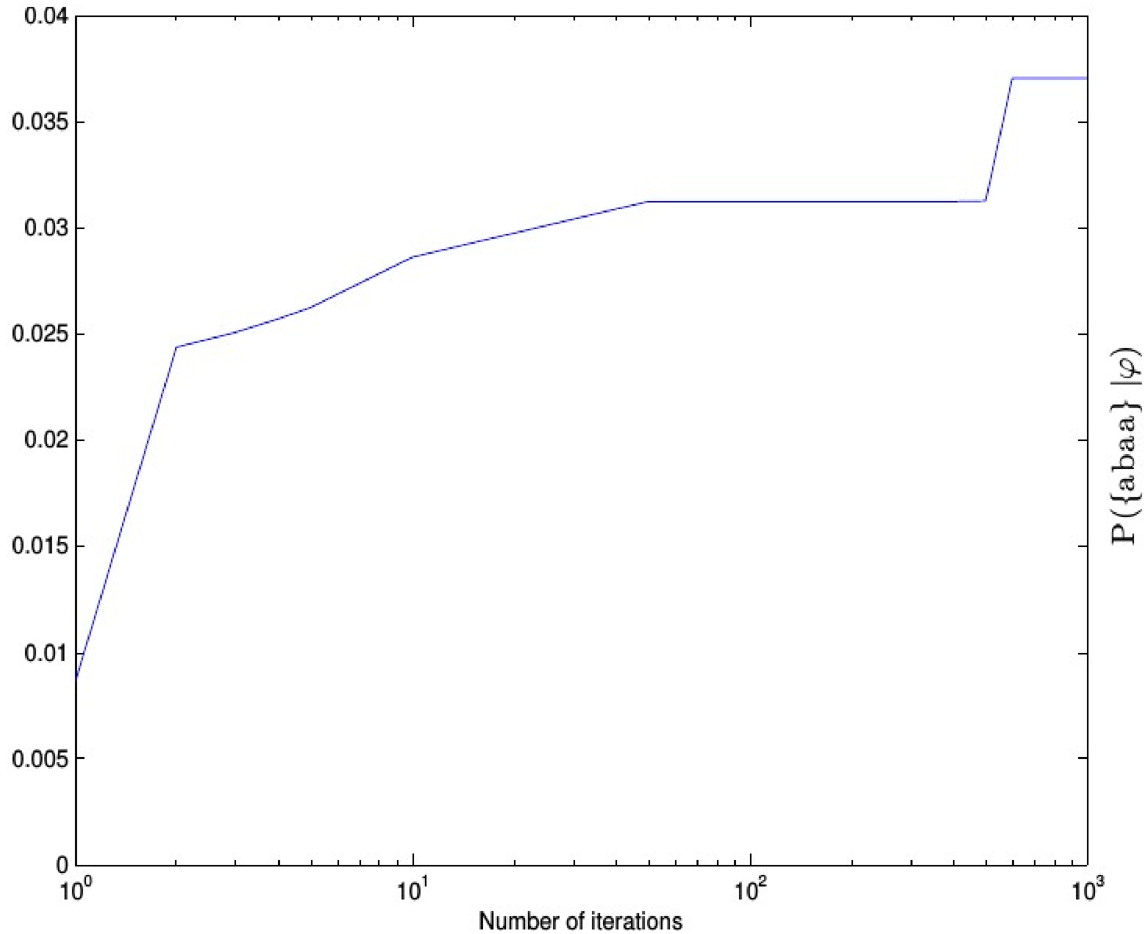
$$P^{(1)}(\{y\}_1^4 = \{abaa\}) = 0.02438$$
$$> P^{(0)}(\{y\}_1^4 = \{abaa\})$$
$$= 0.008632$$

Likelihood increases

Trellis of Example 13.5 with recomputed *a*-posteriori output-transition probabilities

Hidden Markov Models

Training – Baum-Welch (Example – Trellis)

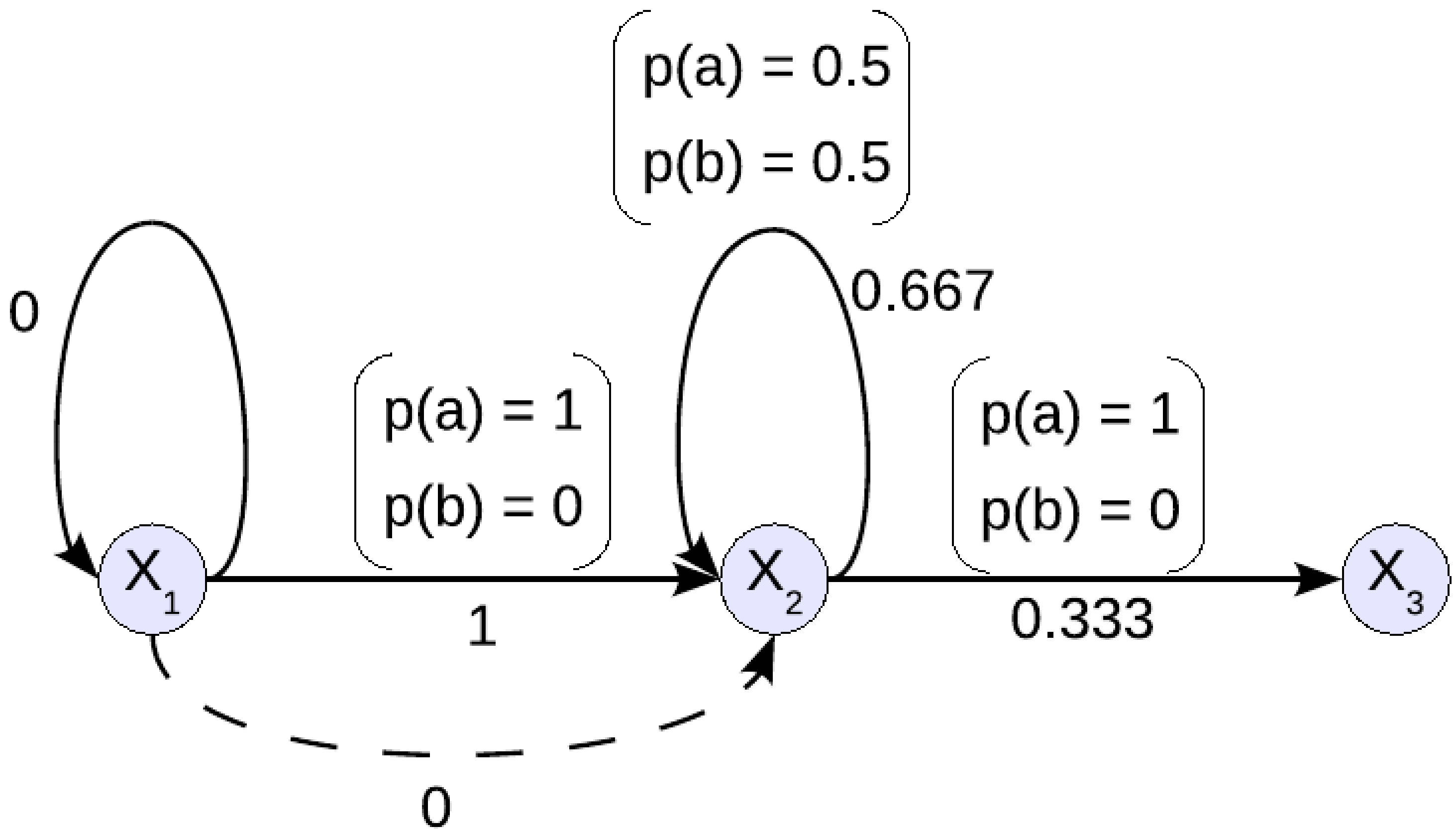


$$P^{(*)}(\{abaa\}|\boldsymbol{\varphi}) = 0.037\overline{037}$$

Convergence of the likelihood of the HMM given the sequence, $\{abaa\}$, to a local maximum, as related to Example 13.5

Hidden Markov Models

Training – Baum-Welch (Example – Local Maximum)



Configuration of the locally converged HMM model for sequence $\{abaa\}$ in Example 13.5

Hidden Markov Models

Training – Baum-Welch (Example – Global Maximum)

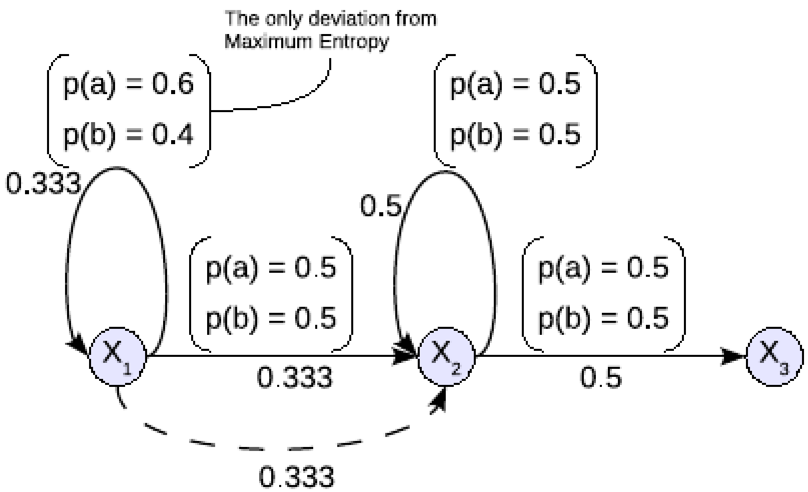


Fig. 13.20: A slight deviation from maximum entropy in the initial distribution of the HMM of Example 13.5

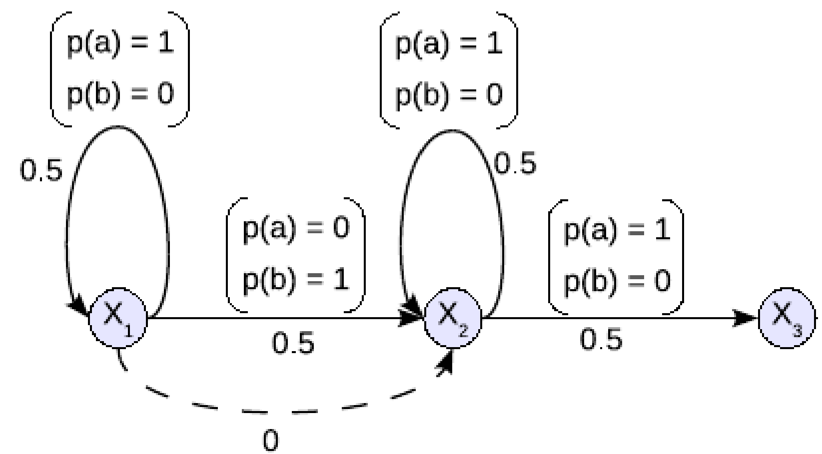


Fig. 13.21: Configuration of the globally converged HMM model for sequence $\{abaa\}$ in Example 13.5

$$P(\{abaa\}|\boldsymbol{\varphi}) = 0.0625$$

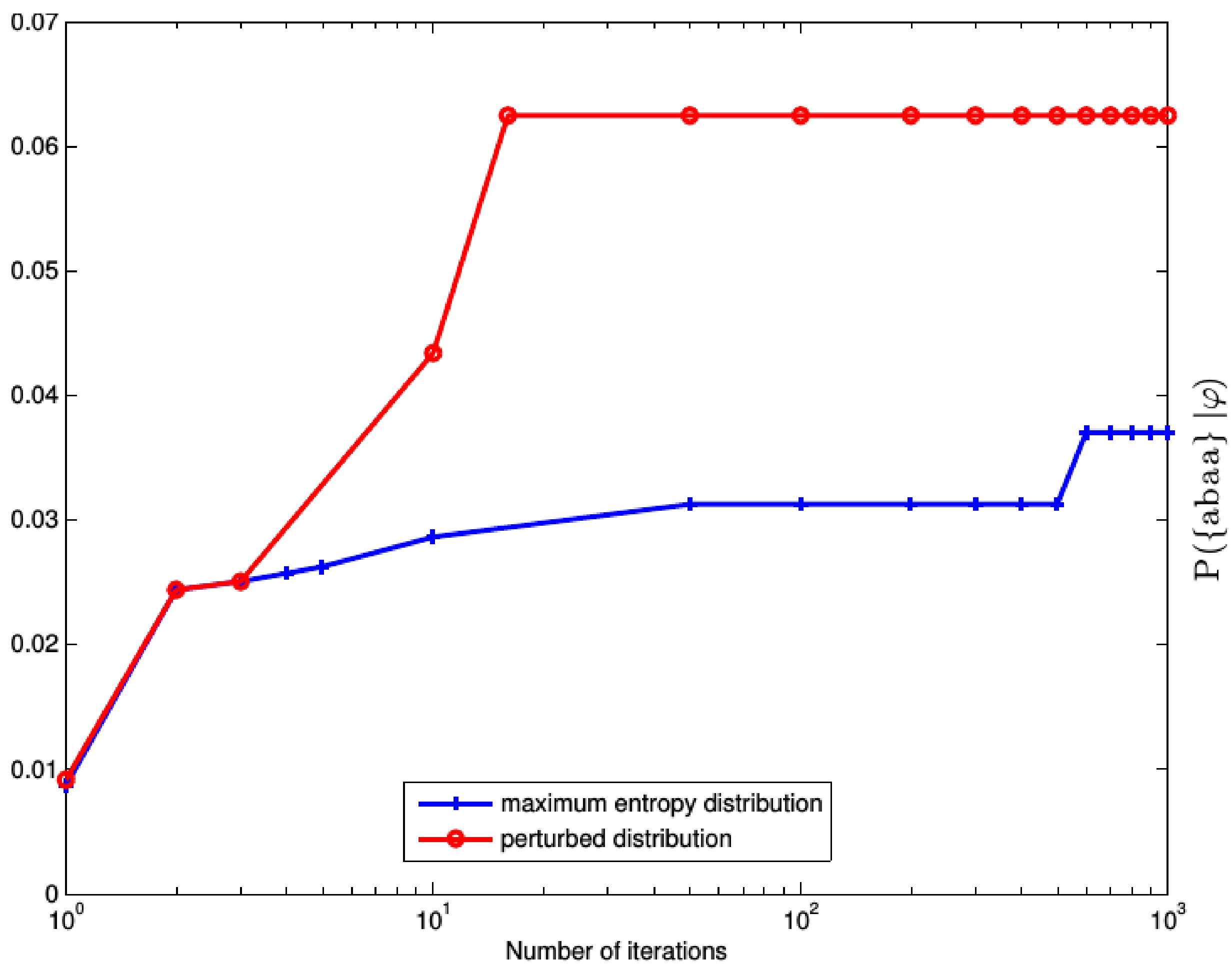
Globally maximum likelihood at 16 iterations

$$P^{(*)}(\{abaa\}|\boldsymbol{\varphi}) = 0.037\overline{037}$$

Local maximum likelihood at 600+ iterations

Hidden Markov Models

Training – Baum-Welch (Example – Global Maximum)



Convergence of the likelihood of the HMM given the sequence, $\{abaa\}$, for two different initial conditions: 1. maximum entropy and 2. slight perturbation

Estimation

Expectation-Maximization (EM)

Expectation Step:

$$\mathbf{v}^{(k)} = \mathscr{E}\left\{\mathbf{v}(\mathbf{x})|\mathbf{y}, \boldsymbol{\varphi}^{(k)}\right\}$$

Maximization Step:
Pick $\boldsymbol{\varphi}^{(k+1)}$ such that,

$$\mathscr{E}\{\mathbf{v}(\mathbf{x})|\boldsymbol{\varphi}\} = \mathbf{v}^{(k)}$$

Gaussian Mixture Model *(GMM)*

$$p(\mathbf{x}|\boldsymbol{\varphi}) = \sum_{\gamma=1}^{\Gamma} p(\mathbf{x}|\boldsymbol{\theta}_\gamma)P(\boldsymbol{\theta}_\gamma)$$

Γ mixture components: $\boldsymbol{\theta}_\gamma, \gamma \in \{1,2,\cdots,\Gamma\}$

mixture weights: $P(\theta = \boldsymbol{\theta}_\gamma), \gamma = \{1,2,\cdots,\Gamma-1\}$

$$\boldsymbol{\theta}_\gamma = \left[\boldsymbol{\mu}_\gamma^T \quad \mathbf{u}^T(\boldsymbol{\Sigma}_\gamma)\right]^T$$

$\mathbf{u}(\boldsymbol{\Sigma}_\gamma)$ is the invertible unique parameter vector function. It converts unique parameters in the covariance matrix to a vector.

$$\boldsymbol{\Sigma}_\gamma = \mathbf{u}_\gamma{}^{-1}$$

$$\left(\mathbf{u}(\boldsymbol{\Sigma}_\gamma)\right)_{[d]} \stackrel{\Delta}{=} \left(\boldsymbol{\Sigma}_\gamma\right)_{[d][d]} \; \forall\, d \in \{1,2,\cdots,D\}$$ (For the diagonal covariance)

Parameter vector:

$$\boldsymbol{\varphi} \stackrel{\Delta}{=} \left[\boldsymbol{\mu}_1^T \cdots \boldsymbol{\mu}_\Gamma^T \;\; \mathbf{u}_1^T \cdots \mathbf{u}_\Gamma^T \;\; P(\boldsymbol{\theta}_1) \cdots P(\boldsymbol{\theta}_{\Gamma-1})\right]^T$$

Only $\Gamma - 1$ mixture weights are included since,

$$\sum_{\gamma=1}^{\Gamma} P(\boldsymbol{\varphi}_\gamma) = 1$$

Gaussian Mixture Model *(GMM)*

maximize,

$$\ell(\boldsymbol{\varphi}|\{\mathbf{x}\}_1^N) = \ln\left(\prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\varphi})\right)$$

$$= \sum_{n=1}^{N} \ln p(\mathbf{x}_n|\boldsymbol{\varphi})$$

$$= \sum_{n=1}^{N} \ln\left(\sum_{\gamma=1}^{\Gamma} p(\mathbf{x}_n|\boldsymbol{\theta}_\gamma)P(\boldsymbol{\theta}_\gamma)\right)$$

Use Expectation Maximization (EM)

Gaussian Mixture Model *(GMM)*

Training

$$\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda) = \sum_{n=1}^{N}\sum_{\gamma=1}^{\Gamma} p(\boldsymbol{\theta}_{\gamma}{}^{(k)}|\mathbf{x}_n) \left(-\frac{D}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\boldsymbol{\Sigma}_{\gamma}\right| - \frac{1}{2}(\mathbf{x}_n - \boldsymbol{\mu}_{\gamma})^T\boldsymbol{\Sigma}_{\gamma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_{\gamma}) + \ln\left(P(\boldsymbol{\theta}_{\gamma})\right)\right)$$
$$-\lambda\left(\sum_{\gamma} P(\boldsymbol{\theta}_{\gamma}) - 1\right)$$

$$\nabla_{\tilde{\boldsymbol{\varphi}}}\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda) = \mathbf{0}$$

Break up the problem, $\gamma \in \{1, 2, \cdots, \Gamma\}$

$$\nabla_{\boldsymbol{\mu}_{\gamma}}\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda) = \mathbf{0}$$
$$\frac{\partial\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda)}{\partial\boldsymbol{\Sigma}_{\gamma}} = \mathbf{0} \qquad \left(\frac{\partial\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda)}{\partial\boldsymbol{\Sigma}_{\gamma}}\right)_{[i][j]} \triangleq \frac{\partial\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda)}{\partial\left(\boldsymbol{\Sigma}_{\gamma}\right)_{[i][j]}} \;\forall\, i, j \in \{1, 2, \cdots, D\}$$
$$\frac{\partial\mathscr{L}(\tilde{\boldsymbol{\varphi}}, \lambda)}{P(\boldsymbol{\theta}_{\gamma})} = \mathbf{0}$$

In splitting the problem, we may speed up convergence by using $\boldsymbol{\mu}_{\gamma}{}^{(k+1)}$ from the first problem in the second and third problems.

Gaussian Mixture Model *(GMM)*

Training Summary

Expectation Step,

$$p(\boldsymbol{\theta}_\gamma|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|\boldsymbol{\theta}_\gamma)P(\boldsymbol{\theta}_\gamma)}{\sum_{\gamma'=1}^{\Gamma} p(\mathbf{x}_n|\boldsymbol{\theta}_{\gamma'})P(\boldsymbol{\theta}_{\gamma'})}$$

Maximization Step,

$$\boldsymbol{\mu}_\gamma{}^{(k+1)} = \frac{\sum_{n=1}^{N} p(\boldsymbol{\theta}_\gamma{}^{(k)}|\mathbf{x}_n)\mathbf{x}_n}{\sum_{n=1}^{N} p(\boldsymbol{\theta}_\gamma{}^{(k)}|\mathbf{x}_n)}$$

$$\boldsymbol{\Sigma}_\gamma{}^{(k+1)} = \frac{\sum_{n=1}^{N} p(\boldsymbol{\theta}_\gamma{}^{(k)}|\mathbf{x}_n)(\mathbf{x}_n - \boldsymbol{\mu}_\gamma{}^{(k+1)})(\mathbf{x}_n - \boldsymbol{\mu}_\gamma{}^{(k+1)})^T}{\sum_{n=1}^{N} p(\boldsymbol{\theta}_\gamma{}^{(k)}|\mathbf{x}_n)}$$

$$P(\boldsymbol{\theta}_\gamma{}^{(k+1)}) = \frac{1}{N}\sum_{n=1}^{N} p(\boldsymbol{\theta}_\gamma{}^{(k)}|\mathbf{x}_n)$$

Gaussian Mixture Model *(GMM)*

Intractability

$$\dim(\boldsymbol{\varphi}) = \Gamma\left(\underbrace{D}_{\boldsymbol{\mu}_\gamma} + \underbrace{\frac{D(D+1)}{2}}_{\boldsymbol{\Sigma}_\gamma} + \underbrace{1}_{P(\boldsymbol{\theta}_\gamma)}\right) \underbrace{-1}_{\sum_\gamma P(\boldsymbol{\theta}_\gamma)=1}$$
$$= \frac{\Gamma}{2}\left(D^2 + 3D + 2\right) - 1$$

If D=45 and $\Gamma = 256$

$$\begin{aligned}\dim(\boldsymbol{\varphi}) &= \frac{\Gamma}{2}\left(D^2 + 3D + 2\right) - 1\\ &= \frac{256}{2}\left(45^2 + 3 \times 45 + 2\right) - 1\\ &= 276,735\end{aligned}$$

Requiring 10 samples per parameter would ask for 2, 767, 350 frames of speech or 561 minutes!

Gaussian Mixture Model *(GMM)*

Intractability

If we use a diagonal Covariance (variances)

$$\begin{aligned}\dim(\boldsymbol{\varphi}) &= \frac{\Gamma}{2}(2D+1)-1\\ &= \frac{256}{2}(2\times 45+1)-1\\ &= 11,647\end{aligned}$$

- A 95.8% reduction in the number of parameters that need to be estimated!

- The 10 sample requirement only calls for less than 2 minutes of audio.

- Other techniques for reducing the parameter space are transformations such as PCA, LDA, and FA.

- Joint Factor Analysis (for GMM & SVM)
- Nuisance Attribute Projection (for SVM)

Support Vector Machine Modeling

- Decision Functions
  - Kernels
  - Direct Decision Functions
  - Indirect Decision Functions
- 2-Class Problem
  - Linearly Separable
  - Linearly Inseparable
- N-Class Problem
  - Unclassifiable Regions
  - Batch Systems
  - Decision Trees
  - Fuzzification
- Variations
  - Least Squares
  - Linear Programming
  - Robust
- Training and Optimization
- Expedited Training
  - Problem Decomposition
  - Data Preselection

Neural Network Modeling

- **The Perceptron**
- **Feedforward Networks**
  - **Auto Associative Neural Networks (AANN)**
  - **Training (Learning)**
    - **Global Solution** – *Simulated Annealing*
- **Recurrent Neural Networks (RNN)**
  - **Long Short-Term Memory (LSTM)**
- **Time-Delay Neural Networks (TDNN)**
- **Hierarchical Mixtures of Experts (HME)**
- **Convolutional Neural Networks (CNN)**
- **Parameter Estimation**
- **Practical Issues**
  - **Over-Training**
  - **Local Traps**

Decision Trees

(Tree construction – binary trees)

1. Establish the best question (the one that maximizes the likelihood of the criterion of interest) for partitioning the data into two equivalence classes and keep doing this on each of the emerging classes until the stopping criterion in the next step is met.

2. Stop when there is either insufficient data with which to proceed or when the best question is not sufficiently informative.

This is a greedy process – local optimality does not necessarily lead to global optimality.

The globally optimal tree construction is NP-complete!

NP-complete means a problem that requires a nondeterministic polynomial time or in other words, it is combinatorially intractable (or it is not practical).

Decision Trees
(Types of questions)

Definition 9.11 (Fixed Question). *A fixed question refers to the selection of a question from a collection of predefined questions that would optimize the objective function of choice for building the tree.*

- Is the phone a stop ($x \in \{/p/, /t/, /k/, /b/, /d/, /g/\}$)?
- Is the phone an unvoiced stop ($x \in \{/p/, /t/, /k/\}$)?
- Is the phone a voiced stop ($x \in \{/b/, /d/, /g/\}$)?
- Is the phone /p/?
- Is the phone /t/?
- Is the phone /k/?
- Is the phone /b/?
- Is the phone /d/?
- Is the phone /g/?

Definition 9.12 (Dynamic Question). *A dynamic question refers to the generation of a question dynamically as the tree nodes are traversed. A search problem is solved to create the question that optimizes the equivalence classes given the training data.*

Problems:

- Increased complexity

- Can lead to overtraining

Decision Trees

Maximum Likelihood Estimation (MLE)

$$
\begin{aligned}
a_i &\triangleq i^{th}\text{Unique outcome}, i \in \{1,2,\cdots,N\} \\
c_i &\triangleq i^{th}\text{Frequency of occurrence of } a_i \\
P_i &\triangleq i^{th} p(X|a_i) \\
Q &\triangleq \text{A question that partitions the sample into two classes} \\
n_l &\triangleq \text{Number of samples that fall into the class on the left} \\
n_r &\triangleq \text{Number of samples that fall into the class on the right} \\
{}_l c_i &\triangleq \text{Frequency of occurrence of the outcome } a_i \text{ in the left partition} \\
{}_r c_i &\triangleq \text{Frequency of occurrence of the outcome } a_i \text{ in the right partition} \\
{}_l P_i &\triangleq i^{th}\text{Probability of the outcome } a_i \text{ in the left partition} \\
{}_r P_i &\triangleq i^{th}\text{Probability of the outcome } a_i \text{ in the right partition} \\
x_j^K &\triangleq \text{The sample sequence, } x_j x_{j+1} \cdots x_{k-1} x_k
\end{aligned}
$$

Consider an i.i.d. sequence $\{x\}_1^n$

(Independent and Identically Distributed)

$$
\begin{aligned}
\mathscr{L}(Q|\{x\}_1^n) &= p(\{x\}_1^n|Q) \\
&= \prod_{j=1}^{n} p(X = x_j|Q) \\
&= \prod_{j=1}^{n} \mathscr{L}(Q|X = x_j)
\end{aligned}
$$

Decision Trees

Maximum Likelihood Estimation (MLE)

Joint likelihood of the whole sample,

$$\mathscr{L}(Q|\{x\}_1^n) = \prod_{i=1}^{N} P_i^{c_i}$$

Using the maximum likelihood estimate of P_i we have

$$\begin{aligned}\ell(Q|\{x\}_1^n) &= \sum_{i=1}^{N} c_i \log_2 \hat{P}_i \\ &= \sum_{i=1}^{N} c_i \log_2 \frac{c_i}{n}\end{aligned}$$

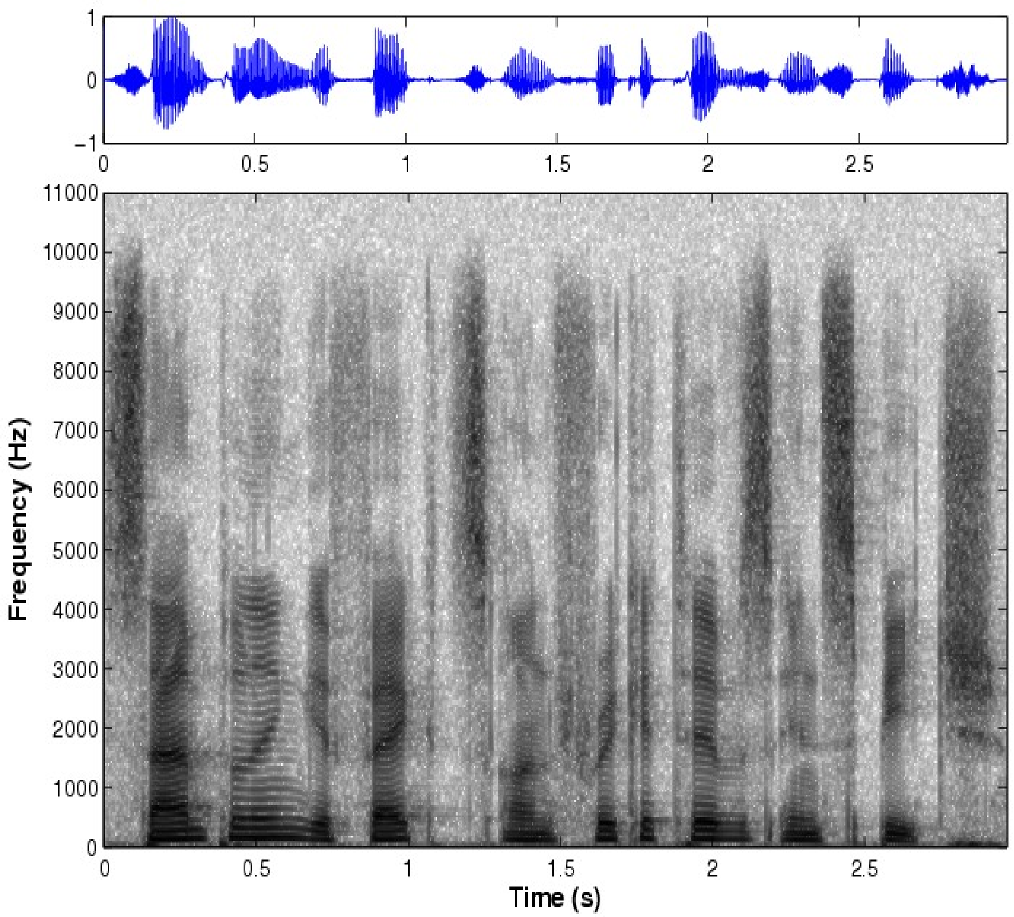
Premultiplying by $-\frac{1}{n}$ gives us a measure of entropy

$$\begin{aligned}-\frac{1}{n}\ell(Q|\{x\}_1^n) &= -\sum_{i=1}^{N} \frac{c_i}{n} \log_2 \frac{c_i}{n} \\ &= -\sum_{i=1}^{N} \hat{P}_i \log_2 \hat{P}_i \\ &= \mathscr{H}\left(\hat{P}\right)\end{aligned}$$

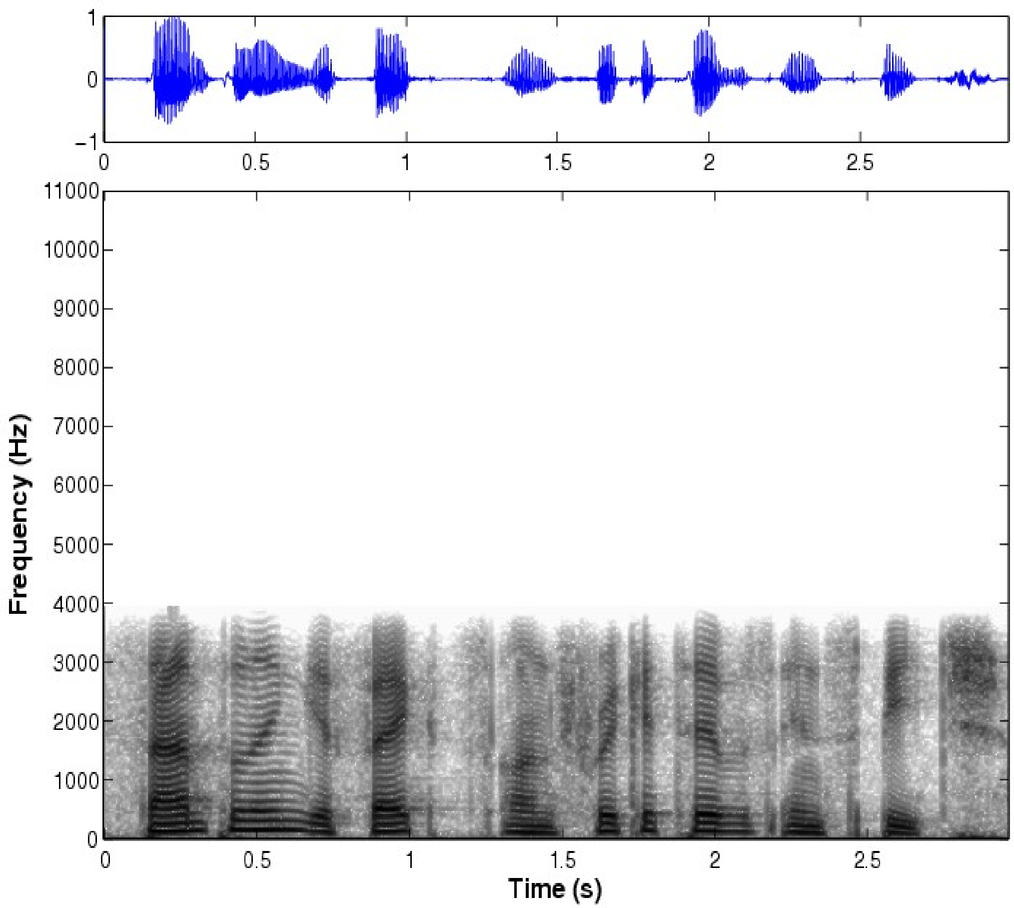
Maximum likelihood is equivalent to minimum Entropy

Channel Mismatch -- Capacity
Band Limitation – Telephony (Landline)

22kHz Sampling Rate



8kHz Sampling Rate



(*from Beigi-2011*)

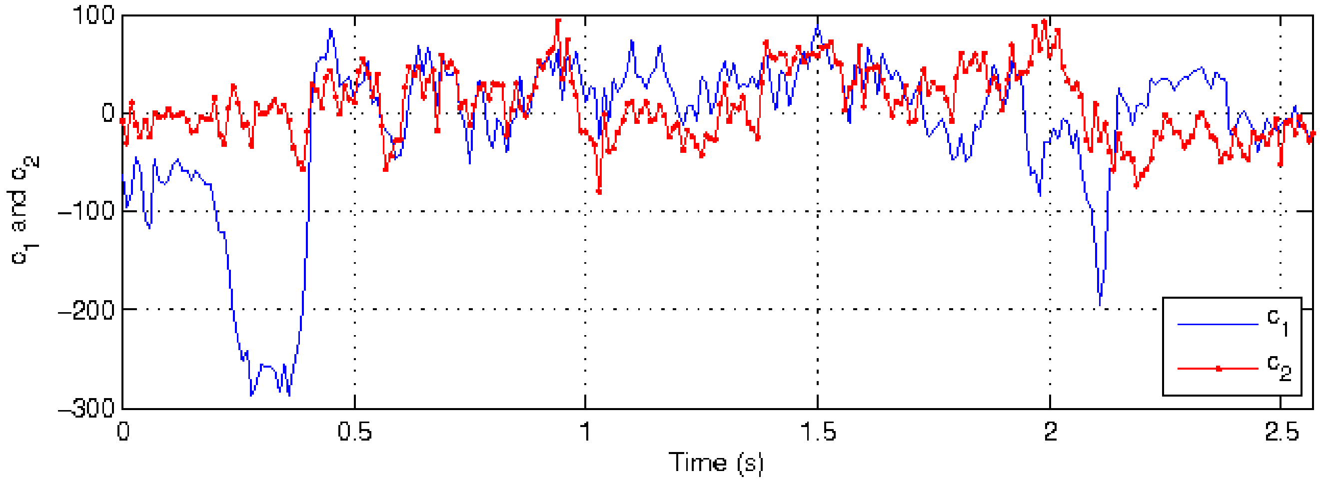
Practical Issues

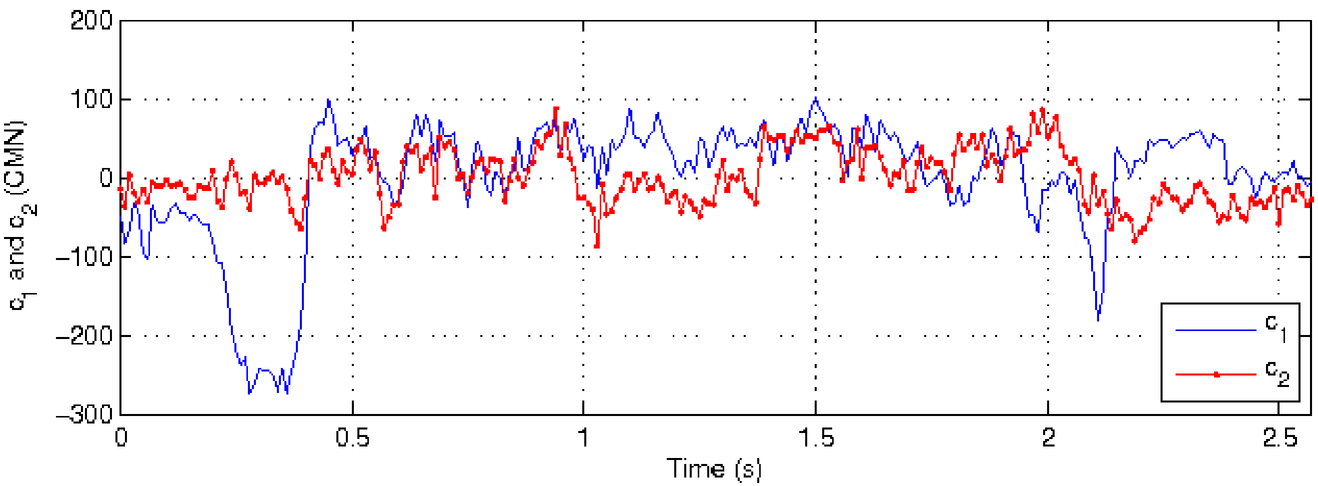
- Channel Mismatch Effects
- Numerical Stability
- Standards
- Data Quality and Quantity
- Speaker Independent Recognition
- Large Vocabulary Unconstrained Recognition

Channel Mismatch -- Handling
Spectral Filtering and Cepstral Liftering

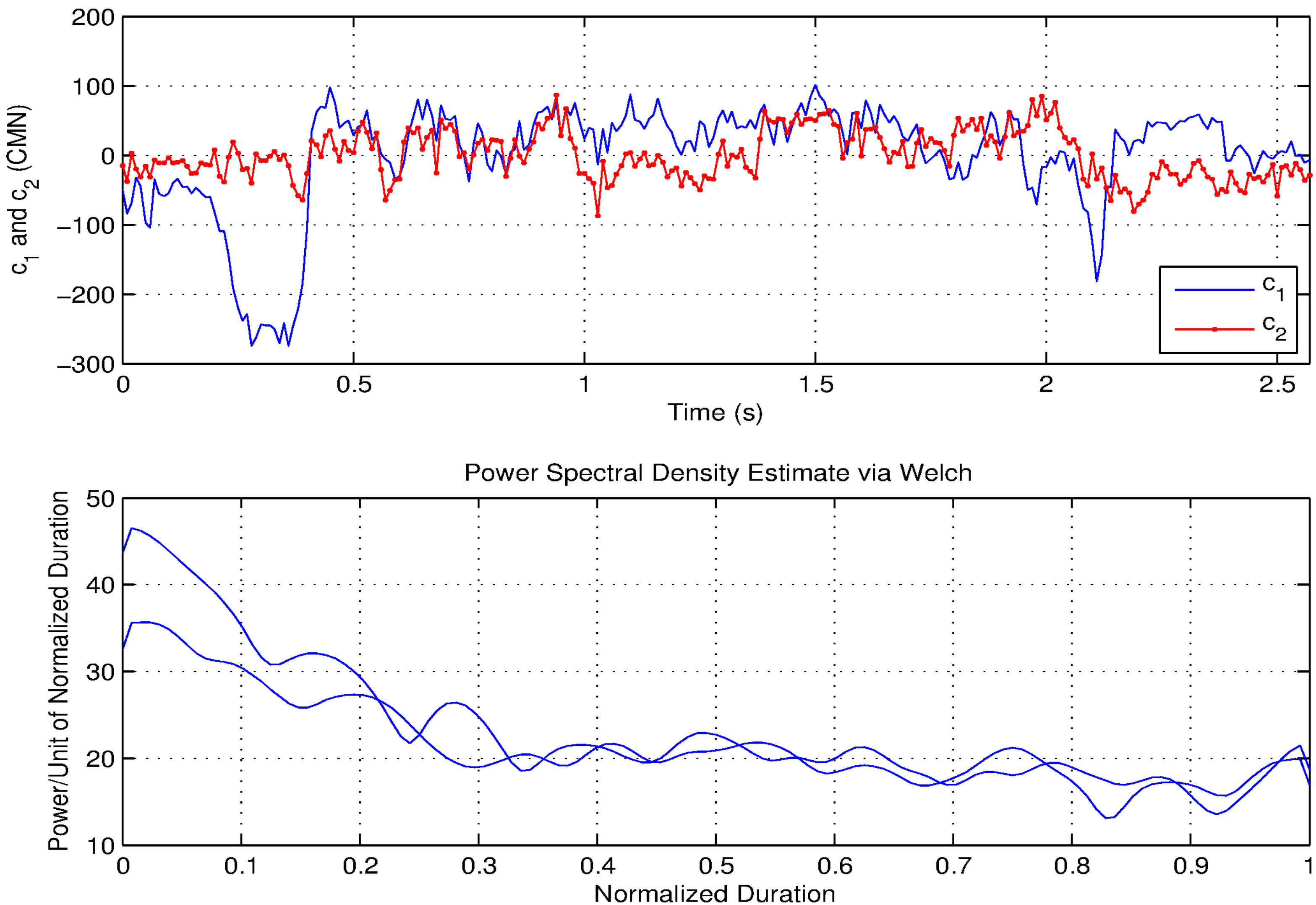
- Cepstral Mean Subtraction (CMS) – aka, Cepstral Mean Normalization (CMN)
- Cepstral Mean and Variance Normalization (CMVN)
- Histogram Equalization (HEQ)
- Cepstral Histogram Normalization (CHN)
- Auto Regressive Moving Average (ARMA)
- RelAtive SpecTrAl (RASTA) Filtering – reduces accuracy in the absence of noise
- J-RASTA (uses an adaptive J factor
  problem: both RASTA and J-RASTA remove slow moving characteristics)
- Kalman Filtering

Channel Mismatch
CMS (CMN)





Channel Mismatch
CMN



Speech Decoding Process

Decoding (Bayes Rule):

Acoustic Model

Language Model

Argmax of P(W|f) is the Search Hypothesis

Model Parametes

$$P(W|f,\Phi) = \frac{p(f|W,\Phi)\ P(W|\ \Phi\ )}{p(f)}$$

Acoustic sequence

$$W^* = \underset{W \in \mathrm{Wordlist}}{\operatorname{argmax}}\ P(W|f,\Phi)$$

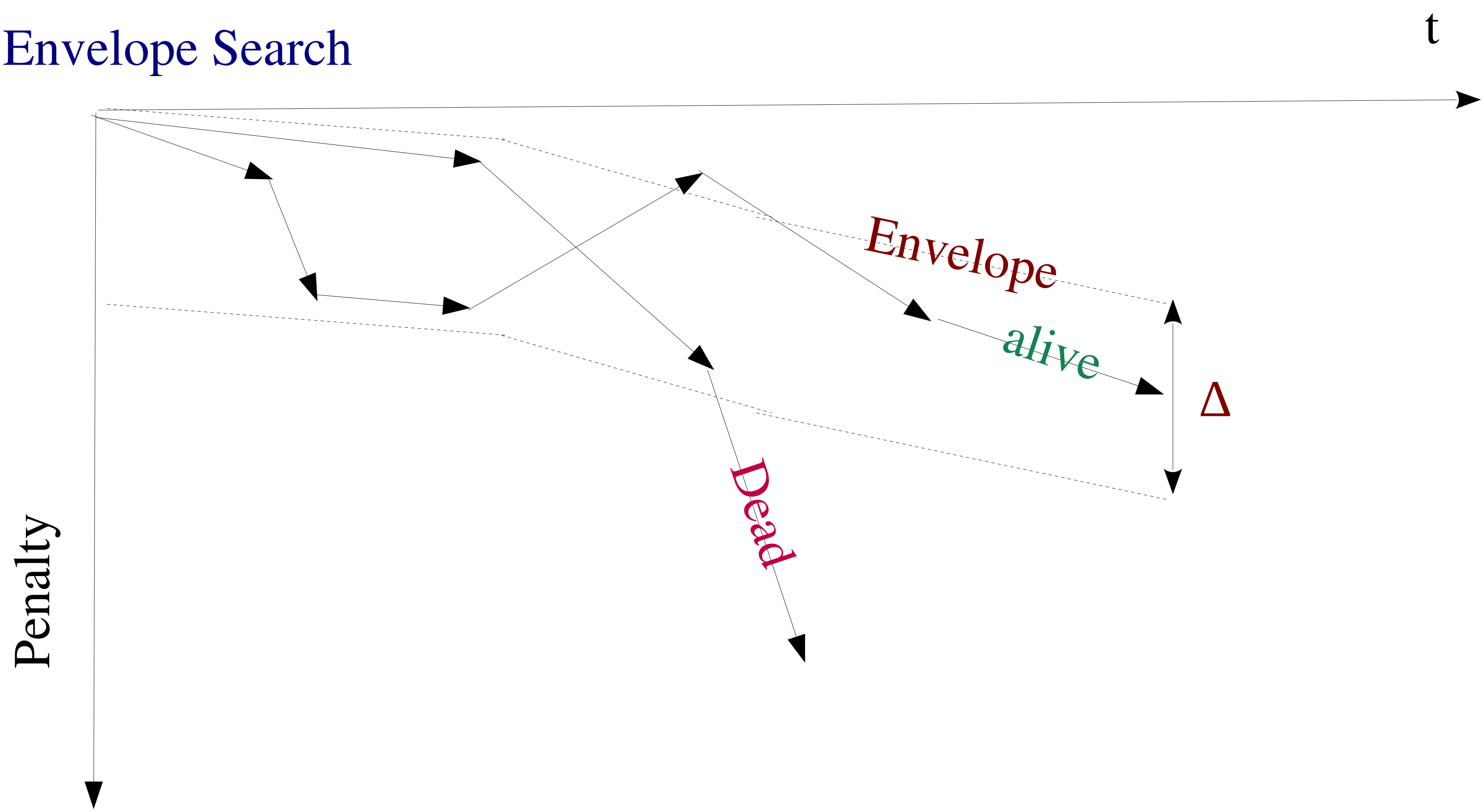
Info@RecoTechnologies.com

Suboptimal Search

A* Search

Beam Search

Envelope Search



N-Gram Language Models

- Estimate probability of a sequence of words, $w_1, w_2, \ldots, w_n$
- Using Chain Rule,

$$P(w_1,w_2,\ldots) = P(w_1)\, P(w_2|w_1)\, P(w_3\, |w_1,w_2)\, \ldots$$

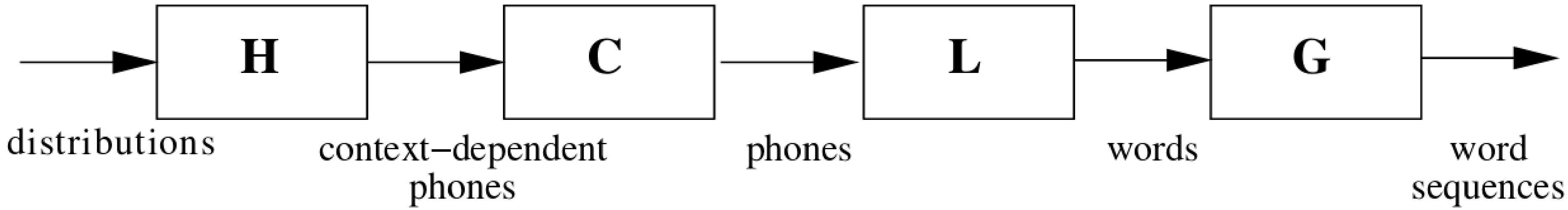
- Bigram approximation,

$$P(w_1,w_2,\ldots) \approx P(w_1|\$)\, P(w_2|w_1)\, P(w_3\, |w_2)\, P(w_4|w_3)\, \ldots$$

- Can be easily represented using a Finite State Transducer (FST)
- Backoff models may be approximated by epsilon transitions or exactly described by failure transitions.

Finite State Transducers

Context-dependent Recognition Transducer

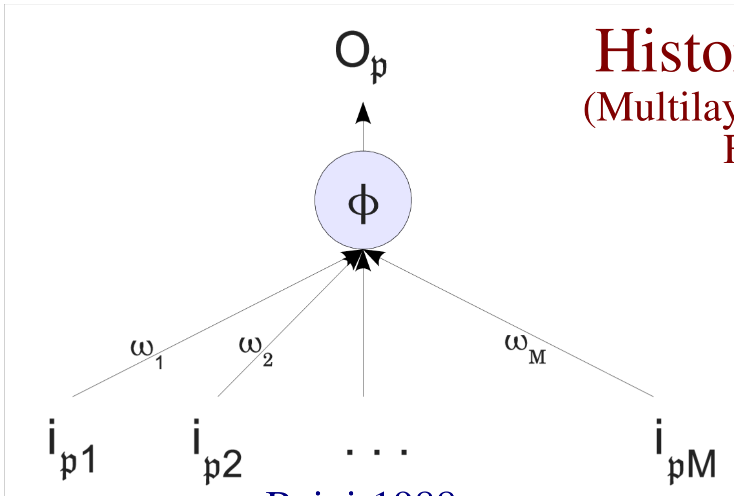


- H: HMM transducer, closure of the union of all HMMs used in acoustic modeling,
- C: context-dependency transducer mapping context-dependent phones to phones,
- L: pronunciation dictionary transducer mapping phonemic transcriptions to word sequences,
- G: language model weighted automaton.

$H \circ C \circ L \circ G$: mapping from sequences of distribution names to word sequences.

History of Neural Networks

(Multilayer Neural Network Generalization Formulation – Beigi-1987)



Beigi-1988

$$\boldsymbol{\phi}^l = \left[\phi^l_1, \phi^l_2, \cdots, \phi^l_{N_l}\right]^T$$ (Activation function parameter vector for level l)

$$\boldsymbol{\omega}^l_{n_l} = \left[\omega^l_{n_l 1}, \omega^l_{n_l 2}, \cdots, \omega^l_{n_l N_{l-1}}\right]^T$$ (Vector of intercellular weights to neuron n_l at layer l)

$$\boldsymbol{\omega}^l = \left[{\boldsymbol{\omega}^l_1}^T, {\boldsymbol{\omega}^l_2}^T, \cdots, {\boldsymbol{\omega}^l_{N_l}}^T\right]^T$$ (Supervector of intercellular weights of level l)

$$\mathbf{x}^l = \left[{\boldsymbol{\phi}^l}^T, {\boldsymbol{\omega}^l}^T\right]^T$$ (State vector for level l)

$$\mathbf{x} = \left[{\mathbf{x}^1}^T, {\mathbf{x}^2}^T, \cdots, {\mathbf{x}^L}^T\right]^T$$ (Super state vector)

$$\frac{\partial E}{\partial (\mathbf{x})_{[j]}} = \sum_{\mathfrak{p}=1}^{P} \frac{\partial E_{\mathfrak{p}}}{\partial (\mathbf{x})_{[j]}}$$

$$\frac{\partial E_{\mathfrak{p}}}{\partial (\mathbf{x})_{[j]}} = 2 \sum_{n_L=1}^{N_L} \left(o^L_{\mathfrak{p}n_l} - t_{\mathfrak{p}n_L}\right) \frac{\partial o^L_{\mathfrak{p}n_L}}{\partial (\mathbf{x})_{[j]}}$$

$$\frac{\partial o^L_{\mathfrak{p}n_L}}{\partial \phi^l_{n_l}} = \frac{\partial o^L_{\mathfrak{p}n_L}}{o^l_{\mathfrak{p}n_l}} \frac{o^l_{\mathfrak{p}n_l}}{\partial \phi^l_{n_l}}$$

$$\frac{\partial o^L_{\mathfrak{p}n_L}}{\partial \omega^l_{n_l n_{l-1}}} = \frac{\partial o^L_{\mathfrak{p}n_L}}{o^l_{\mathfrak{p}n_l}} \frac{o^l_{\mathfrak{p}n_l}}{\partial \omega^l_{n_l n_{l-1}}}$$

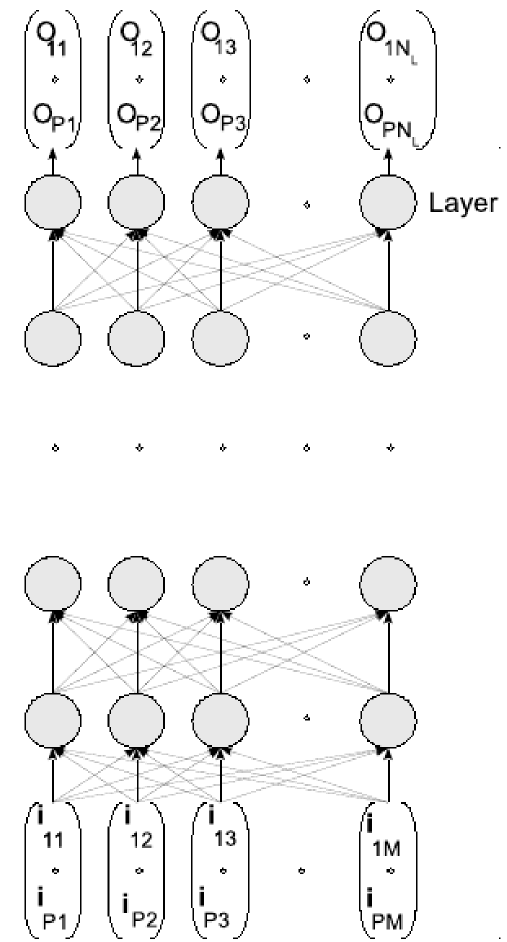
$$\frac{\partial o^L_{\mathfrak{p}n_L}}{o^l_{\mathfrak{p}n_l}} = \prod_{k=l+1}^{L} \frac{\partial o^{(L+l-k+1)}_{\mathfrak{p}n_{(L+l-k+1)}}}{\partial o^{(L+l-k)}_{\mathfrak{p}n_{(L+l-k)}}}$$

Tensor Index Notation

$$\underbrace{\frac{\partial o^{l+1}_{\mathfrak{p}n_{(l+1)}}}{\partial o^l_{\mathfrak{p}n_l}} \frac{\partial o^l_{\mathfrak{p}n_l}}{\partial o^{l-1}_{\mathfrak{p}n_{(l-1)}}}}_{\text{Index Notation}} = \underbrace{\sum_{n_l=1}^{N_l} \frac{\partial o^{l+1}_{\mathfrak{p}n_{(l+1)}}}{\partial o^l_{\mathfrak{p}n_l}} \frac{\partial o^l_{\mathfrak{p}n_l}}{\partial o^{l-1}_{\mathfrak{p}n_{(l-1)}}}}_{\text{Traditional Notation}}$$

$$\mathbf{g} \triangleq \nabla_{\mathbf{x}} E = \sum_{\mathfrak{p}=1}^{P} \nabla_{\mathbf{x}} E_{\mathfrak{p}} \quad \text{(Gradient Vector)}$$

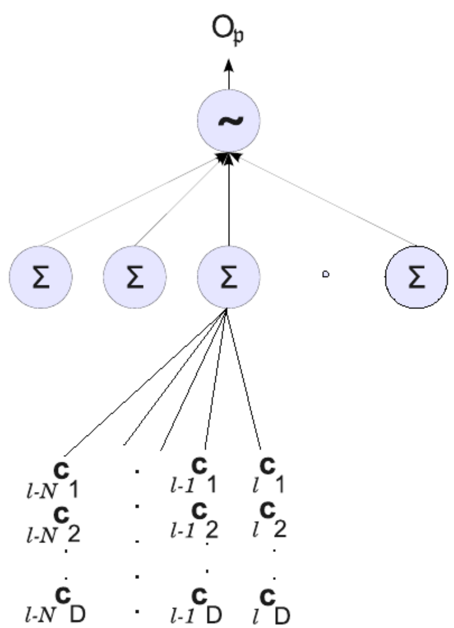
$$\mathbf{G} \triangleq \nabla^2_{\mathbf{x}} E = \sum_{\mathfrak{p}=1}^{P} \nabla^2_{\mathbf{x}} E_{\mathfrak{p}} \quad \text{(Hessian Matrix)}$$



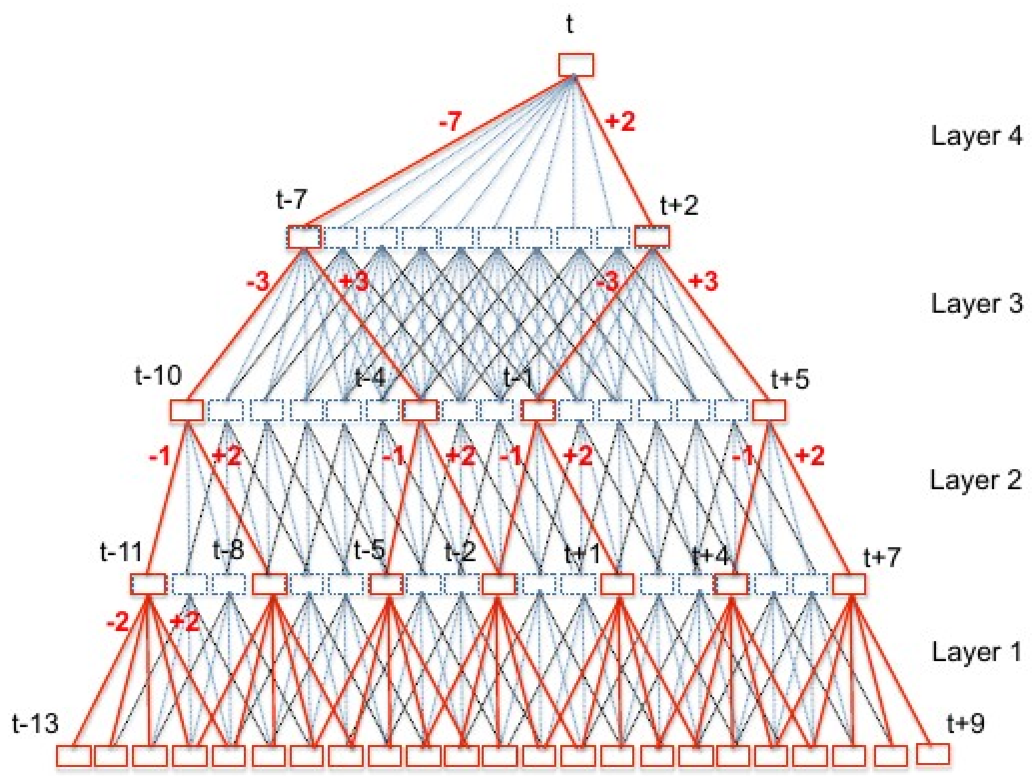
Beigi-1988

Time-Delay Neural Networks

(Time-series Modeling – derived from the work of Alex Waibel in the 1990s)



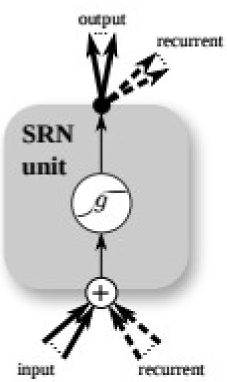
TDNN – Beigi 2011



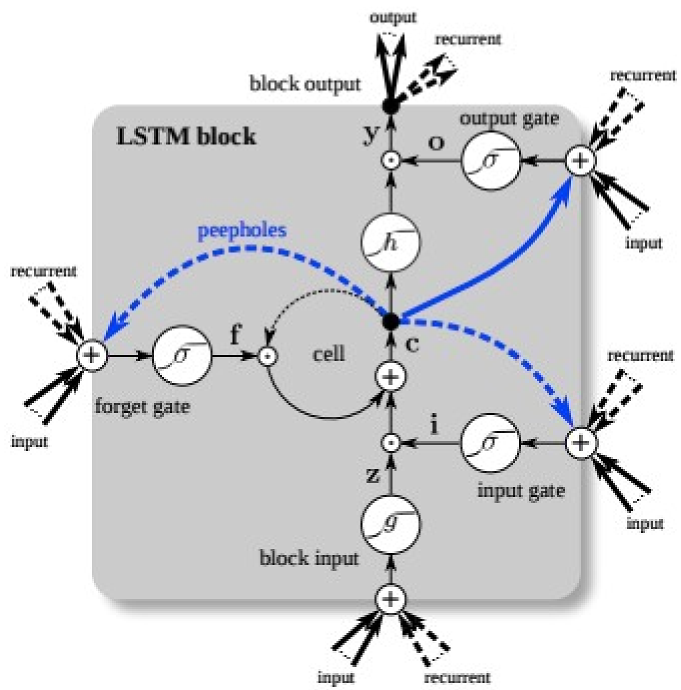
2015 – A time delay neural network architecture for efficient modeling of long temporal contexts

Peddinti, Povey, and Khudanpur
Johns Hopkins University

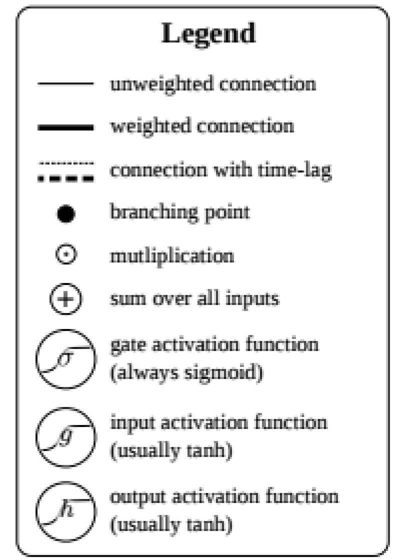
LSTM (Long-Short Term Memory)

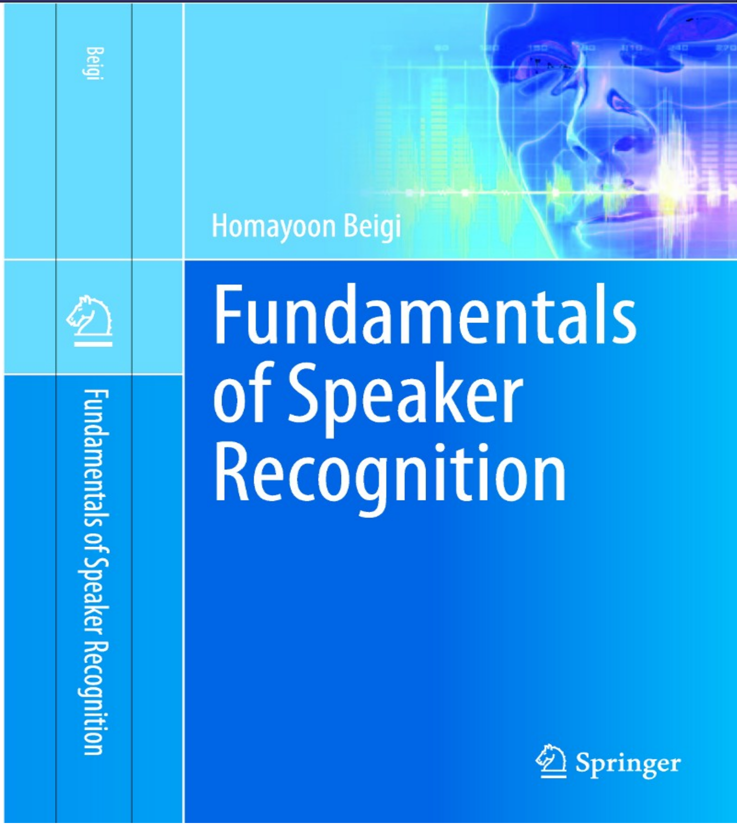


Recurrent Neuron



LSTM Unit using
Recurrenyt neurons





Extremely Multi-Disciplinary

Part I – Basic Theory

1. Introduction
2. Anatomy of Speech
3. Signal Representation of Speech
4. Phonetics and Phonology
5. Signal Proc. & Feature Extraction
6. Probability Theory and Statistics
7. Information Theory
8. Metrics and Divergences
9. Decision Theory
10. Parameter Estimation
11. Unsuperv. Clust. & Learning
12. Transformation
13. Hidden Markov Modeling
14. Neural Networks
15. Support Vector Machines

Part II – Advanced Theory

16. Speaker Modeling
17. Speaker Recognition
18. Signal Enhancement & Comp.

Part III – Practice

19. Evaluation & Representation of Results
20. Time Lapse Effects
21. Adaptation over Time
22. Overall Design

Part IV – Background Material

23. Linear Algebra
24. Integral Transforms
25. Optimization Theory
26. Standards