CS 224S / Linguist 285
Spoken Language Processing

Andrew Maas | Stanford University | Spring 2024

Lecture 7: Speech Recognition Overview

Outline

- Early speech recognition research
- Noisy channel model & ASR architecture
- ASR with Hidden Markov Models

Early speech recognition research

Arc of Recent History

- In 2013:
  - ASR, TTS, dialog all used specialized, hard-to-build modeling approaches
  - Industry application of SLU systems limited. ASR “didn’t quite work well enough”
- Today:
  - ASR, TTS, dialog all use deep learning approaches. Less specialized and better performance
  - Spoken language systems are everywhere!
  - New tools enable building full systems

History: Foundational Insights 1900s-1950s

- **Automaton:**
  - Markov 1911
  - Turing 1936
  - McCulloch-Pitts neuron (1943)
    - http://marr.bsee.swin.edu.au/~dtl/het704/lecture10/ann/node1.html
    - http://diwww.epfl.ch/mantra/tutorial/english/mcpits/html/
  - Shannon (1948) link between automata and Markov models
- **Human speech processing**
  - Fletcher at Bell Labs (1920's)
- **Probabilistic/Information-theoretic models**
  - Shannon (1948)

Early Recognition

1920's Radio Rex

- Celluloid dog with iron base held within house by electromagnet against force of spring
- Current to magnet flowed through bridge which was sensitive to energy at 500 Hz
- 500 Hz energy caused bridge to vibrate, interrupting current, making dog spring forward
- The sound "e" (ARPAbet [eh]) in Rex has 500 Hz component

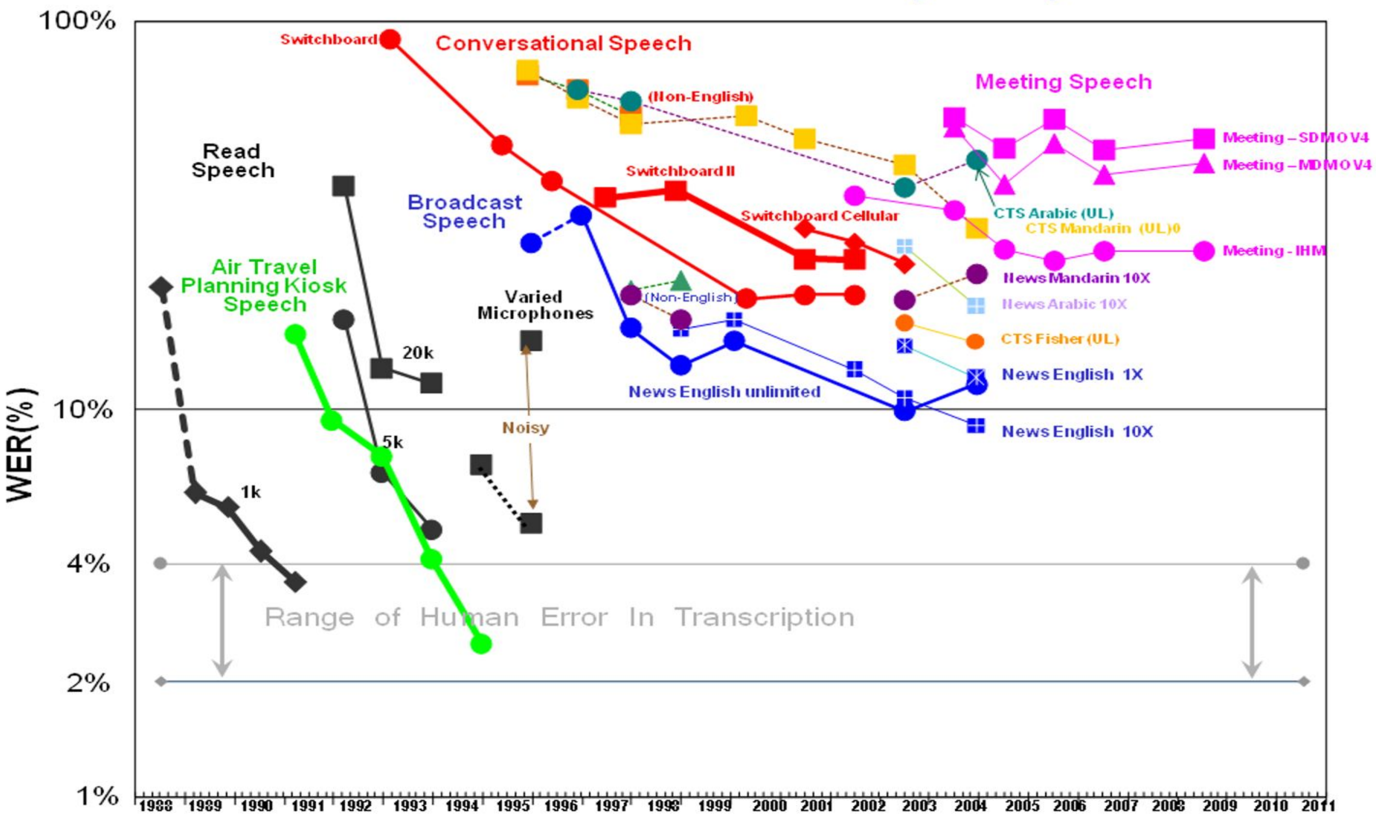
ASR: 1950's - Early Speech Recognizers

- 1952: Bell Labs single-speaker digit recognizer
  - Measured energy from two bands (formants)
  - Built with analog electrical components
  - 2% error rate for single speaker, isolated digits
- 1958: Dudley built classifier that used continuous spectrum rather than just formants
- 1959: Denes ASR combining grammar and acoustic probability

ASR: 1970's and 1980's

- Hidden Markov Model 1972
    - Independent application of Baker (CMU) and Jelinek/Bahl/Mercer lab (IBM) following work of Baum and colleagues at IDA
- ARPA project 1971-1976
    - 5-year speech understanding project: 1000 word vocab, continuous speech, multi-speaker
    - SDC, CMU, BBN
    - Only 1 CMU system achieved goal
- 1980's +
    - Annual ARPA "Bakeoffs"
    - Large corpus collection
        - TIMIT
        - Resource Management
        - Wall Street Journal

NIST STT Benchmark Test History - May 2009



More recent ASR Improvements

Hub5'00 Evaluation (Switchboard / CallHome)

(Possibly trained on more data than SWB, but test set = full Hub5

WER (SWB)	WER (CH)	Paper	Published
4.9%	9.5%	An investigation of phone-based subword units for end-to-end speech recognition	April 2020
5.0%	9.1%	The CAPIO 2017 Conversational Speech Recognition System	December 2017
5.1%	9.9%	Language Modeling with Highway LSTM	September 2017
5.1%		The Microsoft 2017 Conversational Speech Recognition System	August 2017

WSJ

(Possibly trained on more data than WSJ.)

WER eval'92	WER eval'93	Paper	Published
5.03%	8.08%	*Humans* Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015
2.9%		End-to-end Speech Recognition Using Lattice-Free MMI	September 2018
3.10%		Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015

LibriSpeech

(Possibly trained on more data than LibriSpeech.)

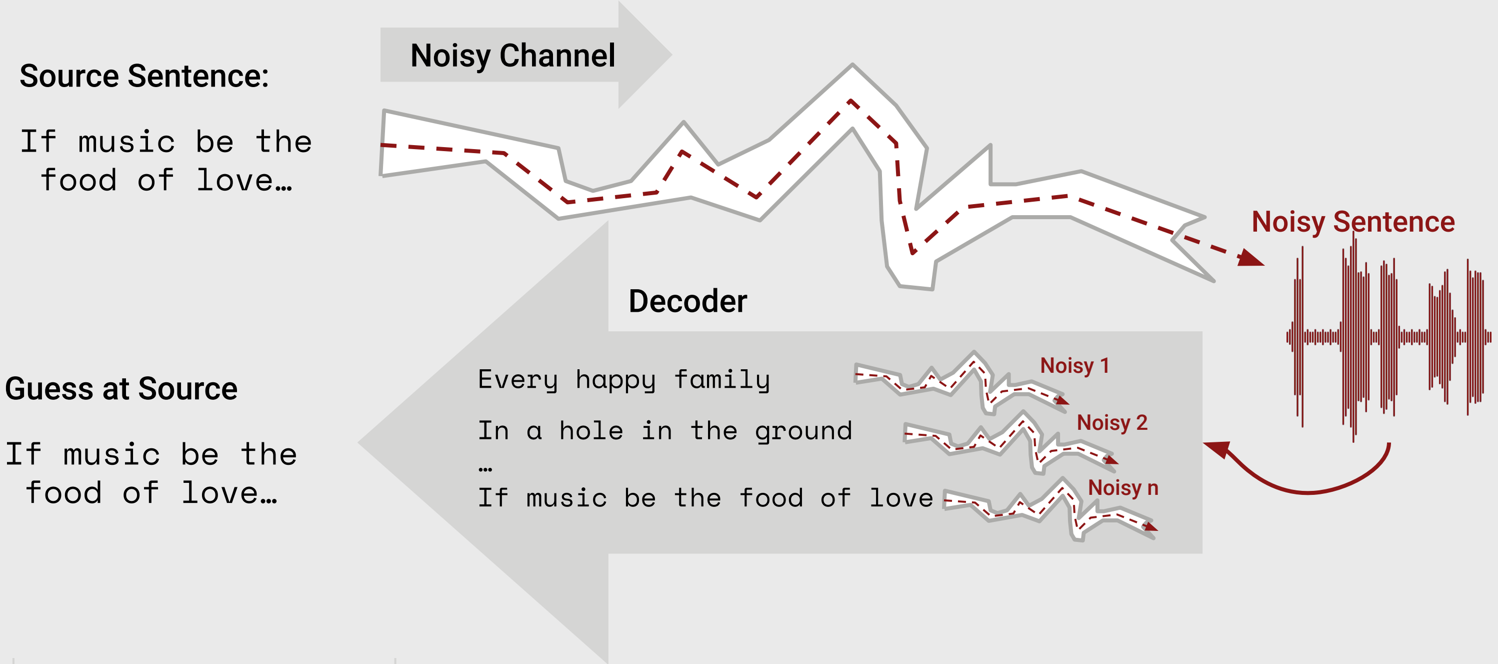
WER test-clean	WER test-other	Paper	Published
5.83%	12.69%	*Humans* Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015
1.9%	3.9%	Conformer: Convolution-augmented Transformer for Speech Recognition	May 2020
1.9%	4.1%	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context	May 2020

https://github.com/syhw/wer_are_we

Noisy Channel Model & ASR Architecture

The Noisy Channel Model

- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform



The Noisy Channel Model

- What is the most likely sentence out of all sentences in the language L that generated some given some acoustic input O?
- Treat acoustic input O as sequence of individual observations
  - $O = o_1, o_2, o_3, \ldots, o_t$
- Define a sentence as a sequence of words:
  - $W = w_1, w_2, w_3, \ldots, w_n$

The Noisy Channel Model

- Probabilistic implication: Pick the highest prob word sequence W:

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(W \mid O)$$

- We can use Bayes rule to rewrite this:

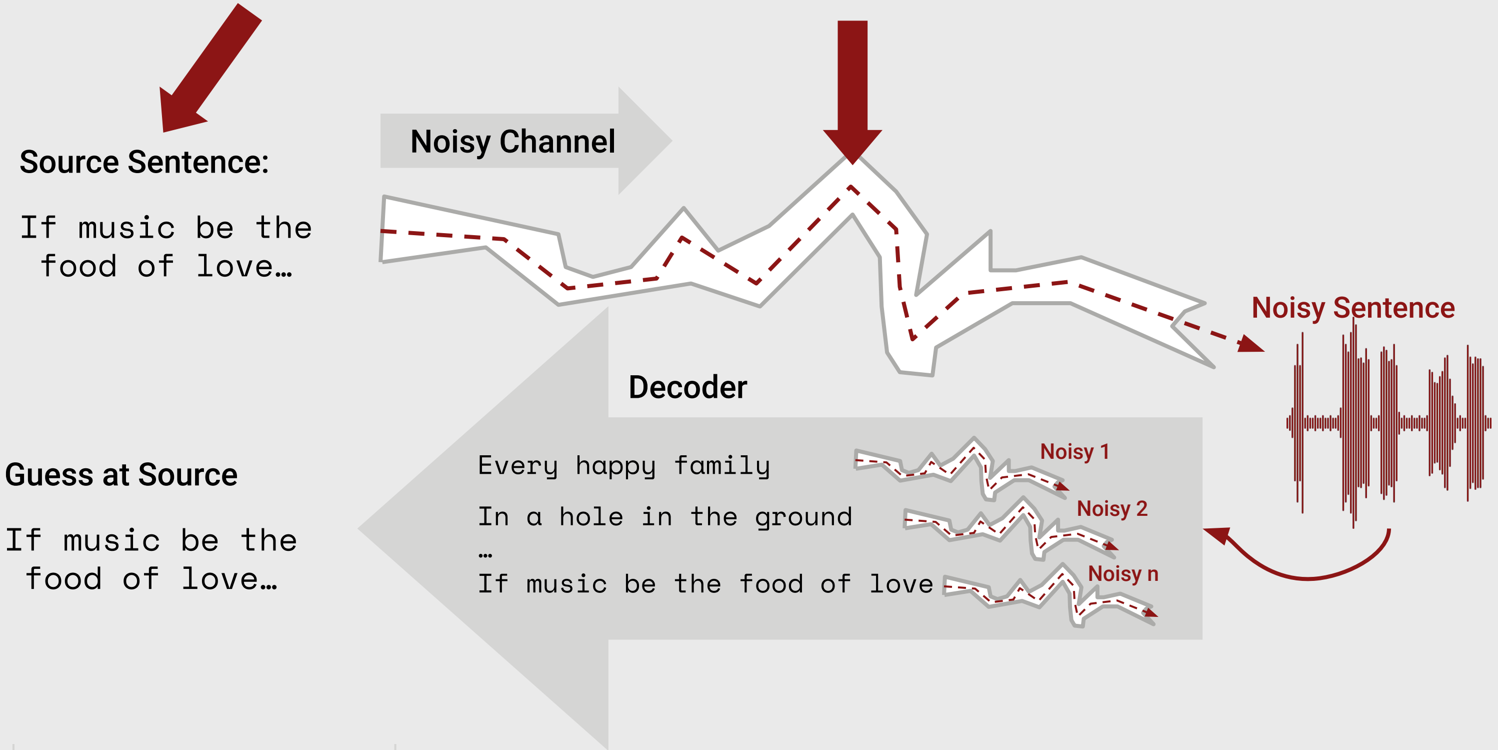
$$\hat{W} = \underset{W \in L}{\arg\max}\, \frac{P(O \mid W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W, we can ignore it for the argmax:

$$\hat{W} = \underset{W \in L}{\arg\max}\, P(O \mid W)P(W)$$

The Noisy Channel Model

- Ignoring the denominator leaves us with two factors: P(Source) and P(Signal|Source)

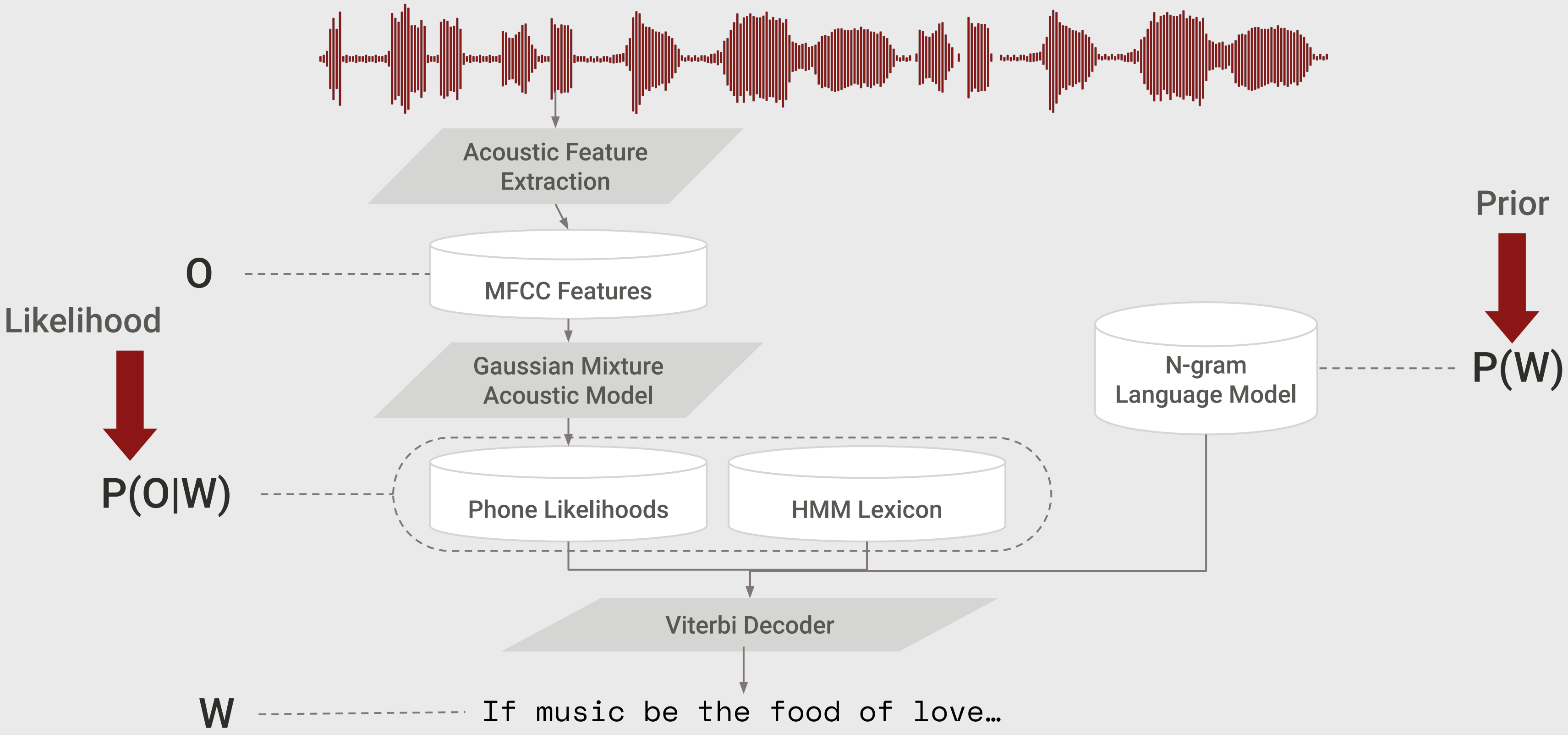


The Noisy Channel Model

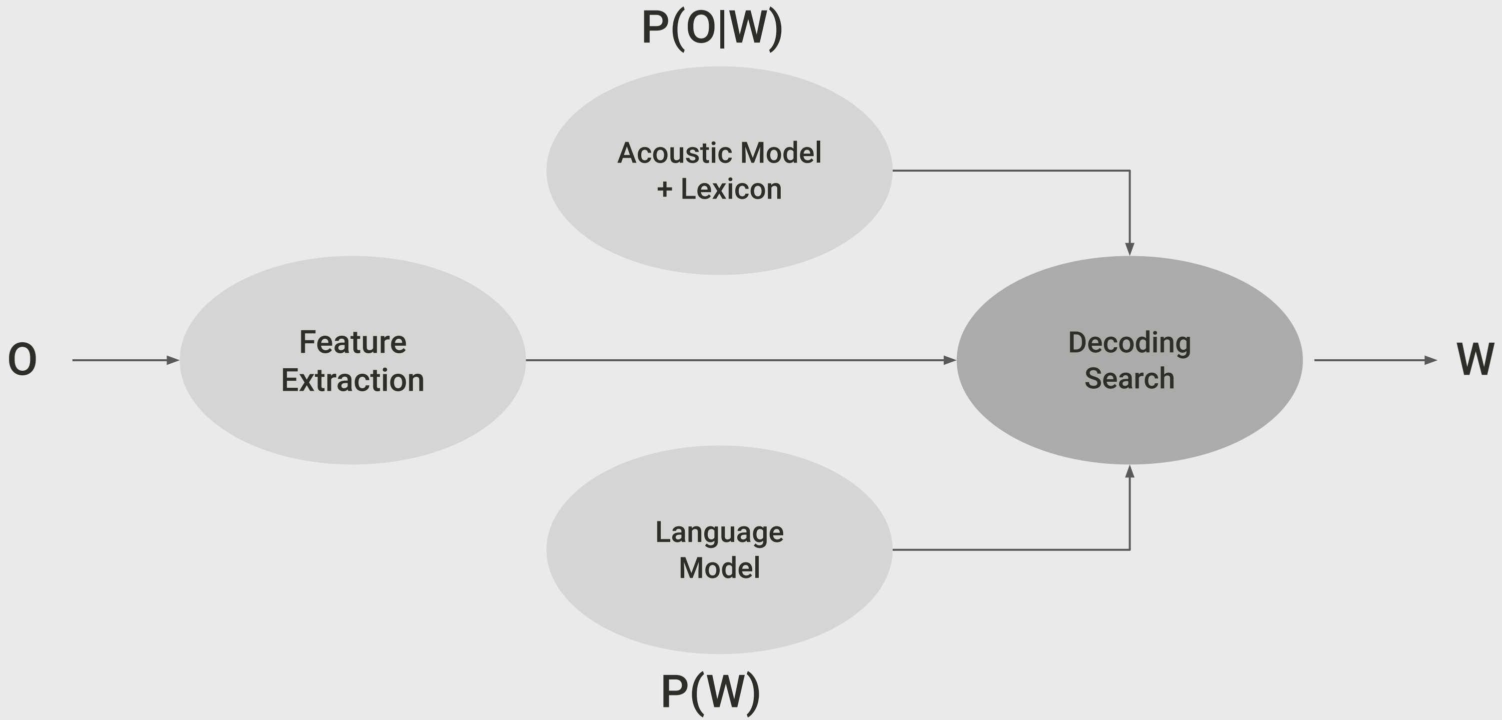
Likelihood Prior

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} \underbrace{P(O \mid W)}_{\text{Likelihood}} \underbrace{P(W)}_{\text{Prior}}$$

Speech Recognition Architecture



Speech Architecture Meets Noisy Channel



Word error rate (WER)

$$\text{Word Error Rate} = 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

Can be >100%.
Doesn't distinguish between function words (of, they, he, she) and more important content words

Compute best alignment of reference and hypothesis to count errors:

REF:	i	***	**	UM	the	PHONE	IS	i	LEFT	THE	portable	****	PHONE	UPSTAIRS	last night
HYP:	i	GOT	IT	TO	the	*****	FULLEST	i	LOVE	TO	portable	FORM	OF	STORES	last night
Eval:		I	I	S		D	S		S	S		I	S	S	

Word error rate (WER)

$$\text{Word Error Rate} \;=\; 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

Can be >100%.
Doesn't distinguish between function words (of, they, he, she) and more important content words

Comparing aligned systems for deeper error analysis:

```
             I              II               III                  IV
REF:    |it was|the best|of|times it|was the worst|of times|  |it was
        |      |        |  |        |             |        |  |
SYS A:|ITS    |the best|of|times it|IS the worst |of times|OR|it was
        |      |        |  |        |             |        |  |
SYS B:|it was|the best|  |times it|WON the TEST |of times|  |it was
```

ASR with Hidden Markov Models & Gaussian Mixture Models (HMM-GMM)

HMM-GMM Decoding Architecture: Main components

- **Feature Extraction:**
  - 39 MFCC features
- **Acoustic Model:**
  - Gaussians for computing p(o|q)
- **Lexicon/Pronunciation Model**
  - HMM: what phones can follow each other
- **Language Model**
  - N-grams for computing $p(w_i|w_{i-1})$
- **Decoder**
  - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech

HMM-GMM System

Transcription: `Samson`

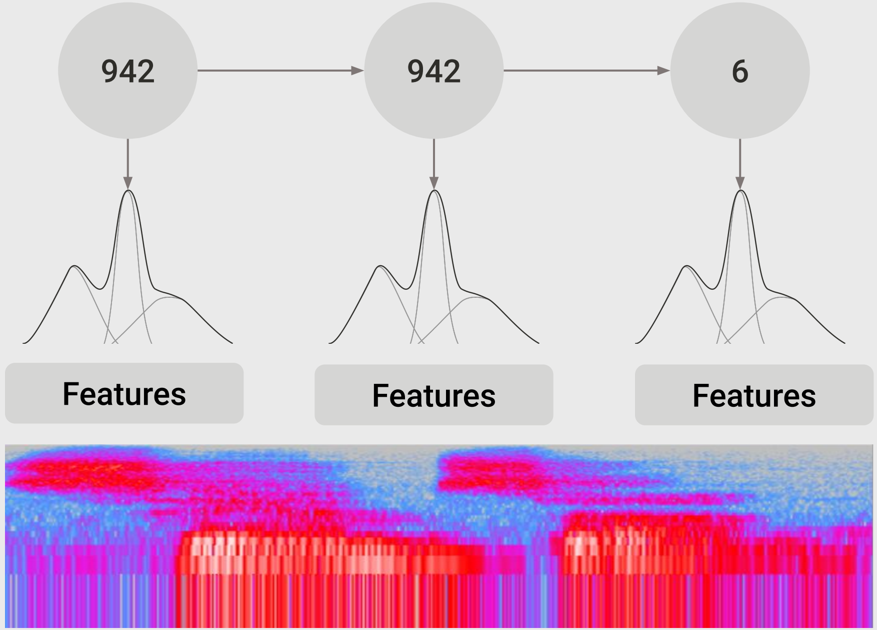
Pronunciation: `S – AE – M – S –AH – N`

Sub-phones: `942 – 6 – 37 – 8006 – 4422 …`

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:



GMM models:

P(o|q)

o: input features

q: HMM state

Hidden Markov Model (HMM) basics

- Markov assumption:

$$P(q_i \mid q_1 \cdots q_{i-1}) = P(q_i \mid q_{i-1})$$

- Output-independence assumption:

$$P(o_t \mid O_1^{t-1}, q_1^t) = P(o_t \mid q_t)$$

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state

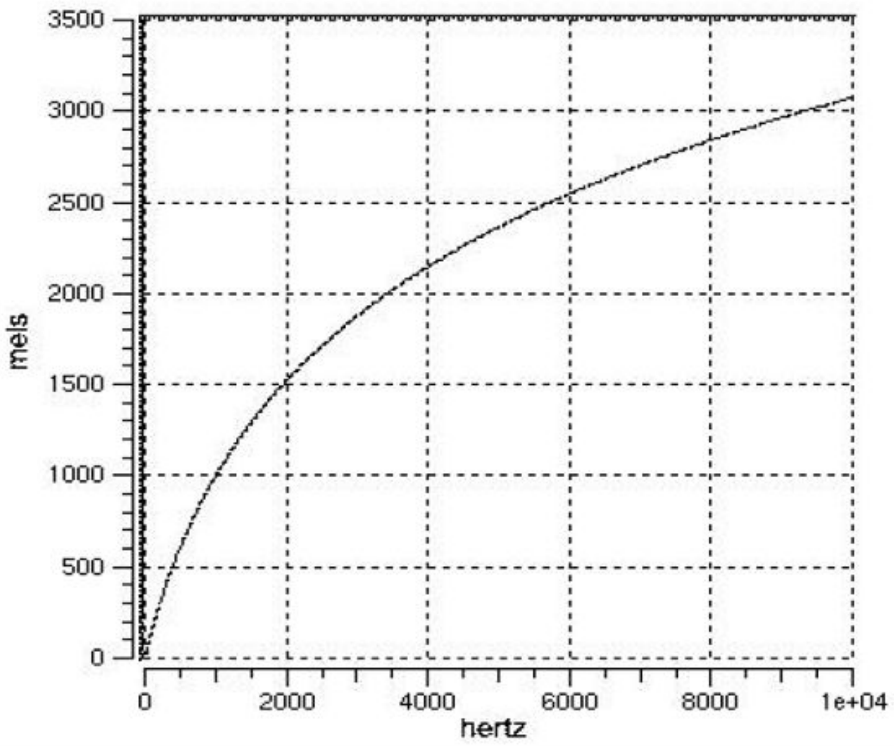
Feature Extraction (MFCCs)

Mel-scale

Human hearing is not equally sensitive to all frequency bands

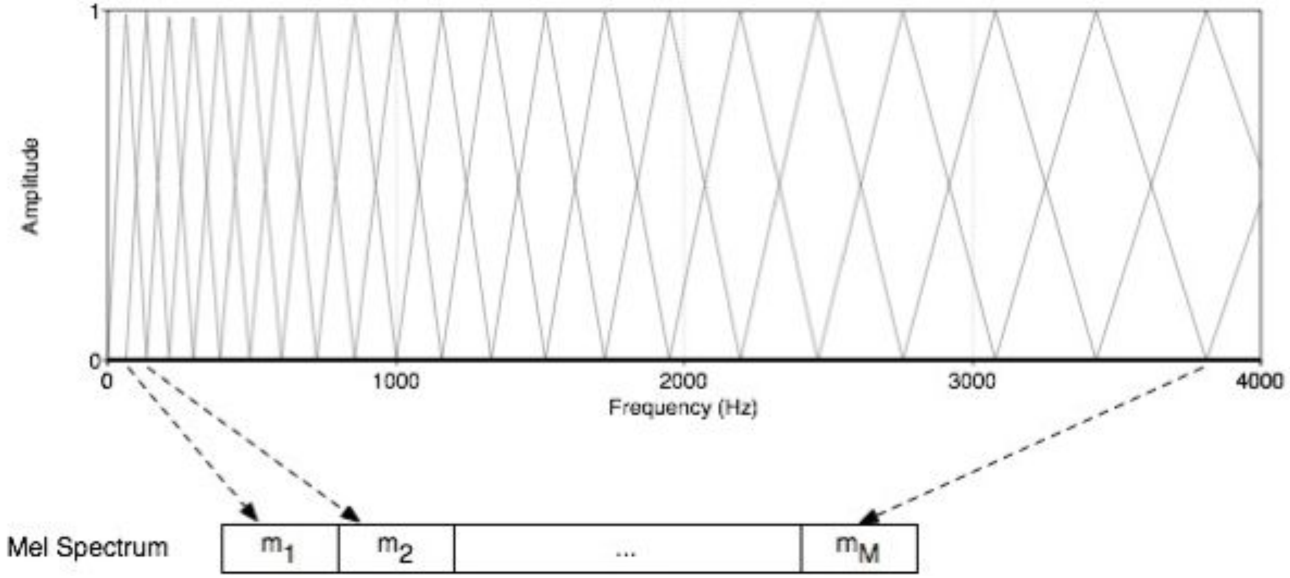
Less sensitive at higher frequencies, > 1000 Hz

I.e. human perception of frequency is non-linear:



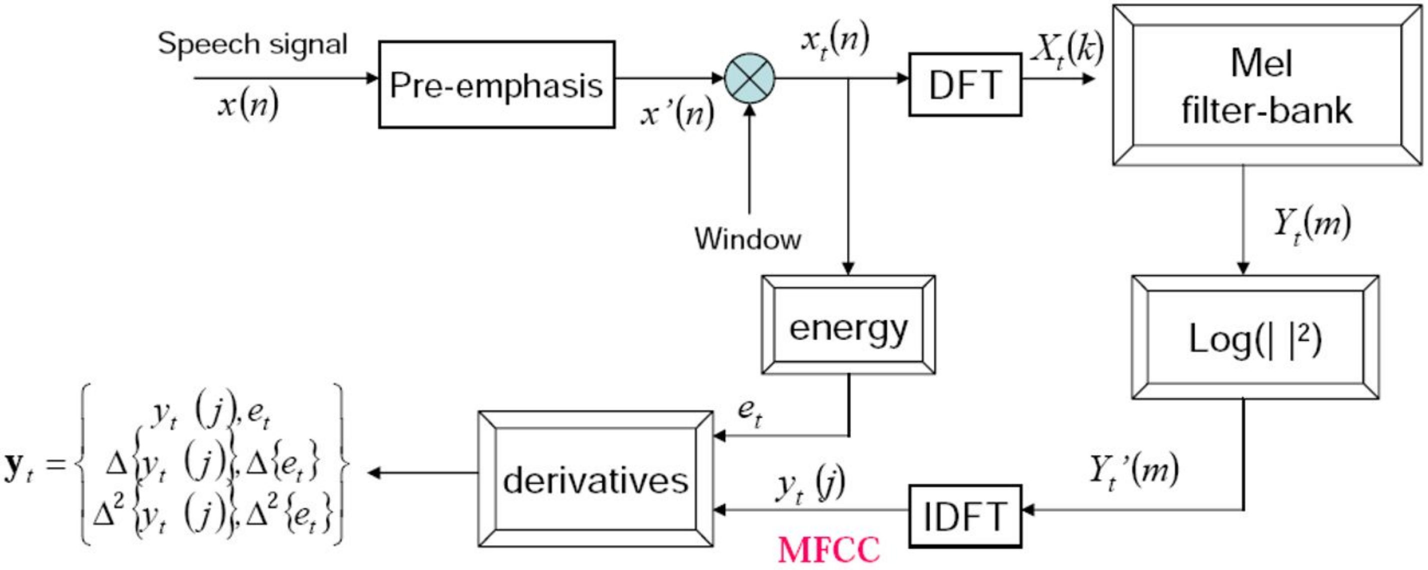
Mel Filter Bank Processing

- Mel Filter bank
  - Roughly uniformly spaced before 1 kHz
  - logarithmic scale after 1 kHz



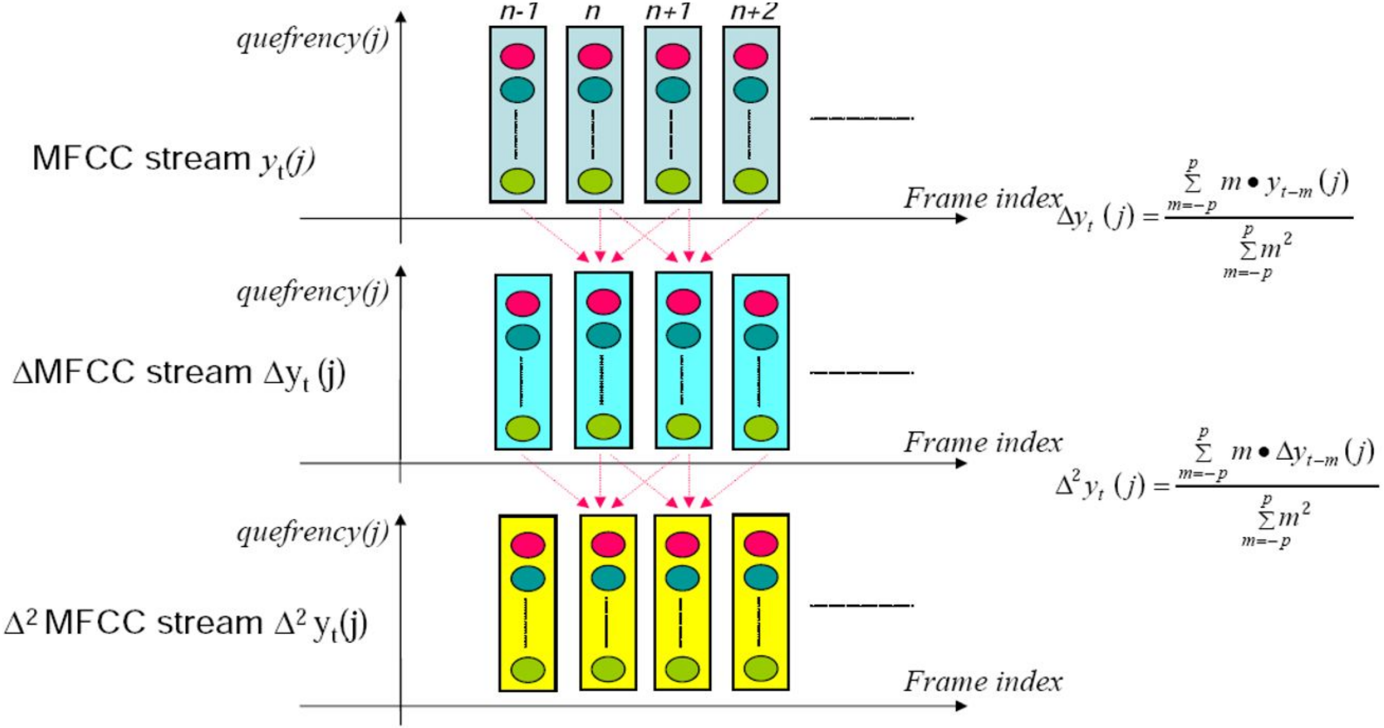
MFCC

- Mel-Frequency Cepstral Coefficient (MFCC)
- Most widely used spectral representation in ASR



Delta and Double-delta

- Derivative: in order to obtain temporal information



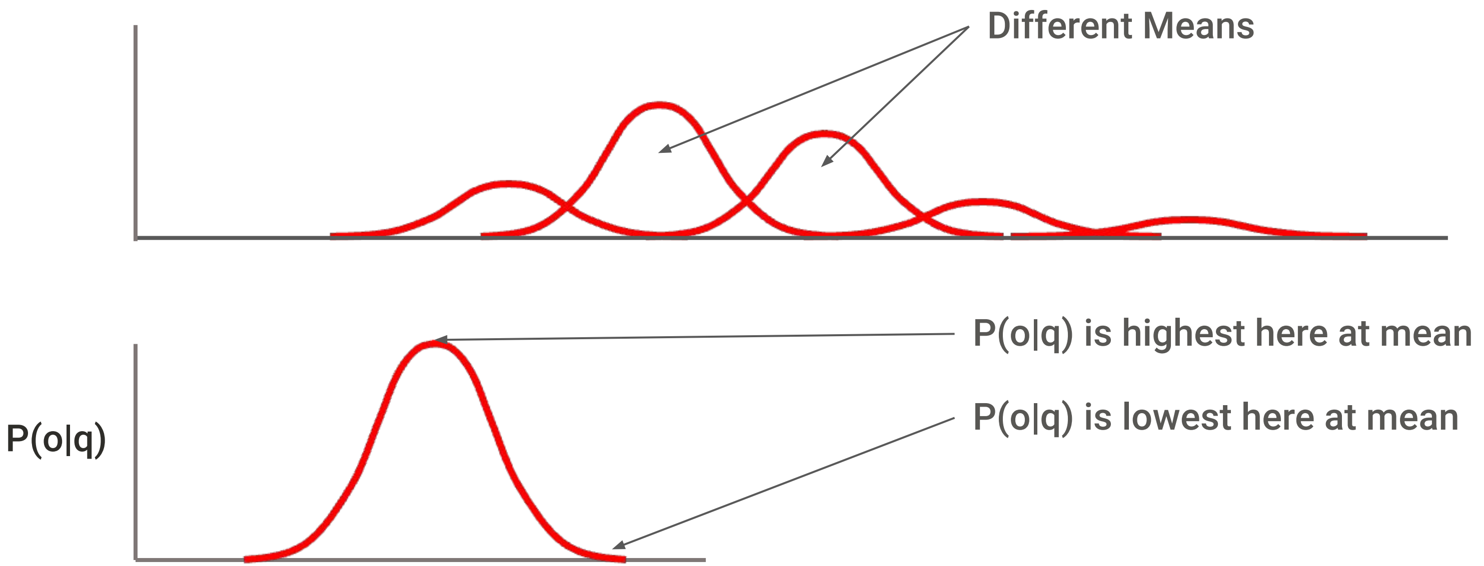
Typical MFCC Features

- Window size: 25ms
- Window shift: 10ms
- Pre-emphasis coefficient: 0.97
- MFCC:
  - 12 MFCC (mel frequency cepstral coefficients)
  - 1 energy feature
  - 12 delta MFCC features
  - 12 double-delta MFCC features
  - 1 delta energy feature
  - 1 double-delta energy feature
- Total 39-dimensional features

GMM Acoustic Models

Gaussians for Acoustic Modeling

- P(o|q): A Gaussian parameterized by mean and variance:

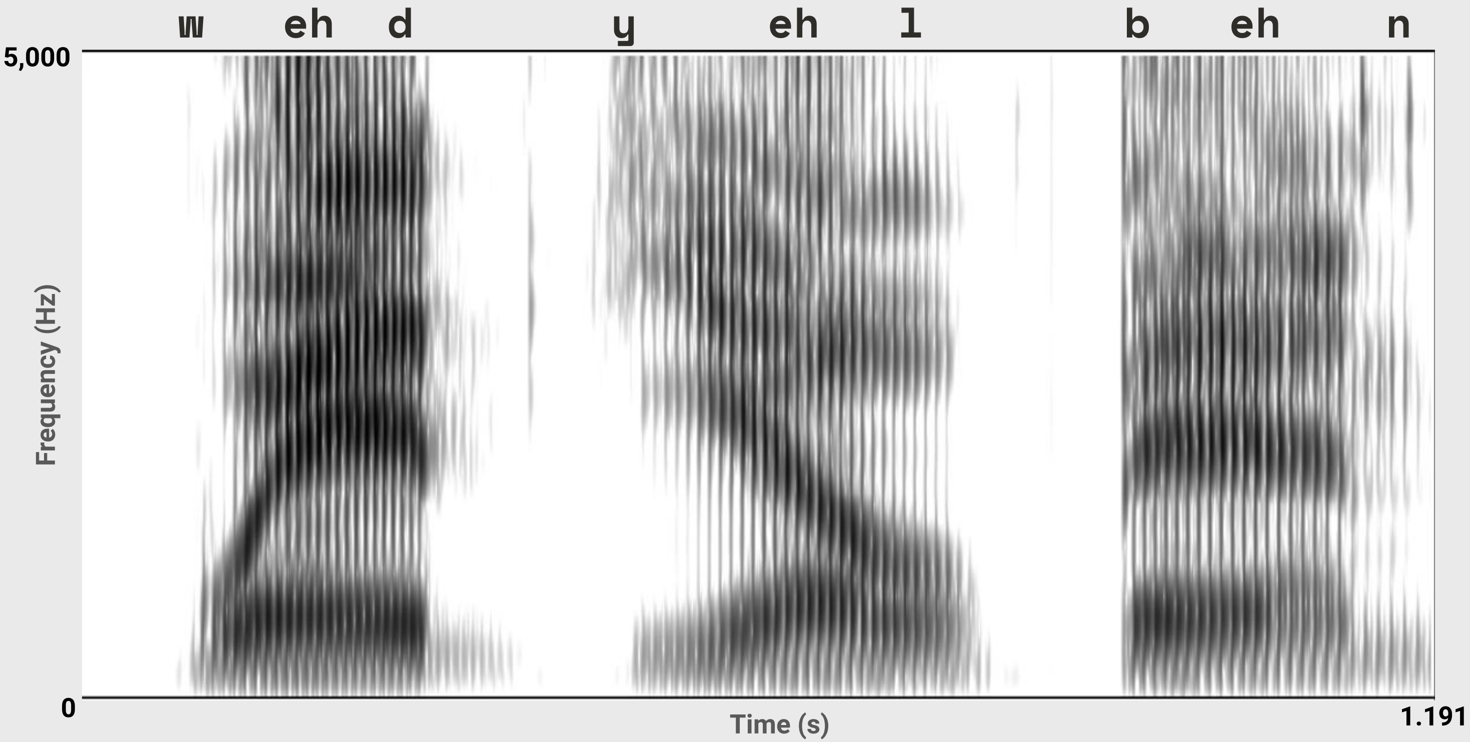


GMMs

Summary: each state has a likelihood function parameterized by:

- *M* Mixture weights
- *M* Mean Vectors of dimensionality *D*
- either:
    - *M* Covariance Matrices of *DxD*
- or more likely:
    - M Diagonal Covariance Matrices of *DxD*
    - which is equivalent to
    - M Variance Vectors of dimensionality *D*

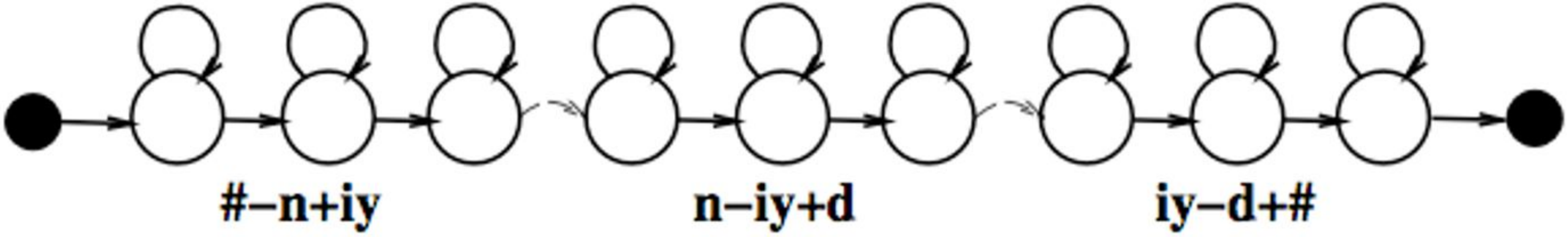
Phonetic Context: Different "eh"s



Context Dependent (CD) Phones: Triphones

- The strongest factor affecting phonetic variability is the neighboring phone
  - HMMs assume *the opposite*: per-state observation likelihoods are conditionally independent
- Idea: have phone models which are specific to context. Context-Dependent (CD) phones
  - Instead of Context-Independent (CI) phones
- Each triphone captures facts about preceding and following phone
- Monophone:
  - p, t, k
- Triphone:
  - ly-p+aa
  - a-b+c means “phone b, preceding by phone a, followed by phone c”
- AND for each triphone, we use 3 separate sub-states (beginning, middle, end) to further split the categories and reduce within-state variance of observations

“Need” with Triphone Models



Word-Boundary Modeling

- Word-Internal Context-Dependent Models

‘OUR LIST’:
SIL AA+R AA-R L+IH L-IH+S IH-S+T S-T

- Cross-Word Context-Dependent Models

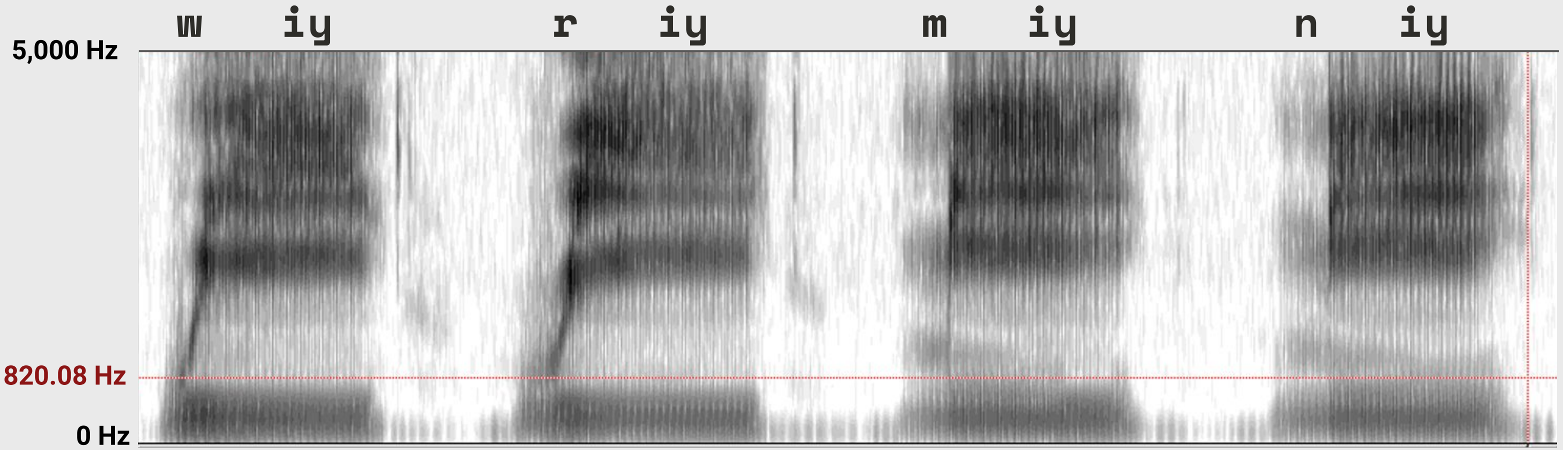
‘OUR LIST’:
SIL-AA+R AA-R+L R-L+IH L-IH+S IH-S+T S-T+SIL

- Dealing with cross-words makes decoding harder!

Implications of Cross-Word Triphones

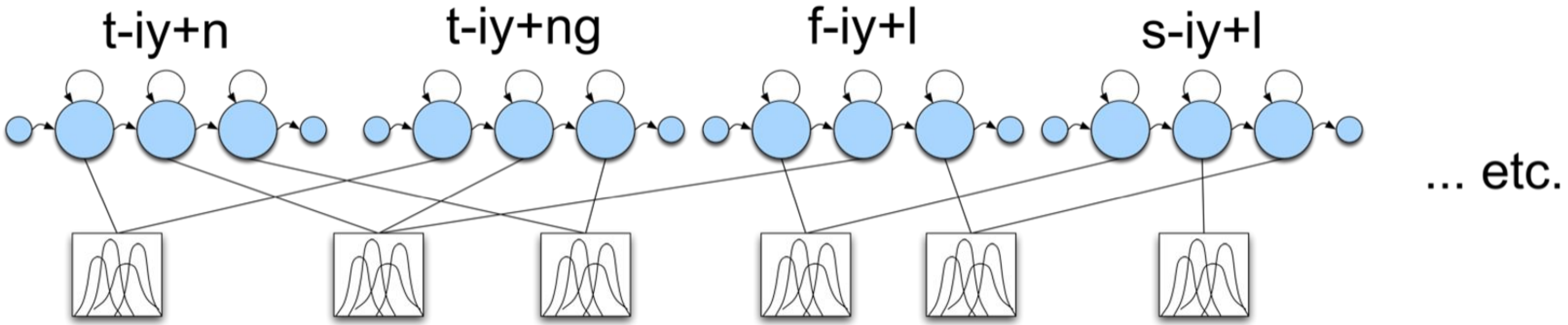
- Possible triphones: 50x50x50=125,000
- How many triphone types actually occur?
- 20K word WSJ Task, numbers from Young et al
- Cross-word models: need 55,000 triphones
- But in training data only 18,500 triphones occur!
- Need to generalize models

Modeling Phonetic Context: Some Contexts Look Similar

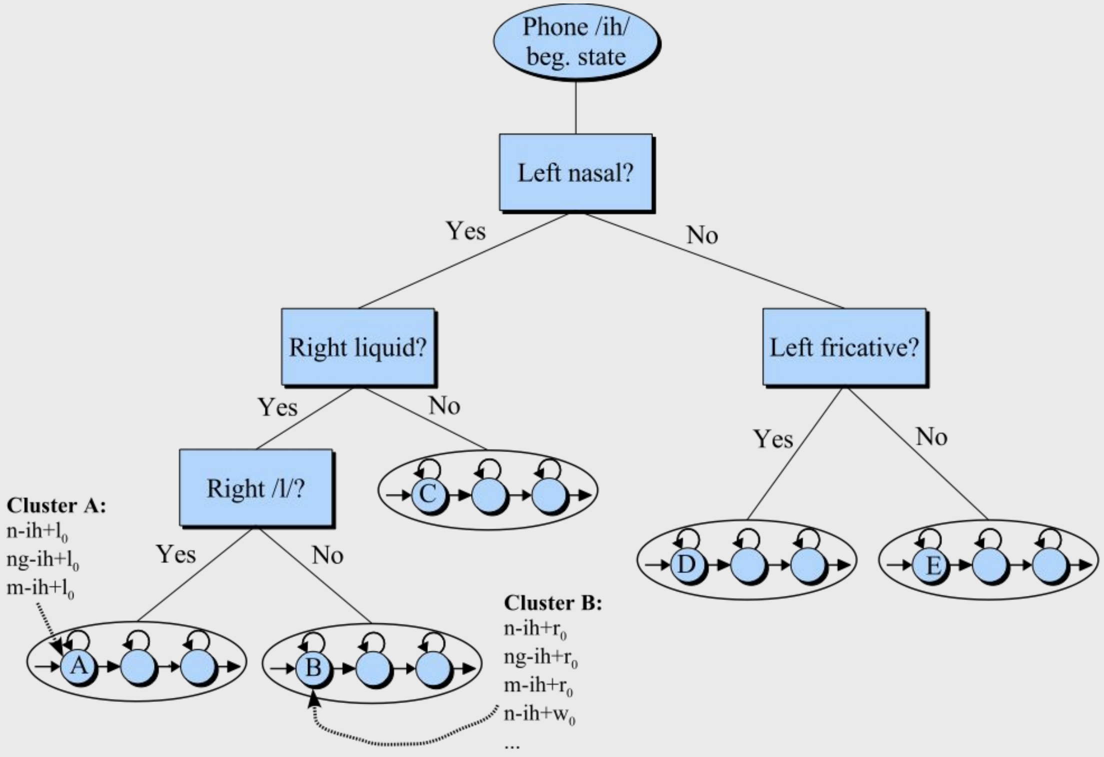


Solution: State Tying

- Young, Odell, Woodland 1994
- Decision-Tree based clustering of triphone states
- States which are clustered together will share their Gaussians
- We call this “state tying”, since these states are “tied together” to the same Gaussian.



Triphone Decision Tree Clustering



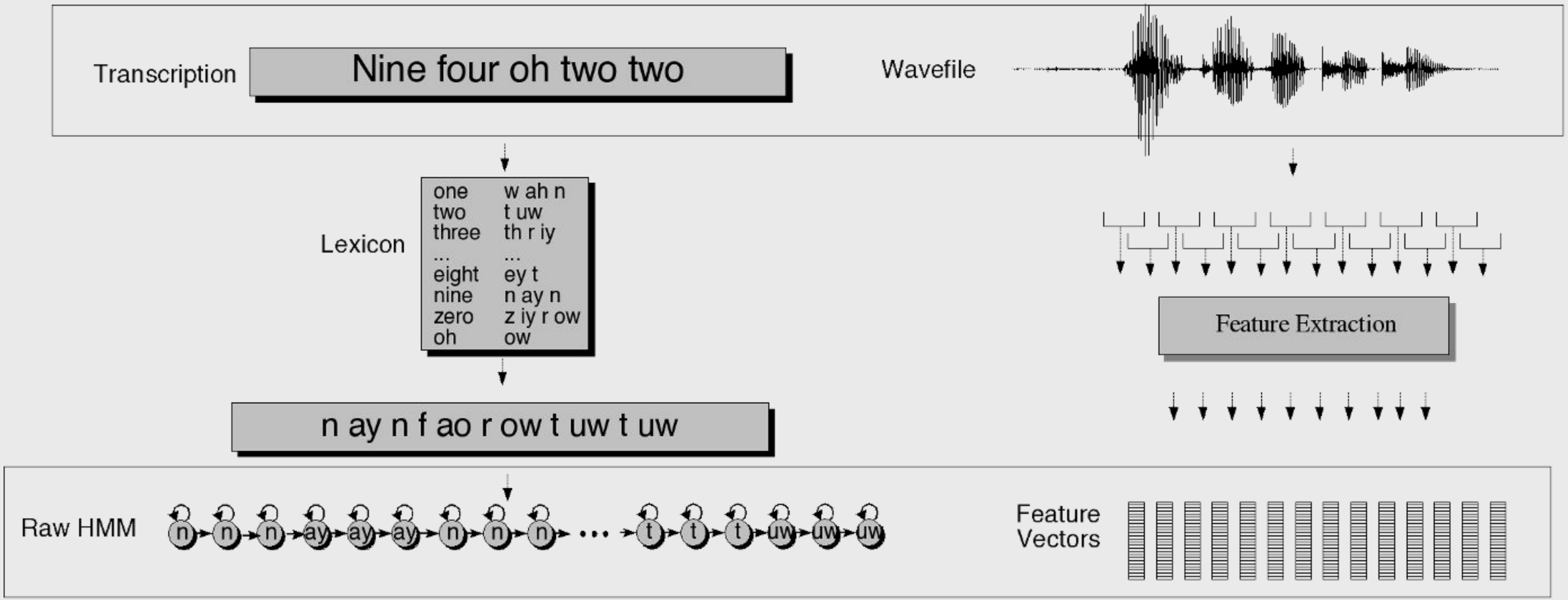
Triphone Decision Tree Clustering

Feature	Phones
Stop	b d g k p t
Nasal	m n ng
Fricative	ch dh f jh s sh th v z zh
Liquid	l r w y
Vowel	aa ae ah ao aw ax axr ay eh er ey ih ix iy ow oy uh uw
Front Vowel	ae eh ih ix iy
Central Vowel	aa ah ao axr er
Back Vowel	ax ow uh uw
High Vowel	ih ix iy uh uw
Rounded	ao ow oy uh uw w
Reduced	ax axr ix
Unvoiced	ch f hh k p s sh t th
Coronal	ch d dh jh l n r s sh t th z zh

Iterative expectation maximization training

- **We initially have no alignments between audio and transcripts**
- **General process. Iteratively improve alignments and train more complex models**
  - Use current HMM-GMM system to produce a “forced alignment”.
    Given the transcripts (ground truth phoneme sequence) produce the phoneme-time alignments
  - Use aligned data as ground truth.
    Throw away old GMMs. Fit more complex GMMs or increase number of states for a more accurate model
  - Repeat the iterative process above to progress to GMM acoustic models with clustered CD states.
- **Progression towards GMMs for each state:**
  - Gaussians
  - Multivariate Gaussians
  - Mixtures of Multivariate Gaussians
- **Make more expressive states progressively:**
  - CI Phone
  - CI Subphone (3ish per phone)
  - CD phone (=triphones)
  - State-tying of CD phone
- **This results in a “training recipe” and there is some art in getting the right progression.**
  A clunky optimization process for the full system

HMM-GMM Embedded Training



Training an HMM system (Viterbi)

- **Given our lexicon + HMM structure, and some acoustic model, we can:**
  - Generate the best alignment of HMM states to acoustic observations
- **With an alignment of HMM states to observations:**
  - Build a new acoustic model. Treat current state/obs mapping as training data+labels
  - This acoustic model is hopefully better than previous one
- **Repeat the align -> rebuild acoustic model process until convergence**
  - Add parameters / complexity to acoustic model each iteration

Forced Alignment

- Computing the “Viterbi path” over the training data is called “forced alignment”
- Because we know which word string to assign to each observation sequence.
- We just don’t know the state sequence.
- So we use a_{ij} to constrain the path to go through the correct words
- And otherwise do normal Viterbi
- Result: state sequence!

Initialization: “Flat start”

- **Transition probabilities:**
  - Set to zero any that you want to be “structurally zero” (lexicon/pronunciation)
  - Set the rest to identical values

- **Likelihoods:**
  - Initialize GMM and of each state to global mean and variance of all training data

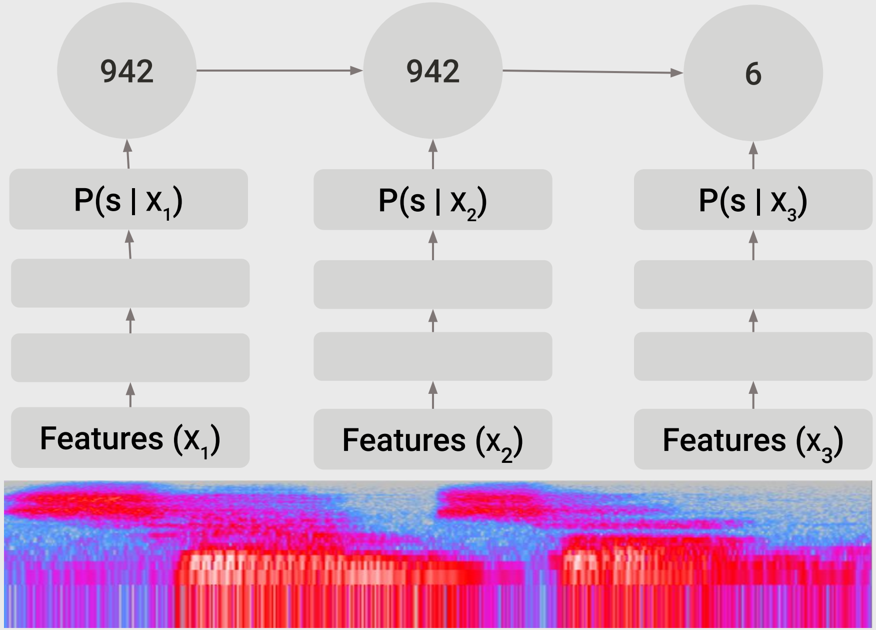
DNN Hybrid Acoustic Models

Transcription: Samson
Pronunciation: S – AE – M – S –AH – N
Sub-phones: 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:



Use a DNN to approximate:
$P(s|x)$

Apply Bayes' Rule:
$P(x|s) = P(s|x) * P(x) / P(s)$

DNN * Constant / State prior

Objective Function for Learning

- Supervised learning, minimize our classification errors
- Standard choice: Cross entropy loss function
  - Straightforward extension of logistic loss for binary

$$Loss(x, y; W, b) = -\sum_{k=1}^{K} (y = k) \log f(x)_k$$

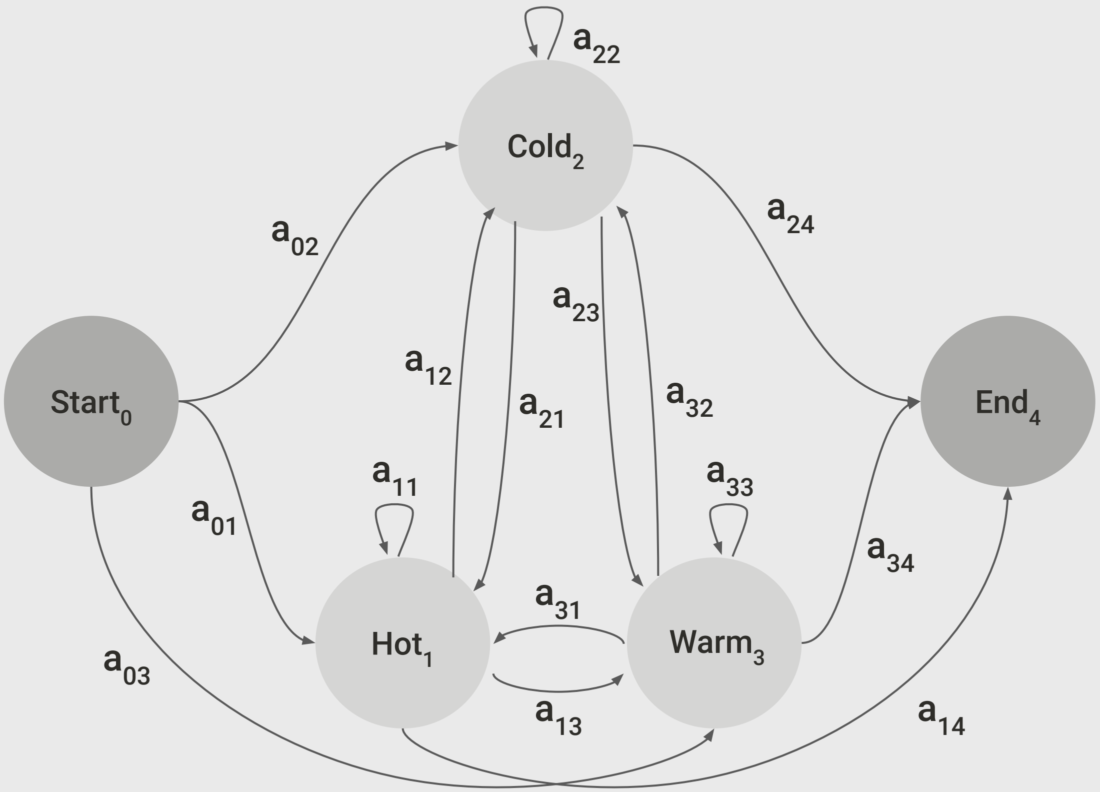
- This is a **frame-wise** loss. We use a label for each frame from a forced alignment
- Other loss functions possible. Can get deeper integration with the HMM or word error rate

Appendix: HMMs for Speech

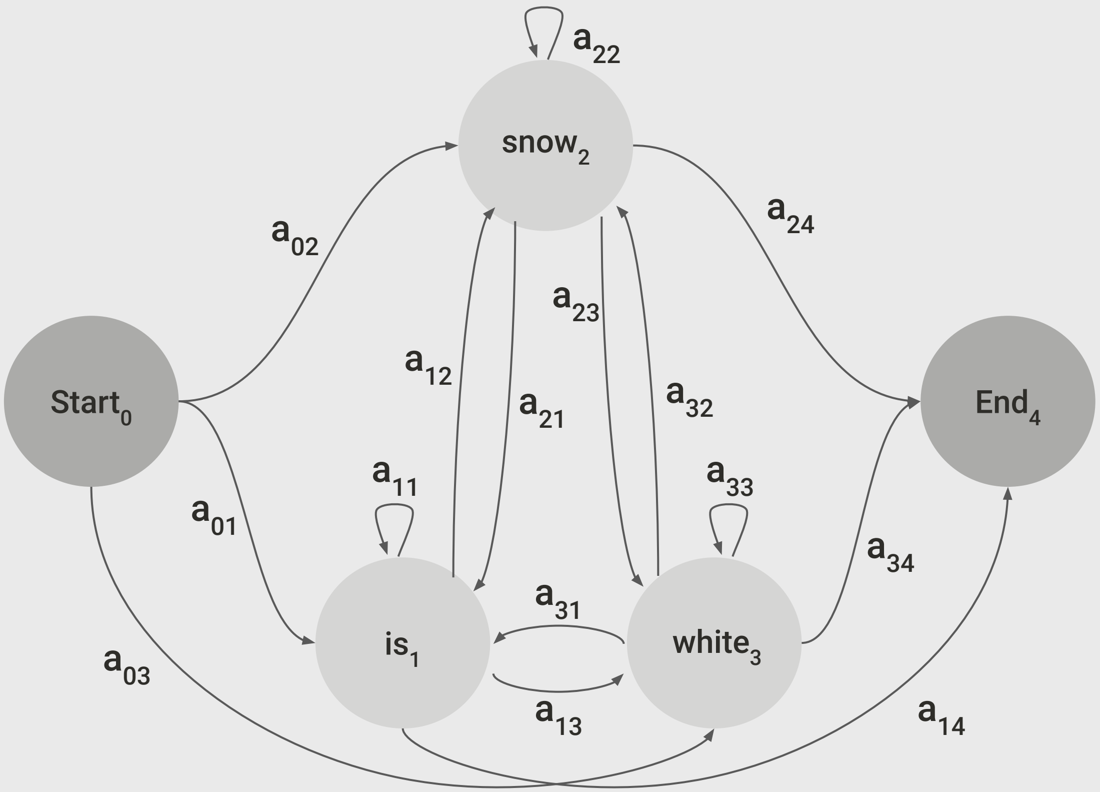
Lexicon

- A list of words
- Each one with a pronunciation in terms of phones
- We get these from an existing pronunciation dictionary
  - Default academic resource: **CMU dictionary:** 127K words
- We represent the lexicon as an HMM

Markov Chain for Weather



Markov Chain for Words



Markov Chain = First-order Observable Markov Model

- A set of states
  - $Q = q_1, q_2 \ldots q_{N;}$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities $A = a_{01}a_{02}\ldots a_{n1}\ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i,j \le N$$

$$\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \le i \le N$$

- Distinguished start and end states

Markov Chain = First-order Observable Markov Model

- Current state only depends on previous state
- Markov Assumption:

$$P(q_i \mid q_1 \cdots q_{i-1}) = P(q_i \mid q_{i-1})$$

Another Representation for Start State

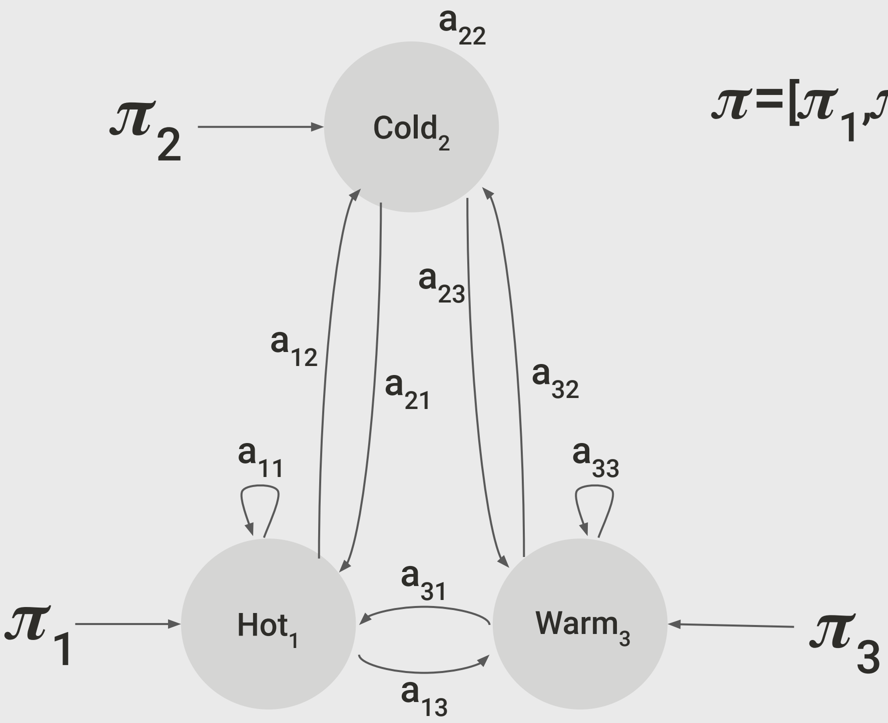
- Instead of start state
- Special initial probability vector π
  - An initial distribution over probability of start states

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

- Constraints:

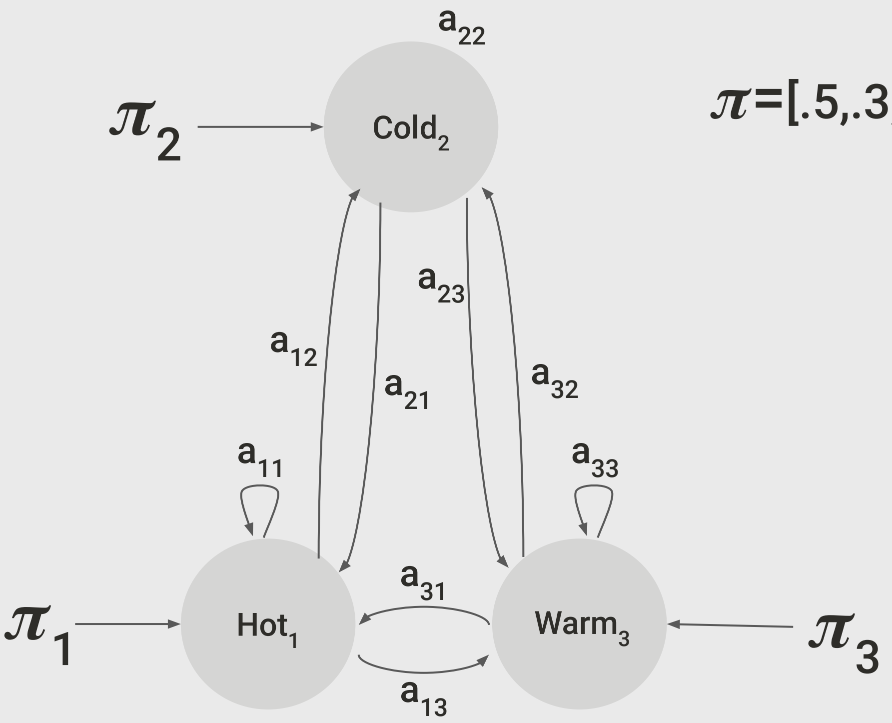
$$\sum_{j=1}^{N} \pi_j = 1$$

The Weather Figure Using π



$$\pi = [\pi_1, \pi_2, \pi_3]$$

The Weather Figure Using π



π=[.5,.3,.2]

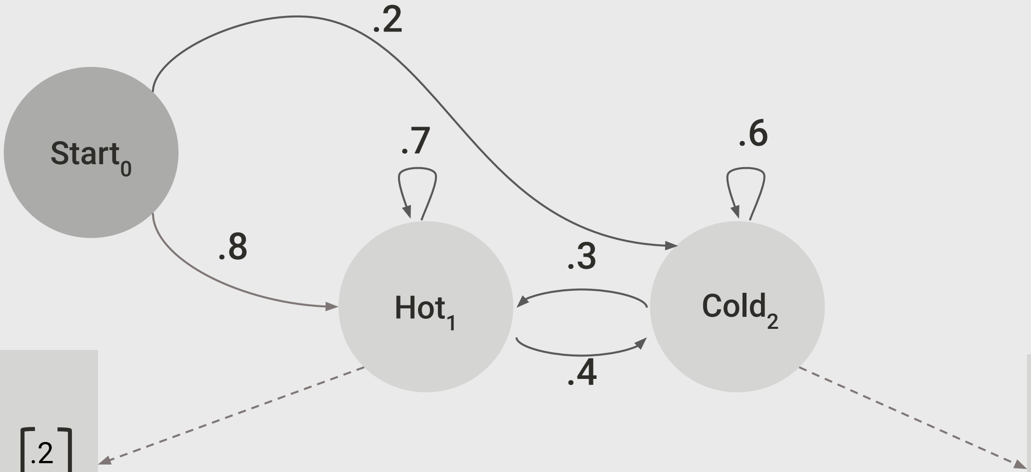
Hidden Markov Model

- For Markov chains, output symbols = state symbols
  - See **hot** weather: we're in state **hot**
- But not in speech recognition
  - Output symbols: vectors of acoustics (**cepstral features**)
  - Hidden states: **phones**
- So we need an extension!
- A **Hidden Markov Model** is an extension of a Markov chain in which the input symbols are not the same as the states
- This means **we don't know which state we are in**

HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can't find any records of the weather in Baltimore, MD for summer of 2008
- But you find Jason Eisner's diary
- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was

HMM for Ice Cream



B_1

$$\begin{bmatrix} P(1 \mid HOT) \\ P(2 \mid HOT) \\ P(3 \mid HOT) \end{bmatrix} = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix}$$

B_2

$$\begin{bmatrix} P(1 \mid COLD) \\ P(2 \mid COLD) \\ P(3 \mid COLD) \end{bmatrix} = \begin{bmatrix} .5 \\ .4 \\ .1 \end{bmatrix}$$

The Three Basic Problems for HMMs

Jack Ferguson at IDA in the 1960s

- **Problem 1 (Evaluation):** Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model F = (A,B), how do we efficiently compute P(O| Φ), the probability of the observation sequence, given the model?
- **Problem 2 (Decoding):** Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model Φ = (A,B), how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)?
- **Problem 3 (Learning):** How do we adjust the model parameters Φ = (A,B) to maximize P(O| Φ)?

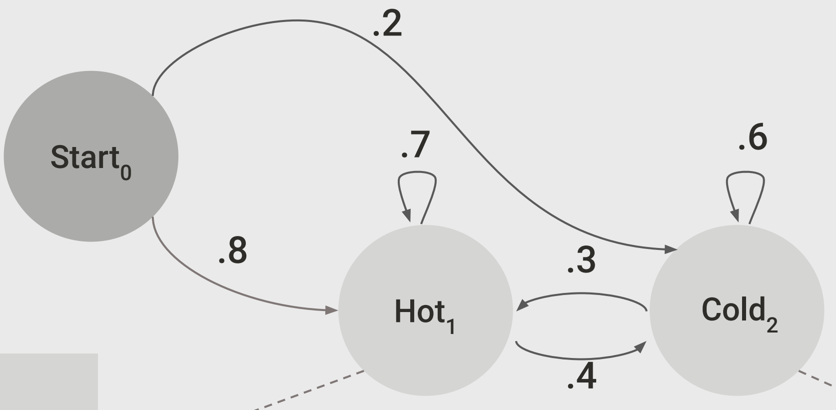
Decoding

- Given an observation sequence
  - 3 1 3
- And an HMM
- The task of the **decoder**
  - To find the best **hidden** state sequence
- Given the observation sequence $O=(o_1 o_2 \ldots o_T)$, and an HMM model $\Phi = (A,B)$, how do we choose a corresponding state sequence $Q=(q_1 q_2 \ldots q_T)$ that is optimal in some sense (i.e., best explains the observations)

HMM for Ice Cream Eisner Task

- **Given**
  - Observed Ice Cream Sequence:
    - 1,2,3,2,2,2,3…
- **Produce:**
  - Hidden Weather Sequence:
    - H,C,H,H,H,C…

HMM for Ice Cream



B_1

$$\begin{bmatrix} P(1 \mid HOT) \\ P(2 \mid HOT) \\ P(3 \mid HOT) \end{bmatrix} = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix}$$

B_2

$$\begin{bmatrix} P(1 \mid COLD) \\ P(2 \mid COLD) \\ P(3 \mid COLD) \end{bmatrix} = \begin{bmatrix} .5 \\ .4 \\ .1 \end{bmatrix}$$

Decoding

- **One possibility:**
  - For each hidden state sequence Q
  - HHH, HHC, HCH,
- **Compute P(O|Q)**
- **Pick the highest one**
- **Why not?**
  - N^T
- **Instead:**
  - The Viterbi algorithm
  - Is a dynamic programming algorithm
  - Uses a similar trellis to the Forward algorithm

Viterbi Intuition

- We want to compute the joint probability of the observation sequence together with the best state sequence

$$v_t(j) = \max_{q_0,q_1,\ldots,q_{t-1}} P(q_0,q_1 \ldots q_{t-1},o_1,o_2 \ldots o_t,q_t = j|\lambda)$$

$$v_t(j) \;=\; \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

Viterbi Recursion

1. **Initialization:**

$$v_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N$$
$$bt_1(j) = 0$$

2. **Recursion** (recall that states 0 and q_F are non-emitting):

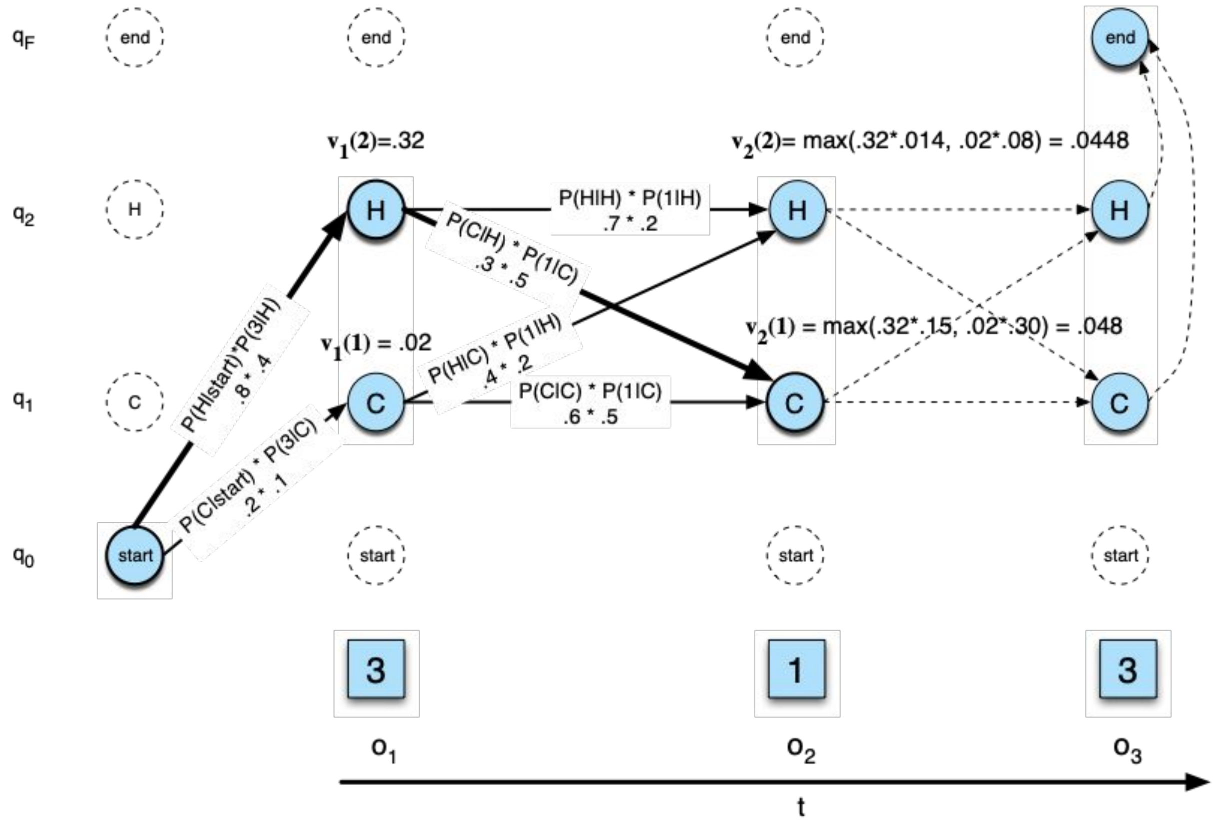
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$
$$bt_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. **Termination:**

The best score: $P* = v_t(q_F) = \max_{i=1}^{N} v_T(i) * a_{i,F}$

The start of backtrace: $q_T* = bt_T(q_F) = \operatorname*{argmax}_{i=1}^{N} v_T(i) * a_{i,F}$

The Viterbi Trellis



Viterbi Intuition

- **Process observation sequence left to right**
  - Filling out the trellis
  - Each cell:

$$v_t(j) = \max_{q_0, q_1, \ldots, q_{t-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j | \lambda)$$

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

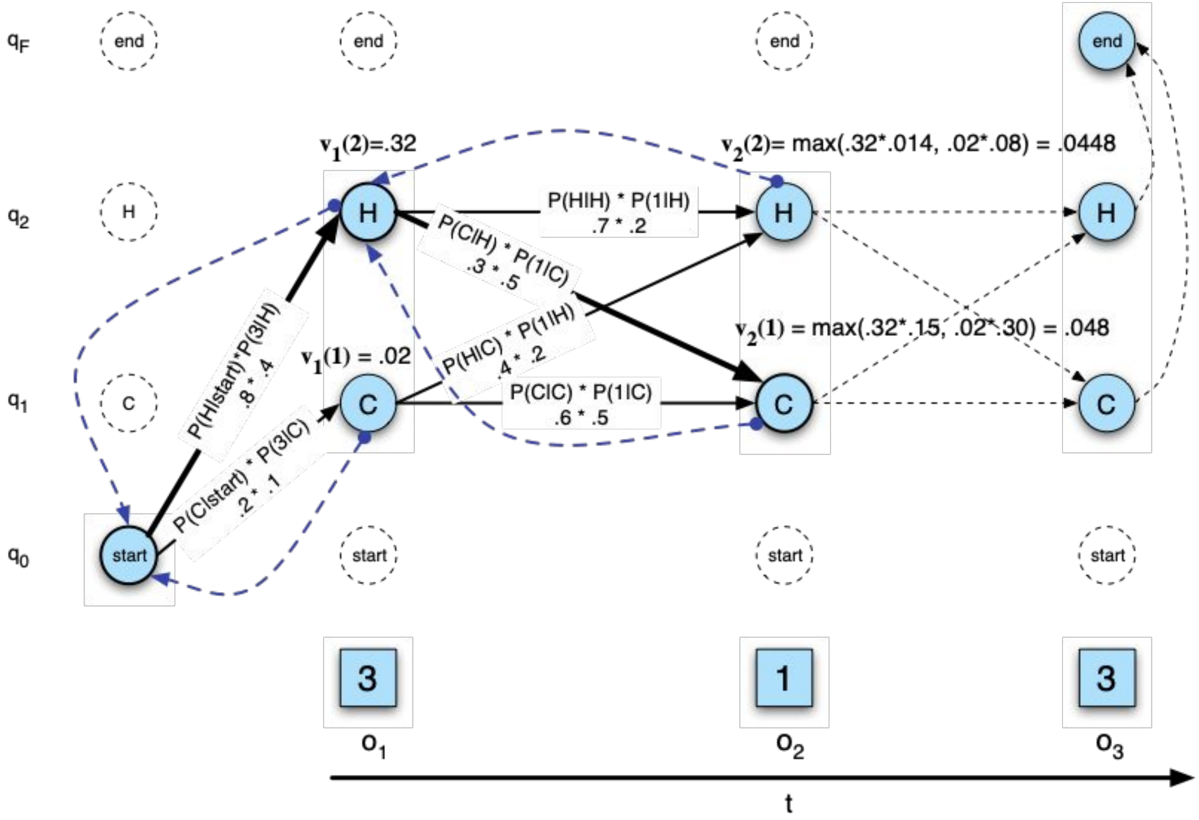
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Viterbi Algorithm

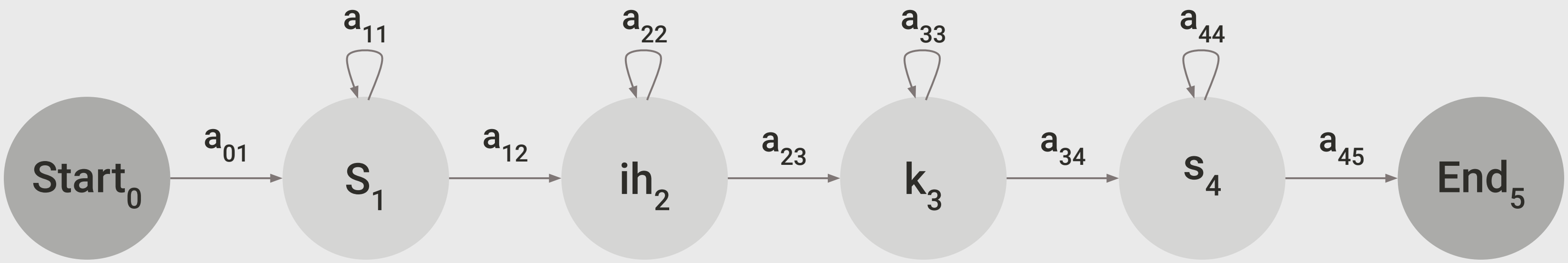
function VITERBI(*observations* of len T, *state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$$
$$backpointer[s,1] \leftarrow 0$$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$$
$$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$$; termination step
$$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

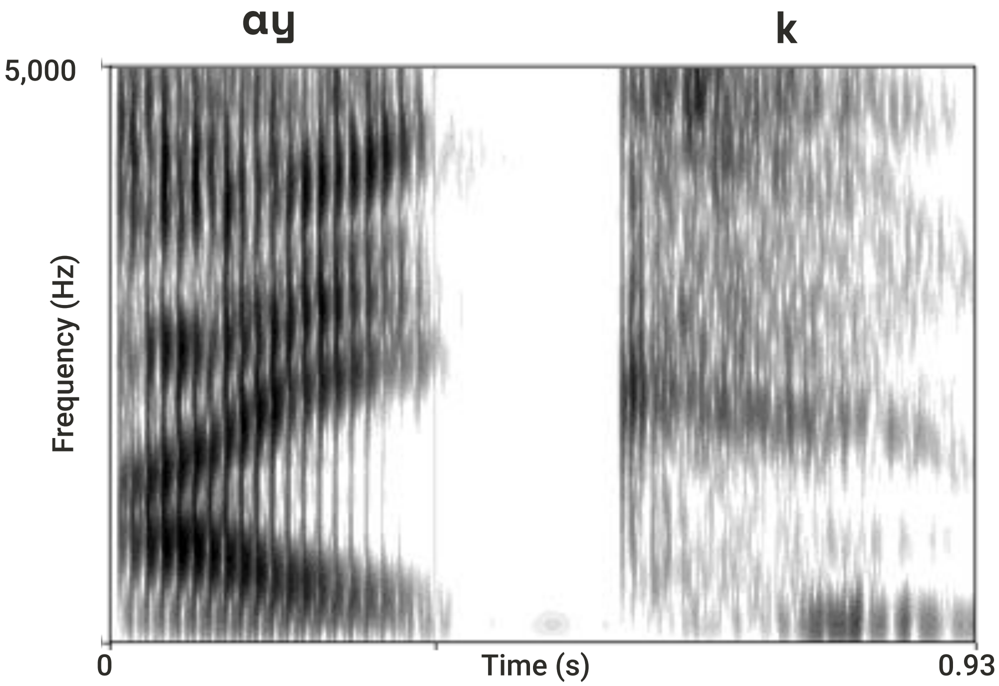
The Viterbi Trellis



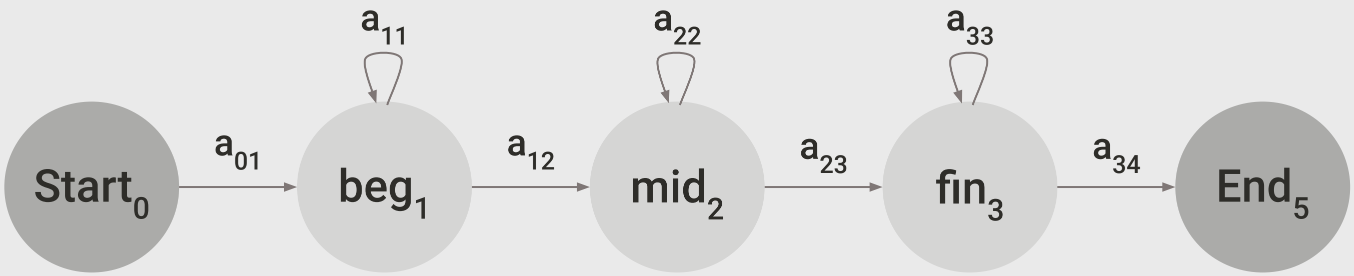
HMMs for Speech



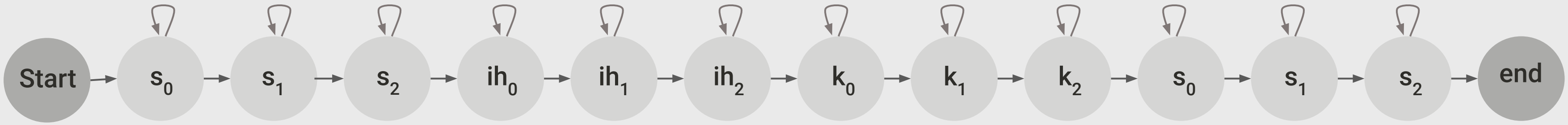
Phones are Not Homogeneous!
Phone-level HMMs Not Enough



Each Phone Has 3 Subphones



Resulting HMM Word Model for “six”



Viterbi Intuition

- **Process observation sequence left to right**
  - Filling out the trellis
  - Each cell:

$$v_t(j) = \max_{q_0,q_1,\ldots,q_{t-1}} P(q_0,q_1\ldots q_{t-1},o_1,o_2\ldots o_t,q_t = j|\lambda)$$

$$v_t(j) \;=\; \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Appendix: MFCC computation details

Discrete Representation of Signal

Represent continuous signal into discrete form

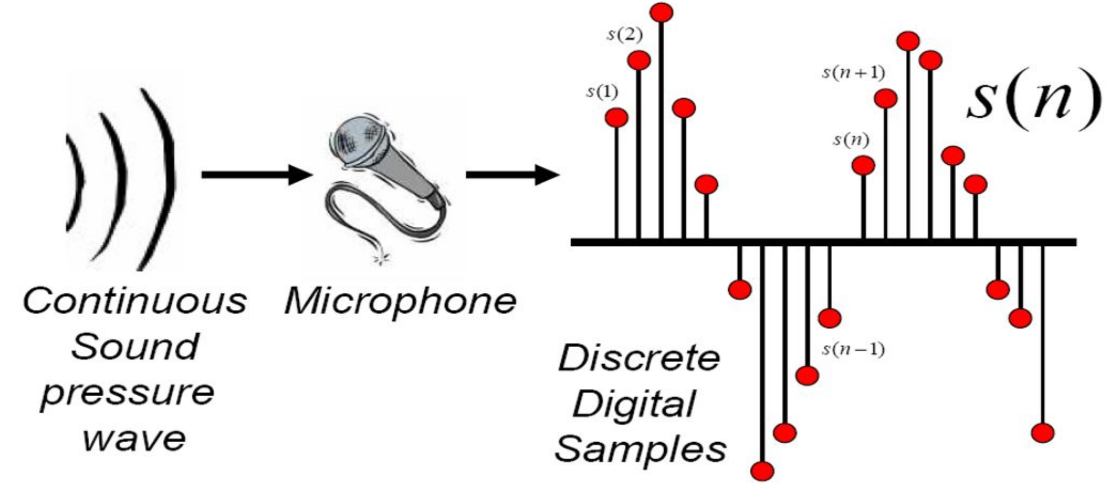
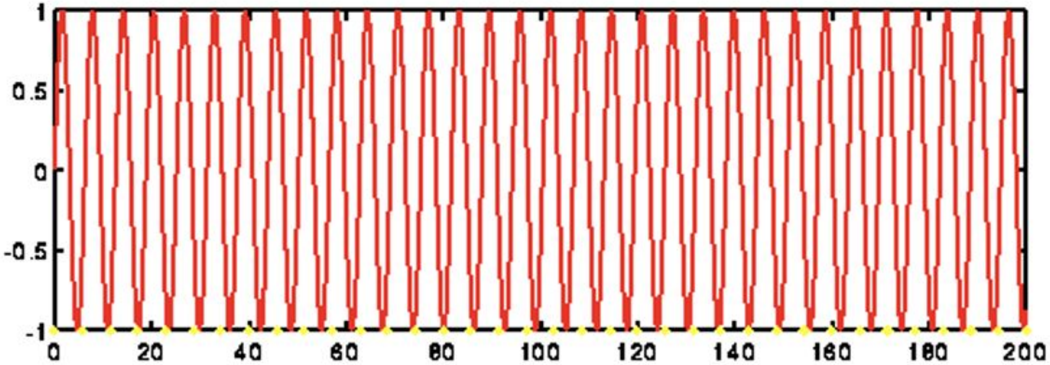


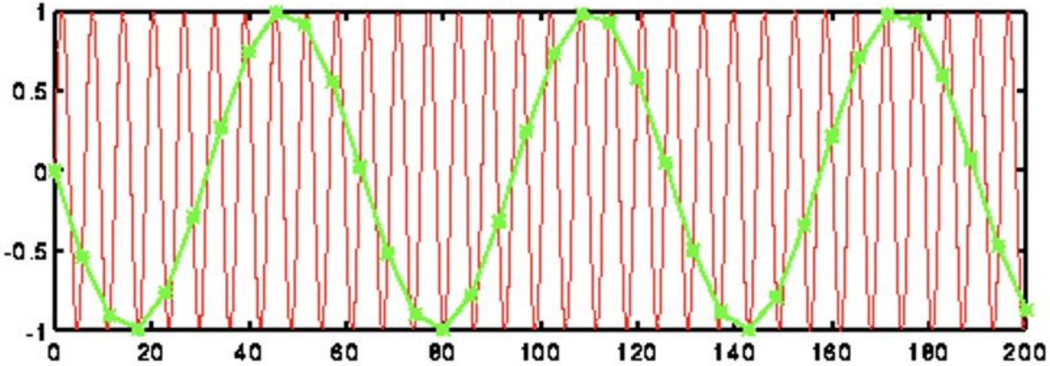
Figure: Bryan Pellom

Discrete Representation of Signal

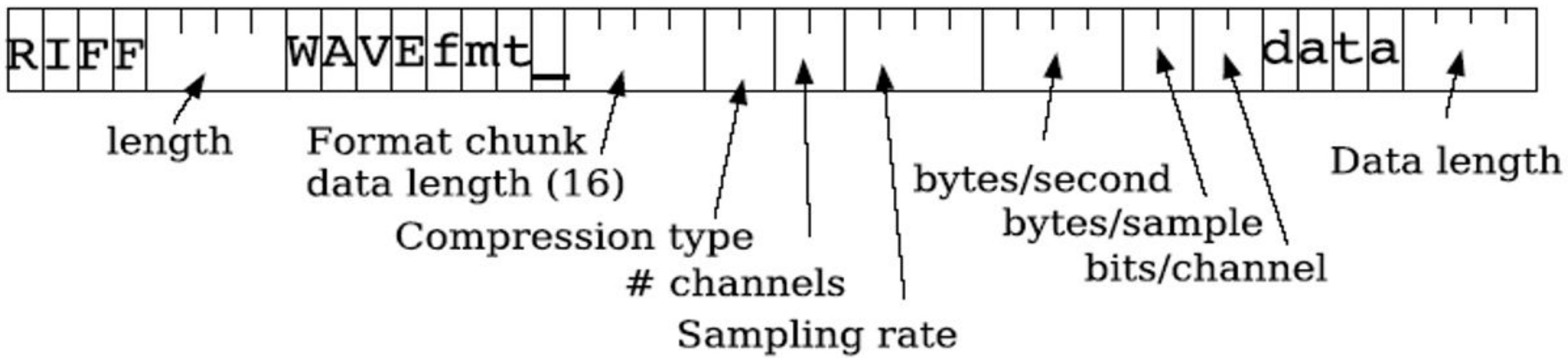
If measure at green dots, will see a lower frequency wave and miss the correct higher frequency one!



Original signal in red



WAV Format



- Many formats, trade-offs in compression, quality
- Nice sound manipulation tool: Sox
  - http://sox.sourceforge.net/
  - convert speech formats

Windowing

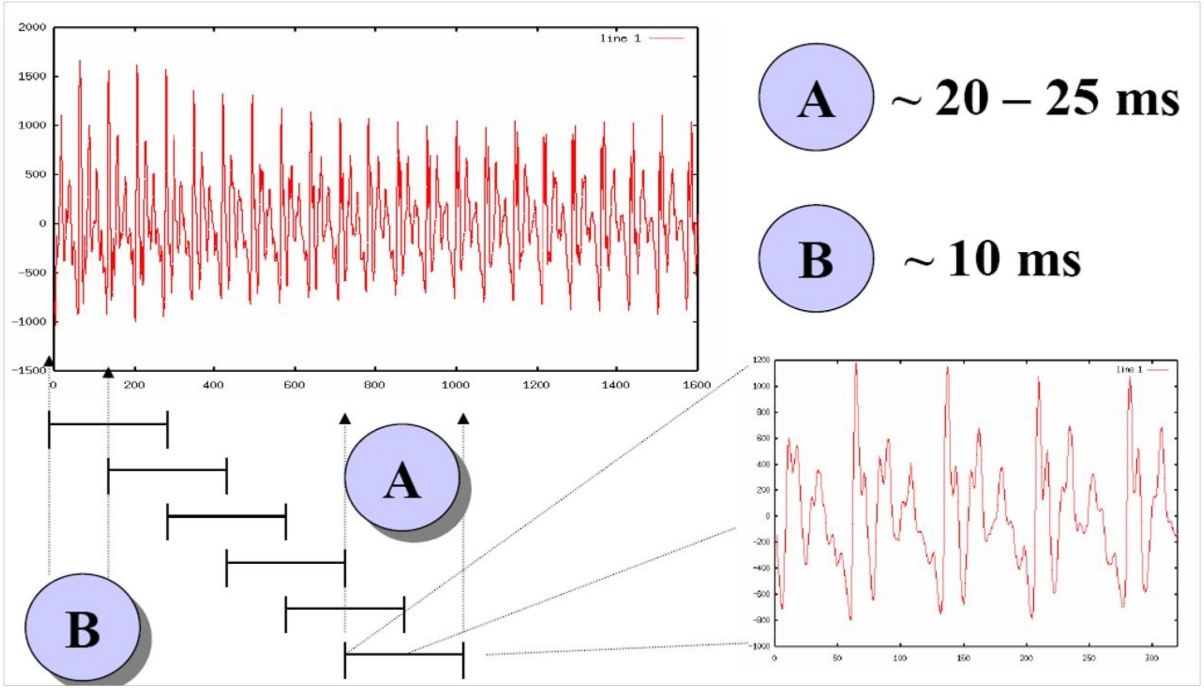
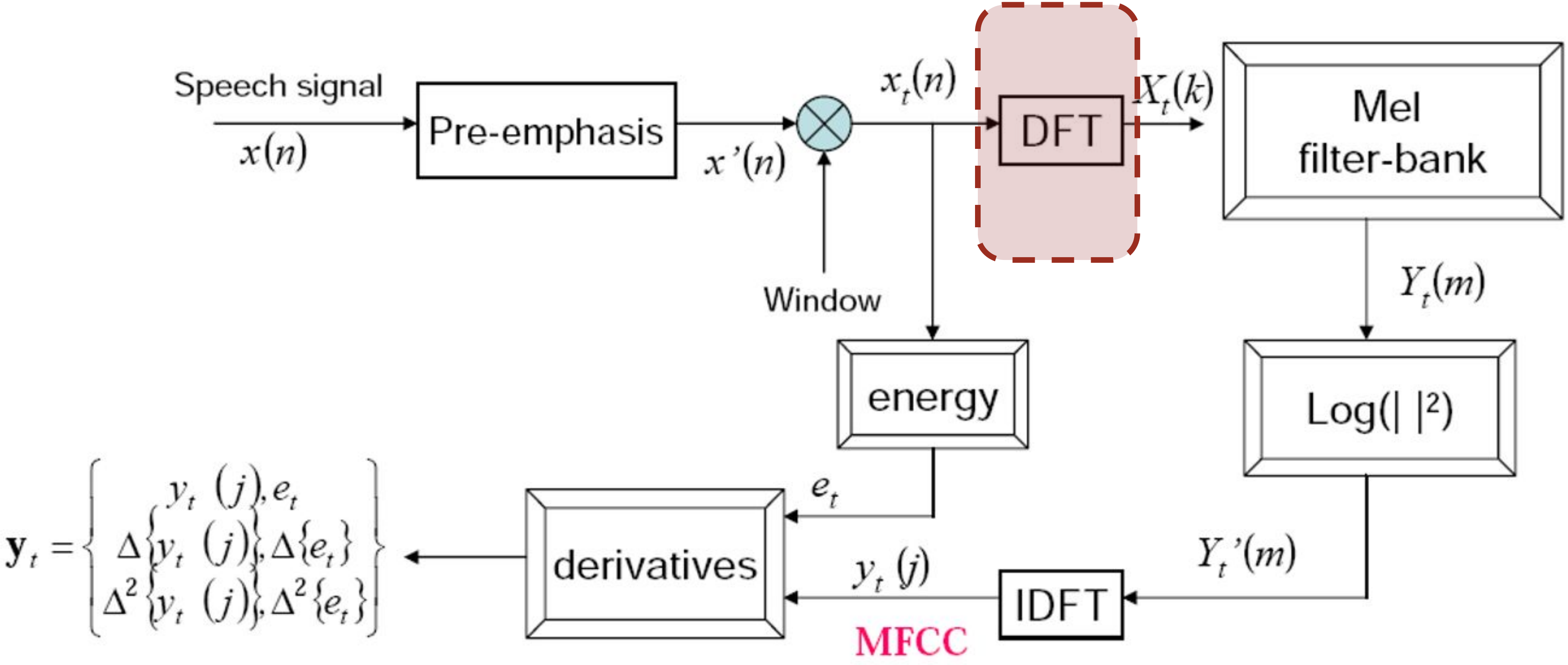


Figure: Bryan Pellom

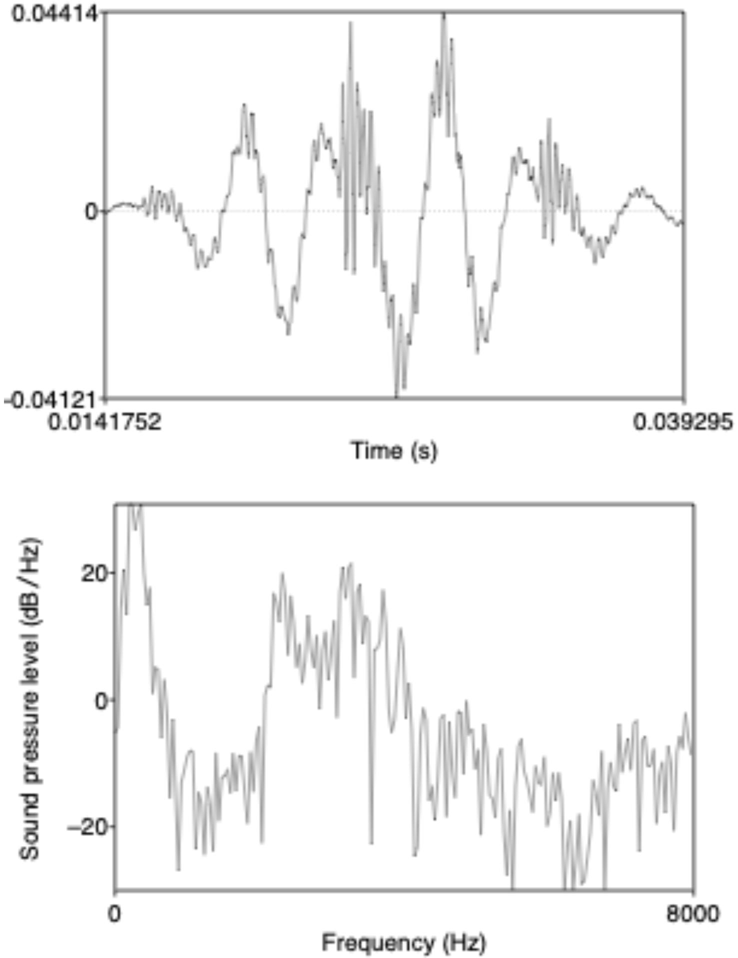
MFCC


Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
energy
Log(| |²)
e_t
derivatives
$Y_t'(m)$
$y_t(j)$
IDFT
MFCC
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$

Discrete Fourier Transform Computing a Spectrum

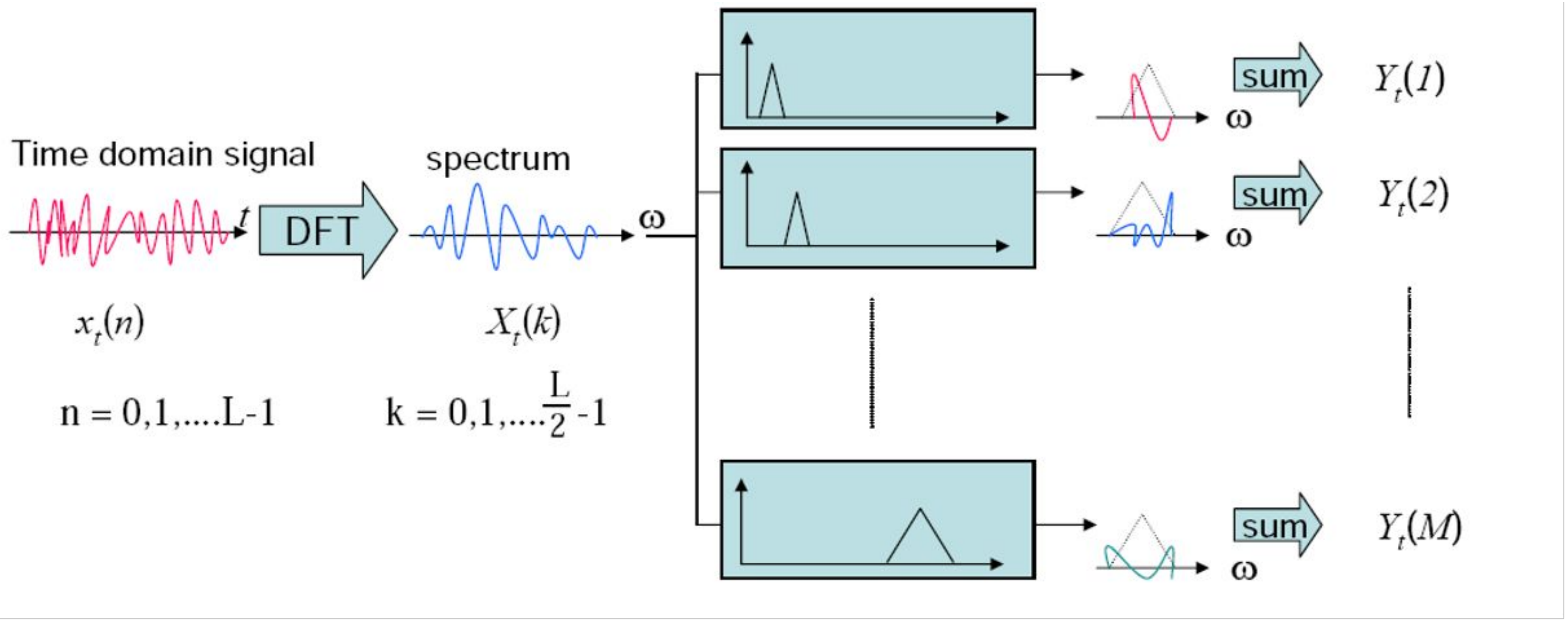
A 25 ms Hamming-windowed signal from [iy]

- And its spectrum as computed by DFT (plus other smoothing)

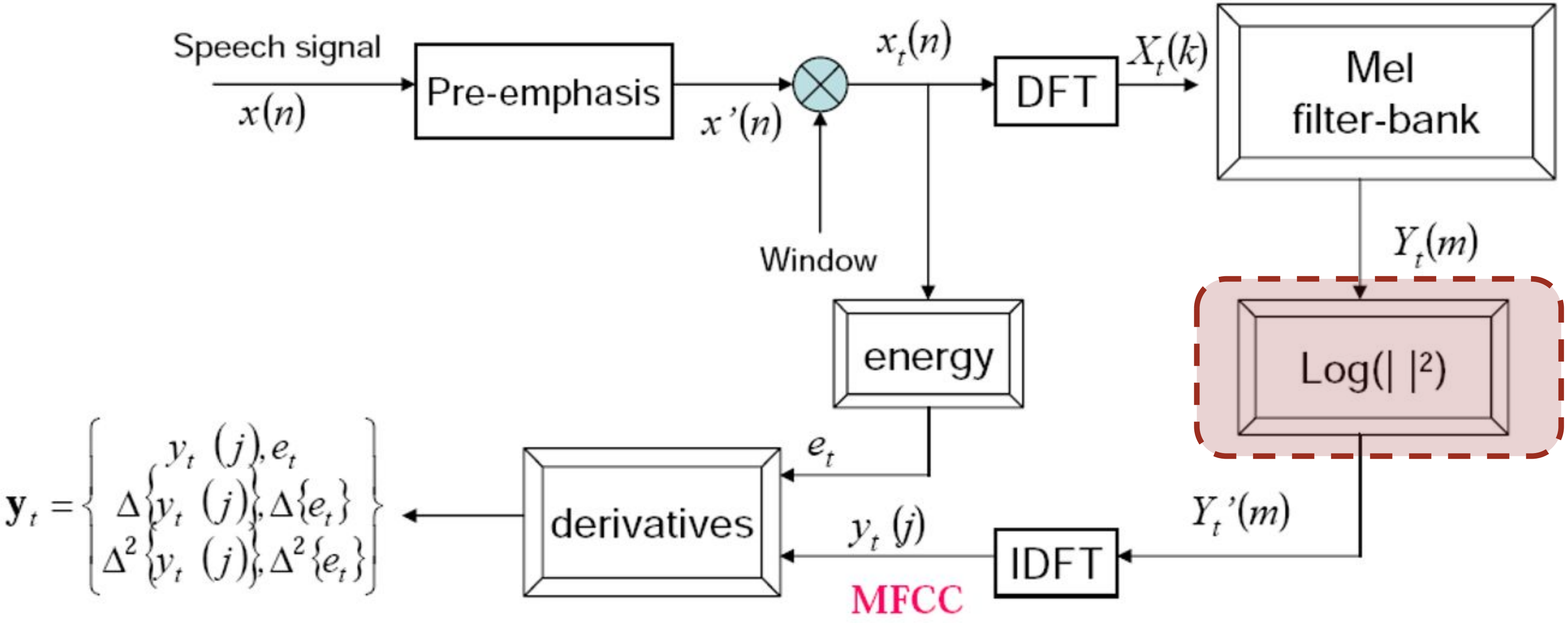


Mel-filter Bank Processing

- Apply the bank of Mel-scaled filters to the spectrum
- Each filter output is the sum of its filtered spectral components

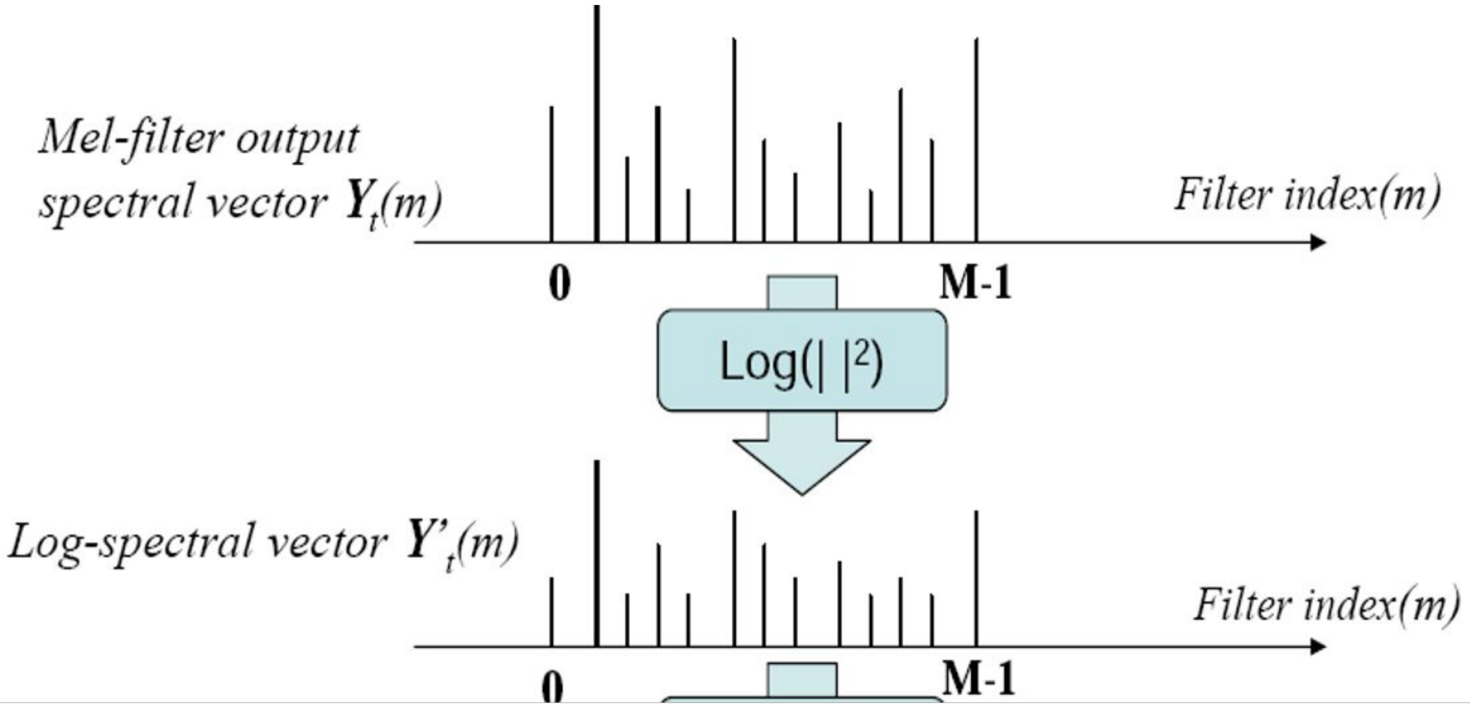


MFCC

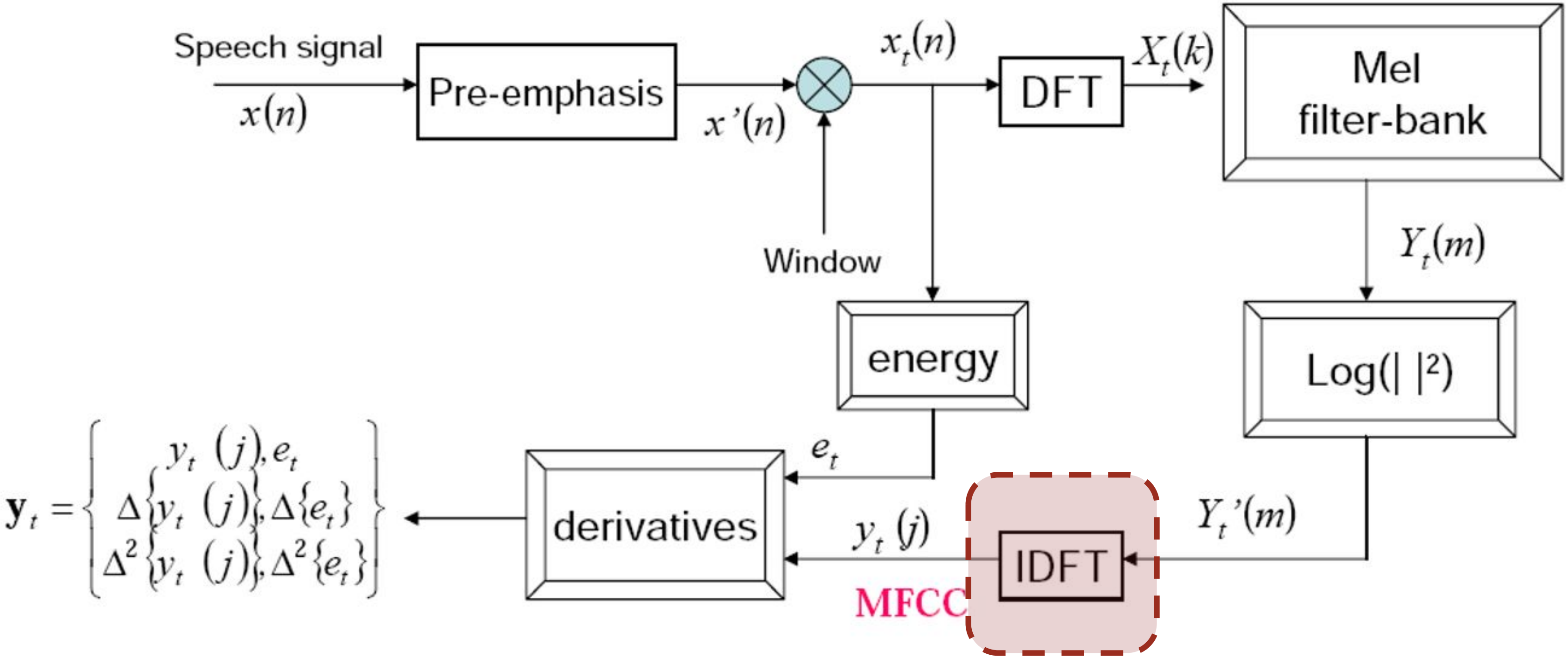

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \left\{ \begin{array}{c} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{array} \right\}$

Log Energy Computation

- Compute the logarithm of the square magnitude of the output of Mel-filter bank



MFCC

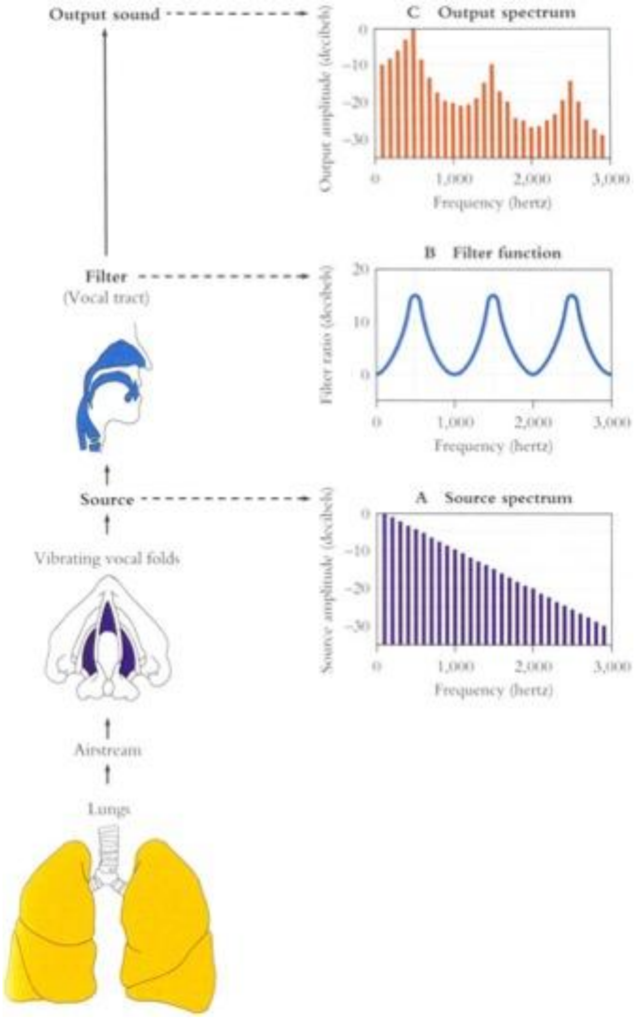

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
MFCC
$y_t(j)$
energy
e_t
derivatives
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$

The Cepstrum

One way to think about this:

- **Separating the source and filter**
- **Speech waveform is created by**
  - A glottal source waveform
  - Passes through a vocal tract which because of its shape has a particular filtering characteristic
- **Remember articulatory facts from lecture 2:**
  - The vocal cord vibrations create harmonics
  - The mouth is an amplifier
  - Depending on shape of oral cavity, some harmonics are amplified more than others

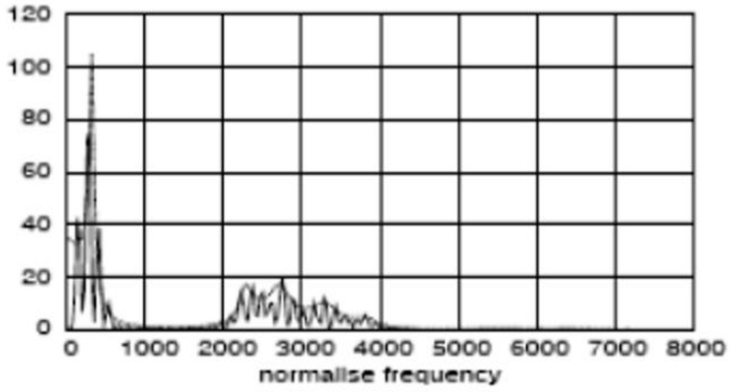
George Miller Figure

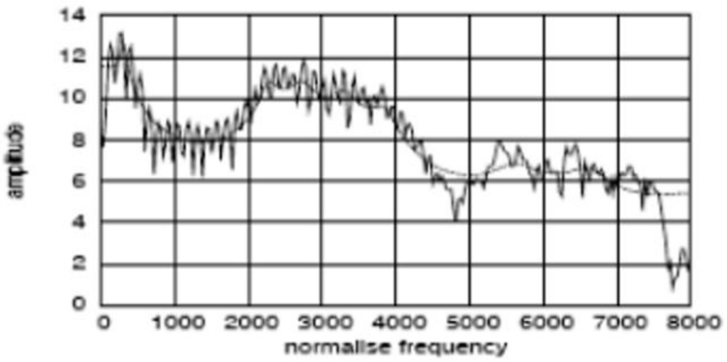


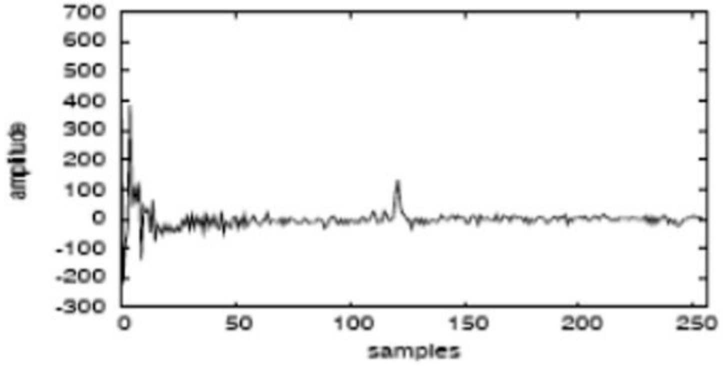
We Care About the FIlter Not the Source

- Most characteristics of the source
  - F0
  - Details of glottal pulse
- Don’t matter for phone detection
- What we care about is the filter
  - The exact position of the articulators in the oral tract
- So we want a way to separate these
  - And use only the filter function

The Cepstrum







Another Advantage of the Cepstrum

- MDCT produces highly uncorrelated features
- If we use only the diagonal covariance matrix for our Gaussian mixture models, we can only handle uncorrelated features.
- In general we'll just use the first 12 cepstral coefficients (we don't want the later ones which have e.g. the F0 spike)