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)
  - 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 Hub5’00)

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

WSJ
(Possibly trained on more data than WSJ.)

<table>
<thead>
<tr>
<th>WER eval/’92</th>
<th>WER eval/’93</th>
<th>Paper</th>
<th>Published</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.03%</td>
<td>8.08%</td>
<td>Humans Deep Speech 2: End-to-End Speech Recognition in English and Mandarin</td>
<td>December 2015</td>
</tr>
<tr>
<td>2.9%</td>
<td></td>
<td>End-to-end Speech Recognition Using Lattice-Free MMI</td>
<td>September 2018</td>
</tr>
<tr>
<td>3.10%</td>
<td></td>
<td>Deep Speech 2: End-to-End Speech Recognition in English and Mandarin</td>
<td>December 2018</td>
</tr>
</tbody>
</table>

LibriSpeech
(Possibly trained on more data than LibriSpeech.)

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

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

Source Sentence:
If music be the food of love...

Guess at Source
If music be the food of love...

Noisy Channel

Noisy 1
Noisy 2
Noisy n

Decoder

Every happy family
In a hole in the ground
... If music be the food of love
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
  - \[ O = o_1,o_2,o_3,...,o_t \]

- Define a sentence as a sequence of words:
  - \[ W = w_1,w_2,w_3,...,w_n \]
The Noisy Channel Model

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

$$\hat{W} = \arg\max_{W \in L} P(W \mid O)$$

- We can use Bayes rule to rewrite this:

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

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

$$\hat{W} = \arg\max_{W \in L} P(O \mid W)P(W)$$
The Noisy Channel Model

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

Source Sentence:
If music be the food of love…

Guess at Source
If music be the food of love…

Noisy Channel

Decoder

Every happy family
In a hole in the ground
If music be the food of love
The Noisy Channel Model

\[ \hat{W} = \arg \max_{W \in L} P(O \mid W) P(W) \]
Speech Recognition Architecture

Prior \( P(W) \)

\( P(O|W) \)

\( O \)

MFCC Features

Gaussian Mixture
Acoustic Model

Phone Likelihoods

HMM Lexicon

Viterbi Decoder

If music be the food of love…
Speech Architecture Meets Noisy Channel

P(O|W)

Acoustic Model + Lexicon

Feature Extraction

Language Model

Decoding Search

P(W)

O W
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:

<table>
<thead>
<tr>
<th>REF:</th>
<th>i *** ** UM the PHONE IS</th>
<th>i LEFT THE portable **** PHONE UPSTAIRS last night</th>
</tr>
</thead>
<tbody>
<tr>
<td>HYP:</td>
<td>i GOT IT TO the ***** FULLEST i LOVE TO portable FORM OF STORES last night</td>
<td></td>
</tr>
<tr>
<td>Eval:</td>
<td>I I S D S S S I S S</td>
<td></td>
</tr>
</tbody>
</table>
Word error rate (WER)

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:

<table>
<thead>
<tr>
<th>REF:</th>
<th>it was</th>
<th>the best of times</th>
<th>it was the worst of times</th>
<th>it was</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS A:</td>
<td>ITS</td>
<td>the best of times</td>
<td>IS the worst of times</td>
<td>OR it was</td>
</tr>
<tr>
<td>SYS B:</td>
<td>it was</td>
<td>the best</td>
<td>times it</td>
<td>WON the TEST of times</td>
</tr>
</tbody>
</table>
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
Sub-phones: 942 – 6 – 37 – 8006 – 4422 ...

Hidden Markov Model (HMM):
- 942
- 942
- 6

Acoustic Model:
Audio Input:
- Features
- Features
- Features

GMM models:
\[ P(o|q) \]
- o: input features
- q: HMM state
Hidden Markov Model (HMM) basics

- Markov assumption:
  \[ P(q_t | q_1 \ldots q_{t-1}) = P(q_t | q_{t-1}) \]

- Output-independence assumption:
  \[ P(o_t | O_1^{t-1}, q_1^t) = P(o_t | q_t) \]

- A set of \( N \) states

- A transition probability matrix \( A \), each \( a_{ij} \) representing the probability of moving from state \( i \) to state \( j \), s.t. \( \sum_{j=1}^{n} a_{ij} = 1 \) \( \forall i \)

- A sequence of \( T \) observations, each one drawn from a vocabulary \( V = v_1, v_2, \ldots, v_V \)

- A sequence of observation likelihoods, also called emission probabilities, each expressing the probability of an observation \( o_t \) being generated from a state \( i \)

- 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

\[
\Delta y_t(j) = \frac{\sum_{m=p}^{p} m \cdot y_{t-m}(j)}{\sum_{m=p}^{p} m^2}
\]

\[
\Delta^2 y_t(j) = \frac{\sum_{m=p}^{p} m \cdot \Delta y_{t-m}(j)}{\sum_{m=p}^{p} m^2}
\]
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:

Different Means

$P(o|q)$ is highest here at mean

$P(o|q)$ is lowest here at mean
GMMs

Summary: each state has a likelihood function parameterized by:

- $M$ Mixture weights
- $M$ Mean Vectors of dimensionality $D$
- either:
  - $M$ Covariance Matrices of $D \times D$
- or more likely:
  - $M$ Diagonal Covariance Matrices of $D \times D$
  - which is equivalent to
  - $M$ Variance Vectors of dimensionality $D$
Phonetic Context: Different “eh”s

![Speech Waveform Diagram](image-url)
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: $50 \times 50 \times 50 = 125,000$
- 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

5,000 Hz

w iy r iy m iy n iy

820.08 Hz

0 Hz
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

Phone /ih/ beg. state

Left nasal?

Yes

Left fricative?

No

Right liquid?

Yes

Cluster A:
n-ih+l₀
ng-ih+l₀
m-ih+l₀

No

Cluster B:
n-ih+r₀
ng-ih+r₀
m-ih+r₀
n-ih+w₀

...
# Triphone Decision Tree Clustering

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

Transcription: Nine four oh two two

Wavefile:

Lexicon:
- one
- two
- three
- eight
- nine
- zero
- oh

Feature Extraction:

Raw HMM:

Feature Vectors:
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

Use a DNN to approximate: 
\[ P(s|x) \]

Apply Bayes' Rule:
\[ P(x|s) = P(s|x) \times P(x) / P(s) \]

DNN * Constant / State prior

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

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input: Features \( x_1 \) Features \( x_2 \) Features \( x_3 \)
Objective Function for Learning

- Supervised learning, minimize our classification errors
- Standard choice: Cross entropy loss function
  - Straightforward extension of logistic loss for binary
  
  \[
  \text{Loss}(x, y; W, b) = -\sum_{k=1}^{K} (y = k) \log f(x)_k
  \]

- 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 \( A = a_{01}, a_{02}, \ldots, a_{n1}, a_{n2}, \ldots, a_{nn} \)
  - Each \( a_{ij} \) represents the probability of transitioning from state \( i \) to state \( j \)
  - The set of these is the transition probability matrix \( A \)

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

\[
\sum_{j=1}^{N} a_{ij} = 1; \quad 1 \leq i \leq N
\]

- 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 \( \pi \)
  - An initial distribution over probability of start states

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

- Constraints:

\[ \sum_{j=1}^{N} \pi_j = 1 \]
The Weather Figure Using $\pi$

$\pi = [\pi_1, \pi_2, \pi_3]$
The Weather Figure Using $\pi$

$\pi_2 \rightarrow \text{Cold}_2$  

$\pi = [0.5, 0.3, 0.2]$  

$\pi_1 \rightarrow \text{Hot}_1$  

$\pi_3 \rightarrow \text{Warm}_3$
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

\[
\begin{align*}
B_1 &= \begin{bmatrix}
P(1 \mid \text{HOT}) \\
P(2 \mid \text{HOT}) \\
P(3 \mid \text{HOT})
\end{bmatrix} = \begin{bmatrix}
.2 \\
.4 \\
.4
\end{bmatrix} \\
B_2 &= \begin{bmatrix}
P(1 \mid \text{COLD}) \\
P(2 \mid \text{COLD}) \\
P(3 \mid \text{COLD})
\end{bmatrix} = \begin{bmatrix}
.5 \\
.4 \\
.1
\end{bmatrix}
\end{align*}
\]
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 \( P(O|\Phi) \), 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 \( \Phi = (A, B) \), 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)?

- **Problem 3 (Learning):** How do we adjust the model parameters \( \Phi = (A, B) \) to maximize \( P(O|\Phi) \)?
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:
HMM for Ice Cream

\[
B_1 = \begin{bmatrix}
P(1 \mid \text{HOT}) \\
P(2 \mid \text{HOT}) \\
P(3 \mid \text{HOT})
\end{bmatrix} = \begin{bmatrix}
.2 \\
.4 \\
.4
\end{bmatrix}
\]

\[
B_2 = \begin{bmatrix}
P(1 \mid \text{COLD}) \\
P(2 \mid \text{COLD}) \\
P(3 \mid \text{COLD})
\end{bmatrix} = \begin{bmatrix}
.5 \\
.4 \\
.1
\end{bmatrix}
\]
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

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

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

\[ v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, 1 < t \leq T \]
\[ bt_t(j) = \arg\max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 \leq j \leq N, 1 < t \leq T \]

3. Termination:

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

The start of backtrace: \( q_T^* = bt_T(q_F) = \arg\max_{i=1}^{N} v_T(i) * a_{i,F} \)
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

```python
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] ← a_{0,s} * b_s(o_1)
    backpointer[s,1] ← 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] ← \max_{s' = 1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)
        backpointer[s,t] ← \arg\max_{s' = 1}^{N} viterbi[s',t-1] * a_{s',s}
    viterbi[q_F,T] ← \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F} ; termination step
    backpointer[q_F,T] ← \arg\max_{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

\[ v_1(2) = 0.32 \]

\[ v_2(1) = \max(0.32 \times 0.14, 0.02 \times 0.08) = 0.0448 \]

\[ v_2(2) = \max(0.32 \times 0.15, 0.02 \times 0.30) = 0.048 \]
HMMs for Speech
Phones are Not Homogeneous!
Phone-level HMMs Not Enough

\[ \alpha y \]

\[ k \]
Each Phone Has 3 Subphones

```
Start₀ → a₀₁ → beg₁ → a₁₂ → mid₂ → a₂₃ → fin₃ → a₃₄ → End₅
```
Resulting HMM Word Model for “six”
Viterbi Intuition

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

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

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

- \(v_{t-1}(i)\) the previous \textbf{Viterbi path probability} from the previous time step
- \(a_{ij}\) the \textbf{transition probability} from previous state \(q_i\) to current state \(q_j\)
- \(b_j(o_t)\) the \textbf{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!
WAV Format

- Many formats, trade-offs in compression, quality
- Nice sound manipulation tool: Sox
  - convert speech formats
Windowing

Figure: Bryan Pellom

A \sim 20 - 25 \text{ ms}

B \sim 10 \text{ ms}
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

\[
x_t(n) \quad \quad X_t(k)
\]

Time domain signal

\[
0, 1, \ldots, L-1 \quad 0, 1, \ldots, \frac{L}{2} - 1
\]

DFT

\[
Y_t(1) \quad Y_t(2) \quad \ldots \quad Y_t(M)
\]

Sum

\[
\sum
\]

Spectrum
Speech signal $x(n)$ is processed through pre-emphasis to become $x'(n)$, which is then transformed using the DFT to become $X_t(k)$. The Mel filter-bank takes $X_t(k)$ and produces $Y_t(m)$. The energy of $Y_t(m)$ is calculated, and the logarithm of the squared magnitude is taken to form the MFCC $Y_t'(m)$. The MFCC is obtained from $Y_t'(m)$ using the IDFT.
Log Energy Computation

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

\[ \text{Log Energy Computation} \]

\[ \text{Mel-filter output spectral vector } Y_t(m) \]

\[ \text{Filter index(m)} \]

\[ \text{Log-spectral vector } Y'_t(m) \]

\[ \text{Filter index(m)} \]
MFCC

Speech signal \( x(n) \) → Pre-emphasis → \( x'(n) \) → DFT → \( X_t(k) \)

Window → energy → Log(\(| |^2\)) → \( Y_t(m) \)

\[ y_t = \left\{ y_t(j), e_t, \Delta y_t(j), \Delta \{ e_t \}, \Delta^2 y_t(j), \Delta^2 \{ e_t \} \right\} \]

\[ e_t, y_t(j), Y_t'(m) \]

IDFT → MFCC
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)