Lecture 8: ASR: Noisy Channel, HMMs, Evaluation
Outline for Today

- ASR Architecture
- Hidden Markov Model (HMM) introduction
- HMMs for speech
- Evaluation with word error rate

- HW1 grades published on Gradescope
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**

**decoder**

Every happy family In a hole in the ground...
If music be the food of love...

**noisy sentence**

noisy 1
noisy 2
noisy N
The Noisy Channel Model

- What is the most likely sentence out of all sentences in the language L 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:
  - \( W = w_1, w_2, w_3, ..., w_n \)
Noisy Channel Model (III)

- Probabilistic implication: Pick the highest prob $S$:

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

- 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:

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

Speech recognition involves several key components:

1. **Cepstral feature extraction**
2. **MFCC features**
3. **Gaussian Acoustic Model**
4. **Phone likelihoods**
5. **HMM lexicon**
6. **Viterbi Decoder**
7. **N-gram language model**

The process starts with the input waveform (O) and proceeds through each step to output the most probable sequence of words (W) as in the phrase: "if music be the food of love..."
Noisy channel model

\[ \hat{W} = \arg \max_{W \in L} P(O | W) P(W) \]
The noisy channel model

Ignoring the denominator leaves us with two factors: $P(\text{Source})$ and $P(\text{Signal} | \text{Source})$
Speech Architecture meets Noisy Channel
**HMM-GMM System**

**Transcription:** Samson

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

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

**Hidden Markov Model (HMM):**

- **Features:** 942 → 942 → 6

**Acoustic Model:**

GMM models:

\[ P(o|q) \]

- o: input features
- q: HMM state

**Audio Input:**

Features Features Features
Decoding Architecture: five easy pieces

- 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
Lexicon

• A list of words
• Each one with a pronunciation in terms of phones
• We get these from on-line pronunciation dictionary
• CMU dictionary: 127K words
  • http://www.speech.cs.cmu.edu/cgi-bin/cmudict
• We’ll represent the lexicon as an HMM
HMMs for speech
Markov chain for weather
Markov chain for words
Markov chain =
First-order observable Markov Model

- a set of states
  - \( Q = q_1, q_2...q_N; \) the state at time \( t \) is \( q_t \)

- Transition probabilities:
  - a set of probabilities \( A = a_{01}a_{02}...a_{n1}...a_{nn}. \)
  - Each \( a_{ij} \) represents the probability of transitioning from state \( i \) to state \( j \)
  - The set of these is the transition probability matrix \( A \)
    \[
    a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \leq i, j \leq N
    \]
    \[
    \sum_{j=1}^{N} a_{ij} = 1; \quad 1 \leq i \leq N
    \]

- 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: specific example
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.
Hidden Markov Models

\( Q = q_1 q_2 \ldots q_N \)  
\( A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nm} \)  
\( O = o_1 o_2 \ldots o_T \)  
\( B = b_i(o_t) \)  

- 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 \( i \)

\( q_0, q_F \)

- 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
Assumptions

- **Markov assumption:**

  \[ P(q_i | q_1 \cdots q_{i-1}) = P(q_i | q_{i-1}) \]

- **Output-independence assumption**

  \[ P(o_t | O_{1}^{t-1}, q_1^t) = P(o_t | q_t) \]
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
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

\[
\begin{align*}
B_1 &= \begin{bmatrix} P(1 \mid \text{HOT}) & P(2 \mid \text{HOT}) & P(3 \mid \text{HOT}) \end{bmatrix} = \begin{bmatrix} .2 & .4 & .4 \end{bmatrix} \\
B_2 &= \begin{bmatrix} P(1 \mid \text{COLD}) & P(2 \mid \text{COLD}) & P(3 \mid \text{COLD}) \end{bmatrix} = \begin{bmatrix} .5 & .4 & .1 \end{bmatrix}
\end{align*}
\]
HMMs for speech
Phones are not homogeneous! Phone-level HMMs not enough.
Each phone has 3 subphones
Resulting HMM word model for “six”
HMM for the digit recognition task
Different types of HMM structure

Bakis = left-to-right

Ergodic = fully-connected
HMM-GMM System

Transcription: Samson
Sub-phones: 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:

GMM models: P(x|s)
x: input features
s: HMM state
Search space with bigrams
The Three Basic Problems for HMMs

Jack Ferguson at IDA in the 1960s

Problem 1 (Evaluation): Given the observation sequence $O=\langle o_1 o_2 \ldots o_T \rangle$, and an HMM model $\Phi = (A, B)$, how do we efficiently compute $P(O \mid \Phi)$, the probability of the observation sequence, given the model?

Problem 2 (Decoding): Given the observation sequence $O=\langle o_1 o_2 \ldots o_T \rangle$, and an HMM model $\Phi = (A, B)$, how do we choose a corresponding state sequence $Q=\langle q_1 q_2 \ldots q_T \rangle$ 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 $P(O \mid \Phi)$?
Evaluation

- How to evaluate the word string output by a speech recognizer?
Word Error Rate

• Word Error Rate =
  \[
  100 \left( \text{Insertions} + \text{Substitutions} + \text{Deletions} \right)
  
  \text{Total Word in Correct Transcript}
  
• Alignment example:
REF:  portable *** PHONE UPSTAIRS last night so
HYP:  portable FORM OF STORES last night so
Eval    I   S   S

• WER = 100 (1+2+0)/6 = 50%
NIST sctk scoring software: Computing WER with sclite

- Sclite aligns a hypothesized text (HYP) (from the recognizer) with a correct or reference text (REF) (human transcribed)

id: (2347-b-013)
Scores: (#C #S #D #I) 9 3 1 2
REF: was an engineer SO I i was always with **** **** MEN UM and they
HYP: was an engineer ** AND i was always with THEM THEY ALL THAT and they
Eval: D S I I S S
Sclite output for error analysis

<table>
<thead>
<tr>
<th>CONFUSION PAIRS</th>
<th>Total</th>
<th>With &gt;= 1 occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(972)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: 6 -&gt; (%hesitation) ==&gt; on</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: 6 -&gt; the ==&gt; that</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3: 5 -&gt; but ==&gt; that</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4: 4 -&gt; a ==&gt; the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5: 4 -&gt; four ==&gt; for</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6: 4 -&gt; in ==&gt; and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7: 4 -&gt; there ==&gt; that</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8: 3 -&gt; (%hesitation) ==&gt; and</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9: 3 -&gt; (%hesitation) ==&gt; the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10: 3 -&gt; (a-) ==&gt; i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11: 3 -&gt; and ==&gt; i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12: 3 -&gt; and ==&gt; in</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13: 3 -&gt; are ==&gt; there</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14: 3 -&gt; as ==&gt; is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15: 3 -&gt; have ==&gt; that</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16: 3 -&gt; is ==&gt; this</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Better metrics than WER?

- WER is useful, but <10% or so and errors might be function words (and, of, uh-huh,...)
- But should we be more concerned with meaning ("semantic error rate")?
  - Good idea, but hard to agree on
  - Has been applied in dialogue systems, where desired semantic/task output is more clear
Computing total likelihood of 3 1 3

- We would need to sum over
  - Hot hot cold
  - Hot hot hot
  - Hot cold hot
  - ....

\[ P(O) = \sum_{Q} P(O, Q) = \sum_{Q} P(O|Q)P(Q) \]

- How many possible hidden state sequences are there for this sequence?
  \[ P(3 \ 1 \ 3) = P(3 \ 1 \ 3, \text{cold cold cold}) + P(3 \ 1 \ 3, \text{cold cold hot}) + P(3 \ 1 \ 3, \text{hot hot cold}) + ... \]

- How about in general for an HMM with N hidden states and a sequence of T observations?
  - \( N^T \)

- So we can’t just do separate computation for each hidden state sequence.
Instead: the Forward algorithm

- A **dynamic programming** algorithm
  - Just like Minimum Edit Distance or CKY Parsing
  - Uses a table to store intermediate values

- Idea:
  - Compute the likelihood of the observation sequence
  - By summing over all possible hidden state sequences
  - But doing this efficiently
    - By folding all the sequences into a single **trellis**
The forward algorithm

- The goal of the forward algorithm is to compute

\[ P(o_1, o_2, \ldots, o_T, q_T = q_F \mid \lambda) \]

- We’ll do this by recursion
The forward algorithm

- Each cell of the forward algorithm trellis $\alpha_t(j)$
  - Represents the probability of being in state $j$
  - After seeing the first $t$ observations
  - Given the automaton
- Each cell thus expresses the following probability

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j | \lambda)$$
1. Initialization:
\[ \alpha_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N \]

2. Recursion (since states 0 and F are non-emitting):
\[ \alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t) ; \quad 1 \leq j \leq N, 1 < t \leq T \]

3. Termination:
\[ P(O|\lambda) = \alpha_T(q_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF} \]
We update each cell

\[ \alpha_t(i) \] the **previous forward 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 \( j \)

\[ \alpha_t(j) = \sum_i \alpha_{t-1}(i) a_{ij} b_j(o_t) \]
The Forward Algorithm

\textbf{function} \textsc{forward} (observations of len } T, \text{ state-graph of len } N \text{) \textbf{returns} forward-prob

create a probability matrix \textit{forward}[N+2,T]

\textbf{for} each state } s \textbf{ from 1 to } N \textbf{ do} \quad ; \text{ initialization step}

\quad \textit{forward}[s,1] \leftarrow a_{0,s} \times b_s(o_1)

\textbf{for} each time step } t \textbf{ from 2 to } T \textbf{ do} \quad ; \text{ recursion step}

\quad \textbf{for} each state } s \textbf{ from 1 to } N \textbf{ do}

\quad \quad \textit{forward}[s,t] \leftarrow \sum_{s' = 1}^{N} \textit{forward}[s',t - 1] \times a_{s',s} \times b_s(o_t)

\quad \textit{forward}[q_F,T] \leftarrow \sum_{s = 1}^{N} \textit{forward}[s,T] \times a_{s,q_F} \quad ; \text{ termination step}

\textbf{return} \textit{forward}[q_F,T]
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 $O=(o_1o_2...o_T)$, and an HMM model $\Phi = (A,B)$, **how do we choose a corresponding state sequence $Q=(q_1q_2...q_T)$ that is optimal in some sense (i.e., best explains the observations)**
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 again 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:**

\[
v_1(j) = a_{0j} b_j(o_1) \quad 1 \leq j \leq N
\]
\[
b_{t1}(j) = 0
\]

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

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

3. **Termination:**

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

The start of backtrace: 
\[
q_{T*} = b_{tT}(q_F) = \arg\max_{i=1}^{N} v_T(i) \cdot a_{i,F}
\]
The Viterbi trellis

\( v_1(2) = 0.32 \)

\( v_2(2) = \text{max}(0.32 \times 0.014, 0.02 \times 0.08) = 0.0448 \)

\( v_1(1) = 0.02 \)

\( v_2(1) = \text{max}(0.32 \times 0.15, 0.02 \times 0.30) = 0.048 \)
Viterbi intuition

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

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

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

- the previous Viterbi path probability from the previous time step
- the transition probability from previous state \(q_i\) to current state \(q_j\)
- 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] ← 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] ← \argmax_{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] ← \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]
Viterbi backtrace

\[
v_1(2) = 0.32
\]

\[
v_2(2) = \max(0.32 \times 0.14, 0.02 \times 0.08) = 0.0448
\]

\[
v_1(1) = 0.02
\]

\[
v_2(1) = \max(0.32 \times 0.15, 0.02 \times 0.30) = 0.048
\]
The weather figure: specific example
Markov chain for weather

- What is the probability of 4 consecutive warm days?
- Sequence is warm-warm-warm-warm-warm
- I.e., state sequence is 3-3-3-3
- $P(3,3,3,3) =$
  - $\pi_3 a_{33} a_{33} a_{33} a_{33} = 0.2 \times (0.6)^3 = 0.0432$
Problem 1: computing the observation likelihood

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence $O$, determine the likelihood $P(O|\lambda)$.

How likely is the sequence 3 1 3?
How to compute likelihood

- For a Markov chain, we just follow the states 3 1 3 and multiply the probabilities
- But for an HMM, we don’t know what the states are!
- So let’s start with a simpler situation.
- Computing the observation likelihood for a given hidden state sequence
  - Suppose we knew the weather and wanted to predict how much ice cream Jason would eat.
  - i.e., \( P(3 1 3 \mid H H C) \)
Computing likelihood of 3 1 3 given hidden state sequence

\[ P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i) \]

\[ P(3 1 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \]
Computing joint probability of observation and state sequence

\[ P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1}) \]

\[ P(3, 1, 3, \text{hot hot cold}) = P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold}) \]
HMMs for Speech

- We haven’t yet shown how to learn the A and B matrices for HMMs;
  - we’ll do that on Thursday
  - The Baum-Welch (Forward-Backward alg)
- But let’s return to think about speech
Reminder: a word looks like this:

\[ Q = q_1 q_2 \ldots q_N \]
\[ A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn} \]
\[ B = b_i(o_t) \]

A set of states corresponding to subphones, a transition probability matrix \( A \), each \( a_{ij} \) representing the probability for each subphone of taking a self-loop or going to the next subphone. Together, \( Q \) and \( A \) implement a pronunciation lexicon, an HMM state graph structure for each word that the system is capable of recognizing.

A set of observation likelihoods:, also called emission probabilities, each expressing the probability of a cepstral feature vector (observation \( o_t \)) being generated from subphone state \( i \).
HMM for digit recognition task
The Evaluation (forward) problem for speech

- The observation sequence $O$ is a series of MFCC vectors
- The hidden states $W$ are the phones and words
- For a given phone/word string $W$, our job is to evaluate $P(O|W)$
- Intuition: how likely is the input to have been generated by just that word string $W$?
Evaluation for speech: Summing over all different paths!

- f ay ay ay ay v v v v
- f f ay ay ay ay v v v
- f f f f ay ay ay ay v
- f f ay ay ay ay ay ay v
- f f ay ay ay ay ay ay ay ay v
- f f ay v v v v v v v
The forward lattice for “five”
The forward trellis for “five”

<table>
<thead>
<tr>
<th>V</th>
<th>0</th>
<th>0</th>
<th>0.008</th>
<th>0.0093</th>
<th>0.0114</th>
<th>0.00703</th>
<th>0.00345</th>
<th>0.00306</th>
<th>0.00206</th>
<th>0.00117</th>
</tr>
</thead>
<tbody>
<tr>
<td>AY</td>
<td>0</td>
<td>0.04</td>
<td>0.054</td>
<td>0.0664</td>
<td>0.0355</td>
<td>0.016</td>
<td>0.00676</td>
<td>0.00208</td>
<td>0.000532</td>
<td>0.000109</td>
</tr>
<tr>
<td>F</td>
<td>0.8</td>
<td>0.32</td>
<td>0.112</td>
<td>0.0224</td>
<td>0.00448</td>
<td>0.000896</td>
<td>0.000179</td>
<td>4.48e-05</td>
<td>1.12e-05</td>
<td>2.8e-06</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
<td>$f$</td>
</tr>
<tr>
<td></td>
<td>0.8</td>
<td>0.8</td>
<td>0.7</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
<td>$ay$</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>B</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
<td>$v$</td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.6</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
<td>$p$</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
<td>$iy$</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Viterbi trellis for “five”
Viterbi trellis for “five”

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>0</td>
<td>0</td>
<td>0.008</td>
<td>0.0072</td>
<td>0.00672</td>
<td>0.00403</td>
<td>0.00188</td>
<td>0.00161</td>
<td>0.000667</td>
<td>0.000493</td>
<td></td>
</tr>
<tr>
<td>AY</td>
<td>0</td>
<td>0.04</td>
<td>0.048</td>
<td>0.0448</td>
<td>0.0269</td>
<td>0.0125</td>
<td>0.00538</td>
<td>0.00167</td>
<td>0.000428</td>
<td>8.78e-05</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>0.8</td>
<td>0.32</td>
<td>0.112</td>
<td>0.0224</td>
<td>0.00448</td>
<td>0.000896</td>
<td>0.000179</td>
<td>4.48e-05</td>
<td>1.12e-05</td>
<td>2.8e-06</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>f</td>
<td>0.8</td>
<td>f</td>
<td>0.8</td>
<td>f</td>
<td>0.7</td>
<td>f</td>
<td>0.4</td>
<td>f</td>
<td>0.4</td>
<td>f</td>
</tr>
<tr>
<td>AY</td>
<td>ay</td>
<td>0.1</td>
<td>ay</td>
<td>0.1</td>
<td>ay</td>
<td>0.3</td>
<td>ay</td>
<td>0.8</td>
<td>ay</td>
<td>0.8</td>
<td>ay</td>
</tr>
<tr>
<td>F</td>
<td>v</td>
<td>0.6</td>
<td>v</td>
<td>0.6</td>
<td>v</td>
<td>0.4</td>
<td>v</td>
<td>0.3</td>
<td>v</td>
<td>0.3</td>
<td>v</td>
</tr>
<tr>
<td>B</td>
<td>p</td>
<td>0.4</td>
<td>p</td>
<td>0.4</td>
<td>p</td>
<td>0.2</td>
<td>p</td>
<td>0.1</td>
<td>p</td>
<td>0.1</td>
<td>p</td>
</tr>
<tr>
<td></td>
<td>iy</td>
<td>0.1</td>
<td>iy</td>
<td>0.1</td>
<td>iy</td>
<td>0.3</td>
<td>iy</td>
<td>0.6</td>
<td>iy</td>
<td>0.6</td>
<td>iy</td>
</tr>
</tbody>
</table>
Search space with bigrams

\begin{align*}
p(\text{one} | \text{one}) & \quad \text{w} \quad \text{w} \quad \text{w} \quad \text{ah} \quad \text{ah} \quad \text{ah} \quad \text{n} \quad \text{n} \quad \text{n} \quad p(\text{two} | \text{one}) \\
p(\text{one} | \text{two}) & \quad \text{w} \quad \text{w} \quad \text{ah} \quad \text{ah} \quad \text{ah} \quad \text{n} \quad \text{n} \quad \text{n} \quad p(\text{one} | \text{two}) \\
p(\text{one} | \text{zero}) & \quad \text{t} \quad \text{t} \quad \text{t} \quad \text{uw} \quad \text{uw} \quad \text{uw} \\
p(\text{two} | \text{zero}) & \quad \text{t} \quad \text{t} \quad \text{t} \quad \text{uw} \quad \text{uw} \quad \text{uw} \\
p(\text{two} | \text{two}) & \quad \text{t} \quad \text{t} \quad \text{t} \quad \text{uw} \quad \text{uw} \quad \text{uw} \\
p(\text{zero} | \text{one}) & \\
p(\text{zero} | \text{two}) & \\
p(\text{zero} | \text{zero}) & \\
\end{align*}
Viterbi trellis
Viterbi backtrace
Summary: ASR Architecture

- Five easy pieces: ASR Noisy Channel architecture
  - 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