Outline for Today

- Speech Recognition Architectural Overview
- Hidden Markov Models in general
  - Forward
  - Viterbi Decoding
  - Baum-Welch
- Applying HMMs to speech
- How this fits into the ASR component of course
  - July 6: Language Modeling
  - July 19 (today): HMMs, Forward, Viterbi, Start of Baum-Welch (EM) training
  - July 23: Feature Extraction, MFCCs, and Gaussian Acoustic modeling
  - July 26: Evaluation, Decoding, Advanced Topics

LVCSR

- Large Vocabulary Continuous Speech Recognition
- ~20,000-64,000 words
- Speaker independent (vs. speaker-dependent)
- Continuous speech (vs isolated-word)

Current error rates

<table>
<thead>
<tr>
<th>Task</th>
<th>Vocabulary</th>
<th>Error Rate%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digits</td>
<td>11</td>
<td>0.5</td>
</tr>
<tr>
<td>WSJ read speech</td>
<td>5K</td>
<td>3</td>
</tr>
<tr>
<td>WSJ read speech</td>
<td>20K</td>
<td>3</td>
</tr>
<tr>
<td>Broadcast news</td>
<td>64,000+</td>
<td>10</td>
</tr>
<tr>
<td>Conversational Telephone</td>
<td>64,000+</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task</th>
<th>Vocab</th>
<th>ASR</th>
<th>Hum SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous digits</td>
<td>11</td>
<td>.5</td>
<td>.009</td>
</tr>
<tr>
<td>WSJ 1995 clean</td>
<td>5K</td>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td>WSJ 1995 w/noise</td>
<td>5K</td>
<td>9</td>
<td>1.1</td>
</tr>
<tr>
<td>SWBD 2004</td>
<td>65K</td>
<td>20</td>
<td>4</td>
</tr>
</tbody>
</table>

Conclusions:
- Machines about 5 times worse than humans
- Gap increases with noisy speech
- These numbers are rough, take with grain of salt

HSR versus ASR

<table>
<thead>
<tr>
<th>Task</th>
<th>Vocab</th>
<th>ASR</th>
<th>Hum SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous digits</td>
<td>11</td>
<td>.5</td>
<td>.009</td>
</tr>
<tr>
<td>WSJ 1995 clean</td>
<td>5K</td>
<td>3</td>
<td>0.9</td>
</tr>
<tr>
<td>WSJ 1995 w/noise</td>
<td>5K</td>
<td>9</td>
<td>1.1</td>
</tr>
<tr>
<td>SWBD 2004</td>
<td>65K</td>
<td>20</td>
<td>4</td>
</tr>
</tbody>
</table>

LVCSR Design Intuition

- Build a statistical model of the speech-to-words process
- Collect lots and lots of speech, and transcribe all the words.
- Train the model on the labeled speech
- Paradigm: Supervised Machine Learning + Search
The Noisy Channel Model (II)

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

Noisy Channel Model (III)

- Probabilistic implication: Pick the highest probability \( S \):
  \[ \hat{W} = \arg \max_{W \in L} P(W | O) \]
- We can use Bayes rule to rewrite this:
  \[ \hat{W} = \arg \max_{W \in L} \frac{P(O | W)P(W)}{P(O)} \]
- Since denominator is the same for each candidate sentence \( W \), we can ignore it for the argmax:
  \[ \hat{W} = \arg \max_{W \in L} P(O | W)P(W) \]

Noisy channel model

- Ignoring the denominator leaves us with two factors:
  \[ \hat{W} = \arg \max_{W \in L} P(O | W)P(W) \]

The noisy channel model

- Probabilistic implication: Pick the one that is most probable given the waveform.
Speech Architecture meets Noisy Channel

Architecture: Five easy pieces (only 2 for today)
- Feature extraction
- Acoustic Modeling
- HMMs, Lexicons, and Pronunciation
- Decoding
- Language Modeling

HMMs for speech

Phones are not homogeneous!

Each phone has 3 subphones

Resulting HMM word model for “six”
HMMs more formally

- Markov chains
- A kind of weighted finite-state automaton

\[ Q = q_1 q_2 \ldots q_N \] a set of states

\[ A = a_{01} a_{02} \ldots a_{0N} \ldots a_{m1} \ldots a_{mN} \] a transition probability matrix \( A \), each \( a_{ij} \) representing the probability of moving from state \( i \) to state \( j \), i.e., \( \sum_{j=1}^{N} a_{ij} = 1 \) \( \forall i \)

A special start and end state which are not associated with observations.

Markov Assumption: \( P(q_t|q_{t-1}) = P(q_t|q_{t-1}) \)

Another Markov chain

\[ \pi = \pi_1, \pi_2, \ldots, \pi_N \] an initial probability distribution over states. \( \pi_i \) is the probability that the Markov chain will start in state \( i \). Some states \( j \) may have \( \pi_j = 0 \), meaning that they cannot be initial states. Also, \( \sum_{i=1}^{N} \pi_i = 1 \)

\[ QA = \{ q_0, q_1, \ldots \} \] a set \( QA \subset Q \) of legal accepting states

An example with numbers:

- What is probability of:
  - Hot, hot, hot, hot
  - Cold, hot, cold, hot

Hidden Markov Models

\[ Q = q_0 q_1 \ldots q_N \] a set of \( N \) states

\[ A = a_{01} a_{02} \ldots a_{0N} \ldots a_{m1} \ldots a_{mN} \] a transition probability matrix \( A \), each \( a_{ij} \) representing the probability of moving from state \( i \) to state \( j \), i.e., \( \sum_{j=1}^{N} a_{ij} = 1 \) \( \forall i \)

\[ O = o_1 o_2 \ldots o_T \] a sequence of \( T \) observations, each one drawn from a vocabulary \( V = \{v_1, v_2, \ldots, v_V\} \)

\[ B = b_i(o_j) \] A sequence of observation likelihoods; also called emission probabilities, each expressing the probability of an observation \( o_j \) being generated from a state \( i \)

\[ \psi_0 \psi_T \] a special start state and end (final) state which are not associated with observations; together with transition probabilities \( o_{01}, o_{02}, \ldots, o_{mT} \), out of the start state and \( o_{T0}, o_{T1}, \ldots, o_{Tm} \) into the end state.
Hidden Markov Models

Markov Assumption:  \( P(q_t | q_{t-1}, q_{t-2}, \ldots) = P(q_t | q_{t-1}) \)
Output Independence Assumption:  \( P(o_t | q_1, q_2, \ldots, q_t) = P(o_t | q_t) \)

The Jason Eisner task

You are a climatologist in 2799 studying the history of global warming. YOU can’t find records of the weather in Baltimore for summer 2006. But you do find Jason Eisner’s diary. Which records how many ice creams he ate each day. Can we use this to figure out the weather?

Given a sequence of observations \( O \),
- each observation an integer = number of ice creams eaten
- Figure out correct hidden sequence \( Q \) of weather states \( (H \text{ or } C) \) which caused Jason to eat the ice cream

HMMs more formally

- Three fundamental problems
  - Jack Ferguson at IDA in the 1960s
  1) Given a specific HMM, determine likelihood of observation sequence.
  2) Given an observation sequence and an HMM, discover the best (most probable) hidden state sequence
  3) Given only an observation sequence, learn the HMM parameters \( (A, B) \) matrix

The Three Basic Problems for HMMs

- Problem 1 (Evaluation): Given the observation sequence \( O = (o_1, o_2, \ldots, o_T) \), and an HMM model \( \Phi = (A,B) \), 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 \( P(O | \Phi) \)?
Problem 1: computing the observation likelihood

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

= Given the following HMM:

= 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. $P(3\ 1\ 3\ |\ HH\ C)$

Computing likelihood for 1 given hidden state sequence

$$P(O|Q) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=0}^{n} P(q_i|q_{i-1})$$

$P(3\ 1\ 3|\ hot\ hot\ cold)$ = $P(\text{hot}\ |\ \text{start}) \times P(\text{hot}\ |\ \text{hot}) \times P(\text{cold}\ |\ \text{hot})$

$\times P(3\ |\ \text{hot}) \times P(1\ |\ \text{hot}) \times P(3\ |\ \text{cold})$

Computing total likelihood of 3 1 3

We would need to sum over:

- Hot hot cold
- Hot hot hot
- Hot cold hot

How many possible hidden state sequences are there for this sequence?

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 kind of dynamic programming algorithm
- 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 Trellis
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 $\lambda$, each cell thus expresses the following probability:

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

We update each cell

$\alpha_{t-1}(i)$: the previous forward path probability from the previous time step
$\alpha_t(j)$: the transition probability from previous state $q_t$ to current state $q_j$
$h_j(o_t)$: the state observation likelihood of the observation symbol $o_t$ given the current state $j$

The Forward Recursion

1. Initialization:
$$\alpha_1(j) = a_j \beta_j(o_1), \quad 1 \leq j \leq N$$

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

3. Termination:
$$P(O | \lambda) = \alpha_T(\Phi) = \sum_{i=1}^{N} \alpha_T(i) \beta_i(o_T)$$

The Forward Algorithm

function FORWARD(observations of len T, state-graph of len N) returns forward-prob
create a probability matrix forward[N+2,T]
for each state $i$ from 1 to $N$ do
  forward[i,1] = $a_i \beta_i(o_1)$
end for
for each time step $t$ from 2 to $T$ do
  for each state $i$ from 1 to $N$ do
    forward[i,t] = $\sum_{j=1}^{N} forward[i-1,t-1] \cdot a_{ij} \cdot h_j(o_t)$
  end for
end for
forward[0,T] = $\sum_{i=1}^{N} forward[i,T] \cdot a_i \beta_i(o_T)$; termination step
return forward[0,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 $O = (o_1, o_2, \ldots, o_T)$, and an HMM model $\Phi = (A, B)$, how do we choose a corresponding state sequence $Q = (q_1, q_2, q_T)$ that is optimal in some sense (i.e., best explains the observations)?

Decoding

- One possibility:
  - For each hidden state sequence $-HHH, HHC, HCH$
  - Run the forward algorithm to compute $P(\Phi | O)$
- Why not?
  - $N^T$
- Instead:
  - The Viterbi algorithm
  - Is again a dynamic programming algorithm
  - Uses a similar trellis to the Forward algorithm
The Viterbi trellis

Viterbi Algorithm

function VITERBI(Observations of size T, state graph of size N) returns best path
create a path probability matrix viterbi(N+2,T)
for each state j from 1 to N do
    viterbi(j,1) = a0 * bj(0)
    backpointer[j,1] = 0
for each time step t from 2 to T do
    for each state j from 1 to N do
        viterbi[j,t] = max_{j'} viterbi[j,t-1] * a_{ij'} * bj(t)
        backpointer[j,t] = argmax_{j'} viterbi[j,t-1] * a_{ij'}
    viterbi[all, T] = max_{j} viterbi[j, T] * a_{oT}
    backpointer[all, T] = argmax_{j} viterbi[j, T] * a_{oT}
return the backtrace path by following back pointers to states back in time from backpointer[all, T]

Viterbi intuition

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

\[ \nu_t(j) = P(q_0, q_1, ..., q_{t-1}, a_1, a_2, ..., a_t | q_t = i) \]

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

- \(\nu_t(j)\) the previous Viterbi path probability from the previous time step
- \(a_{ij}\) the transition probability from previous state \(q_{t-1}\) to current state \(q_t\)
- \(b_j(o_t)\) the state observation likelihood of the observation symbol \(o_t\) given the current state \(j\)

Viterbi backtrace

Why “Dynamic Programming”

I spent the Fall quarter of 1950 at RAND. My first task was to find a name for multi-stage decision processes. An interesting question is, Where did the name, dynamic programming, come from? The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I’m not using the term lightly; I’m using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporation. What title, what name, could I choose? In the first place I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, “programming.” I wanted to get across the idea that this was dynamic, this was multi-stage, I was trying to think, I was thinking of two birds with one stone. Let us take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is in its impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It’s impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities.” Richard Bellman, “Eye of the Hurricane: an autobiography” 1984.
HMMs for Speech

- We haven’t yet shown how to learn the $A$ and $B$ matrices for HMMs; we’ll do that later today or possibly on Monday.
- But let’s return to think about speech.

Reminder: a word looks like this:

$Q = q_1, q_2, \ldots, q_N$

a set of states corresponding to subphones

$A = a_{ij} = \Pr(q_i \rightarrow q_j)$

a transition probability matrix; $a_{ij}$ represents 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.

$B = b_i(o)$

A set of observation likelihoods; also called emission probabilities, each expressing the probability of a cepstral feature vector (observation $o$) being generated from subphone state $q_i$.

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!

- $f\ ay\ ay\ ay\ ay\ ay\ vv\ vv\ vv$
- $f\ f\ ay\ ay\ ay\ vv\ vv$
- $f\ f\ f\ ay\ ay\ ay\ vv$
- $f\ ay\ ay\ ay\ ay\ ay\ ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$ $ay$
- $f\ ay\ vv\ vv\ vv\ vv$

The forward lattice for “five”
The forward trellis for “five”

Viterbi trellis for “five”

Search space with bigrams

Viterbi trellis with 2 words and uniform LM

Viterbi backtrace
Evaluation

How to evaluate the word string output by a speech recognizer?

Word Error Rate

Word Error Rate =

\[
\frac{100 \times (\text{Insertions} + \text{Substitutions} + \text{Deletions})}{\text{Total Word in Correct Transcript}}
\]

Alignment example:

REF:   portable *** PHONE UPSTAIRS last night so
HYP:   portable FORM OF STORES last night so
Eval I S S
WER = 100 \times (1+2+0)/6 = 50%

NIST sctk-1.3 scoring software:
Computing WER with sclite

http://www.nist.gov/speech/tools/

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 S I I S

Sclite output for error analysis

1:    6  ->  (%hesitation) ==> on
2:    6  ->  the ==> that
3:    5  ->  but ==> that
4:    4  ->  a ==> the
5:    4  ->  four ==> for
6:    4  ->  in ==> and
7:    4  ->  there ==> that
8:    3  ->  (%hesitation) ==> and
9:    3  ->  (%hesitation) ==> the
10:    3  ->  (a-) ==> i
11:    3  ->  and ==> i
12:    3  ->  and ==> in
13:    3  ->  are ==> there
14:    3  ->  as ==> is
15:    3  ->  have ==> that
16:    3  ->  is ==> this
17:    3  ->  it ==> that
18:    3  ->  mouse ==> most
19:    3  ->  was ==> is
20:    3  ->  was ==> this
21:    2  ->  (%hesitation) ==> it
22:    2  ->  (%hesitation) ==> that
23:    2  ->  (%hesitation) ==> to
24:    2  ->  (%hesitation) ==> yeah
25:    2  ->  a ==> all
26:    2  ->  a ==> know
27:    2  ->  a ==> you
28:    2  ->  along ==> well
29:    2  ->  along ==> well
30:    2  ->  and ==> in
31:    2  ->  and ==> we
32:    2  ->  and ==> you
33:    2  ->  are ==> i
34:    2  ->  are ==> were

Sclite output for error analysis
**Better metrics than WER?**

- WER has been useful
- 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 output is more clear

**Summary: ASR Architecture**

- Five easy pieces: ASR Noisy Channel architecture
  1) Feature Extraction:
     - MFCC features
  2) Acoustic Model:
     - Gaussians for computing $p(o|q)$
  3) Lexicon/Pronunciation Model
     - HMM: what phones can follow each other
  4) Language Model
     - N-grams for computing $p(w_i|w_{i-1})$
  5) Decoder
     - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!

**ASR Lexicon: Markov Models for pronunciation**

**Summary**

- Speech Recognition Architectural Overview
- Hidden Markov Models in general
  - Forward
  - Viterbi Decoding
- Hidden Markov models for Speech
- Evaluation