Where Are We?

1. Review
2. Technical Details
3. Continuous Word Recognition
4. Discussion
What is an HMM?

- Has states $S$ and arcs/ transitions $a$.
- Has *start* state $S_0$ (or start distribution).
- Has *transition* probabilities $p_a$.
- Has *output* probabilities $P(\tilde{x}|a)$ on arcs (or states).
  - Discrete: multinomial or single output.
  - Continuous: GMM or other.
Assigns probabilities $P(\mathbf{x})$ to observation sequences:

$$\mathbf{x} = \bar{x}_1, \ldots, \bar{x}_T$$

Each $\mathbf{x}$ can be output by many *paths* through HMM.

- Path consists of sequence of arcs $A = a_1, \ldots, a_T$.

Compute $P(\mathbf{x})$ by summing over path likelihoods.

$$P(\mathbf{x}) = \sum_{\text{paths } A} P(\mathbf{x}, A)$$

Compute path likelihood by . . .

- Multiplying transition and output probs along path.

$$P(\mathbf{x}, A) = \prod_{t=1}^{T} p_{a_t} \times P(\bar{x}_t | a_t)$$
HMM’s and ASR

- One HMM per word.
- A standard topology.

Use diagonal covariance GMM’s for output distributions.

\[
P(\bar{x} | a) = \sum_{\text{comp} j} p_{a,j} \prod_{\text{dim} d} \mathcal{N}(x_d; \mu_{a,j,d}, \sigma_{a,j,d})
\]
The Full Model

\[ P(x) = \sum_{\text{paths } A} P(x, A) \]

\[ = \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \times P(\vec{x}_t | a_t) \]

\[ = \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} \prod_{\text{dim } d} \mathcal{N}(x_{t,d}; \mu_{a_t,j,d}, \sigma_{a_t,j,d}^2) \]

- \( p_a \) — transition probability for arc \( a \).
- \( p_{a,j} \) — mixture weight, \( j \)th component of GMM on arc \( a \).
- \( \mu_{a,j,d} \) — mean, \( d \)th dim, \( j \)th component, GMM on arc \( a \).
- \( \sigma_{a,j,d}^2 \) — variance, \( d \)th dim, \( j \)th component, GMM on arc \( a \).
The Viterbi and Forward Algorithms

- The Forward algorithm.

\[ P(x) = \sum_{\text{paths } A} P(x, A) \]

- The Viterbi algorithm.

\[ \text{bestpath}(x) = \arg \max_{\text{paths } A} P(x, A) \]

- Can handle exponential number of paths \( A \) . . .
  - In time linear in number of states, number of frames.*

*Assuming fixed number of arcs per state.
Decoding

- Given trained HMM for each word $\omega$.
- Use Forward algorithm to compute $P_\omega(x_{\text{test}})$ for each $\omega$.
- Pick word that assigns highest likelihood.

$$\omega^* = \arg \max_{\omega \in \text{vocab}} P_\omega(x_{\text{test}})$$
Recap

- HMM/GMM framework can model arbitrary distributions . . .
  - Over sequences of continuous vectors.
- Can train and decode efficiently.
  - Forward, Viterbi, Forward-Backward algorithms.
Probabilities and Log Probabilities

\[ P(x) = \sum_{\text{paths } A} \prod_{t=1}^{T} p_{a_t} \sum_{\text{comp } j} p_{a_t,j} \prod_{\text{dim } d} \mathcal{N}(x_t,d; \mu_{a_t,j,d}, \sigma^2_{a_t,j,d}) \]

- 1 sec of data \( \Rightarrow T = 100 \Rightarrow \) Multiply 4,000 likelihoods.
  - Easy to generate values below \( 10^{-307} \).
  - Cannot store in C/C++ 64-bit double.
- Solution: store log probs instead of probs.
  - e.g., in Forward algorithm, instead of storing \( \alpha(S, t) \), \( \ldots \)
  - Store values log \( \alpha(S, t) \)
\[ \hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \hat{\alpha}(S', t - 1) \]

\[
\log \hat{\alpha}(S, t) = \max_{S' \xrightarrow{x_t} S} \left[ \log P(S' \xrightarrow{x_t} S) + \log \hat{\alpha}(S', t - 1) \right]
\]
Forward Algorithm and Sum is Tricky

\[
\alpha(S, t) = \sum_{S' \xrightarrow{x_t} S} P(S' \xrightarrow{x_t} S) \times \alpha(S', t - 1)
\]

\[
\log \alpha(S, t) = \log \sum_{S' \xrightarrow{x_t} S} \exp \left[ \log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t - 1) \right]
\]

\[
= \log \sum_{S' \xrightarrow{x_t} S} \exp \left[ \log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t - 1) - C \right] \times e^C
\]

\[
= C + \log \sum_{S' \xrightarrow{x_t} S} \exp \left[ \log P(S' \xrightarrow{x_t} S) + \log \alpha(S', t - 1) - C \right]
\]

- How to pick \( C \)?
What we said:

- Use Forward algorithm to compute $P_\omega(x_{\text{test}})$ …
- Separately for each word HMM.
- Pick word that assigns highest likelihood.

$$\omega^* = \arg \max_{\omega \in \text{vocab}} P_\omega(x_{\text{test}})$$

Reality:

- Merge HMM for all words into “one big HMM”.
- Use Viterbi algorithm to find best path given $x_{\text{test}}$.
- In backtrace, collect word label on path.
The One Big HMM Paradigm: Before
The One Big HMM Paradigm: After
Recovering the Word Sequence
Part II

Language Modeling
Wreck a Nice Beach?

- Demo.

  THIS IS OUR ROOM FOR A FOUR HOUR PERIOD.
  THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD
  IT IS EASY TO RECOGNIZE SPEECH.
  IT IS EASY TO WRECK A NICE BEACH.

- How does it get it right . . .
  - Even though acoustics for pair is same?
  - (What if want other member of pair?)
Pick word sequence $\omega$ which assigns highest likelihood . . .

To test sample $\mathbf{x}$.

$$\omega^* = \arg \max_\omega P_\omega(\mathbf{x}) = \arg \max_\omega P(\mathbf{x}|\omega)$$

What about $\omega_1 = \text{SAMPLE}$, $\omega_2 = \text{SAM PULL}$?

- $P(\mathbf{x}|\omega_1) \approx P(\mathbf{x}|\omega_2)$
- Intuitively, much prefer $\omega_1$ to $\omega_2$.

Something’s missing.
What Do We Really Want?

What HMM/GMM’s give us: $P(x|\omega)$.

$$\omega^* \overset{?}{=} \arg\max_{\omega} P(x|\omega)$$

$$\omega^* \overset{!}{=} \arg\max_{\omega} P(\omega|x)$$
Bayes’ rule:

\[ P(x, \omega) = P(\omega)P(x|\omega) = P(x)P(\omega|x) \]

\[ P(\omega|x) = \frac{P(\omega)P(x|\omega)}{P(x)} \]

Substituting:

\[ \omega^* = \arg \max_{\omega} P(\omega|x) \]

\[ = \arg \max_{\omega} \frac{P(\omega)P(x|\omega)}{P(x)} \]

\[ = \arg \max_{\omega} P(\omega)P(x|\omega) \]
The Fundamental Equation of ASR

- Old way: maximum likelihood classification.
  \[ \omega^* = \arg \max_{\omega} P(x|\omega) \]

- New way: maximum \textit{a posteriori} classification.
  \[ \omega^* = \arg \max_{\omega} P(\omega|x) = \arg \max_{\omega} P(\omega)P(x|\omega) \]

- What’s new?
  - Prior distribution \( P(\omega) \) over word sequences.
  - How frequent each word sequence \( \omega \) is.
Does This Fix Our Problem?

\[ \omega^* = \arg \max_{\omega} P(\omega) P(x|\omega) \]

- What about homophones?
  
  THIS IS OUR ROOM FOR A FOUR HOUR PERIOD.
  THIS IS HOUR ROOM FOUR A FOR OUR . PERIOD

- What about confusible sequences in general?
  
  IT IS EASY TO RECOGNIZE SPEECH .
  IT IS EASY TO WRECK A NICE BEACH .
Terminology

\[ \omega^* = \arg \max_{\omega} P(\omega)P(x|\omega) \]

- \( P(x|\omega) = \text{acoustic model.} \)
  - Models frequency of acoustic feature vectors \( x \ldots \)
  - Given word sequence \( \omega \).
  - \( i.e. \), HMM/GMM’s.

- \( P(\omega) = \text{language model.} \)
  - Models frequency of each word sequence \( \omega \).
  - The rest of this lecture.
Language Modeling: Goals

- Specific to *domain***!!
- Describe which word sequences are *allowed*.
  - *e.g.*, restricted domains like digit strings.
- Describe which word sequences are *likely*.
  - *e.g.*, unrestricted domains like web search.
  - *e.g.*, BRITNEY SPEARS vs. BRIT KNEE SPEARS.
- Analogy: multiple-choice test.
  - LM restricts choices given to acoustic model.
  - The fewer choices, the better you do.
What is language model training data?
- Must match domain!

**Grammars** — hidden Markov models.
- Restricted domain.
- Little or no training data available.
- *e.g.*, airline reservation app.

**n-gram models** — Markov models of order $n - 1$.
- Unrestricted domain.
- Lots of training data available.
- *e.g.*, web search app.
Idea: Hidden Markov Models

- Like in acoustic modeling.
- What topology?
  - Is there logical topology like for word HMM?
- Learn topology from data?
  - e.g., fully interconnected topology; learn parameters?
- Issues:
  - Local minima issue, FB algorithm.
  - Quadratic in number of states; e.g., 1M states?
- Bottom line: hasn’t worked.
Idea: (Non-Hidden) Markov Models

- Review: Markov property order $n - 1$ holds if

\[
P(w_1, \ldots, w_L) = \prod_{i=1}^{L} P(w_i | w_1, \ldots, w_{i-1})
= \prod_{i=1}^{L} P(w_i | w_{i-n+1}, \ldots, w_{i-1})
\]

- i.e., if data satisfies this property . . .
  - No loss from just remembering past $n - 1$ items!
Markov Model, Order 1: Bigram Model

$$P(w_1, \ldots, w_L) = \prod_{i=1}^{L} P(w_i|w_{i-1}) = \prod_{i=1}^{L} p_{w_{i-1},w_i}$$

- Separate multinomial $P(w_i|w_{i-1})$ ...
- For each word *history* $w_{i-1}$.
- Model $P(w_i|w_{i-1})$ with parameter $p_{w_{i-1},w_i}$. 
Markov Model, Order 2: Trigram Model

\[ P(w_1, \ldots, w_L) = \prod_{i=1}^{L} P(w_i|w_{i-2}w_{i-1}) = \prod_{i=1}^{L} p_{w_{i-2},w_{i-1},w_i} \]

- Separate multinomial \( P(w_i|w_{i-2}w_{i-1}) \) . . .
- For each bigram \textit{history} \( w_{i-2}w_{i-1} \).
- Model \( P(w_i|w_{i-2}w_{i-1}) \) with parameter \( p_{w_{i-2},w_{i-1},w_i} \).
\[ P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1}) \]

- Pad with beginning-of-sentence token: \( w_{-1} = w_0 = \triangleright \).
\[ P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L} P(w_i|w_{i-2}w_{i-1}) \]

- Want probabilities to normalize: \( \sum_{\omega} P(\omega) = 1 \)
- Consider sum of probabilities of one-word sequences.

\[ \sum_{w_1} P(\omega = w_1) = \sum_{w_1} p_{\text{>,},w_1} = 1 \]

- Fix: introduce end-of-sentence token \( w_{L+1} = \langle \)

\[ P(\omega = w_1 \cdots w_L) = \prod_{i=1}^{L+1} P(w_i|w_{i-2}w_{i-1}) \]

In fact, \( \sum_{\omega:|\omega|=L} P(\omega) = 1 \) for all \( L \) \( \Rightarrow \) \( \sum_\omega P(\omega) = \infty \)
Maximum Likelihood Estimation

- Optimize likelihood of each multinomial independently.
  - One multinomial per history.
- ML estimate for multinomials: count and normalize!
  - e.g., trigram model:

\[
p_{w_{i-2},w_{i-1},w_i}^{\text{MLE}} = \frac{c(w_{i-2}w_{i-1}w_i)}{\sum_w c(w_{i-2}w_{i-1}w)} = \frac{c(w_{i-2}w_{i-1}w_i)}{c(w_{i-2}w_{i-1})}
\]
Training data:

JOHN READ MOBY DICK
MARY READ A DIFFERENT BOOK
SHE READ A BOOK BY CHER

What is $P(\text{JOHN READ A BOOK})$?
Bigram Model Example

\[ P(\text{JOHN READ A BOOK}) \]
\[ = P(\text{JOHN} | \triangleright) P(\text{READ} | \text{JOHN}) P(\text{A} | \text{READ}) P(\text{BOOK} | \text{A}) P(\triangleright | \text{BOOK}) \]
\[ = \frac{1}{3} \times 1 \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2} \approx 0.06 \]
Recap: \( N \)-Gram Models

- Simple formalism.
- Easy to train.
  - Just count and normalize.
  - Can train on vast amounts of data; just gets better.
Does Markov Property Hold For English?

- Not for small $n$.

$$P(w_i \mid \text{OF THE}) \neq P(w_i \mid \text{KING OF THE})$$

- Make $n$ larger?

  FABIO, WHO WAS NEXT IN LINE, ASKED IF THE TELLER SPOKE . . .

- For vocabulary size $V = 20,000$ . . .
  - How many parameters ($p_{w_{i-1}, w_i}$) in bigram model?
  - In trigram model?

- Vast majority of trigrams not present in training data!
Where Are We?

1. N-Gram Models
2. Technical Details
3. Smoothing
4. Discussion
Decoding without LM’s.
- Start with word HMM encoding allowable word sequences.
- Replace each word with its HMM.
Point: $n$-gram model is (hidden) Markov model.
- Can be expressed as word HMM.
- Replace each word with its HMM.
- Leave in language model probabilities.

How do LM’s impact acoustic model training?
One Puny Prob versus Many?
The Language Model Weight

- This doesn’t look like fair fight.
- Solution: language (or acoustic) model weight.

\[ \omega^* = \arg \max_{\omega} P(\omega)^\alpha P(x|\omega) \]

- \(\alpha\) usually somewhere between 10 and 20.
- Important to tune for each LM, AM.
- Theoretically inelegant.
  - Empirical performance trumps theory any day of week.
Where Are We?

1. N-Gram Models
2. Technical Details
3. Smoothing
4. Discussion
An Experiment

- Take 50M words of WSJ; shuffle sentences; split in two.
- “Training” set: 25M words.

NONCOMPETITIVE TENDERS MUST BE RECEIVED BY NOON EASTERN TIME THURSDAY AT THE TREASURY OR AT FEDERAL RESERVE BANKS OR BRANCHES. PERIOD NOT EVERYONE AGREED WITH THAT STRATEGY. PERIOD

... 

... 

- “Test” set: 25M words.

NATIONAL PICTURE AMPERSAND FRAME –DASH INITIAL TWO MILLION, COMMA TWO HUNDRED FIFTY THOUSAND SHARES, COMMA VIA WILLIAM BLAIR. PERIOD THERE WILL EVEN BE AN EIGHTEEN-HYPHEN HOLE GOLF COURSE. PERIOD

... 

...
An Experiment

- Count how often each word occurs in training; sort by count.

<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>,COMMA</td>
<td>1156259</td>
</tr>
<tr>
<td>THE</td>
<td>1062057</td>
</tr>
<tr>
<td>.PERIOD</td>
<td>877624</td>
</tr>
<tr>
<td>OF</td>
<td>520374</td>
</tr>
<tr>
<td>TO</td>
<td>510508</td>
</tr>
<tr>
<td>A</td>
<td>455832</td>
</tr>
<tr>
<td>AND</td>
<td>417364</td>
</tr>
<tr>
<td>IN</td>
<td>385940</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZZZZ</td>
<td>2</td>
</tr>
<tr>
<td>AAAAAAHHH</td>
<td>1</td>
</tr>
<tr>
<td>AAB</td>
<td>1</td>
</tr>
<tr>
<td>AACHENER</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ZYPLAST</td>
<td>1</td>
</tr>
<tr>
<td>ZYUGANOV</td>
<td>1</td>
</tr>
</tbody>
</table>
An Experiment

- For each word that occurs exactly once in training . . .
  - Count how often occurs in test set.
  - Average this count across all such words.
- What does ML estimate predict?
- What is actual value?
  1. Larger than 1.
  2. Exactly 1, more or less.
  3. Between 0.5 and 1.
  4. Between 0.1 and 0.5.
- What if do this for trigrams, not unigrams?
Why?

- What percentage of words/trigrams in test set . . .
  - Had no counts in training set?
  - 0.2%/31%.
In theory, ML estimate is as good as it gets …
- In limit of lots of data.
In practice, sucks when data is sparse.
- Can be off by large factor.
According to MLE trigram model . . .
- What is probability of sentence $\omega$ if $\omega$ contains . . .
- Trigram with no training counts?

How common are unseen trigrams?
- (Brown et al., 1992): 350M word training set
- In test set, what percentage of trigrams unseen?

How does this affect WER? Perplexity?
Smoothing

- How to adjust ML estimates to better match test data?
- How to avoid zero probabilities?
- Also called *regularization*. 
The Basic Idea, Bigram Model

- For each history word $w_{i-1} \ldots$
  - Estimate conditional distribution $P(w_i|w_{i-1})$.
- Maximum likelihood estimates.

$$p^\text{MLE}_{w_{i-1},w_i} = \frac{c(w_{i-1}w_i)}{c(w_{i-1})}$$

- Give prob to zero counts by discounting nonzero counts.

$$p^\text{sm}_{w_{i-1},w_i} = \frac{c(w_{i-1}w_i) - d(w_{i-1}w_i)}{c(w_{i-1})}$$

- How much to discount?
The Good-Turing Estimate

- How often word with $k$ counts in training data . . .
  - Occurs in test set of equal size?

\[
\text{(avg. count)} \approx \frac{\text{(\# words w/ } k + 1 \text{ counts)} \times (k + 1)}{\text{(\# words w/ } k \text{ counts)}}
\]

- How accurate is this?

<table>
<thead>
<tr>
<th>$k$</th>
<th>GT estimate</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td>2</td>
<td>1.26</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>2.24</td>
<td>2.24</td>
</tr>
<tr>
<td>4</td>
<td>3.24</td>
<td>3.23</td>
</tr>
<tr>
<td>5</td>
<td>4.22</td>
<td>4.21</td>
</tr>
</tbody>
</table>
The Basic Idea, Bigram Model (cont’d)

- Give prob to zero counts by discounting nonzero counts.
  Can use GT estimate to determine discounts $d(w_{i-1} \ w_i)$.

$$p^{sm}_{w_{i-1}, w_i} = \frac{c(w_{i-1} \ w_i) - d(w_{i-1} \ w_i)}{c(w_{i-1})}$$

- Total prob freed up for zero counts:

$$P^{sm}(\text{unseen}|w_{i-1}) = \frac{\sum_{w_i\text{seen}} d(w_{i-1} \ w_i)}{c(w_{i-1})}$$

- How to divvy up between words unseen after $w_{i-1}$?
Task: divide up some probability mass . . .
- Among words not occurring after some history $w_{i-1}$.

Idea: uniformly?

Better idea: according to unigram distribution.
- e.g., give more mass to THE than FUGUE.

$$P(w) = \frac{c(w)}{\sum_w c(w)}$$

Backoff: use lower-order distribution . . .
- To fill in probabilities for unseen words.
Putting It All Together: Katz Smoothing

Katz (1987)

\[ P_{\text{Katz}}(w_i|w_{i-1}) = \begin{cases} 
P_{\text{MLE}}(w_i|w_{i-1}) & \text{if } c(w_{i-1}w_i) \geq k \\
P_{\text{GT}}(w_i|w_{i-1}) & \text{if } 0 < c(w_{i-1}w_i) < k \\
\alpha_{w_{i-1}} P_{\text{Katz}}(w_i) & \text{otherwise}
\end{cases} \]

- If count high, no discounting (GT estimate unreliable).
- If count low, use GT estimate.
- If no count, use scaled backoff probability.

Choose \( \alpha_{w_{i-1}} \) so \( \sum_{w_i} P_{\text{Katz}}(w_i|w_{i-1}) = 1 \).

Most popular smoothing technique for about a decade.
Recap: Smoothing

- No smoothing (MLE estimate): performance will suck.
  - Zero probabilities will kill you.
- Key aspects of smoothing algorithms.
  - How to discount counts of seen words.
  - Estimating mass of unseen words.
  - Backoff to get information from lower-order models.
- Lots and lots of smoothing algorithms developed.
  - Will talk about newer algorithms in Lecture 11.
  - Gain: $\sim 1\%$ absolute in WER over Katz.
- No downside to good smoothing (except implementing).
Where Are We?

1. N-Gram Models
2. Technical Details
3. Smoothing
4. Discussion
N-Gram Models

- Workhorse of language modeling for ASR for 30 years.
  - Used in great majority of deployed systems.
- Almost no linguistic knowledge.
  - Totally data-driven.
- Easy to build.
  - Fast and scalable.
The Fundamental Equation of ASR

\[ \omega^* = \arg \max_{\omega} P(\omega|x) = \arg \max_{\omega} P(\omega)P(x|\omega) \]

- **Source-channel** model.
  - Source model \( P(\omega) \) [language model].
  - (Noisy) channel model \( P(x|\omega) \) [acoustic model].
  - Recover \( \omega \) despite corruption from noisy channel.

- Many other applications follow same framework.
Where Else Are Language Models Used?

\[ \omega^* = \arg\max_\omega P(\omega | x) = \arg\max_\omega P(\omega) P(x | \omega) \]

- Handwriting recognition.
- Optical character recognition.
- Spelling correction.
- Machine translation.
- Natural language generation.
- Information retrieval.
- Any problem involving sequences?