Lecture 11:
ASR: Training & Systems

1. Hidden Markov Model review
2. Training HMMs
3. Language modeling
4. Discrimination & adaptation

Dan Ellis  <dpwe@ee.columbia.edu>
http://www.ee.columbia.edu/~dpwe/e6820/
1. **HMM review**

- **HMM** $M_j$ is defined by:
  
  - states $q^i$
  
  - transition probabilities $a_{ij}$
  
  - emission distributions $b_i(x)$

  (+ initial distribution $p(q_1) \equiv \pi_i$)

- **HMMs are a generative model:**

  recognition is inference of $p(M_j | X)$

- (see e6820/papers/Rabiner89-hmm.pdf)
HMM probability inference

- Given states $Q = \{q_1, q_2, \ldots, q_N\}$

& observations $X = X_1^N = \{x_1, x_1, \ldots, x_N\}$

- Markov assumption makes
  
  $$p(X, Q|M) = \prod_{n} p(x_n|q_n)p(q_n|q_{n-1})$$

- Then $p(X|M) = \sum_{all Q} p(X, Q|M)$

- Full calculation via forward recursion:
  
  $$p(X_1^n, q_n^j) = \alpha_n(j) = \left[ \sum_{i=1}^{S} \alpha_{n-1}(i) a_{ij} \right] \cdot b_j(x_n)$$

- Viterbi (best path) approximation
  
  $$\alpha_n^*(j) = \max_i \left\{ \alpha_{n-1}^*(i) a_{ij} \right\} \cdot b_j(x_n)$$

  - then backtrace...
Summing over all paths

Model $M_1$

<table>
<thead>
<tr>
<th>States</th>
<th>S</th>
<th>A</th>
<th>B</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>.9</td>
<td>.1</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>A</td>
<td>.7</td>
<td>.2</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>B</td>
<td>.8</td>
<td>.2</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>E</td>
<td>.1</td>
<td>.1</td>
<td>.1</td>
<td>1</td>
</tr>
</tbody>
</table>

Observations

$p(x|B)$

$p(x|A)$

Observation likelihoods

$p(x|q)$

$\{ q \}$

A

2.5 0.2 0.1

B

0.1 2.2 2.3

Paths

All possible 3-emission paths $Q_k$ from S to E

| $q_0$ | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $p(Q \mid M) = \prod_n p(q_n|q_{n-1})$ | $p(X \mid Q,M) = \prod_n p(x_n|q_n)$ | $p(X,Q \mid M)$ |
|-------|-------|-------|-------|-------|---------------------------------|---------------------------------|-----------------|
| S A A A E | .9 x .7 x .7 x .1 = 0.0441 | 2.5 x .2 x .1 = 0.05 | 0.0022 |
| S A A B E | .9 x .7 x .2 x .2 = 0.0252 | 2.5 x .2 x 2.3 = 1.15 | 0.0290 |
| S A B B E | .9 x .2 x .8 x .2 = 0.0288 | 2.5 x 2.2 x 2.3 = 12.65 | 0.3643 |
| S B B B E | .1 x .8 x .8 x .2 = 0.0128 | 0.1 x 2.2 x 2.3 = 0.506 | 0.0065 |
| $\Sigma = 0.1109$ |

$\Sigma = p(X \mid M) = 0.4020$
HMM example: Different state sequences

Model $M_1$

Model $M_2$

Emission distributions

$p(x \mid q)$
Model inference:
emission probabilities

Model $M_1$

- $\log p(X \mid M) = -32.1$
- **state alignment**
  - $\log p(X,Q^* \mid M) = -33.5$
- **log trans.prob**
  - $\log p(Q^* \mid M) = -7.5$
- **log obs.l'hood**
  - $\log p(X \mid Q^*,M) = -26.0$

Model $M_2$

- $\log p(X \mid M) = -47.0$
- **state alignment**
  - $\log p(X,Q^* \mid M) = -47.5$
- **log trans.prob**
  - $\log p(Q^* \mid M) = -8.3$
- **log obs.l'hood**
  - $\log p(X \mid Q^*,M) = -39.2$
Model inference:
transition probabilities

Model $M'_1$

Model $M'_2$

state alignment

log obs.l'hood

$\log p(X \mid Q^*, M) = -26.0$

$\log p(X, Q^* \mid M) = -33.6$

$\log p(Q^* \mid M) = -7.6$

Model $M'_1$

$\log p(X \mid M) = -32.2$

$\log p(X, Q^* \mid M) = -33.6$

$\log p(Q^* \mid M) = -7.6$

Model $M'_2$

$\log p(X \mid M) = -33.5$

$\log p(X, Q^* \mid M) = -34.9$

$\log p(Q^* \mid M) = -8.9$
Recognition with HMMs

- **Isolated word**
  - choose best $p(M | X) \propto p(X | M) p(M)$

- **Continuous speech**
  - Viterbi decoding of one large HMM gives words
Outline

1. Hidden Markov Model review
2. Training HMMs
   - Viterbi training
   - EM for HMM parameters
   - Forward-backward (Baum-Welch)
3. Language modeling
4. Discrimination & adaptation
Training HMMs

- Probabilistic foundation allows us to train HMMs to ‘fit’ training data
  - i.e. estimate $a_{ij}$, $b_i(x)$ given interpretations

- Algorithms to improve $p(M \mid X)$ are key to success of HMMs

- Problem arises because state alignments $Q$ of training examples are generally unknown
  - Viterbi training
    - choose ‘best’ labels (heuristic)
  - EM training
    - ‘fuzzy labels’ (guaranteed convergence)
Overall training procedure

Labelled training data
- "two one"
- "five"
- "four three"
- ...

Word models
- one
- two
- three
- ...

Data

Models

Fit models to data
Re-estimate model parameters

Repeat until convergence
Viterbi training

- “Fit models to data”
  = Viterbi best-path alignment

\[
\text{Data} \quad \begin{array}{c}
\text{Viterbi} \\
\text{labels} \end{array} \begin{array}{c} Q* \end{array} \quad \text{th} \quad r \quad iy
\]

- “Re-estimate model parameters”:

  pdf e.g. 1D Gauss: \( \mu_i = \frac{\sum_{n \in q^i} x_n}{\#(q_n^i)} \)

  count transitions: \( a_{ij} = \frac{\#(q_{n-1}^i \rightarrow q_n^j)}{\#(q_n^i)} \)

- And repeat...
- But: converges only if good initialization
EM for HMMs

- **Expectation-Maximization (EM):** optimizes models with unknown parameters
  - finds locally-optimal parameters $\Theta$ to maximize data likelihood $p(x_{\text{train}} | \Theta)$
  - makes sense for decision rules like $p(x | M_j) \cdot p(M_j)$

- **Principle:** adjust $\Theta$ to maximize expected log likelihood of known $x$ & unknown $u$:

$$E[\log p(x, u | \Theta)] = \sum_u p(u | x, \Theta_{\text{old}}) \log [p(x | u, \Theta) p(u | \Theta)]$$

  - for GMMs, unknowns $= \text{mix assignments } k$
  - for HMMs, unknowns $= \text{hidden state } q_n$
  (take $\Theta$ to include $M_j$)

- **Interpretation:** “fuzzy” values for unknowns
EM for HMMs (2)

- Expected log likelihood for HMM:

\[
\sum_{\text{all } Q_k} p(Q_k \mid X, \Theta_{\text{old}}) \log \left[ p(X \mid Q_k, \Theta) p(Q_k \mid \Theta) \right]
\]

\[
= \sum_{\text{all } Q_k} p(Q_k \mid X, \Theta_{\text{old}}) \log \left[ \prod_{n} p(x_n \mid q_n) \cdot p(q_n \mid q_{n-1}) \right]
\]

\[
= \sum_{n=1}^{N} \sum_{i=1}^{S} p(q_n^i \mid X, \Theta_{\text{old}}) \log p(x_n \mid q_n^i, \Theta)
\]

\[
+ \sum_{i=1}^{S} p(q_1^i \mid X, \Theta_{\text{old}}) \log p(q_1^i \mid \Theta)
\]

\[
+ \sum_{n=2}^{N} \sum_{i=1}^{S} \sum_{j=1}^{S} p(q_{n-1}^i, q_n^j \mid X, \Theta_{\text{old}}) \log p(q_n^j \mid q_{n-1}^i, \Theta)
\]

- closed-form maximization by differentiation etc.
EM update equations

- For acoustic model (e.g. 1-D Gauss):
  \[
  \mu_i = \frac{\sum_n p(q_n^i|X, \Theta_{\text{old}}) \cdot x_n}{\sum_n p(q_n^i|X, \Theta_{\text{old}})}
  \]

- For transition probabilities:
  \[
  p(q_n^j|q_{n-1}^i) = a_{ij}^{\text{new}} = \frac{\sum_n p(q_{n-1}^i, q_n^j|X, \Theta_{\text{old}})}{\sum_n p(q_{n-1}^i|X, \Theta_{\text{old}})}
  \]

- Fuzzy versions of Viterbi training
  - reduce to Viterbi if \( p(q|X) = 1/0 \)

- Require ‘state occupancy probabilities’,
  \[
  p(q_n^i|X_1^N, \Theta_{\text{old}})
  \]
The forward-backward algorithm

• We need \( p(q_n^i \mid X_1^N) \) for EM updates (\( \Theta \) implied)

• Forward algorithm gives \( \alpha_n(i) = p(q_n^i, X_1^n) \)
  - excludes influence of remaining data \( X_{n+1}^N \)

• Hence, define \( \beta_n(i) = p(X_{n+1}^N \mid q_n^i, X_1^n) \)
  so that \( \alpha_n(i) \cdot \beta_n(i) = p(q_n^i, X_1^n) \)

  then \( p(q_n^i \mid X_1^N) = \frac{\alpha_n(i) \cdot \beta_n(i)}{\sum_j \alpha_n(j) \cdot \beta_n(j)} \)

• Recursive definition for \( \beta \):
  \( \beta_n(i) = \sum_j \beta_{n+1}(j) a_{ij} b_j(x_{n+1}) \)
  - recurses \emph{backwards} from final state \( N \)
Estimating $a_{ij}$ from $\alpha$ & $\beta$

- From EM equations:

$$p(q_n^i | q_{n-1}^i) = a_{ij}^{\text{new}} = \frac{\sum_n p(q_{n-1}^i, q_n^j | X, \Theta_{\text{old}})}{\sum_n p(q_{n-1}^i | X, \Theta_{\text{old}})}$$

- prob. of transition normalized by prob. in first

- Obtain from $p(q_{n-1}^i, q_n^j, X | \Theta_{\text{old}})$

$$= p(X_n^N | q_n^j) p(x_n | q_n^j) p(q_n^j | q_{n-1}^i) p(q_{n-1}^i, X_1^{n-1})$$

$$= \beta_n(j) \cdot b_j(x_n) \cdot a_{ij} \cdot \alpha_{n-1}(i)$$
GMM-HMMs in practice

- GMMs as acoustic models:
  train by including mixture indices as unknowns
  - just more complicated equations...

\[
\mu_{ik} = \frac{\sum_n p(m_k|q^i, x_n, \Theta_{old}) p(q^i_n|X, \Theta_{old}) \cdot x_n}{\sum_n p(m_k|q^i, x_n, \Theta_{old}) p(q^i_n|X, \Theta_{old})}
\]

- Practical GMMs:
  - 9 to 39 feature dimensions
  - 2 to 64 Gaussians per mixture depending on number of training examples

- Lots of data → can model more classes
  - e.g context-independent (CI): \(q^i = ae\ aa\ ax\ ...
  \rightarrow context-dependent (CD): \(q^i = b-ae-b\ b-ae-k\ ...

Training summary

• Training data + basic model topologies → derive fully-trained models
  - alignment all handled implicitly

• What do the states end up meaning?
  - not necessarily what you intended; whatever locally maximizes data likelihood

• What if the models or transcriptions are bad?
  - slow convergence, poor discrimination in models

• Other kinds of data, transcriptions
  - less constrained initial models...

ONE = w ah n
TWO = t uw
Outline

1. Hidden Markov Models review
2. Training HMMs
3. Language modeling
   - Pronunciation models
   - Grammars
   - Decoding
4. Discrimination & adaptation
Language models

- Recall, MAP recognition criterion:
  \[ M^* = \arg\max_{M_j} p(M_j|X, \Theta) \]
  
  \[ = \arg\max_{M_j} p(X|M_j, \Theta_A)p(M_j|\Theta_L) \]

- So far, looked at \( p(X|M_j, \Theta_A) \)

- What about \( p(M_j|\Theta_L) \)?
  - \( M_j \) is a particular word sequence
  - \( \Theta_L \) are parameters related to the language

- Two components:
  - link state sequences to words \( p(Q|w_i) \)
  - priors on word sequences \( p(w_i|M_j) \)
HMM Hierarchy

- **HMMs support composition**
  - can handle time dilation, pronunciation, grammar within the same framework

\[
p(q|M) = p(q, \Phi, w|M) = p(q|\Phi) \cdot p(\phi|w) \cdot p(w_n|w_1^{n-1}, M)
\]
Pronunciation models

- Define states within each word \( p(Q|w_i) \)
- Can have unique states for each word (‘whole-word’ modeling), or ...
- Sharing (tying) subword units between words to reflect underlying phonology
  - more training examples for each unit
  - generalizes to unseen words
  - (or can do it automatically...)
- Start e.g. from pronouncing dictionary:

  ZERO\( (0.5) \)  z iy r ow
  ZERO\( (0.5) \)  z ih r ow
  ONE\( (1.0) \)  w ah n
  TWO\( (1.0) \)  tcl t uw
  ...
Learning pronunciations

- ‘Phone recognizer’ transcribes training data as phones
  - align to ‘canonical’ pronunciations

\[ \text{Baseform Phoneme String} \]
\[ f \text{ay} \text{v} \text{y iy r ow l d} \]
\[ f \text{ah ay v y uh r ow l} \]

\[ \text{Surface Phone String} \]
- infer modification rules
- predict other pronunciation variants

- e.g. ‘d deletion’:
  \[ d \rightarrow \emptyset / l _ { [\text{stop}] } \quad p = 0.9 \]

- Generate pronunciation variants; use forced alignment to find weights
Grammar

• Account for different likelihoods of different words and word sequences \( p(w_i | M_j) \)

• ‘True’ probabilities are very complex for LVCSR
  - need parses, but speech often agrammatic

→ **Use n-grams:**
\[
p(w_n | w_1^L) = p(w_n | w_{n-K}, \ldots, w_{n-1})
\]
  - e.g. n-gram models of Shakespeare:

  n=1  To him swallowed confess hear both. Which. Of save on ...
  n=2  What means, sir. I confess she? then all sorts, he is trim, ...
  n=3  Sweet prince, Falstaff shall die. Harry of Monmouth's grave...
  n=4  King Henry. What! I will go seek the traitor Gloucester. ...

• **Big win in recognizer WER**
  - raw recognition results often highly ambiguous
  - grammar guides to ‘reasonable’ solutions
Smoothing LVCSR grammars

- n-grams (n=3 or 4) are estimated from large text corpora
  - 100M+ words
  - but: not like spoken language

- 100,000 word vocabulary → $10^{15}$ trigrams!
  - never see enough examples
  - unobserved trigrams should NOT have Pr=0!

- Backoff to bigrams, unigrams
  - $p(w_n)$ as an approx to $p(w_n | w_{n-1})$ etc.
  - interpolate 1-gram, 2-gram, 3-gram with learned weights?

- Lots of ideas e.g. category grammars
  - e.g. $p(\text{PLACE} | \text{“went”, “to”}) \cdot p(w_n | \text{PLACE})$
  - how to define categories?
  - how to tag words in training corpus?
Decoding

- How to find the MAP word sequence?
- States, prons, words define one big HMM
  - with 100,000+ individual states for LVCSR!

→ Exploit hierarchic structure
  - phone states independent of word
  - next word (semi) independent of word history

```
root
  d uw
    iy

k
  oy
    DECOY

ow
  d
    DECODE

z
  axr
    DECODER

s
  DECODES

DO DECOY DECODES DECODES DECODER
```
Decoder pruning

- **Searching ‘all possible word sequences’?**
  - need to restrict search to most promising ones: *beam search*
  - sort by estimates of total probability
    - $Pr(\text{so far}) + \text{lower bound estimate of remains}$
    - trade search errors for speed

- **Start-synchronous algorithm:**
  - extract top hypothesis from queue:
    $$[P_n, \{w_1, \ldots, w_k\}, n]$$
    - pr. so far     words     next time frame
  - find plausible words $\{w^i\}$ starting at time $n$
    $\rightarrow$ new hypotheses:
    $$[P_n \cdot \left( X_n^{n+N-1} \right | w^i) \cdot P(w^i \mid w_k \ldots), \{w_1, \ldots, w_k, w^i\}, n+N]$$
    - discard if too unlikely, or queue is too long
    - else re-insert into queue and repeat
Outline

1. Hidden Markov Models review
2. Training HMMs
3. Language modeling
4. Discrimination & adaptation
   - Discriminant models
   - Neural net acoustic models
   - Model adaptation
4 Discriminant models

- **EM training of HMMs is *maximum likelihood***
  - i.e. local max $p(X_{trn} \mid \Theta)$
  - converges to Bayes optimum ... in the limit

- **Decision rule is** $\max p(X \mid M) \cdot p(M)$
  - training will increase $p(X \mid M_{\text{correct}})$
  - may also increase $p(X \mid M_{\text{wrong}})$ ...more?

- **Discriminant training tries directly to increase discrimination between right & wrong models**
  - e.g. Maximum Mutual Information (MMI)

\[
I(M_j, X \mid \Theta) = \log \frac{p(M_j, X \mid \Theta)}{p(M_j \mid \Theta) p(X \mid \Theta)} \\
= \log \frac{p(X \mid M_j, \Theta)}{\sum p(X \mid M_k, \Theta) p(M_k \mid \Theta)}
\]
Neural Network Acoustic Models

- Single model generates posteriors directly for all classes at once = frame-discriminant
- Use regular HMM decoder for recognition
  - set $b_i(x_n) = p(x_n | q^i) \propto p(q^i | x_n)/p(q^i)$
- Nets are less sensitive to input representation
  - skewed feature distributions
  - correlated features
- Can use temporal context window to let net ‘see’ feature dynamics:

$$p(q^i | X)$$
Neural nets: Practicalities

- **Typical net sizes:**
  - input layer: 9 frames x 9-40 features ~ 300 units
  - hidden layer: 100-8000 units, dep. train set size
  - output layer: 30-60 context-independent phones

- **Hard to make context dependent**
  - problems training many classes that are similar?

- **Representation is opaque:**

  - *Hidden -> Output weights*
  - *Input -> Hidden #187*
Model adaptation

- **Practical systems often suffer from mismatch**
  - test conditions are not like training data:
    accent, microphone, background noise ...

- **Desirable to continue tuning during recognition**
  = adaptation
  - but: no ‘ground truth’ labels or transcription

- **Assume that recognizer output is correct;**
  **Estimate a few parameters from those labels**
  - e.g. Maximum Likelihood Linear Regression (MLLR)
Recap: Recognizer Structure

- Now we have it all!
Summary

• **Hidden Markov Models**
  - state transitions and emission likelihoods in one
  - best path (Viterbi) performs recognition

• **HMMs can be trained**
  - Viterbi training makes intuitive sense
  - EM training is guaranteed to converge
  - acoustic models (e.g. GMMs) train at same time

• **Language modeling captures higher structure**
  - pronunciation, word sequences
  - fits directly into HMM state structure
  - need to ‘prune’ search space in decoding

• **Further improvements...**
  - discriminant training moves models ‘apart’
  - adaptation adjusts models in new situations