Hidden Markov Models use for speech recognition

Contents:
- Viterbi training
- Acoustic modeling aspects
- Isolated-word recognition
- Connected-word recognition
- Token passing algorithm
- Language models

Phoneme HMM

- Each phoneme is represented by a left-to-right HMM with 3 states

  \[
  \begin{align*}
  & s_1 \quad & a_{11} \quad & a_{12} \quad & a_{21} \quad & a_{22} \quad & a_{32} \quad & a_{33} \\
  & s_2 \quad & a_{12} \quad & a_{22} \quad & a_{23} \quad & a_{33} \\
  & s_3 
  \end{align*}
  \]

- Word and sentence HMMs are constructed by concatenating the phoneme-level HMMs

  "ONE"

  \[
  \begin{align*}
  & W \quad & A \quad & X \quad & N 
  \end{align*}
  \]

Viterbi training

- In training, the goal is to "assign" extracted feature vectors to HMM states
- Forward-backward algorithm assigns a probability that a feature vector was emitted from an HMM state
- Viterbi training: we construct the composite HMM from the phoneme units and use Viterbi algorithm to find the best state-sequence → assignment of feature vectors to HMM states

  \[
  \begin{align*}
  & \sigma_1 \quad & \sigma_t \quad & \sigma_T 
  \end{align*}
  \]

  For each training example, use current HMM models to assign feature vectors to HMM states
    - Using Viterbi algorithm, find the most likely path through the composite HMM model
    - This is called Viterbi forced alignment

  Group the feature vectors assigned to each HMM state and estimate new parameters for each HMM (for example using the GMM update equations)

  Repeat alignment and parameter reestimation
Acoustic models

An ideal acoustic model is:

- **Accurate**
  - It accounts for context dependency (phonetic context)

- **Compact**
  - It provides a compact representation, trainable from finite amounts of data

- **General**
  - It is a general representation that allows new words to be modeled, even if they were not seen in the training data

Whole-word HMMs

- Each word is modeled as a whole
- Each word is assigned an HMM with a number of states

Is it a good acoustic model?
- **Accurate**
  - Yes, if there is enough data and the system has a small vocabulary; No, if trying to model context changes between words
- **Compact**
  - No. It needs many states as the vocabulary increases, and there might not be enough training data to model EVERY word.
- **General**
  - No. It cannot be used to build new words.

Phoneme HMMs

- Each phoneme is modeled using an HMM with M states
- "monophone" models

Is it a good acoustic model?
- **Accurate**
  - No. It does not model well coarticulation.
- **Compact**
  - Yes. The complete system will have M states and N phonemes, a total of MxN states, not so many parameters to be estimated
- **General**
  - Yes. Any new word can be formed by concatenating the units.

Modeling phonetic context

- **Monophone**
  - A single model is used to represent a phoneme in all contexts

- **Biphone**
  - One model represents a particular left or right context
  - Notation:
    - left context biphone: (a-b)
    - right context biphone: (b+c)

- **Triphone**
  - One model represents a particular left and right context
  - Notation: (a-b+c)
Context-dependent model examples

- Monophone
  - SPEECH S P I Y CH
- Biphone
  - Left context:
  - Right context:
- Triphone

Context-dependent triphone HMMs

- Each phoneme unit within the immediate left and right context is modeled using an HMM with M states
- Is it a good acoustic model?
  - Accurate – Yes. Takes into account coarticulation.
  - Compact – Yes. Trainable – No. For N phonemes there are NxNxN triphone models, too many parameters to estimate!
  - General – Yes. New words can be formed by concatenating units
- Training issues
  - Not all triphones appear in training data ("unseen" triphones)
  - Many triphones occur infrequently – not enough training data
  - Solution: clustering of HMM states which have similar statistical distributions, to estimate HMM parameters using pooled data

Isolated word recognition

- Whole-word model
  - Collect many examples of each word spoken in isolation
  - Assign a number of states to each word model based on word duration
  - Estimate HMM model parameters
- Subword-unit model
  - Collect a large corpus of speech and estimate phonetic unit HMMs
  - Construct word-level HMMs from phoneme-level HMMs
  - This is more general than the whole-word approach
Whole-word HMM

HMM for word "one"

Viterbi algorithm through a model

Isolated word recognition system

- \( P(O|W) \) calculated using Viterbi algorithm rather than forward algorithm
- Viterbi provides the probability of the path represented by the most likely state sequence

Connected-word recognition

- Boundaries of utterance are unknown
- Number of words spoken is unknown – position of word boundaries is often unclear, difficult to determine
- Example: two word network
Connected-words Viterbi search

At each node we must compute $\tilde{\delta}_t(j)$ - the probability of the best state sequence up to that point, and keep the information about where it came from – this will allow back-tracing to find the best state sequence.

During back-tracing we will find the word boundaries.

Beam pruning:
- at each point determine the log-probability of the absolute best Viterbi path
  \[ \tilde{\delta}_t^{MAX} = \max_{1 \leq i \leq N} [\tilde{\delta}_t(i)] \]
- Prune away paths which fall below a "beam width" from the maximum probable path. "Deactivate" state $j$ if
  \[ \tilde{\delta}_t(j) < \tilde{\delta}_t^{MAX} - BW \]

Beam pruning illustration

Token passing approach
- Assume each HMM state can hold multiple tokens
- Token is an object that can move from state to state in the HMM network
- Each token carries with it the log scale Viterbi path score $s$
- At each time $t$ we examine tokens assigned to the nodes
- We propagate tokens to reachable positions at time $t+1$:
  - Make a copy of the token
  - Adjust path score to account for the transition within the HMM network and observation probability
- Merge tokens according to Viterbi algorithm
  - Select the token with maximum score
  - Discard all other competing tokens
Token passing algorithm

- **Initialization (t=0)**
  - Initialize each initial state to hold a token with score \( s = 0 \)
  - All other states are initialized with a token with \( s = \infty \)

- **Algorithm (t>0)**
  - Propagate tokens to all possible next states (all connecting states) and increment
    \[ s = s + \tilde{a}_{ij} + \tilde{b}_j (o_i) \]
  - In each state, find the token with the largest \( s \) and discard the rest of the tokens in that state (Viterbi)

- **Termination (t=T)**
  - Examine the tokens in all possible final states, find the one with the largest Viterbi path score
  - This is the probability of the most likely state sequence

Token passing for connected-word recognition

- Individual word models are connected into a composite model – can transition from final state of word \( m \) to initial state of word \( n \)
- Path scores are maintained by the tokens
- Path sequence also maintained by the tokens, allowing recovery of the best word sequence

\[ s = s + P(W_i) \]

Tokens emitted from last state of each word propagate to initial state of each word

Probability of entering the initial state of each word \( P(W_i) \) is the probability of that word given by the language model

Bayes formulation revisited

- Recall the Bayes rule applied to speech recognition

\[
P(W \mid O) = \frac{P(O \mid W)P(W)}{P(O)}
\]

\[
\hat{W} = \arg \max_{W} P(W \mid O) = \arg \max_{W} P(O \mid W)P(W)
\]

- In practice, we use log-probabilities:

\[
\hat{W} = \arg \max_{W} \log(P(O \mid W)P(W))
\]

Probabilities of word sequences, given by the language model
Language models

- Usually the language model is also scaled by a grammar scale factor $s$ and word transition penalty $p$

$$\hat{W} = \arg \max_w \left\{ \log(P(O \mid W)P(W)) \right\}$$

$$\hat{W} = \arg \max_w \left\{ \log(P(O \mid W)) + s \cdot \log(P(W)) + p \right\}$$

Language models

- Assign probabilities to word sequences $P(W)$
- The additional information provides help to reduce the search space
- Language models resolve homonyms:
  
  Write a letter to Mr. Wright right away.
- Tradeoff between constraint and flexibility

Statistical language models

- We want to estimate

$$P(W) = P(w_1, w_2, \ldots, w_N)$$

- We can decompose this probability left-to-right:

$$P(W) = P(w_1, w_2, \ldots, w_N)$$

$$= P(w_1)P(w_2 \mid w_1) \cdots P(w_N \mid w_1, w_2, \ldots, w_{N-1})$$

$$= \prod_{n=1}^{N} P(w_n \mid w_1, w_2, \ldots, w_{n-1})$$

How does this work?

- $P(W) = P(\text{analysis of audio, speech and music signals})$
  
  $= P(\text{analysis}) \ P(\text{of | analysis}) \ P(\text{audio | analysis of})$ …

  … $P(\text{signals | analysis of audio speech and music})$

- How can we model the entire word sequence? There is never enough training data!
- Consider restricting the word history
Practical training

Consider word-histories ending in the same last N-1 words, and treat it as a Markov model.

- N = 1
  \[ P(w_n | w_1, w_2 \ldots w_{n-1}) = P(w_n) \]

- N = 2
  \[ P(w_n | w_1, w_2 \ldots w_{n-1}) = P(w_n | w_{n-1}) \]

- N = 3
  \[ P(w_n | w_1, w_2 \ldots w_{n-1}) = P(w_n | w_{n-1}, w_{n-2}) \]

n-gram language models

- Probability of a word based on the previous N-1 words:
  - N=1 – unigram
  - N=2 – bigram
  - N=3 – trigram

- Training: probabilities are estimated from a corpus of training data (a large amount of text)
- Once the model is trained, it can be used to generate new sentences randomly
- Syntax is roughly encoded by the obtained model, but generated sentences are often ungrammatical and semantically strange

Trigram example

...the United States of ???

\[ P(\text{states} | \text{the united}) = \ldots \]
\[ P(\text{of} | \text{states of}) = \ldots \]
\[ P(\text{America} | \text{states of}) = \ldots \]
\[ P(\text{India} | \text{states of}) = \ldots \]
\[ P(\text{matter} | \text{states of}) = \ldots \]
\[ P(\text{Australia} | \text{states of}) = \ldots \]
\[ P(\text{Mexico} | \text{states of}) = \ldots \]

Estimating the n-gram probabilities

- Given a text corpus, define:
  - Count of occurrences of word \( n \)
    \[ C(w_n) \]
  - Count of occurrences of word \( n-1 \) followed by word \( n \)
    \[ C(w_{n-1}, w_n) \]
  - Count of occurrences of word \( n-2 \) followed by word \( n-1 \) and word \( n \)
    \[ C(w_{n-2}, w_{n-1}, w_n) \]
Estimating the n-gram probabilities

- Based on the count frequency of occurrence for the word sequences, the maximum likelihood estimates of word probabilities are calculated:

\[
P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}, w_n)}{C(w_{n-1})}
\]

\[
P(w_n \mid w_{n-2}, w_{n-1}) = \frac{C(w_{n-2}, w_n, w_{n-1})}{C(w_{n-2}, w_{n-1})}
\]

n-grams in the decoding process

- The goal of the search is to find the most likely string of symbols (phonemes, words, etc) to account for the observed speech waveform:

\[
\hat{W} = \arg \max_w P(O \mid W)P(W)
\]

- Connected-word example:

Beam search revisited

- At each node we must compute

\[
\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + a_{ij} + \beta_j(y_t)] + b_j(y_t)
\]

where \( \beta_{ij} \) is the log language model score

\[
\beta_j = \begin{cases} 
  s \tilde{P}(W_k) + p & \text{if "i" is the last state of any word} \\
  0 & \text{otherwise} 
\end{cases}
\]

- \( s \) is the grammar scale factor and \( p \) is the (log) word transition penalty
Language model in the search

- The language model scores are applied at the point where there is a transition INTO a word.
- As the number of words increases, the number of states and interconnections increases too.
- N-grams are easier to incorporate into the token passing algorithm.

\[ s = s + gP(W_i) + p \]

The language model score is added to the path score upon word entry, so the token keeps the combined acoustic and language model information.

*Note: here g is the grammar scale factor, as s was used to denote the path score.

Lyrics recognition from singing

<table>
<thead>
<tr>
<th>Correct transcription</th>
<th>Recognized</th>
</tr>
</thead>
<tbody>
<tr>
<td>yesterday</td>
<td>yes today</td>
</tr>
<tr>
<td>seemed so far away</td>
<td>seem to find away</td>
</tr>
<tr>
<td>my my</td>
<td>mama</td>
</tr>
<tr>
<td>finding the answer</td>
<td>fighting the answer</td>
</tr>
<tr>
<td>the distance in your eyes</td>
<td>from this is in your eyes</td>
</tr>
<tr>
<td>all the way</td>
<td>all way</td>
</tr>
<tr>
<td>cause it's a bittersweet</td>
<td>cause I said bittersweet</td>
</tr>
<tr>
<td>this life</td>
<td>this our life</td>
</tr>
<tr>
<td>trying to make ends meet</td>
<td>trying to maintain sweetest</td>
</tr>
<tr>
<td>you're a slave to the</td>
<td>ain't gettin' money</td>
</tr>
<tr>
<td>money</td>
<td>then you die</td>
</tr>
</tbody>
</table>

Y EH S T ER D EY vs Y EH S . T AH D EY
MAY . MAY vs MAA M AH
AO L . DH AH . W EY vs AO L . AH W EY