Agenda

• Speech Recognition
  – Hidden Markov Models
    • Uncertain observations
    • Recognition: Viterbi, Stack/A*
    • Training the model: Baum-Welch
Speech Recognition Model

• Question: Given signal, what words?
• Problem: uncertainty
  – Capture of sound by microphone, how phones produce sounds, which words make phones, etc
• Solution: Probabilistic model
  – $P(\text{words}|\text{signal}) = \frac{P(\text{signal}|\text{words})P(\text{words})}{P(\text{signal})}$
  – Idea: Maximize $P(\text{signal}|\text{words})*P(\text{words})$
    • $P(\text{signal}|\text{words})$: acoustic model; $P(\text{words})$: lang model
Hidden Markov Models (HMMs)

• An HMM is:
  – 1) A set of states: \( Q = q_0, q_1, \ldots, q_k \)
  – 2) A set of transition probabilities: \( A = a_{01}, \ldots, a_{mn} \)
    • Where \( a_{ij} \) is the probability of transition \( q_i \rightarrow q_j \)
  – 3) Observation probabilities: \( B = b_i(o_t) \)
    • The probability of observing \( o_t \) in state \( i \)
  – 4) An initial probability dist over states: \( \pi_i \)
    • The probability of starting in state \( i \)
  – 5) A set of accepting states
Acoustic Model

• 3-state phone model for [m]
  – Use Hidden Markov Model (HMM)
  
  Observation probabilities

  
  Transition probabilities

  
  Onset 0.3
  C1: 0.5  
  C2: 0.2  
  C3: 0.3  
  Mid 0.9
  C3: 0.2  
  C4: 0.7  
  End 0.4
  C4: 0.1  
  C6: 0.4  
  Final 0.6
  C6: 0.5

  – Probability of sequence: sum of prob of paths
Viterbi Algorithm

• Find BEST word sequence given signal
  – Best $P(\text{words}|\text{signal})$
  – Take HMM & VQ sequence
    • $\Rightarrow$ word seq (prob)

• Dynamic programming solution
  – Record most probable path ending at a state $i$
    • Then most probable path from $i$ to end
    • $O(bMn)$
Viterbi Code

Function Viterbi(observations length T, state-graph) returns best-path
Num-states<-num-of-states(state-graph)
Create path prob matrix viterbi[num-states+2,T+2]
Viterbi[0,0]<- 1.0
For each time step t from 0 to T do
  for each state s from 0 to num-states do
    for each transition s’ from s in state-graph
      new-score<-viterbi[s,t]*at[s,s’]*bs’(ot)
      if ((viterbi[s’,t+1]=0) || (viterbi[s’,t+1]<new-score))
        then
          viterbi[s’,t+1] <- new-score
          back-pointer[s’,t+1]<-s
    Backtrace from highest prob state in final column of viterbi[] & return
Enhanced Decoding

• Viterbi problems:
  – Best phone sequence not necessarily most probable word sequence
    • E.g. words with many pronunciations less probable
  – Dynamic programming invariant breaks on trigram

• Solution 1:
  – Multipass decoding:
    • Phone decoding -> n-best lattice -> rescoring (e.g. tri)
Enhanced Decoding: A*

- Search for highest probability path
  - Use forward algorithm to compute acoustic match
  - Perform **fast match** to find next likely words
    - Tree-structured lexicon matching phone sequence
  - Estimate path cost:
    - Current cost + underestimate of total
  - Store in priority queue
  - Search best first
Modeling Sound, Redux

- **Discrete VQ codebook values**
  - Simple, but inadequate
  - Acoustics highly variable

- **Gaussian pdfs over continuous values**
  - Assume normally distributed observations
    - Typically sum over multiple shared Gaussians
      - “Gaussian mixture models”
      - Trained with HMM model

\[
b_j(o_t) = \frac{1}{\sqrt{(2\pi)|\sum_j|}} e^{-\frac{1}{2}((o_t-\mu_j)^T)(\sum_j^{-1})(o_t-\mu_j))}
\]
Learning HMMs

- Issue: Where do the probabilities come from?
- Solution: Learn from data
  - Trains transition \((a_{ij})\) and emission \((b_j)\) probabilities
    - Typically assume structure
  - Baum-Welch aka forward-backward algorithm
    - Iteratively estimate counts of transitions/emitted
    - Get estimated probabilities by forward comput’n
      - Divide probability mass over contributing paths
Forward Probability

$$\alpha_j(1) = a_{1j}b_j(o_1), 1 < j < N$$

$$\alpha_j(t) = \left[ \sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij} \right] b_j(o_t)$$

$$P(O \mid \lambda) = \alpha_N(T) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN}$$

Where $\alpha$ is the forward probability, $t$ is the time in utterance, $i, j$ are states in the HMM, $a_{ij}$ is the transition probability, $b_j(o_t)$ is the probability of observing $o_t$ in state $bj$, $N$ is the final state, $T$ is the last time, and $1$ is the start state.
Backward Probability

\[ \beta_i (T) = a_{iN} \]

\[ \beta_i (t) = \sum_{i=2}^{N-1} a_{ij} b_j (o_{t+1}) \beta_j (t+1) \]

\[ P(O | \lambda) = \alpha_N (T) = \beta_1 (T) = \sum_{j=2}^{N-1} a_{1j} b_j (o_1) \beta_j (1) \]

Where \( \beta \) is the backward probability, \( t \) is the time in utterance, \( i,j \) are states in the HMM, \( a_{ij} \) is the transition probability, \( b_j(o_t) \) is the probability of observing \( o_t \) in state \( b_j \), \( N \) is the final state, \( T \) is the last time, and \( 1 \) is the start state.
Re-estimating

- Estimate transitions from i->j

\[ \tau_t(i, j) = \frac{\alpha_i(t)a_{ij}b_j(o_t)\beta_j(t+1)}{\alpha_N(T)} \]

\[ \hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \tau_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \tau_t(i, j)} \]

- Estimate observations in j

\[ \sigma_j(t) = \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)} = \frac{\alpha_j(t)\beta_j(t)}{P(O | \lambda)} \]

\[ \hat{b}_j(v_k) = \frac{\sum_{t=1, s.t. o_t = v_k}^{T} \sigma_j(t)}{\sum_{t=1}^{T} \sigma_j(t)} \]
ASR Training

• Models to train:
  – Language model: typically tri-gram
  – Observation likelihoods: B
  – Transition probabilities: A
  – Pronunciation lexicon: sub-phone, word

• Training materials:
  – Speech files – word transcription
  – Large text corpus
  – Small phonetically transcribed speech corpus
Training

• Language model:
  – Uses large text corpus to train n-grams
    • 500 M words

• Pronunciation model:
  – HMM state graph
  – Manual coding from dictionary
    • Expand to triphone context and sub-phone models
HMM Training

• Training the observations:
  – E.g. Gaussian: set uniform initial mean/variance
    • Train based on contents of small (e.g. 4hr) phonetically labeled speech set (e.g. Switchboard)

• Training A&B:
  – Forward-Backward algorithm training
Does it work?

• Yes:
  – 99% on isolate single digits
  – 95% on restricted short utterances (air travel)
  – 80+% professional news broadcast

• No:
  – 55% Conversational English
  – 35% Conversational Mandarin
  – ?? Noisy cocktail parties
Speech Synthesis

• Text to speech produces
  – Sequence of phones, phone duration, phone pitch

• Most common approach:
  – Concatentative synthesis
    • Glue waveforms together

• Issue: Phones depend heavily on context
  – Diphone models: mid-point to mid-point
    • Captures transitions, few enough contexts to collect (1-2K)
Speech Synthesis: Prosody

• Concatenation intelligible but unnatural
• Model duration and pitch variation
  – Could extract pitch contour directly
  – Common approach: TD-PSOLA
    • Time-domain pitch synchronous overlap and add
      – Center frames around pitchmarks to next pitch period
      – Adjust prosody by combining frames at pitchmarks for desired pitch and duration
      – Increase pitch by shrinking distance b/t pitchmarks
      – Can be squeaky
Speech Recognition as Modern AI

• Draws on wide range of AI techniques
  – Knowledge representation & manipulation
    • Optimal search: Viterbi decoding
  – Machine Learning
    • Baum-Welch for HMMs
    • Nearest neighbor & k-means clustering for signal id
  – Probabilistic reasoning/Bayes rule
    • Manage uncertainty in signal, phone, word mapping

• Enables real world application