Entropy &
Hidden Markov Models

Natural Language Processing
CMSC 35100
April 22, 2003
Agenda

• Evaluating N-gram models
  – Entropy & perplexity
    • Cross-entropy, English

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

• Entropy & Perplexity
  – Information theoretic measures
  – Measures information in grammar or fit to data
  – Conceptually, lower bound on # bits to encode

• Entropy: $H(X)$: $X$ is a random var, $p$: prob fn

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

– E.g. 8 things: number as code => 3 bits/trans
– Alt. short code if high prob; longer if lower
  • Can reduce

• Perplexity: $2^H$
  – Weighted average of number of choices
Entropy of a Sequence

• Basic sequence
  \[ \frac{1}{n} H(W_1^n) = -\frac{1}{n} \sum_{W_1^n \in L} p(W_1^n) \log_2 p(W_1^n) \]

• Entropy of language: infinite lengths
  – Assume stationary
    & ergodic
    \[ H(L) = \lim_{n \to \infty} -\frac{1}{n} \sum_{W \in L} p(w_1, \ldots, w_n) \log p(w_1, \ldots, w_n) \]
    \[ H(L) = \lim_{n \to \infty} -\frac{1}{n} \log p(w_1, \ldots, w_n) \]
Cross-Entropy

• Comparing models
  – Actual distribution unknown
  – Use simplified model to estimate
    • Closer match will have lower cross-entropy
    \[ H(p, m) = \lim_{n \to \infty} -\frac{1}{n} \sum_{w \in L} p(w_1, \ldots, w_n) \log m(w_1, \ldots, w_n) \]
    \[ H(p, m) = \lim_{n \to \infty} -\frac{1}{n} \log m(w_1, \ldots, w_n) \]
Entropy of English

- Shannon’s experiment
  - Subjects guess strings of letters, count guesses
  - Entropy of guess seq = Entropy of letter seq
  - 1.3 bits; Restricted text

- Build stochastic model on text & compute
  - Brown computed trigram model on varied corpus
  - Compute (pre-char) entropy of model
  - 1.75 bits
Speech Recognition

• Goal:
  – Given an acoustic signal, identify the sequence of words that produced it
  – Speech understanding goal:
    • Given an acoustic signal, identify the meaning intended by the speaker

• Issues:
  – Ambiguity: many possible pronunciations,
  – Uncertainty: what signal, what word/sense produced this sound sequence
Decomposing Speech Recognition

• Q1: What speech sounds were uttered?
  – Human languages: 40-50 phones
    • Basic sound units: b, m, k, ax, ey, ...(arpabet)
    • Distinctions categorical to speakers
      – Acoustically continuous
    • Part of knowledge of language
      – Build per-language inventory
      – Could we learn these?
Decomposing Speech Recognition

• Q2: What words produced these sounds?
  – Look up sound sequences in dictionary
  – Problem 1: Homophones
    • Two words, same sounds: too, two
  – Problem 2: Segmentation
    • No “space” between words in continuous speech
    • “I scream”/”ice cream”, “Wreck a nice beach”/”Recognize speech”

• Q3: What meaning produced these words?
  – NLP (But that’s not all!)
Signal Processing

• Goal: Convert impulses from microphone into a representation that
  – is compact
  – encodes features relevant for speech recognition

• Compactness: Step 1
  – Sampling rate: how often look at data
    • 8KHz, 16KHz,(44.1KHz= CD quality)
  – Quantization factor: how much precision
    • 8-bit, 16-bit (encoding: u-law, linear…)
(A Little More) Signal Processing

• Compactness & Feature identification
  – Capture mid-length speech phenomena
    • Typically “frames” of 10ms (80 samples)
      – Overlapping
  – Vector of features: e.g. energy at some frequency
  – Vector quantization:
    • n-feature vectors: n-dimension space
      – Divide into m regions (e.g. 256)
      – All vectors in region get same label - e.g. C256
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
  - \( P(\text{words}|\text{signal}) = \)
  - \( P(\text{signal}|\text{words})P(\text{words})/P(\text{signal}) \)
  - Idea: Maximize \( P(\text{signal}|\text{words})*P(\text{words}) \)
    - \( P(\text{signal}|\text{words}): \) acoustic model; \( P(\text{words}): \) lang model
Probabilistic Reasoning over Time

• Issue: Discrete models
  – Speech is continuously changing
  – How do we make observations? States?

• Solution: Discretize
  – “Time slices”: Make time discrete
  – Observations, States associated with time: Ot, Qt
Modelling Processes over Time

• Issue: New state depends on preceding states
  – Analyzing sequences

• Problem 1: Possibly unbounded # prob tables
  – Observation+State+Time

• Solution 1: Assume stationary process
  – Rules governing process same at all time

• Problem 2: Possibly unbounded # parents
  – Markov assumption: Only consider finite history
  – Common: 1 or 2 Markov: depend on last couple
Language Model

• Idea: some utterances more probable

• Standard solution: “n-gram” model
  – Typically tri-gram: $P(w_i|w_{i-1},w_{i-2})$
    • Collect training data
      – Smooth with bi- & uni-grams to handle sparseness
  – Product over words in utterance
Acoustic Model

- $P(\text{signal}|\text{words})$
  - words $\rightarrow$ phones + phones $\rightarrow$ vector quantiz’n
- Words $\rightarrow$ phones
  - Pronunciation dictionary lookup
    - Multiple pronunciations?
      - Probability distribution
        - Dialect Variation: tomato
        - +Coarticulation
      - Product along path
Acoustic Model

• $P(\text{signal}|\ \text{phones})$:
  – Problem: Phones can be pronounced differently
    • Speaker differences, speaking rate, microphone
    • Phones may not even appear, different contexts
  – Observation sequence is uncertain

• Solution: Hidden Markov Models
  – 1) Hidden $\Rightarrow$ Observations uncertain
  – 2) Probability of word sequences $\Rightarrow$
    • State transition probabilities
  – 3) $1^{\text{st}}$ order Markov $\Rightarrow$ use 1 prior state
Hidden Markov Models (HMMs)

• An HMM is:
  – 1) A set of states: \( Q = q_o, q_1, ..., q_k \)
  – 2) A set of transition probabilities: \( A = a_{01}, ..., a_{mn} \)
    • 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)

  ![Diagram of transition and observation probabilities]

  - Transition probabilities:
    - Onset: 0.3, 0.7
    - Mid: 0.9, 0.1, 0.2
    - End: 0.4, 0.6
  - Observation probabilities:
    - C1: 0.5, C2: 0.2, C3: 0.3, C4: 0.7, C5: 0.1, C6: 0.4

  - 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

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

\[ \alpha_t(i) = P(o_1, o_2, \ldots, o_t, q_t = j \mid \lambda) \]

\[ \alpha_j(1) = a_{1j} b_j(o_t), 1 < j < N \]

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

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

\[ \beta_i(t) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = j, \lambda) \]
\[ \beta_i(T) = a_{iN} \]
\[ \beta_i(t) = \sum_{i=2}^{N-1} a_{ij} b_j(o_{t+1}) \beta_j(t+1) \]

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

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

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