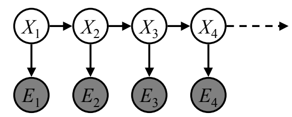
CS 188: Artificial Intelligence
Fall 2007

Lecture 22: Viterbi
11/13/2007

Dan Klein – UC Berkeley

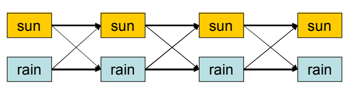
Hidden Markov Models



- An HMM is
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X|X_{-1})$
  - Emissions: $P(E|X)$

Most Likely Explanation

- Remember: weather Markov chain



- Tracking: $P(x_t) \Rightarrow P(x_t|e_{1:t})$
- Viterbi: $\arg\max_{x_{1:t}} P(x_{1:t}) \Rightarrow \arg\max_{x_{1:t}} P(x_{1:t}|e_{1:t})$

Most Likely Explanation

- Question: most likely sequence ending in x at t?
  - E.g. if sun on day 4, what's the most likely sequence?
  - Intuitively: probably sun all four days
- Slow answer: enumerate and score

$$\max_{x_1 \ldots x_{t-1}} P(x_1, \ldots x_{t-1}, sun) =$$

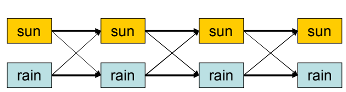
$$P(X_1 = sun)P(X_2 = sun|X_1 = sun)P(X_3 = sun|X_2 = sun)P(X_4 = sun|X_3 = sun)$$

$$P(X_1 = sun)P(X_2 = rain|X_1 = sun)P(X_3 = sun|X_2 = rain)P(X_4 = sun|X_3 = sun)$$

$$\vdots$$

Mini-Viterbi Algorithm

- Better answer: cached incremental updates

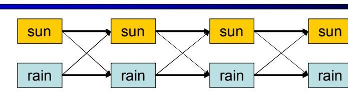


- Define: $m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x)$

$$a_t[x] = \arg\max_{x_{1:t-1}} P(x_{1:t-1}, x)$$

- Read best sequence off of m and a vectors

Mini-Viterbi



$$m_t[x] = \max_{x_{1:t-1}} P(x_{1:t-1}, x)$$

$$= \max_{x_{1:t-1}} P(x_{1:t-1}) P(x|x_{t-1})$$

$$= \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1})$$

$$= \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x]$$

$$m_1[x] = P(x_1)$$

Viterbi Algorithm

- Question: what is the most likely state sequence given the observations $e_{1:t}$?
  - Slow answer: enumerate all possibilities
  - Better answer: incremental updates

$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T}) = \arg\max_{x_{1:T}} P(x_{1:T}, e_{1:T})$$

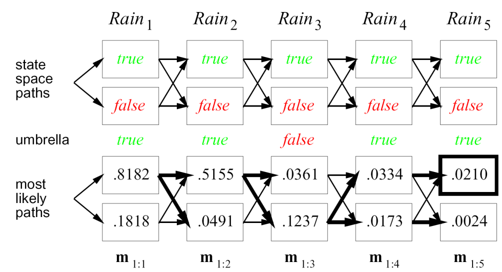
$$m_t[x_t] = \max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})$$

$$= \max_{x_{1:t-1}} P(x_{1:t-1}, e_{1:t-1}) P(x_t|x_{t-1}) P(e_t|x_t)$$

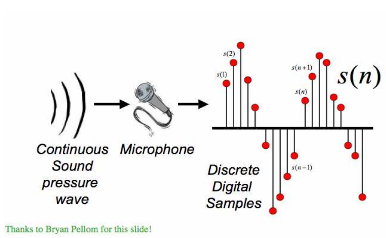
$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) \max_{x_{1:t-2}} P(x_{1:t-1}, e_{1:t-1})$$

$$= P(e_t|x_t) \max_{x_{t-1}} P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]$$

Example

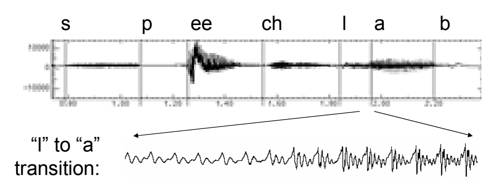


Digitizing Speech



Speech in an Hour

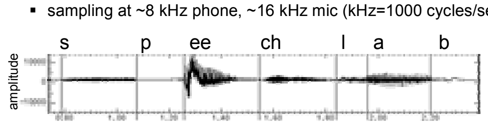
- Speech input is an acoustic wave form



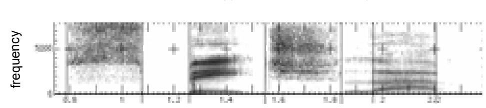
Graphs from Simon Arnfield's web tutorial on speech, Sheffield: http://www.psyc.leeds.ac.uk/research/cogn/speech/tutorial/

Spectral Analysis

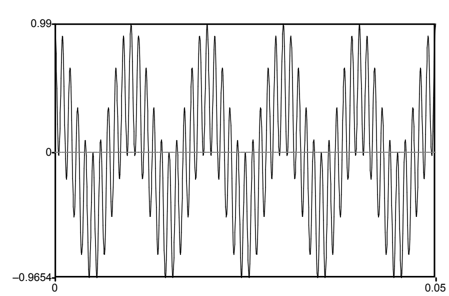
- Frequency gives pitch; amplitude gives volume
  - sampling at ~8 kHz phone, ~16 kHz mic (kHz=1000 cycles/sec)



- Fourier transform of wave displayed as a spectrogram
  - darkness indicates energy at each frequency

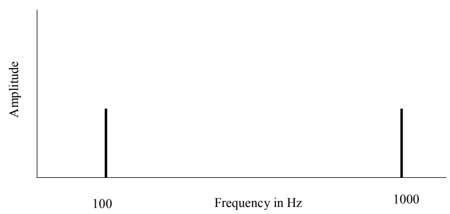


Adding 100 Hz + 1000 Hz Waves

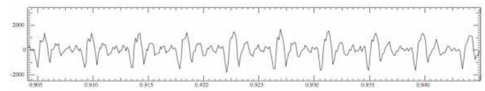


Spectrum

Frequency components (100 and 1000 Hz) on x-axis



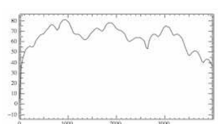
Part of [ae] from "lab"



- Note complex wave repeating nine times in figure
- Plus smaller waves which repeats 4 times for every large pattern
- Large wave has frequency of 250 Hz (9 times in .036 seconds)
- Small wave roughly 4 times this, or roughly 1000 Hz
- Two little tiny waves on top of peak of 1000 Hz waves

Back to Spectra

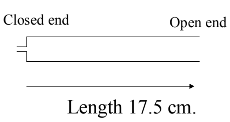
- Spectrum represents these freq components
- Computed by Fourier transform, algorithm which separates out each frequency component of wave.



- x-axis shows frequency, y-axis shows magnitude (in decibels, a log measure of amplitude)
- Peaks at 930 Hz, 1860 Hz, and 3020 Hz.

Resonances of the vocal tract

- The human vocal tract as an open tube



- Air in a tube of a given length will tend to vibrate at resonance frequency of tube.
- Constraint: Pressure differential should be maximal at (closed) glottal end and minimal at (open) lip end.

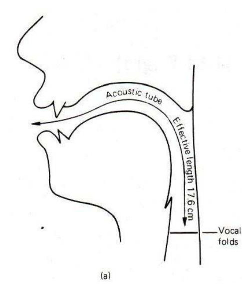
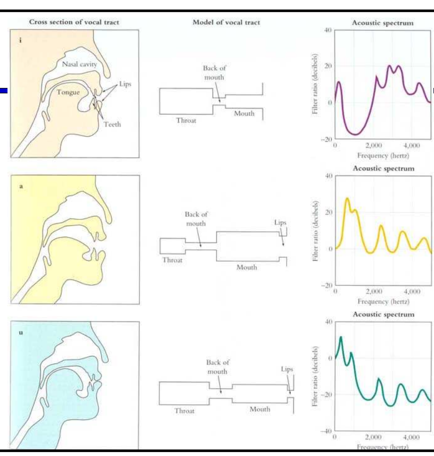


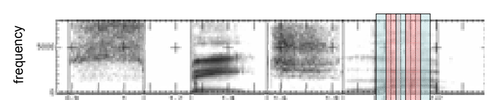
Figure from W. Barry Speech Science slides

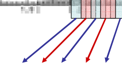


From Mark Liberman's website

Acoustic Feature Sequence

- Time slices are translated into acoustic feature vectors (~39 real numbers per slice)



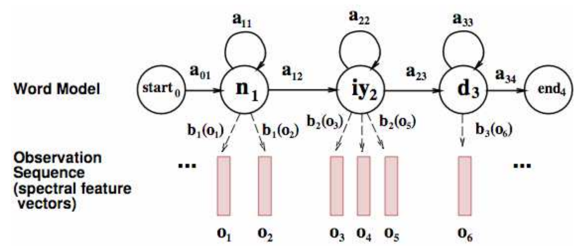


- These are the observations, now we need the hidden states X

State Space

- P(E|X) encodes which acoustic vectors are appropriate for each phoneme (each kind of sound)
- P(X|X') encodes how sounds can be strung together
- We will have one state for each sound in each word
- From some state x, can only:
  - Stay in the same state (e.g. speaking slowly)
  - Move to the next position in the word
  - At the end of the word, move to the start of the next word
- We build a little state graph for each word and chain them together to form our state space X

HMMs for Speech



Markov Process with Bigrams

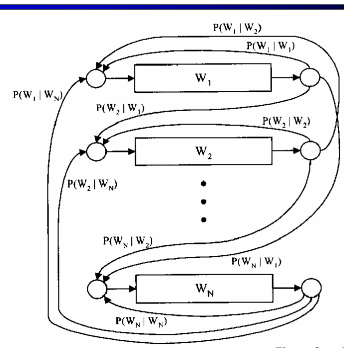


Figure from Huang et al page 618

Decoding

- While there are some practical issues, finding the words given the acoustics is an HMM inference problem
- We want to know which state sequence $x_{1:T}$ is most likely given the evidence $e_{1:T}$:

$$x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|e_{1:T})$$
$$= \arg\max_{x_{1:T}} P(x_{1:T}, e_{1:T})$$

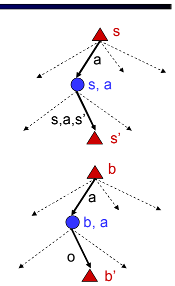
- From the sequence x, we can simply read off the words

POMDPs

- Up until now:
  - MDPs: decision making when the world is fully observable (even if the actions are non-deterministic
  - Probabilistic reasoning: computing beliefs in a static world
- What about acting under uncertainty?
  - In general, the formalization of the problem is the partially observable Markov decision process (POMDP)
  - A simple case: value of information

POMDPs

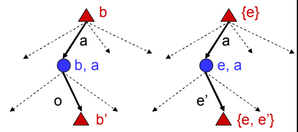
- MDPs have:
  - States S
  - Actions A
  - Transition fn P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s')
- POMDPs add:
  - Observations O
  - Observation function P(o|s,a) (or O(s,a,o))
- POMDPs are MDPs over belief states b (distributions over S)



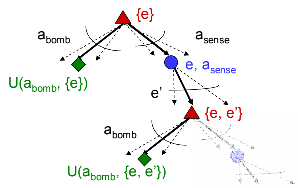
$$B'(s') \propto \sum_{s'} P(o|s,a,s')P(s'|s,a)B(s)$$

Example: Battleship

- In (static) battleship:
  - Belief state determined by evidence to date {e}
  - Tree really over evidence sets
  - Probabilistic reasoning needed to predict new evidence given past evidence



- Solving POMDPs
  - One way: use truncated expectimax to compute approximate value of actions
  - What if you only considered bombing or one sense followed by one bomb?
  - You get the VPI agent from project 4!



More Generally

- General solutions map belief functions to actions
  - Can divide regions of belief space (set of belief functions) into policy regions (gets complex quickly)
  - Can build approximate policies using discretization methods
  - Can factor belief functions in various ways
- Overall, POMDPs are very (actually PSACE-) hard
- We'll talk more about POMDPs at the end of the course!