Speech Recognition

Chapter 15.1 – 15.3, 23.5

“Markov models and hidden Markov models: A brief tutorial,” E. Fosler-Lussier, 1998

Introduction

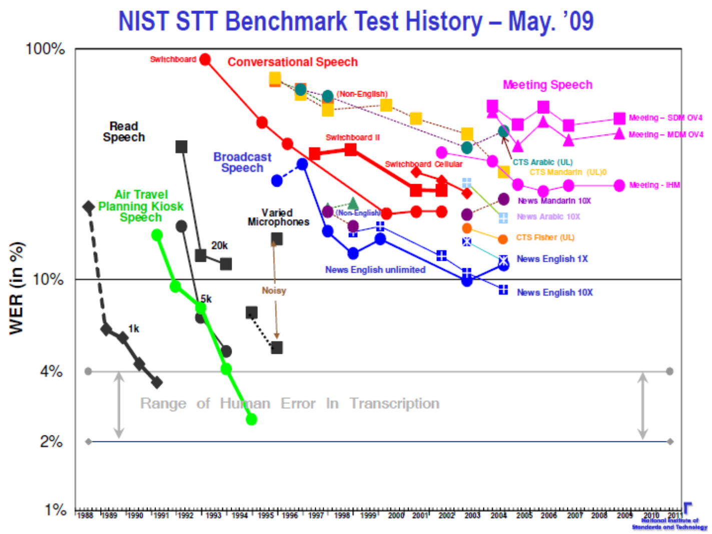
- Speech is a dominant form of communication between humans and is becoming one for humans and machines
- Speech recognition: mapping an acoustic signal into a string of *words*
- Speech understanding: mapping what is said to its *meaning*

Applications

- Medical transcription
- Vehicle control (e.g., fighter aircraft, helicopters)
- Game control
- Intelligent personal assistants (e.g., Siri)
- Smartphone apps
- HCI
- Automatic translation
- Telephony for the hearing impaired
- Air traffic control

Commercial Software

- Nuance Dragon NaturallySpeaking
- Microsoft Windows Speech Recognition
- CMU’s Sphinx-4 (free)
- and many more



Sphinx-4 Performance

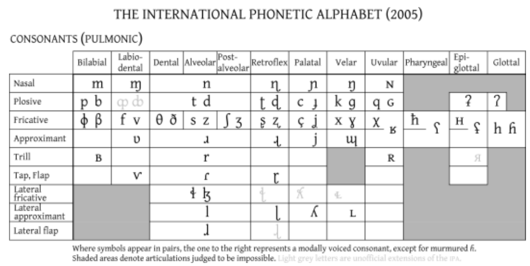
Test	WER (%)	RT	Vocabulary Size	Language Model
TI46	0.168	.02	11	isolated digits recognition
TIDIGITS	0.549	0.05	11	continuous digits
AN4	1.192	0.20	79	trigram
RM1	2.88	0.41	1,000	trigram
WSJ5K	6.97	0.96	5,000	trigram
HUB4	18.756	3.95	60,000	trigram

WER - Word error rate (%) (lower is better)
RT - Real Time - Ratio of processing time to audio time - (lower is better)

Introduction

- Human languages are limited to a set of about 40 to 50 distinct sounds called **phones**, e.g.,
  - [ey] b<u>e</u>t
  - [ah] b<u>u</u>t
  - [oy] b<u>oy</u>
  - [em] bott<u>om</u>
  - [en] butt<u>on</u>
- **Phonemes** are equivalence classes of phones that can't be distinguished from each other in a given language
- These phones are characterized in terms of acoustic features, e.g., frequency and amplitude, that can be extracted from the sound waves

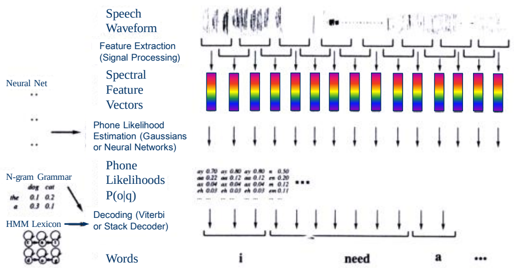
International Phonetic Alphabet



- http://www.yorku.ca/earmstro/ipa/consonants.html
- http://www.youtube.com/watch?v=Nz44WiTVJww&feature=fvw

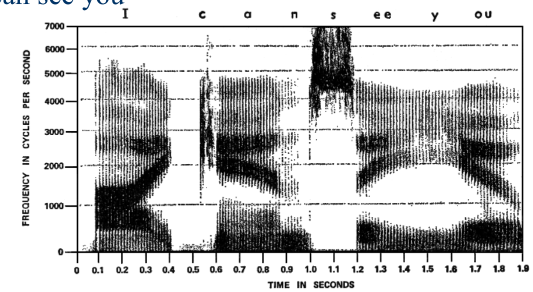
Speech Recognition Architecture

Goal: Large vocabulary, continuous speech (words not separated), speaker-independent

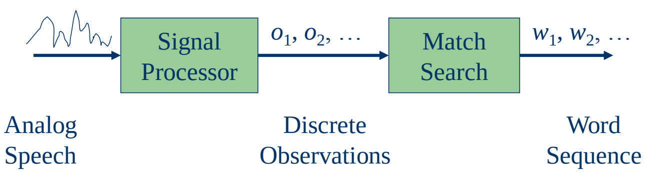


Spectrograph

"I can see you"



Speech Recognition Task



Introduction

- Why isn't this easy?
  - just develop a dictionary of pronunciation
    e.g., coat = [k] + [ow] + [t] = [kowt]
  - but "recognize speech" ≈ "wreck a nice beach"
- Problems:
  - Homophones: different fragments sound the same
    - e.g., "rec" and "wreck"
  - Segmentation: determining breaks between words
    - e.g., "nize speech" and "nice beach"
  - Signal processing problems

Hearing Speech with your Eyes

- McGurk Effect: An audio-visual illusion – what you **see** affects what you hear
- Is he saying “BA BA” or “DA DA”?

Hearing Speech with your Eyes

- McGurk Effect: An audio-visual illusion – what you **see** affects what you hear
- Is he saying “BA BA” or “DA DA”?
- Most adults (98%) think they are hearing "DA" – a so called "fused response" – where the "D" is a result of an ***audio-visual illusion***. In reality you are *hearing* the sound “BA,” while you are *seeing* the lip movements “GA.”

Signal Processing

- Sound is an analog energy source resulting from pressure waves striking an eardrum or microphone
- An analog-to-digital converter is used to capture the speech sounds
  - Sampling: the number of times per second that the sound level is measured
  - Quantization: the number of bits of precision for the sound level measurements
    - Telephone: 3 KHz (3000 times per second)
    - Speech recognizer: 8 KHz with 8 bits/sample so that 1 minute takes about 500K bytes

Signal Processing

- Wave encoding
  - group into ~10 msec frames (larger blocks) that are analyzed individually
  - frames overlap to ensure important acoustical events at frame boundaries aren't lost
  - frames are analyzed in terms of features
    - amount of energy at various frequencies
    - total energy in a frame
    - differences from prior frame
  - vector quantization encodes signal by mapping frames into regions in *n*-dimensional feature space

Input Sequence

- Vector Quantization encodes each region as one of, say, 256 possible observation values (aka labels): C1, C2, …, C256
- Uses **k-Means Clustering** for unsupervised learning of k = 256 "bins"
- Input is a sequence such as
  C82, C44, C63, C44, C25, …
  = o_1, o_2, o_3, o_4, o_5, …

Signal Processing

- *Goal is speaker independence so that representation of sound is independent of a speaker's specific pitch, volume, speed, and other aspects such as dialect*
- Speaker identification does the opposite, i.e., the specific details are needed to decide *who is speaking*
- A significant problem is dealing with background noise that is often other speakers

Speech Recognition Model

- Bayes's Rule is used break up the problem into manageable parts:

$$P(Words \mid Signal) = \frac{P(Signal \mid Words)\, P(Words)}{P(Signal)}$$

$P(Words)$: **Language model**
- likelihood of words being heard
- e.g., "recognize speech" more likely than "wreck a nice beach"

$P(Signal \mid Words)$: **Acoustic model**
- likelihood of a signal given word sequence
- accounts for differences in pronunciation of words
- e.g., given "nice," likelihood that it is pronounced [nuys]

- *Signal* = **observation sequence** $O = o_1, o_2, o_3, \ldots, o_t$
- *Words* = **sequence of words** $W = w_1, w_2, w_3, \ldots, w_n$
- Best match metric: **probability** $\hat{W} = \arg\max_{W \in L} P(W \mid O)$
- Bayes's rule:

$$\hat{W} = \arg\max_{W \in L} \frac{P(O \mid W) P(W)}{P(O)}$$

$$\propto \arg\max_{W \in L} P(O \mid W) P(W)$$

observation likelihood (acoustic model) — prior probability (language model)

Language Model (LM)

- $P(Words)$ *is the joint probability that a sequence of words =* $w_1\ w_2 \dots w_n$ *is likely for a specified natural language*
- This joint probability can be expressed using the chain rule (order reversed):

$$P(w_1\ w_2 \dots w_n) = P(w_1)\ P(w_2 | w_1)\ P(w_3 | w_1\ w_2) \dots P(w_n | w_1 \dots w_{n-1})$$

- Collecting all these probabilities is too complex; it requires statistics for m^{n-1} starting sequences for a sequence of n words in a language of m words
  - Simplification is necessary!

Language Model (LM)

- **First-order Markov Assumption**
  - Probability of a word depends only on the previous word:

$$P(w_i | w_1 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

- The LM simplifies to

$$P(w_1\ w_2 \dots w_n) = P(w_1)\ P(w_2 | w_1)\ P(w_3 | w_2) \dots P(w_n | w_{n-1})$$

  - called the ***bigram model***
  - relates *consecutive pairs of words*

Language Model (LM)

- More context could be used, such as the *two* words before, called the ***trigram** model:*

$$P(w_i | w_1 \dots w_{i-1}) \approx P(w_i | w_{i-1}\ w_{i-2})$$

- A weighted sum of *unigram, bigram, trigram models* could also be used in combination:

$$P(w_1\ w_2 \dots w_n) = \prod (c_1 P(w_i) + c_2 P(w_i | w_{i-1}) + c_3 P(w_i | w_{i-1}\ w_{i-2}))$$

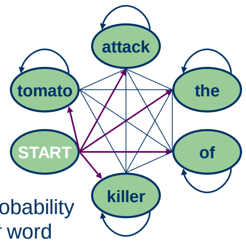
- Bigram and trigram models account for
  - *local* context-sensitive effects
    - e.g., "bag of tricks" vs. "bottle of tricks"
  - some *local* grammar
    - e.g., "we was" vs. "we were"

Language Model (LM)

- Probabilities are obtained by computing statistics of the frequency of all possible pairs of words in a large training set of word strings:
  - if "the" appears in training data 10,000 times and it's followed by "clock" 11 times then $P(clock \mid the) = \mathbf{11/10000 = .0011}$
- These probabilities are stored in
  - a probability table
  - a probabilistic finite state machine

Language Model (LM)

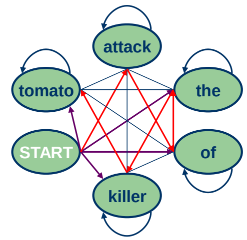
- **Probabilistic finite state machine: a (almost) fully connected directed graph:**



- **nodes:** all possible words and a START state
- **arcs:** labeled with a probability
  - from START to a word is the **prior** probability of the destination word being the *first* word
  - from one word to another is the conditional probability of the destination word following a given source word

Language Model (LM)

- **Probabilistic finite state machine: a (almost) fully connected directed graph:**



  - **joint probability** is estimated for the *bigram* model by starting at START and multiplying the probabilities of the arcs that are traversed for a given sentence:
  - ***P* (“attack of the killer tomato”) =**
    ***P* (attack) *P* (of | attack) *P* (the | of) *P* (killer | the) *P* (tomato | killer)**

Acoustic Model (AM)

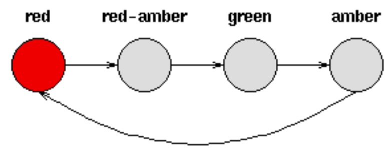
- $P(Signal \mid Words\,)$ is the conditional probability that a signal is likely given a sequence of words for a particular natural language

- This is divided into two probabilities:
  - $P\,(\boldsymbol{phones} \mid \boldsymbol{word}\,)$: probability of a sequence of phones given *word*
  - $P\,(\boldsymbol{signal} \mid \boldsymbol{phone}\,)$: probability of a sequence of vector quantization values from the acoustic signal given *phone*

Finding Patterns

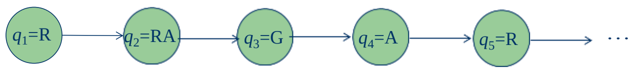
- Speech is an example of a more general problem of *finding patterns over time* (or any discrete sequence)
- *Deterministic* patterns have a *fixed sequence of states*, where the next state is dependent solely on the previous state

European Stop light



Modeling a Sequence of States

- Bayesian Network structure for a sequence of states from {Red (R), Red-Amber (RA), Green (G), Amber (A)}. Each q_i is a random variable indicating the state of the stoplight at time *i*.



Markov Property

- The 1st order Markov assumption called the "Markov property:"
- State q_{t+1} is **conditionally independent** of $\{q_{t-1}, q_{t-2}, \ldots q_1\}$ given q_t. In other words:

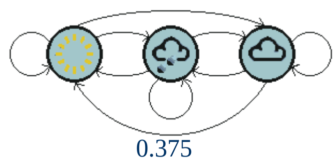
$$P(q_{t+1} = s_j \mid q_t = s_i) = P(q_{t+1} = s_j \mid q_t = s_i, q_{t-1} = s_k, \ldots, q_1 = s_l)$$

Non-Deterministic Patterns

- Assume a discrete set of states, but can't model the sequence deterministically
- Example: Predicting the weather
  - **States**: Sunny, Cloudy, Rainy
  - **Arcs**: Probability, called the **state transition probability**, of moving from one state to another
- ***N*th-order Markov assumption**: **Today's weather can be predicted solely given knowledge of the last *N* days' weather**
- **1st-order Markov assumption**: **Today's weather can be predicted solely given knowledge of yesterday's weather**

1st-Order Markov Model

- Markov process is a process that moves from state to state ***probabilistically*** based on a state transition matrix, **A**, associated with a graph of the possible states
- Sum of values in each *row* is 1
- 1st-order Markov model for weather prediction:

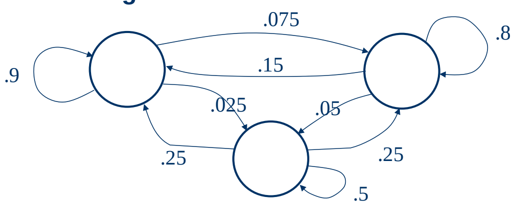


weather today

weather yesterday	Sun	Cloud	Rain
Sun	0.5	0.25	0.25
Cloud	0.375	0.125	0.375
Rain	0.125	0.625	0.375

Matrix Properties

- Sum of values in row = 1 because these are the probabilities of all **outgoing arcs** from a state
- Sum of values in a *column* does NOT necessarily equal 1 because this is the sum of probabilities on all **incoming arcs** to a state



1^{st}-Order Markov Model

- To initialize the process, also need the prior probabilities of the initial state at time t=0, called π. For example, if we know the first day was sunny, then π is a vector =

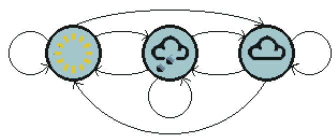
$$\begin{array}{ccc} \text{Sun} & \text{Cloud} & \text{Rain} \\ \end{array}$$
$$\begin{pmatrix} 1.0 & 0.0 & 0.0 \end{pmatrix}$$

- For simplicity, we will often assume a single, given state is the start state

1^{st}-Order Markov Model

- **Markov Model M = (A, π)** consists of
  - Discrete set of states, $s_1, s_2, \ldots, s_N$
  - π vector, where $\pi_i = P(q_1=s_i)$
  - State transition matrix, $\mathbf{A} = \{a_{ij}\}$ where $a_{ij} = P(q_{t+1} = s_j \mid q_t = s_i)$
- The state transition matrix is fixed for all times and describes probabilities associated with a (completely-connected) graph of the states

Example: Using a Markov Model for Weather Prediction



$\mathbf{A}$ =

weather today

weather yesterday	Sun	Cloud	Rain
Sun	0.5	0.25	0.25
Cloud	0.375	0.125	0.375
Rain	0.125	0.625	0.375

$$\Pi = \begin{array}{ccc} \text{Sun} & \text{Cloud} & \text{Rain} \end{array}$$
$$\begin{pmatrix} 1.0 & 0.0 & 0.0 \end{pmatrix}$$

Given that today is sunny, what is the probability of the next two days being sunny and rainy, respectively?

Weather Prediction Example (cont.)

- $P(q_2=\text{Sun}, q_3=\text{Rain} \mid q_1=\text{Sun}) = ?$

- $P(q_3=\text{Rain} \mid q_1=\text{Cloudy}) = ?$

Weather Prediction Example (cont.)

conditionalized chain rule

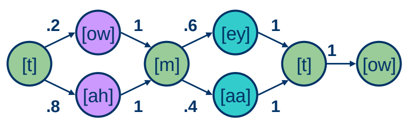
- $P(q_2=\text{Sun}, q_3=\text{Rain} \mid q_1=\text{Sun})$
  $= P(q_3=\text{Rain} \mid q_2=\text{Sun}, q_1=\text{Sun})\, P(q_2=\text{Sun} \mid q_1=\text{Sun})$
  $= P(q_3=\text{Rain} \mid q_2=\text{Sun})\, P(q_2=\text{Sun} \mid q_1=\text{Sun})$
  $= (.25)(.5)$
  $= 0.125$

1st order Markov assumption (conditional independence)

- $P(q_3=\text{Rain} \mid q_1=\text{Cloudy})$
  $= P(q_2=\text{Sun}, q_3=\text{Rain} \mid q_1=\text{Cloudy})$
  $+ P(q_2=\text{Cloudy}, q_3=\text{Rain} \mid q_1=\text{Cloudy})$
  $+ P(q_2=\text{Rain}, q_3=\text{Rain} \mid q_1=\text{Cloudy})$

Acoustic Model (AM)

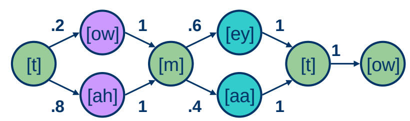
- ***P*(*phones* |*word*)** can be specified as a **Markov model**, which is a way of describing a process that goes through a sequence of states, e.g., “tomato”



- **Nodes:** correspond to the production of a phone
  - sound slurring (coarticulation), e,g., due to quickly pronouncing a word
  - variation in pronunciation of words, e.g., due to dialects
- **Arcs:** probability of transitioning from current state to another

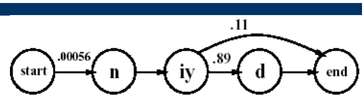
Acoustic Model

- ***P*(*phones* |*word*)** can be specified as a **Markov model**, which is a way of describing a process that goes through a sequence of states, e.g., “tomato:”



- ***P*(*phones* | *word*) is a *path* through the diagram, i.e.,**
  - ***P* ([*towmeytow*] | *tomato*)** = 0.2 * 1 * 0.6 * 1 * 1 = 0.12
  - ***P* ([*towmaatow*] | *tomato*)** = 0.2 * 1 * 0.4 * 1 * 1 = 0.08
  - ***P* ([*tahmeytow*] | *tomato*)** = 0.8 * 1 * 0.6 * 1 * 1 = 0.48
  - ***P* ([*tahmaatow*] | *tomato*)** = 0.8 * 1 * 0.4 * 1 * 1 = 0.32

Acoustic Model for "Need"



Word model for "need"

- **Look at probabilities of various phones as we listen:**
  - In corpus "need" always starts with "n" sound
  - What are the possibilities for the next sound? With probability 1, we know that next sound will be "iy"
  - What are possibilities for next sound? 11% of the time, "d" sound will be omitted
  - Probability of transitioning from "iy" to the "d" sound is .89
- **Circles represent two things—states and observations**
- **In real world, state is hidden: For sound [iy], we don't know whether we are at second phone of the word "knee" or the second phone of the word "need"**

Problem

- We don't know the sequence of phones, we only have the observation sequence $o_1, o_2, o_3, \ldots$
- How do we relate the given input sequence to phone sequences?

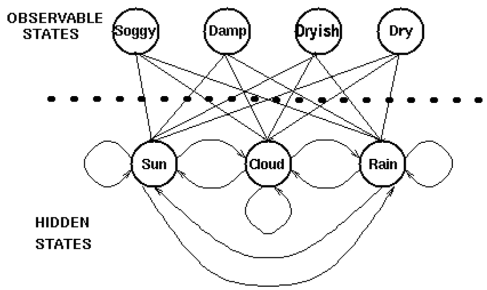
Hidden Markov Models (HMMs)

- Sometimes the states we want to predict are *not directly observable*; only observations available are *indirect* evidence
- Example: A CS major does not have direct access to the weather, but can only observe the state of a piece of corn (dry, dryish, damp, soggy)
- Example: In speech recognition we can observe features of the changing sound, i.e., $o_1, o_2, \ldots$, but there is no direct evidence of the *words* being spoken

HMMs

- Hidden States: The states of real interest, e.g., the true weather or the sequence of words spoken; represented as a 1st-order Markov model
- Observable Values: A discrete set of observable values; the number of observable values is *not*, in general, equal to the number of hidden states. The observable values are related *somehow* to the hidden states (i.e., *not* 1-to-1 correspondence)

Hidden Markov Model



Arcs and Probabilities in HMMs

- Arcs connecting hidden states and observable values represent the probability of generating an observed value given that the Markov process is in a hidden state
- Observation Likelihood matrix, **B**, (aka output probability distribution) stores probabilities associated with arcs from hidden states to observable values, i.e., *P*(Obs **|** Hidden)

Encodes semantic variations, sensor noise, etc.

corn

weather

	Dry	Dryish	Damp	Soggy
Sun	0.60	0.20	0.15	0.05
Cloud	0.25	0.25	0.25	0.25
Rain	0.05	0.10	0.35	0.50

HMM Summary

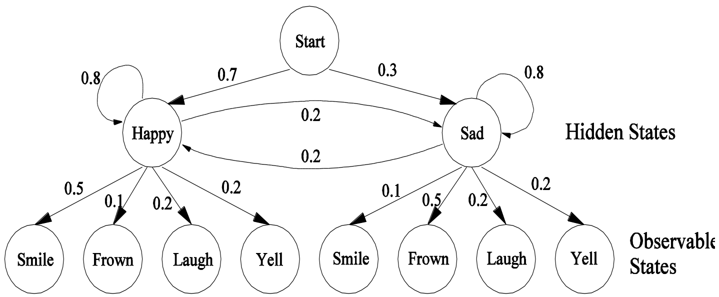
- An HMM contains 2 types of information:
  - Hidden states: $s_1, s_2, s_3, \ldots$
  - Observable values
    - In speech recognition, the vector quantization values in the input sequence $\mathbf{O} = o_1, o_2, o_3, \ldots$
- An HMM, $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$, contains 3 sets of probabilities
  - $\boldsymbol{\pi}$ vector, $\boldsymbol{\pi} = (\pi_i)$
  - State transition matrix, $\mathbf{A} = (a_{ij})$ where $a_{ij} = P(q_t = s_i \mid q_{t-1} = s_j)$
  - Observation likelihood, $\mathbf{B} = b_j(o_k) = P(y_t = o_k \mid q_t = s_j)$

HMM Summary

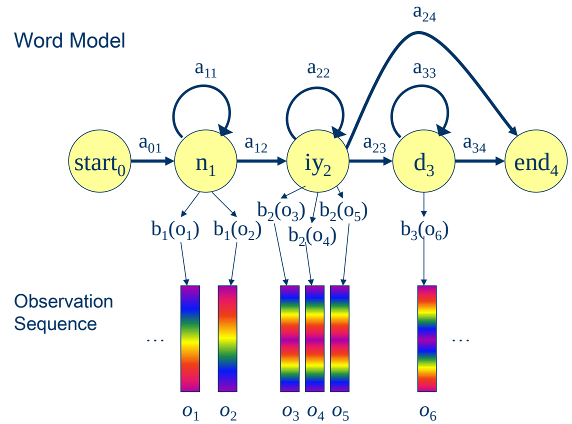
- Markov property says observation o_i is ***conditionally independent*** of hidden states $q_{i-1}, q_{i-2}, \ldots, q_0$ given q_i
- In other words:

$$P(O_t = X \mid q_t = s_i) =$$
$$P(O_t = X \mid q_t = s_i, \text{any earlier history})$$

Example: An HMM



Example: An HMM Word Model



Acoustic Model (AM)

- *P*(***Signal*** | ***Phone***) can be specified as an HMM, e.g., phone model for [m]:

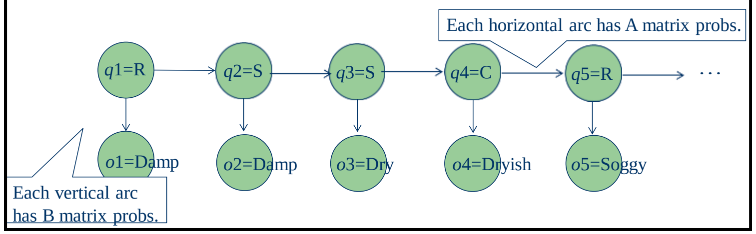
  - **nodes:** probability distribution over a set of vector quantization values
  - **arcs:** probability of transitioning from current hidden state to next hidden state

Generating HMM Observations

- **Choose an initial hidden state, $q_1 = s_i$, based on π**
- **For t = 1 to T do**
  1. Choose output/observation value $z_t = o_k$ according to the symbol probability distribution in hidden state s_i, $b_i(k)$
  2. Transition to a new hidden state $q_{t+1} = s_j$ according to the state transition probability distribution for state s_i, a_{ij}
- So, **transition to new state and *then* output value at the new state**

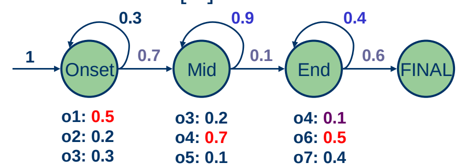
Modeling a Sequence of States

- **Bayesian Network** structure for a sequence of hidden states from {R, S, C}. Each q_i is a "latent" random variable indicating the state of the weather on day *i*. Each o_i is the observed state of the corn on day *i*.



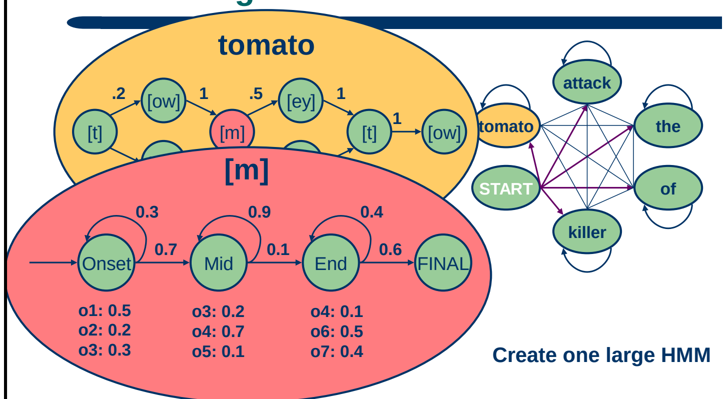
Acoustic Model (AM)

- $P(\textit{Signal} \mid \textit{Phone})$ can be specified as an HMM, e.g., phone model for [m]:



- $P(\textit{Signal} \mid \textit{Phone})$ **is a path through the states, i.e.,**
  - $P([o1,o4,o6] \mid [m]) = (1)(.5)(.7)(.7)(.1)(.5)(.6) = 0.00735$
  - $P([o1,o4,o4,o6] \mid [m]) = (1)(.5)(.7)(.7)(.9)(.7)(.1)(.5)(.6) + (1)(.5)(.7)(.7)(.1)(.1)(.4)(.5)(.6) = 0.0049245$
- Model allows for variation in speed of pronunciation

Combining Models



Create one large HMM

3 Common Tasks using HMMs

- **Evaluation problem**
- **Decoding problem**
- **Learning problem**

Evaluation Problem: $P(O \mid \lambda)$

- Find the probability of an observed sequence given an HMM. For example, given several HMMs such as a "summer HMM" and a "winter HMM" and a sequence of corn observations, *which* HMM most likely generated the given sequence?
- In speech recognition, use one HMM for each word, and an observation sequence from a spoken word. Compute $P(O|\lambda)$. Recognize the word by identifying the most probable HMM.
- Use **Forward algorithm**

Decoding Problem

- Find the most probable sequence (i.e., path) of hidden states given an observation sequence
- Compute $Q^* = \text{argmax}_Q \, P(Q \mid O)$
- Use **Viterbi algorithm**

Learning Problem

- Generate an HMM given a sequence of observations and a set of known hidden states
- Learn the most probable HMM
- Compute $\lambda^* = \text{argmax}_\lambda \, P(O \mid \lambda)$
- Use **Forward-Backward algorithm** or **Expectation-Maximization (EM) algorithm**

Evaluation Problem

- **$P(\mathbf{O} \mid \lambda)$ = ? where $\mathbf{O} = o_1, o_2, \ldots, o_T$ is the observation sequence and λ is an HMM model**
- Let **$\mathbf{Q} = q_1, q_2, \ldots, q_T$ be a specific hidden state sequence**

$$P(\mathbf{O} \mid \lambda) = \sum_{Q} P(\mathbf{O} \mid \mathbf{Q}, \lambda) \, P(\mathbf{Q} \mid \lambda) \quad \textbf{by Conditioning rule}$$

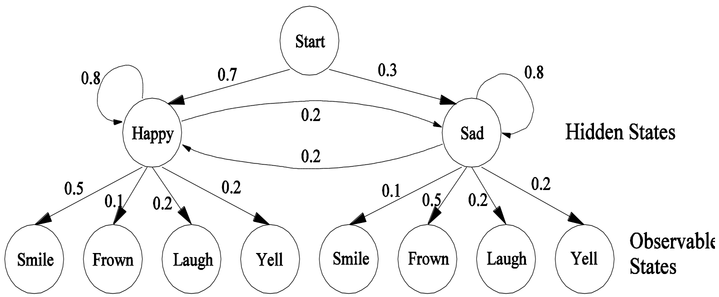
$$P(\mathbf{O} \mid \mathbf{Q}, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda) = b_{q_1}(o_1)\, b_{q_2}(o_2) \ldots b_{q_T}(o_T)$$

$$P(\mathbf{Q} \mid \lambda) = \pi_{q_1} \, a_{q1q_2} \ldots a_{q_{T-1}q_T}$$

$$\text{So, } P(\mathbf{O}|\lambda) = \sum_{Q} \pi_{q_1} b_{q_1}(o_1)\, a_{q_0q_1} b_{q_2}(o_2)\, a_{q_1q_2} b_{q_3}(o_3) \cdots a_{q_{T-1}q_T} b_{q_T}(o_T)$$

- **(π, A, B) known for given HMM λ**

Example: An HMM



$P(q1$ =Happy, q_2=Sad | HMM) = ?

$P(q1$ = H, q_2=S | HMM) = ?

$$P(q_1 = H, q_2 = S) = P(q_2 = S \mid q_1 = H) P(q_1 = H)$$

(by Chain rule)

$$= (.2)(.7) = 0.14$$

$P(q_2$=Happy | HMM) = ?

$P(q_2$=Happy | HMM) = ?

$$P(q_2 = H) = P(q_2 = H \mid q_1 = H)P(q_1 = H) + P(q_2 = H \mid q_1 = S)P(q_1 = S)$$

(by Conditioning rule)

$$= (.8)(.7) + (.2)(.3) = 0.62$$

$P(q_3$=Happy | HMM) = ?

$P(q_3$=Happy | HMM) = ?

$$P(q_3 = H) = P(q_3 = H \mid q_1 = H, q_2 = H)P(q_1 = H, q_2 = H) + P(q_3 = H \mid q_1 = H, q_2 = S)P(q_1 = H, q_2 = S) + \text{ 2 more cases}$$

(by Conditioning rule)

$$= P(q_3 = H \mid q_2 = H)P(q_1 = H, q_2 = H) + \ldots$$

$$= P(q_3 = H \mid q_2 = H)P(q_2 = H \mid q_1 = H)P(q_1 = H) + \ldots$$

(by Chain rule)

$$= (.7)(.8)(.8) + (.7)(.2)(.2) + (.3)(.2)(.8) + (.3)(.8)(.2)$$

$P(o_1$=Laugh, o_2=Frown | q_1=H, q_2=S) = ?

$P(o_1$=Laugh, o_2=Frown | q_1=H, q_2=S) = ?

$$= P(o_1 = L \mid o_2 = F, q_1 = H, q_2 = S) P(o_2 = F \mid q_1 = H, q_2 = S) \text{ by chain rule}$$

$$= P(o_1 = L \mid q_1 = H) P(o_2 = F \mid q_2 = S) \text{ by Markov property}$$

$$= (.2)(.5) = 0.1$$

$P(o_1$=Laugh, o_2=Frown | HMM) = ?

(Evaluation Problem)

$P(o_1$=Laugh, o_2=Frown | HMM) = ?

$$\begin{aligned}
P(o_1 = L, o_2 = F) = {} & P(o_1 = L, o_2 = F \mid q_1 = H, q_2 = H) P(q_1 = H, q_2 = H) \\
& + P(o_1 = L, o_2 = F \mid q_1 = H, q_2 = S) P(q_1 = H, q_2 = S) \\
& + P(o_1 = L, o_2 = F \mid q_1 = S, q_2 = H) P(q_1 = S, q_2 = H) \\
& + P(o_1 = L, o_2 = F \mid q_1 = S, q_2 = S) P(q_1 = S, q_2 = S) \\
& \qquad \text{by Conditioning rule} \\
= {} & P(o_1 = L \mid o_2 = F, q_1 = H, q_2 = H) P(o_2 = F \mid q_1 = H, q_2 = H) P(q_1 = H, q_2 = H) + \ldots \\
& \qquad \text{by Conditionalized Chain rule} \\
= {} & P(o_1 = L \mid q_1 = H) P(o_2 = F \mid q_2 = H) P(q_2 = H \mid q_1 = H) P(q_1 = H) + \ldots \\
& \qquad \text{by Markov property} \\
= {} & P(q_1 = H) P(o_1 = L \mid q_1 = H) P(q_2 = H \mid q_1 = H) P(o_2 = F \mid q_2 = H) + \ldots
\end{aligned}$$

$$= (.7)(.2)(.8)(.1) + \ldots$$

$P(o_1$=L, o_2=F, q_1=H, q_2=S) = ?

$P(o_1=L, o_2=F, q_1=H, q_2=S)$ = ?

$$= P(o_1 = L \mid q_1 = H) P(o_2 = F \mid q_2 = S) P(q_2 = H \mid q_1 = S) P(q_1 = H)$$
$$= (.2)(.5)(.2)(.7)$$

by chain rule plus Markov property

Computation of P(O | λ)

- Since there are N^T sequences (N hidden states and T observations), $O(T\,N^T)$ calculations required!
- For N=5, T=100 $\Rightarrow$ 10^{72} computations!!

$P(q_1=S \mid o_1=F)$ = ?

(State Estimation Problem)

Needed as part of solving the "Decoding Problem:"

Most Probable Path Problem: Find q_1, q_2 such that $P(q_1=X, q_2=Y \mid o_1=L, o_2=F)$ is a *maximum* over all possible values of X and Y, and give the values of X and Y

$P(q_1=S \mid o_1=F)$ = ?

$$P(q_1 = S \mid o_1 = F) = \frac{P(o_1 = F \mid q_1 = S) P(q_1 = S)}{P(o_1 = F)} \quad \text{by Bayes's rule}$$
$$= \frac{(.5)(.3)}{P(o_1 = F \mid q_1 = S) P(q_1 = S) + P(o_1 = F \mid q_1 = H) P(q_1 = H)}$$
$$= \frac{(.5)(.3)}{(.5)(.3) + (.1)(.7)}$$

$P(q_3$=H | o_1=F, o_2=L, o_3=Y)=?

(Decoding Problem)

$P(q_3$=H | o_1=F,o_2=L,o_3=Y)=?

$$
\begin{aligned}
P(q_3 = H \mid o_1 = F, o_2 = L, o_3 = Y) = {} & P(q_3 = H, q_2 = H, q_1 = H \mid \ldots) \\
& + P(q_3 = H, q_2 = H, q_1 = S \mid \ldots) \\
& + P(q_3 = H, q_2 = S, q_1 = H \mid \ldots) \\
& + P(q_3 = H, q_2 = S, q_1 = S \mid \ldots)
\end{aligned}
$$

where

$$
P(q_3 = H, q_2 = H, q_1 = H \mid o_1 = F, o_2 = L, o_3 = Y)
$$

$$
= \frac{P(o_1 = F, o_2 = L, o_3 = Y \mid q_1 = H, q_2 = H, q_3 = H) P(q_1 = H, q_2 = H, q_3 = H)}{P(o_1 = F, o_2 = L, o_3 = Y)}
$$

by Bayes's rule

$$
= \frac{P(o_1 = F \mid q_1 = H) P(o_2 = L \mid q_2 = H) P(o_3 = Y \mid q_3 = H) P(q_3 = H \mid q_2 = H) P(q_2 = H \mid q_1 = H) P(q_1 = H)}{P(o_1 = F, o_2 = L, o_3 = Y)}
$$

P(q_2 = H | q_1 = S, o_2=F) = ?

$P(q_2 = H, q_1 = S \mid o_2 = F) / P(q_1 = S \mid o_2 = F)$ by cond. chain rule

$= P(q_2 = H, q_1 = S \mid o_2 = F) / P(q_1 = S)$ by independence of q_1 and o_2

$= P(o_2 = F \mid q_1 = S, q_2 = H) P(q_1 = S, q_2 = H) / P(q_1 = S) P(o_2 = F)$ Bayes rule

$= P(o_2 = F \mid q_2 = H) P(q_1 = S, q_2 = H) / P(q_1 = S) P(o_2 = F)$ Markov assump.

$= P(o_2 = F \mid q_2 = H) P(q_2 = H \mid q_1 = S) P(q_1 = S) / P(q_1 = S) P(o_2 = F)$

$= P(o_2 = F \mid q_2 = H) P(q_2 = H \mid q_1 = S) / P(o_2 = F)$

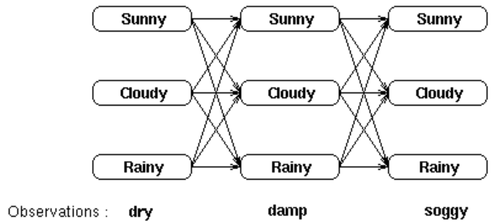
Product rule

$= (.1)(.2) / P(o_2 = F)$

How to Solve HMM Problems Efficiently?

Evaluation using Exhaustive Search

- Given observation sequence (dry, damp, soggy), "unroll" the hidden state sequence as a "trellis" (matrix):



Evaluation using Exhaustive Search

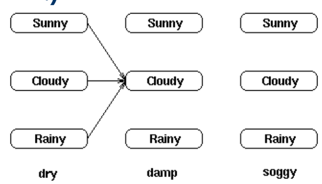
- Each column in the trellis (matrix) shows a possible state of the weather
- Each state in a column is connected to each state in the adjacent columns
- Sum the probabilities of each possible sequence of the hidden states; here, $3^3 = 27$ possible weather sequences
- ***P*(dry,damp,soggy | HMM λ) =**
  - ***P*(dry,damp,soggy | sunny,sunny,sunny) +**
  - ***P*(dry,damp,soggy | sunny,sunny,cloudy) + ⋯ +**
  - ***P*(dry,damp,soggy | rainy,rainy,rainy)**
- Not practical since the number of paths is $O(N^T)$ where N is the number of hidden states and T is number of observations

Forward Algorithm Intuition

- Idea: compute and cache values $\alpha_t(i)$ representing probability of being in state i after seeing first t observations, $o_1, o_2, \ldots, o_t$
- Each cell expresses the probability
  $\alpha_t(i) = P(q_t = i \mid o_1, o_2, \ldots, o_t)$
- $q_t = i$ means "the probability that the t^{th} state in the sequence of hidden states is state i
- Compute α by summing over extensions of all paths leading to current cell
- An extension of a path from a state i at time $t-1$ to state j at t is computed by multiplying together:
  i. previous path probability **from the previous cell $\alpha_{t-1}(i)$**
  ii. transition probability **a_{ij} from previous state i to current state j**
  iii. observation likelihood **b_{jt} that current state j matches observation symbol t**

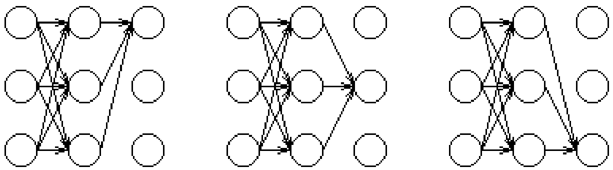
Evaluation using Forward Algorithm

- Compute probability of reaching each intermediate hidden state in the trellis given observation sequence $O = o_1, o_2, \ldots, o_T$ That is, $P(q_t = s_i \mid O)$
- Example: Given O = (dry, damp, soggy), compute $P(O, q_2 = \text{cloudy} \mid \text{HMM } \lambda)$
- $\alpha_2(\text{cloudy})$ =
  $P(O \mid q_2 = \text{cloudy}) \ast$
  $P(\text{all paths to } q_2 = \text{cloudy})$



Forward Algorithm (cont.)

- $P(\mathbf{O}, q_T{=}s_j \mid \lambda)$ = sum of all possible paths through the trellis

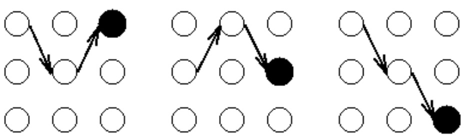


Forward Algorithm (cont.)

- Compute α recursively:
  - $\alpha_1(j) = \pi_j\, b_j(o_1)$ for all states j
  - $\alpha_t(j) = P(\mathbf{O}, q_t{=}\, s_j \mid \lambda)$
  - $= [\sum_{i=0}^{N} \alpha_{t-1}(i) a_{ij}]\, b_j(o_t)$ for $t > 0$
- $P(\mathbf{O} \mid \lambda) = \sum_{j=1}^{N} \alpha_T(s_j)$
- $O(N^2T)$ computation time (i.e., linear in T, the length of the sequence)

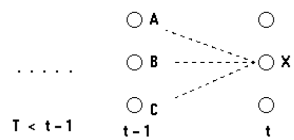
Decoding Problem

- Most probable sequence of hidden states is the state sequence **Q** that **maximizes** $P(\mathbf{O},\mathbf{Q} \mid \lambda)$
- Similar to the Forward Algorithm except uses MAX instead of SUM, thereby computing the probability of the most probable path to each state in the trellis



Decoding using Viterbi Algorithm

- For each state q_i and time t, compute recursively $\delta_t(i)$ = **maximum** probability of all sequences ending at state q_i and time t, and the best path to that state
- Assumption: **Dynamic Programming invariant:**
  If ultimate best path for **O** includes state q_i
  then it includes the best path up to and including q_i



Viterbi Algorithm

- Variant of forward algorithm that considers all words simultaneously and computes most likely *path*
- A type of dynamic programming algorithm
- Input is sequence of observations, and an HMM
- Output is most probable state sequence **Q** = $q_1, q_2, q_3, q_4, \ldots, q_T$ together with its probability
- Works by computing **max** of previous paths instead of sum
- Looks at the whole sequence before deciding on the best final state and then follows back pointers to recover the best path
- **Linear time and linear space algorithm in T**

Viterbi Algorithm

- By the 1st-order Markov assumption:

$$P(X, t) = \max_{i=A,B,C} \{ P(i, t\text{-}1) \times P(X \mid i) \times P(o_t \mid X) \}$$

- Hence, $\delta_t(i) = \max_j \{ \delta_{t-1}(j)\ a_{ji}\ b_j(o_t) \}$
- Record back pointer to best previous state; use this to obtain best path by tracing back from final state
- Linear time and linear space in *t*

Summary of Viterbi Algorithm

- Given an HMM, the Viterbi algorithm finds the most probable sequence of hidden states given a sequence of observed values
- Exploits time invariance of the probabilities to avoid examining every possible path through the trellis
- Looks at the whole sequence before deciding on the best final state and then follows back pointers to recover the best path
- Uses entire context to make its decision and is therefore robust with respect to noise (e.g., a bad observation value in the middle of the sequence)

Summary

- **Speech recognition systems work best if**
  - high SNR (low noise and background sounds)
  - small vocabulary
  - good language model
  - pauses between words
  - trained to a specific speaker
- **Current systems**
  - vocabulary of ~200,000 words for single speaker
  - vocabulary of ~5,000 words for multiple speakers
  - accuracy depends on the task

Error Rates: Machine vs. Human*

Task	Machine	Human
Digits (10)	0.72%	0.009%
Letters (26)	5.0%	1.6%
Transactional speech (1,000)	3.6%	0.1%
Sentences read from *WSJ* (5,000)	12.8 - 7.2%	1.1 - 0.9%
Telephone speech (14,000)	43.0% (19.3% in 2000)	4.0%

* R. Lippmann, *Speech Comm.* **22**(1), **1997**

How Siri Works

- **Apple's Siri Personal Assistant consists of:**
  1. Voice recognition
     Limited vocabulary
  2. Grammar analysis
     Search for key phrases and use them to build a simple model of what user wants to do; integrated with other info on phone such as address book, nicknames, birthdays, GPS
  3. Web service providers
     Tools for mapping to external APIs for Yelp, Zagat, Wikipedia
- **Limited domains: restaurants, sports, movies, travel, weather, …**

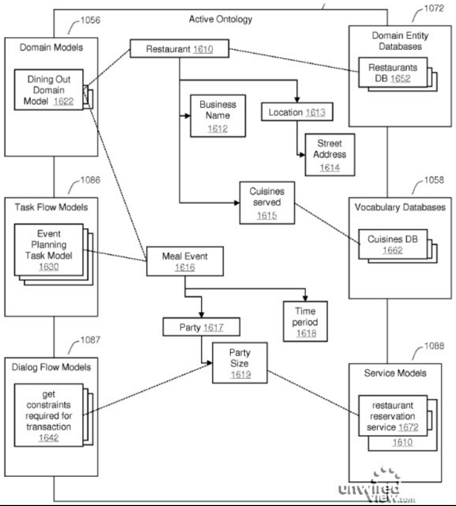
Siri Topics for Limited Domains

- **Ask for a reminder**
- **Send a text message**
- **Ask about the weather**
- **Make a dinner reservation**
- **Ask for information (e.g., from Wikipedia)**
- **Set a meeting**
- **Send e-mail**
- **Get directions**
- **Get a telephone number or make a call**
- **and a few more**

Limited Set of "Active Ontologies"

- **Restaurant/Dining Ontology includes restaurant databases and review services (e.g., Yelp and Zagat)**
- **Dining-related vocabulary database**
- **Model of actions that people usually perform when they decide on dining choices**
- **Domain-specific dialog for interaction with user**
- **Access to online reservation services, e.g., Open Table, and rules for making reservation through it, and entering result into user's calendar**

Restaurant Ontology



Other Applications of HMMs

- **Probabilistic robotics**
  - SLAM: Simultaneous Localization And Mapping
  - Robot control learning
- **Tracking objects in video**
- **Spam deobfuscation (mis-spelling words)**
- **Human Genome Project**
- **Consumer decision modeling**
- **Economics & Finance**
- **And many more …**