Lecture 6

Language Modeling/Pronunciation Modeling

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{picheny,bhuvana,stanchen}@us.ibm.com

15 October 2012

Review: Acoustic Modeling

- **x** — Observations; sequence of $\sim$40d feature vectors.
- ω — word sequence.
- HMM/GMM framework lets us model $P(\mathbf{x}|\omega)$ ...
    - How likely feature vectors are given word sequence.

The Fundamental Equation of ASR

I HATE TO WAIT
EYE HATE TWO WEIGHT

$$\omega^* = \arg\max_{\omega} \ P(\mathbf{x}|\omega)$$

$$\Downarrow$$

$$\omega^* = \arg\max_{\omega} \ P(\omega|\mathbf{x}) = \arg\max_{\omega} \ P(\omega)P(\mathbf{x}|\omega)$$

- What's new?
    - Language model $P(\omega)$ describing ...
    - Frequency of each word sequence ω.

Part I

Language Modeling

Language Modeling: Goals

- Describe which word sequences are *likely*.
  - *e.g.*, BRITNEY SPEARS *vs.* BRIT KNEE SPEARS.
- Analogy: multiple-choice test.
  - LM restricts choices given to acoustic model.
  - The fewer choices, the better you do.

What Type of Model?

- Want probability distribution over sequence of symbols.
- (Hidden) Markov model!
- Hidden or non-hidden?
  - For hidden, too hard to come up with topology.

Where Are We?

1. *N*-Gram Models

2. Technical Details

3. Smoothing

4. Discussion

What's an *n*-Gram Model?

- Markov model of order $n - 1$.
- To predict next word . . .
  - Only need to remember last $n - 1$ words.

What's a Markov Model?

- Decompose probability of sequence ...
  - Into product of conditional probabilities.
- *e.g.*, trigram model $\Rightarrow$ Markov order 2 $\Rightarrow$ ...
  - Remember last 2 words.

$$P(w_1 \cdots w_L) = \prod_{i=1}^{L} P(w_i | w_1 \cdots w_{i-1})$$

$$= \prod_{i=1}^{L} P(w_i | w_{i-2} w_{i-1})$$

$$P(\text{I HATE TO WAIT}) = P(\text{I})P(\text{HATE}|\text{I})P(\text{TO}|\text{I HATE})P(\text{WAIT}|\text{HATE TO})$$

Sentence Begins and Ends

- Pad left with beginning-of-sentence tokens.
  - *e.g.*, $w_{-1} = w_0 = \triangleright$.
  - Always condition on two words to left, even at start.
- Predict end-of-sentence token at end.
  - So true probability, *i.e.*, $\sum_\omega P(\omega) = 1$.

$$P(w_1 \cdots w_L) = \prod_{i=1}^{L+1} P(w_i | w_{i-2} w_{i-1})$$

$$P(\text{I HATE TO WAIT}) = P(\text{I}|\triangleright\triangleright) \times P(\text{HATE}|\triangleright\text{I}) \times P(\text{TO}|\text{I HATE}) \times$$
$$P(\text{WAIT}|\text{HATE TO}) \times P(\triangleleft|\text{TO WAIT})$$

How to Set Probabilities?

- For each *history* $w_{i-2} w_{i-1}$ ...
  - $P(w_i | w_{i-2} w_{i-1})$ is *multinomial* distribution.
- Maximum likelihood estimation for multinomials.
  - Count and normalize!

$$P_{\text{MLE}}(w_i | w_{i-2} w_{i-1}) = \frac{c(w_{i-2} w_{i-1} w_i)}{\sum_w c(w_{i-2} w_{i-1} w)}$$

$$= \frac{c(w_{i-2} w_{i-1} w_i)}{c(w_{i-2} w_{i-1})}$$

Example: Maximum Likelihood Estimation

- 23M words of Wall Street Journal text.

  FEDERAL HOME LOAN MORTGAGE CORPORATION –DASH ONE .POINT FIVE BILLION DOLLARS OF REALESTATE MORTGAGE -HYPHEN INVESTMENT CONDUIT SECURITIES OFFERED BY MERRILL LYNCH &AMPERSAND COMPANY

  NONCOMPETITIVE TENDERS MUST BE RECEIVED BY NOON EASTERN TIME THURSDAY AT THE TREASURY OR AT FEDERAL RESERVE BANKS OR BRANCHES

  ...
  ...

$$P(\text{TO}|\text{I HATE}) = \frac{c(\text{I HATE TO})}{c(\text{I HATE})} = \frac{17}{45} = 0.378$$

Example: Bigram Model

- $P(\text{I HATE TO WAIT}) = ???$
- $P(\text{EYE HATE TWO WEIGHT}) = ???$
- Step 1: Collect all bigram counts, unigram history counts.

	EYE	I	HATE	TO	TWO	WAIT	WEIGHT	◁	*
▷	3	3234	5	4064	1339	8	22	0	892669
EYE	0	0	0	26	1	0	0	52	735
I	0	0	45	2	1	1	0	8	21891
HATE	0	0	0	40	0	0	0	9	246
TO	8	6	19	21	5341	324	4	221	510508
TWO	0	5	0	1617	652	0	0	4213	132914
WAIT	0	0	0	71	2	0	0	35	882
WEIGHT	0	0	0	38	0	0	0	45	643

Example: Bigram Model

$P(\text{I HATE TO WAIT})$

$$= P(\text{I}|\triangleright)P(\text{HATE}|\text{I})P(\text{TO}|\text{HATE})P(\text{WAIT}|\text{TO})P(\triangleleft|\text{WAIT})$$

$$= \frac{3234}{892669} \times \frac{45}{21891} \times \frac{40}{246} \times \frac{324}{510508} \times \frac{35}{882} = 3.05 \times 10^{-11}$$

$P(\text{EYE HATE TWO WEIGHT})$

$$= P(\text{EYE}|\triangleright)P(\text{HATE}|\text{EYE})P(\text{TWO}|\text{HATE})P(\text{WEIGHT}|\text{TWO}) \times$$
$$\quad P(\triangleleft|\text{WEIGHT})$$

$$= \frac{3}{892669} \times \frac{0}{735} \times \frac{0}{246} \times \frac{0}{132914} \times \frac{45}{643} = 0$$

Recap: *N*-Gram Models

- Simple formalism, yet effective.
    - Discriminates between wheat and chaff.
- Easy to train: count and normalize.
- Generalizes.
    - Assigns nonzero probabilities to sentences . . .
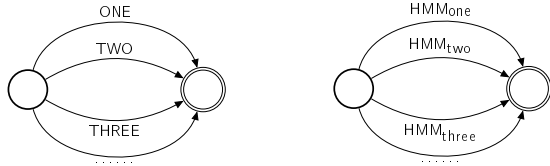   - Not seen in training data, *e.g.*, I HATE TO WAIT.

Where Are We?

1. *N*-Gram Models

2. Technical Details

3. Smoothing
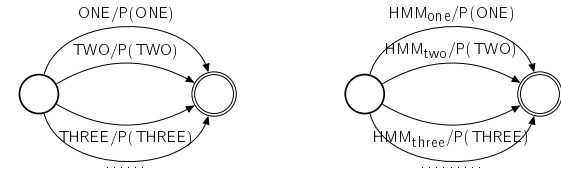
4. Discussion

LM's and Training and Decoding

- Decoding without LM's.
  - Word HMM encoding allowable word sequences.
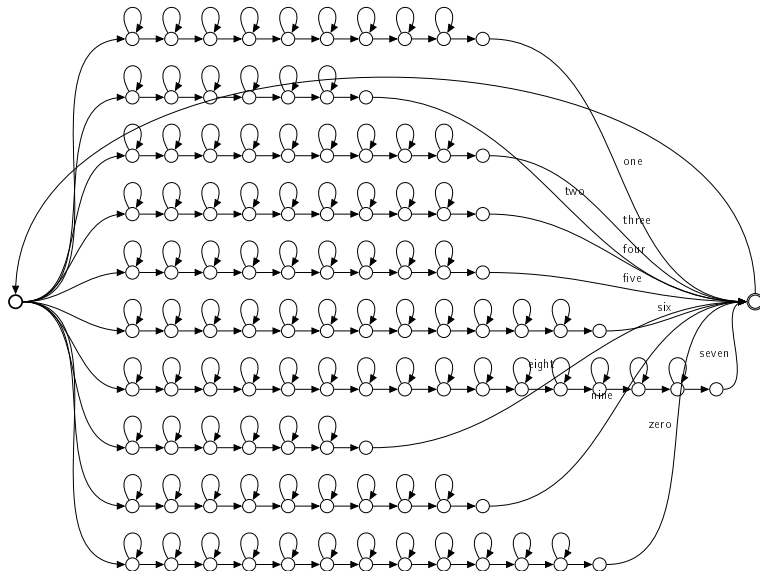 - Replace each word with its HMM.

LM's and Training and Decoding

- Point: *n*-gram model is (hidden) Markov model.
  - Can be expressed as word HMM.
  - Replace each word with its HMM.
  - Leave in language model probabilities.

- Lots more details in lectures 7, 8.
- How do LM's impact acoustic model training?

One Puny Prob versus Many?

The Acoustic Model Weight

- Not a fair fight.
- Solution: acoustic model weight.

$$\omega^* = \arg\max_{\omega}\ P(\omega)P(\mathbf{x}|\omega)^{\alpha}$$

  - α usually somewhere between 0.05 and 0.1.
- Important to tune for each LM, AM.
- Theoretically inelegant.
  - Empirical performance trumps theory any day of week.
- Is it LM weight or AM weight?

Real World Toy Example

- Test set: continuous digit strings.
- *Unigram* language model: $P(\omega) = \prod_{i=1}^{L+1} P(w_i)$.
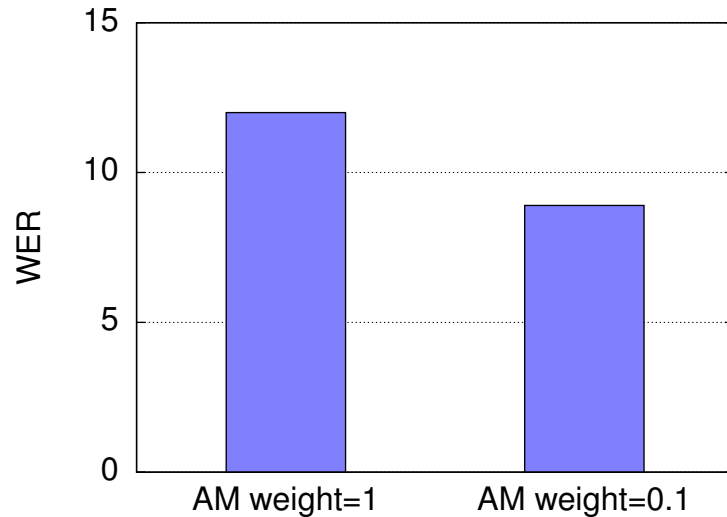
What is This Word Error Rate Thing?

- Most popular evaluation measure for ASR systems
- Divide total number of errors in test set . . .
  - By total number of words.

$$\text{WER} \equiv \frac{\sum_{\text{utts } u}(\text{\# errors in } u)}{\sum_{\text{utts } u}(\text{\# words in reference for } u)}$$

- What is "number of errors" in utterance?
  - Minimum number of word insertions, deletions, and . . .
  - Substitutions to transform reference to hypothesis.

Example: Word Error Rate

- What is the WER?

reference:	THE DOG IS HERE NOW
hypothesis:	THE UH BOG IS NOW

- Can WER be above 100%?
- What algorithm to compute WER?
  - How many ways to transform reference to hypothesis?

Evaluating Language Models

- Best way: plug into ASR system; measure WER.
  - Need ASR system.
  - Expensive to compute (especially in old days).
  - Results depend on acoustic model.
- Is there something cheaper that predicts WER well?

Perplexity

- Basic idea: test set likelihood . . .
  - Normalized so easy to interpret.
- Take (geometric) average probability p_{avg} . . .
  - Assigned to each word in test data.

$$p_{avg} = \left[\prod_{i=1}^{L+1} P(w_i | w_{i-2} w_{i-1}) \right]^{\frac{1}{L+1}}$$

- Invert it: $PP = \frac{1}{p_{avg}}$.
- Interpretation:
  - Given history, how many possible next words . . .
  - (For acoustic model to choose from.)
  - *e.g.*, uniform unigram LM over V words $\Rightarrow$ PP = V.

Example: Perplexity

$P(\text{I HATE TO WAIT})$
$$= P(\text{I}|\triangleright)P(\text{HATE}|\text{I})P(\text{TO}|\text{HATE})P(\text{WAIT}|\text{TO})P(\triangleleft|\text{WAIT})$$
$$= \frac{3234}{892669} \times \frac{45}{21891} \times \frac{40}{246} \times \frac{324}{510508} \times \frac{35}{882} = 3.05 \times 10^{-11}$$

$$p_{avg} = \left[\prod_{i=1}^{L+1} P(w_i | w_{i-1}) \right]^{\frac{1}{L+1}}$$
$$= (3.05 \times 10^{-11})^{\frac{1}{5}} = 0.00789$$
$$PP = \frac{1}{p_{avg}} = 126.8$$

Perplexity: Example Values

type	domain	training data	case+ punct	PP
human[1]	biography			142
machine[2]	Brown	600MW	$\checkmark$	790
ASR[3]	WSJ	23MW		120

- Varies highly across domains, languages. Why?

---

[1] *Jefferson the Virginian*; Shannon game (Shannon, 1951).
[2] Trigram model (Brown *et al.*, 1992).
[3] Trigram model; 20kw vocabulary.

Does Perplexity Predict Word-Error Rate?

- Not *across* different LM types.
  - *e.g.*, word *n*-gram model; class *n*-gram model; . . .
- OK *within* LM type.
  - *e.g.*, vary training set; model order; pruning; . . .

Perplexity and Word-Error Rate
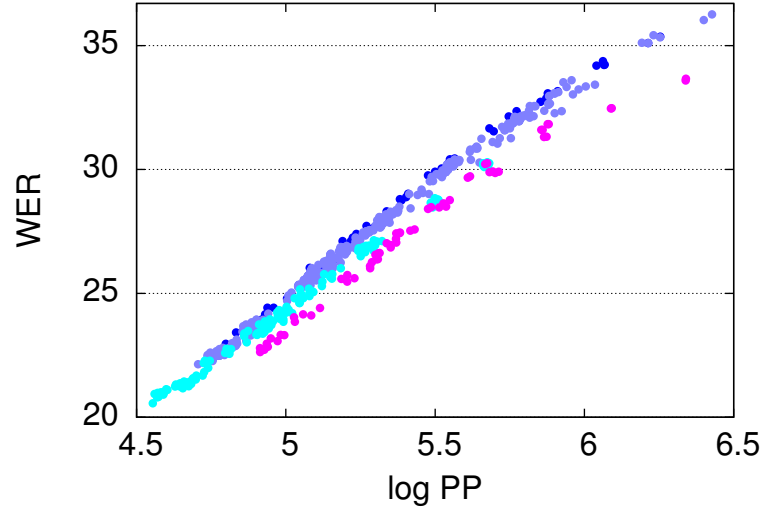
Recap

- LM describes allowable word sequences.
  - Used to build decoding graph.
- Need AM weight for LM to have full effect.
- Best to evaluate LM's using WER . . .
  - But perplexity can be informative.
- Can you think of any problems with word error rate?
  - What do we really care about in applications?

Where Are We?

1. *N-Gram Models*

2. Technical Details

3. Smoothing

4. Discussion

An Experiment

- Take 50M words of WSJ; shuffle sentences; split in two.
- "Training" set: 25M words.

  NONCOMPETITIVE TENDERS MUST BE RECEIVED BY NOON
  EASTERN TIME THURSDAY AT THE TREASURY OR AT
  FEDERAL RESERVE BANKS OR BRANCHES .PERIOD

  NOT EVERYONE AGREED WITH THAT STRATEGY .PERIOD

  . . .
  . . .

- "Test" set: 25M words.

  NATIONAL PICTURE &AMPERSAND FRAME –DASH INITIAL
  TWO MILLION ,COMMA TWO HUNDRED FIFTY THOUSAND
  SHARES ,COMMA VIA WILLIAM BLAIR .PERIOD

  THERE WILL EVEN BE AN EIGHTEEN -HYPHEN HOLE GOLF
  COURSE .PERIOD

  . . .
  . . .

An Experiment

- Count how often each word occurs in training; sort by count.

word	count
,COMMA	1156259
THE	1062057
.PERIOD	877624
OF	520374
TO	510508
A	455832
AND	417364
IN	385940
. . .	. . .
. . .	. . .

word	count
. . .	. . .
. . .	. . .
ZZZZ	2
AAAAAHHH	1
AAB	1
AACHENER	1
. . .	. . .
. . .	. . .
ZYPLAST	1
ZYUGANOV	1

An Experiment

- For each word that occurs exactly once in training . . .
  - Count how often occurs in test set.
  - Average this count across all such words.
- What is actual value?
  1. Larger than 1.
  2. Exactly 1, more or less.
  3. Between 0.5 and 1.
  4. Between 0.1 and 0.5.
- What if do this for trigrams, not unigrams?

Why?

- Q: How many unigrams/trigrams in test set . . .
  - Do not appear in training set?
  - A: 48k/7.4M.
- Q: How many unique unigrams/trigrams in training set?
  - A: 135k/9.4M.
- On average, everything seen in training is discounted!

What Does This Have To Do With Anything?

- Goal: estimate frequencies of n-grams in test data!
- MLE $\Leftrightarrow$ frequency of n-gram in training data!

$$P(\text{TO}|\text{I HATE}) = \frac{c(\text{I HATE TO})}{c(\text{I HATE})} = \frac{17}{45} = 0.378$$

- Point: training and test frequencies can differ a ton!

Maximum Likelihood and Sparse Data

- In theory, ML estimate is as good as it gets ...
  - In limit of lots of data.
- In practice, sucks when data is *sparse*.
  - Can be off by large factor.
  - *e.g.*, for 1-count trigram, MLE = $\frac{1}{25M}$.
  - Average frequency in test data = $\frac{0.25}{25M}$.
  - How bad is it for zero counts?

Maximum Likelihood and Zero Probabilities

- According to MLE bigram model ...
  - What is probability of sentence if contains ...
  - Bigram with no training counts, *e.g.*, HATE TWO?

  $P(\text{I HATE TWO PEOPLE})$
  $= P(\text{I}|\triangleright)P(\text{HATE}|\text{I})P(\text{TWO}|\text{HATE})P(\text{PEOPLE}|\text{TWO}) \times$
  $\qquad P(\triangleleft|\text{PEOPLE})$

- How common are unseen trigrams in test data?
  - (Brown *et al.*, 1992): 350M word training set: 15%.
  - What does this imply about impact on WER?
- Perplexity? (Inverse of geometric average of word probs.)

Smoothing

- Adjusting ML estimates to better match test data.
  - How to decrease probabilities for seen stuff?
  - How to estimate probabilities for unseen stuff?
- Also called *regularization*.

The Basic Idea (Bigram Model)

- Conditional distribution: $P(w|\text{HATE})$.
  - Discount nonzero counts; move mass to zero counts.

w	c	P_{MLE}	c_{smooth}	P_{smooth}
TO	40	0.163	40.0000	0.162596
THE	22	0.089	20.9840	0.085301
IT	15	0.061	14.2573	0.057957
CRIMES	13	0.053	12.2754	0.049900
...	...	...	...	...
AFTER	1	0.004	0.4644	0.001888
ALL	1	0.004	0.4644	0.001888
...	...	...	...	...
A	0	0.000	1.1725	0.004766
AARON	0	0.000	0.0002	0.000001
...	...	...	...	...
total	246	1.000	246	1.000000

How Much To Discount Nonzero Counts?

- The Good-Turing estimate (Good, 1953).
- How often word with k counts in training data . . .
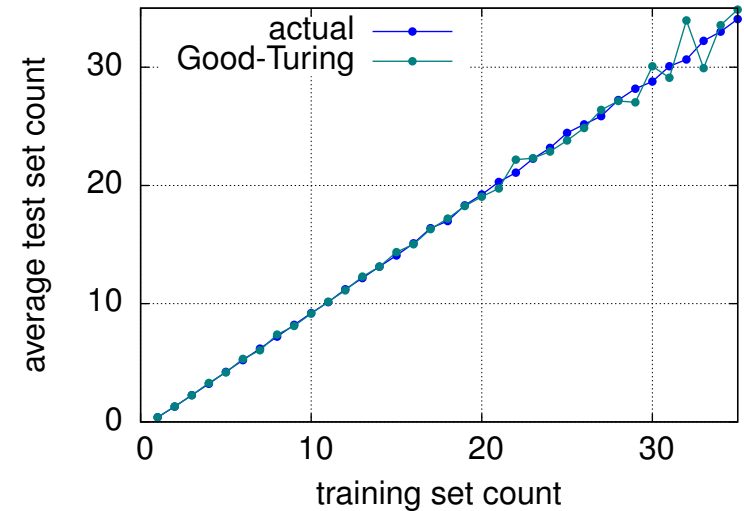 - Occurs in test set of equal size?

$$(\text{avg. count}) \approx \frac{(\text{\# words w/ } k+1 \text{ counts}) \times (k+1)}{(\text{\# words w/ } k \text{ counts})}$$

- Example: 23M words WSJ.
  - How often do 1-count words occur in test set?
  - Number of words with 1 count: 7419143.
  - Number of words with 2 counts: 933493.

$$(\text{avg. count}) \approx \frac{933493 \times 2}{7419143} = 0.252$$

How Accurate Is Good-Turing?

Bigram counts; 10M words WSJ training and test.

The Basic Idea (cont'd)

- Use GT estimate to discount counts of seen words.
  - How to divvy up counts among unseen words?

w	c	P_{MLE}	c_{smooth}	P_{smooth}
TO	40	0.163	40.0000	0.162596
THE	22	0.089	20.9840	0.085301
IT	15	0.061	14.2573	0.057957
CRIMES	13	0.053	12.2754	0.049900
. . .	. . .	. . .	. . .	. . .
AFTER	1	0.004	0.4644	0.001888
ALL	1	0.004	0.4644	0.001888
. . .	. . .	. . .	. . .	. . .
A	0	0.000	???	???????
AARON	0	0.000	???	???????
. . .	. . .	. . .	. . .	. . .
total	246	1.000	246	1.000000

Backoff

- Task: divide up some probability mass . . .
  - Among words not occurring after a history.
- Idea: uniformly?
- Better: according to *unigram* distribution $P(w)$.
  - *e.g.*, give more mass to *A* than *AARON*.

$$P(w) = \frac{c(w)}{\sum_w c(w)}$$

- *Backoff*: use lower-order distribution . . .
  - To fill in probabilities for unseen words.

Putting It All Together: Katz Smoothing

- Katz (1987)

$$P_{\text{Katz}}(w_i|w_{i-1}) = \begin{cases} P_{\text{MLE}}(w_i|w_{i-1}) & \text{if } c(w_{i-1}w_i) \geq k \\ P_{\text{GT}}(w_i|w_{i-1}) & \text{if } 0 < c(w_{i-1}w_i) < k \\ \alpha_{w_{i-1}} P_{\text{Katz}}(w_i) & \text{otherwise} \end{cases}$$

  - If count high, no discounting (GT estimate unreliable).
  - If count low, use GT estimate.
  - If no count, use scaled backoff probability.
- Choose $\alpha_{w_{i-1}}$ so $\sum_{w_i} P_{\text{Katz}}(w_i|w_{i-1}) = 1$.
- Most popular smoothing technique for about a decade.

Example: Katz Smoothing

- Conditional distribution: $P(w|\text{HATE})$.

w	c	P_{MLE}	c_{smooth}	P_{smooth}
TO	40	0.163	40.0000	0.162596
THE	22	0.089	20.9840	0.085301
IT	15	0.061	14.2573	0.057957
CRIMES	13	0.053	12.2754	0.049900
...	...	...	...	...
AFTER	1	0.004	0.4644	0.001888
ALL	1	0.004	0.4644	0.001888
...	...	...	...	...
A	0	0.000	1.1725	0.004766
AARON	0	0.000	0.0002	0.000001
...	...	...	...	...
total	246	1.000	246	1.000000

Recap: Smoothing

- ML estimates: way off for low counts.
  - Zero probabilities kill performance.
- Key aspects of smoothing algorithms.
  - How to discount counts of seen words.
  - Estimating mass of unseen words.
  - Backoff to get information from lower-order models.
- No downside.

Where Are We?

1. *N*-Gram Models

2. Technical Details

3. Smoothing

4. **Discussion**

N-Gram Models

- Workhorse of language modeling for ASR for 30 years.
  - Used in great majority of deployed systems.
- Almost no linguistic knowledge.
  - Totally data-driven.
- Easy to build.
  - Fast and scalable.
  - Can train on vast amounts of data; just gets better.

Smoothing

- Lots and lots of smoothing algorithms developed.
  - Will talk about newer algorithms in Lecture 11.
  - Gain: ∼1% absolute in WER over Katz.
- With good smoothing, don't worry models being too big!
  - Can increase n-gram order w/o loss in performance.
  - Can gain in performance if lots of data.
- Rule of thumb: if ML estimate is working OK . . .
  - Model is way too small.

Does Markov Property Hold For English?

- Not for small n.

$$P(w_i \mid \text{OF THE}) \neq P(w_i \mid \text{KING OF THE})$$

- Make n larger?

  FABIO, WHO WAS NEXT IN LINE, ASKED IF THE TELLER SPOKE . . .

- Lots more to say about language modeling . . .
  - In Lecture 11.

References

- C.E. Shannon, "Prediction and Entropy of Printed English", Bell Systems Technical Journal, vol. 30, pp. 50–64, 1951.

- I.J. Good, "The Population Frequencies of Species and the Estimation of Population Parameters", Biometrika, vol. 40, no. 3 and 4, pp. 237–264, 1953.

- S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 35, no. 3, pp. 400–401, 1987.

- P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.C. Lai, R.L. Mercer, "An Estimate of an Upper Bound for the Entropy of English", Computational Linguistics, vol. 18, no. 1, pp. 31–40, 1992.

Part II

Administrivia

Administrivia

- Clear (7); mostly clear (10); unclear (1).
- Pace: too fast/too much content (4); OK (10); too slow/not enough time on LM's (2).
- Feedback (2+ votes):
  - More demos (2).
  - More examples (2).
  - Post answers to lab/sooner (2).
  - Put administrivia in middle of lecture.
- Muddiest: n-grams (2); . . .

Administrivia

- Lab 1
  - Handed back today?
  - Answers:
    `/user1/faculty/stanchen/e6870/lab1_ans/`
- Lab 2
  - Due two days from now (Wednesday, Oct. 17) at 6pm.
  - Xiao-Ming has extra office hours: Tue 2-4pm.
- Optional non-reading projects.
  - Will be posted Thursday; we'll send out announcement.
  - Proposal will be due week from Wednesday (Oct. 24).
  - For reading projects, oral presentation $\Rightarrow$ paper.

Part III

Pronunciation Modeling

In the beginning...

- ... . was the whole word model.
- For each word in the vocabulary, decide on a topology.
- Often the number of states in the model is chosen to be proportional to the number of phonemes in the word.
- Train the observation and transition parameters for a given word using examples of that word in the training data.
- Good domain for this approach: digits.

Example topologies: Digits

- Vocabulary consists of ("zero", "oh", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine").
- Assume we assign two states per phoneme.
- Must allow for different durations
- Models look like:
- "zero".

- "oh".

"one"   W AA N

"two"   T UW

"three"   TH R IY

"four"   F AO R

"five"   F AY V

"six"

"seven"

"eight"

"nine"
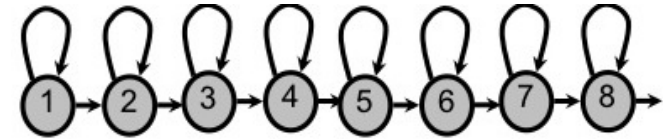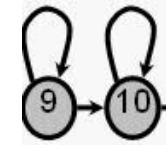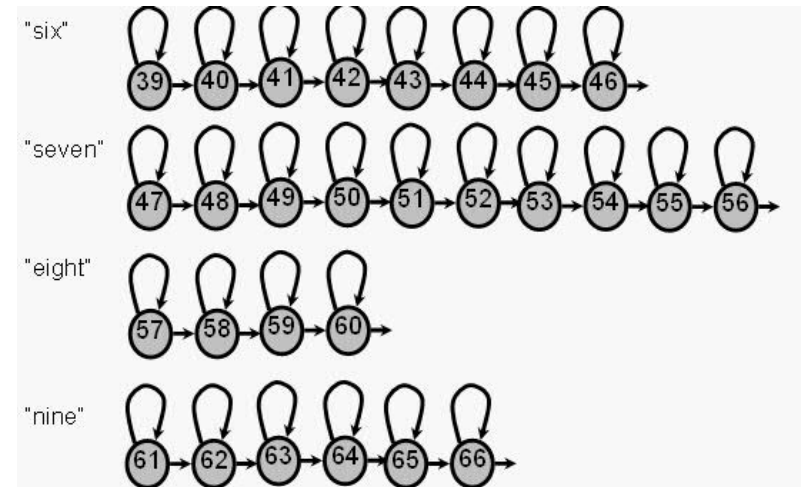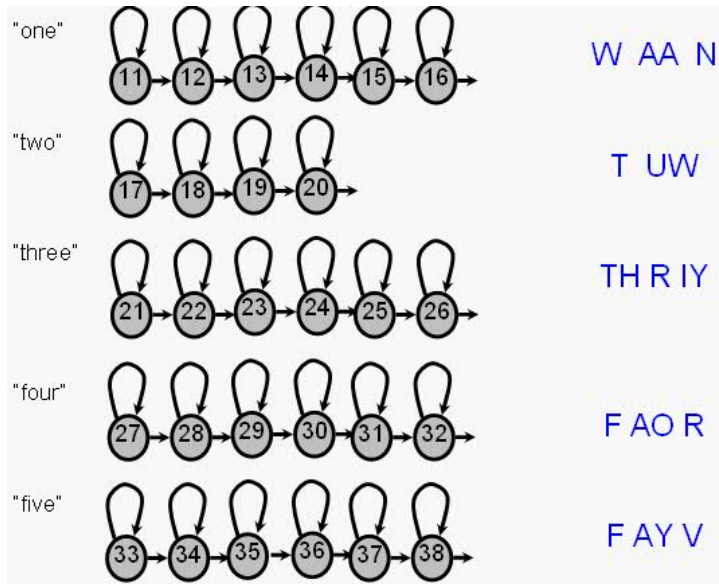
How to represent any sequence of digits?
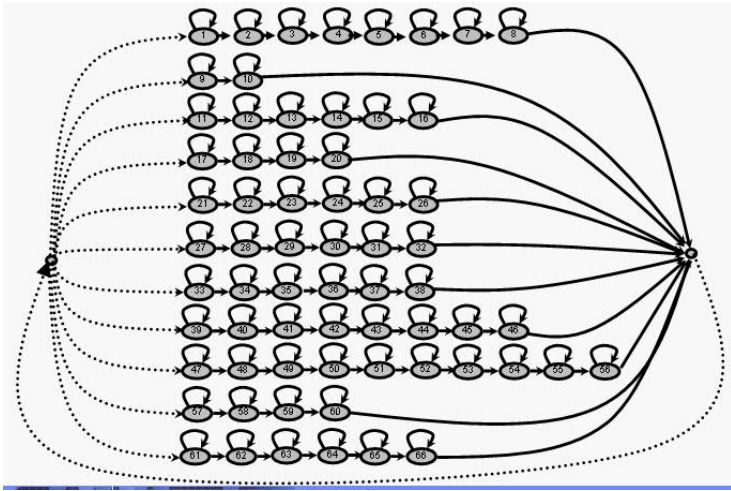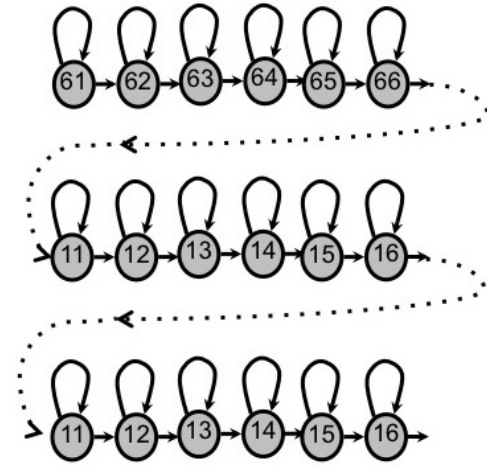
"911"

Trellis Representation
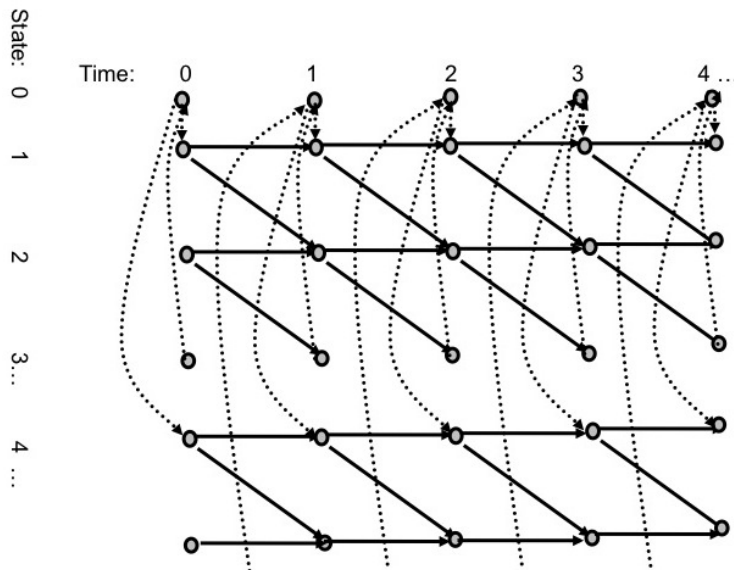
Whole-word model limitations

- The whole-word model suffers from two main problems.
  - Cannot model unseen words. In fact, we need several samples of each word to train the models properly.
  - Cannot share data among models – data sparseness problem.
  - The number of parameters in the system is proportional to the vocabulary size.
- Thus, whole-word models are best on small vocabulary tasks.

Subword Units

- To reduce the number of parameters, we can compose word models from sub-word units.
- These units can be shared among words. Examples include

Units	Approximate number
Phones	50.
Diphones	2000.
Syllables	5,000.

- Each unit is small.
- The number of parameters is proportional to the number of units (not the number of words in the vocabulary as in whole-word models.).

Phonetic Models

- We represent each word as a sequence of phonemes. This representation is the "baseform" for the word.

$$\text{BANDS} \ \rightarrow \ \text{B AE N D Z}$$

- Some words need more than one baseform.

$$\begin{aligned} \text{THE} \ &\rightarrow \ \text{DH UH} \\ &\rightarrow \ \text{DH IY} \end{aligned}$$

Baseform Dictionary

- To determine the pronunciation of each word, we look it up in a dictionary.
- Each word may have several possible pronunciations.
- Every word in our training script and test vocabulary must be in the dictionary.
- The dictionary is generally written by hand.
- Prone to errors and inconsistencies.

Phonetic Models, cont'd

- We can allow for a wide variety of phonological variation by representing baseforms as graphs.

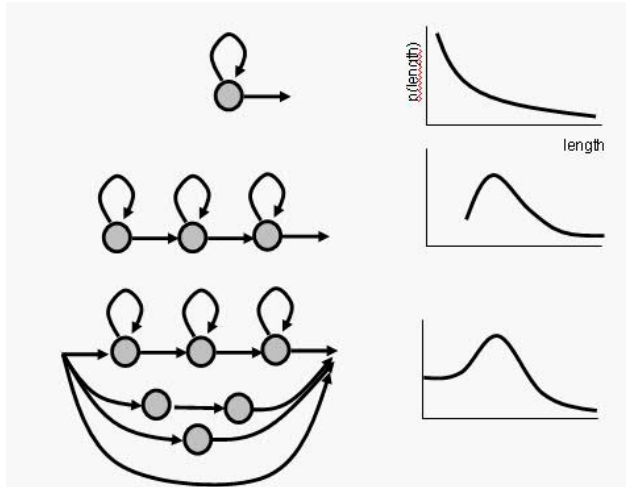| acapulco	AE K AX P AH L K OW
acapulco	AA K AX P UH K OW
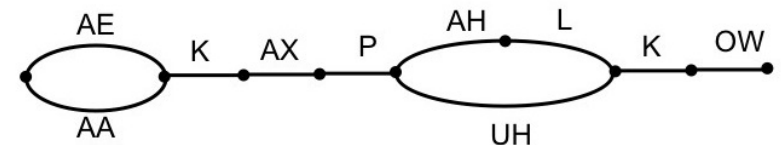
Phonetic Models, cont'd
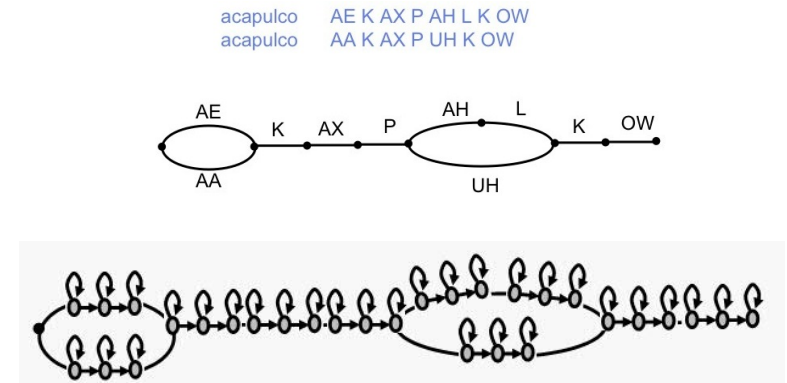
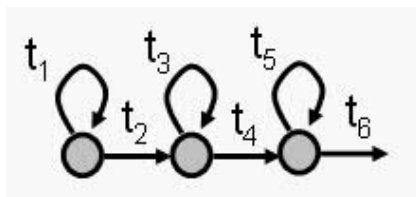- Now, construct a Markov model for each phone.
- Examples:

Embedding

- Replace each phone by its Markov model to get a word model.
- N.b. The model for each phone will have different parameter values.

acapulco   AE K AX P AH L K OW
acapulco   AA K AX P UH K OW

Reducing Parameters by Tying

- Consider the three-state model.

- Note that.
  - t_1 and t_2 correspond to the beginning of the phone.
  - t_3 and t_4 correspond to the middle of the phone.
  - t_5 and t_6 correspond to the end of the phone.
- If we force the output distributions for each member of those pairs to be the same, then the training data requirements are reduced.

Tying

- A set of arcs in a Markov model are tied to one another if they are constrained to have identical output distributions.
- Similarly, states are tied if they have identical transition probabilities.
- Tying can be explicit or implicit.

Implicit Tying

- Occurs when we build up models for larger units from models of smaller units.
- Example: when word models are made from phone models.
- First, consider an example without any tying.
  - Let the vocabulary consist of digits 0,1,2,... 9.
- We can make a separate model for each word.
- To estimate parameters for each word model, we need several samples for each word.
- Samples of "0" affect only parameters for the "0" model.

Implicit Tying, cont'd
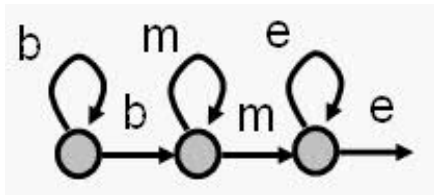
- Now consider phone-based models for this vocabulary.

- Training samples of "0" will also affect models for "3" and "4".
- Useful in large vocabulary systems where the number of words is much greater than the number of phones.

Explicit Tying

- Example:

- 6 non-null arcs, but only 3 different output distributions because of tying.
- Number of model parameters is reduced.
- Tying saves storage because only one copy of each distribution is saved.
- Fewer parameters mean less training data needed.

Variations in realizations of phonemes

- The broad units, phonemes, have variants known as allophones
  - Example: p and p^h (un-aspirated and aspirated p).
  - Exercise: Put your hand in front of your mouth and pronounce *spin* and then *pin* Note that the p in *pin* has a puff of air,. while the p in *spin* does not.
- Articulators have inertia, thus the pronunciation of a phoneme is influenced by surrounding phonemes. This is known as co-articulation
  - Example: Consider k and g in different contexts.
    - In *key* and *geese* the whole body of the tongue has to be pulled up to make the vowel.
    - Closure of the k moves forward compared to *caw* and *gauze*.
- Phonemes have canonical articulator target positions that may or may not be reached in a particular utterance.
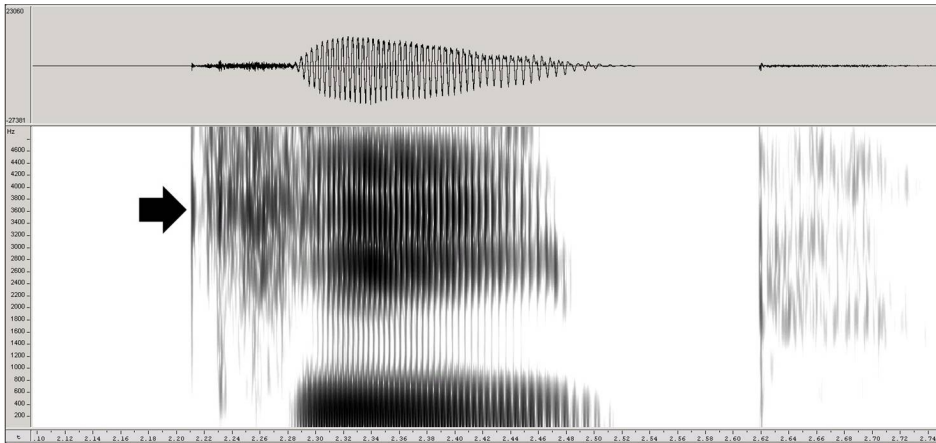
keep
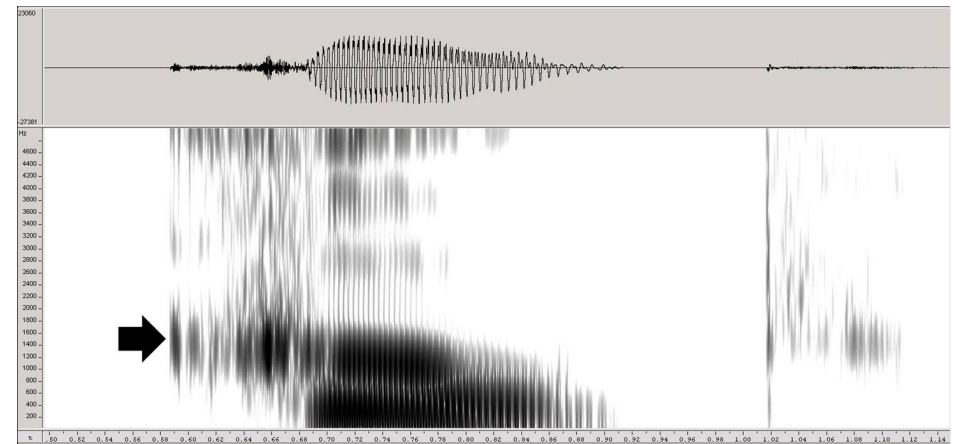
coop

Context-dependent models

- We can model phones in context.
- Two approaches: "triphones" and "Decision Trees".
- Both methods use clustering. "Triphones" use bottom-up clustering, "Decision trees" implement top-down clustering.
- Typical improvements of speech recognizers when introducing context dependence: 30% - 50% fewer errors.

Triphone models

- Model each phoneme in the context of its left and right neighbor.
- E.g. K-IY+P is a model for IY when K is its left context phoneme and P is its right context phoneme.
- If we have 50 phonemes in a language, we could have as many as 50^3 triphones to model.
- Not all of these occur.
- Still have data sparsity issues.
- Try to solve these issues by agglomerative clustering.

Agglomerative / "Bottom-up" Clustering

- Start with each item in a cluster by itself.
- Find "closest" pair of items.
- Merge them into a single cluster.
- Iterate.
- Different results based on distance measure used.
  - Single-link: dist(A,B) = min dist(a,b) for aA, bB.
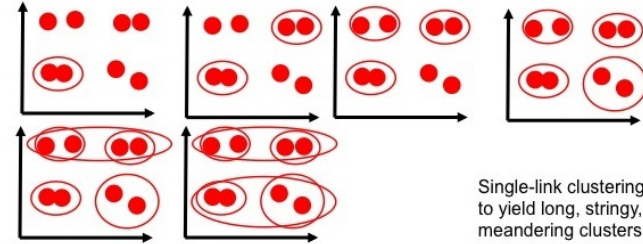 - Complete-link: dist(A,B) = max dist(a,b) for aA, bB.

Bottom-up clustering / Single Link

- Assume our data points look like:
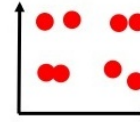
Single-link: clusters are close if any of their points are:
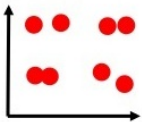dist(A,B) = min dist(a,b) for a∈A, b∈B

- Single-link clustering into 2 groups proceeds as:

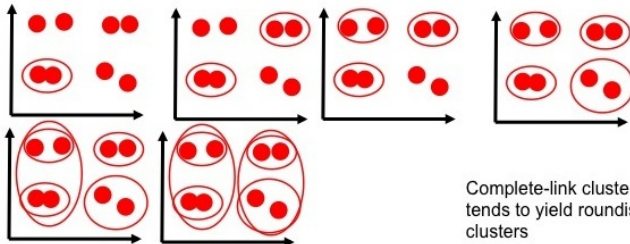Single-link clustering tends to yield long, stringy, meandering clusters

Bottom-up clustering / Complete Link

- Again, assume our data points look like:

Complete-link: clusters are close only if ALL of their points are:
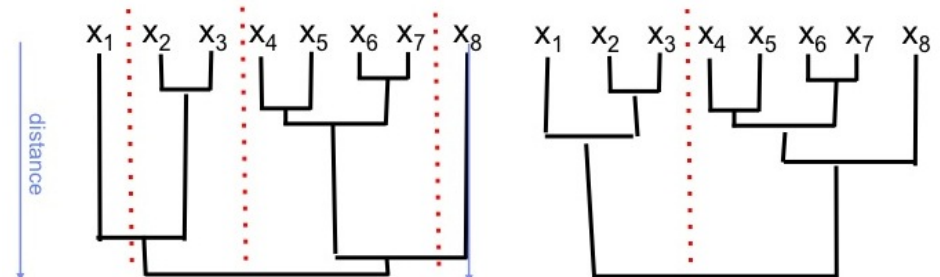dist(A,B) = max dist(a,b) for a∈A, b∈B

- Single-link clustering into 2 groups proceeds as:

Complete-link clustering tends to yield roundish clusters

Dendrogram

- A natural way to display clusters is through a "dendrogram".
- Shows the clusters on the x-axis, distance between clusters on the y-axis.
- Provides some guidance as to a good choice for the number of clusters.

Triphone Clustering

- We can use e.g. complete-link clustering to cluster triphones.
- Helps with data sparsity issue.
- Still have an issue with unseen data.
- To model unseen events, we need to "back-off" to lower order models such as bi-phones and uni-phones.

Decision Trees

- Goal of any clustering scheme is to find equivalence classes among our training samples.
- A decision tree maps data tagged with set of input variables into equivalence classes.
- Asks questions about the input variables to designed to improve some criterion function associated with the training data.
    - Output data may be labels - criteria could be entropy
    - Output data may be real numbers or vector - criteria could be mean-square error
- The goal when constructing a decision tree is significantly improve the criterion function (relative to doing nothing)

Decision Trees - A Form of Top-Down Clustering

- DTs perform top-down clustering because constructed by asking series of questions that recursively split the training data.
- In our case,
    - The input features will be phonetic context (the phones to left and right of phone for which we are creating a context-dependent model;
    - The output data will be the feature vectors associated with each phone
    - The criterion function will be the likelihood of the output features.
- **Classic text:** L. Breiman et al. Classification and Regression Trees. Wadsworth & Brooks. Monterey, California. 1984.

What does a decision tree look like?
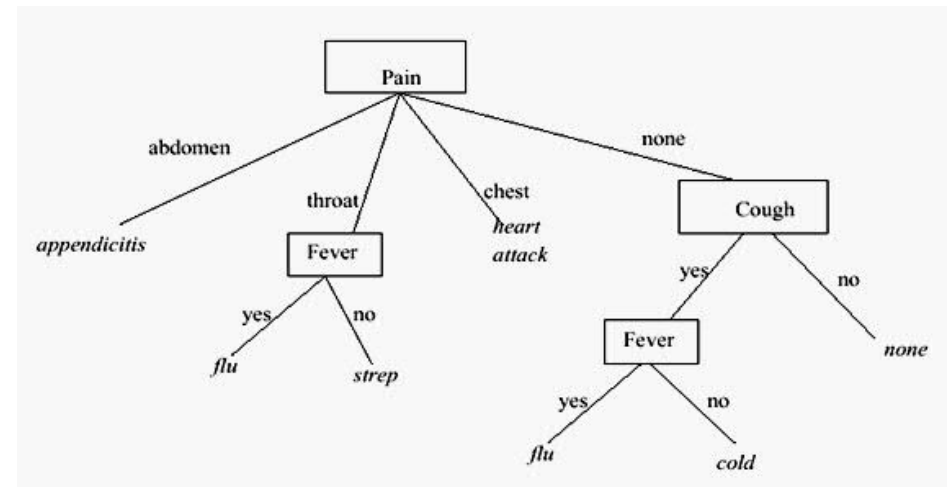
Types of Input Attributes/Features

- Numerical: Domain is ordered and can be represented on the real line (e.g., age, income).
- Nominal or categorical: Domain is a finite set without any natural ordering (e.g., occupation, marital status, race).
- Ordinal: Domain is ordered, but absolute differences between values is unknown (e.g., preference scale, severity of an injury).

The Classification Problem

- If the dependent variable is categorical, the problem is a *classification problem*.
- Let C be the class label of a given data point $X = \{X_1, \ldots, X_k\}$
- Let $d()$ be the predicted class label
- Define the *misclassification rate* of d:

$$P(d(X = \{X_1, \ldots, X_k\}) \neq C$$

- **Problem definition:** Given a dataset, find the classifier d such that the misclassification rate is minimized.

The Regression Problem

- If the dependent variable is numerical, the problem is a *regression problem.*.
- The tree d maps observation X to prediction Y' of Y and is called a *regression function.*.
- Define mean squared error of d as:

$$E[(Y - d(X = \{X_1, \ldots, X_k\}))^2]$$

- **Problem definition:** Given dataset, find regression function d such that mean squared error is minimized.

Goals & Requirements

- Traditional Goals of Decision Trees
  - To produce an accurate classifier/regression function.
  - To understand the structure of the problem.
- Traditional Requirements on the model:
  - High accuracy.
  - Understandable by humans, interpretable.
  - Fast construction for very large training databases.
- For speech recognition, understandibility quickly goes out the window....
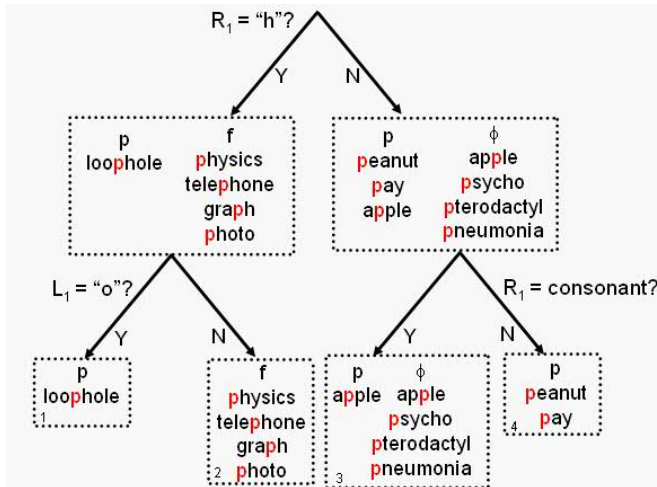
Decision Trees: Letter-to-Sound Example

- Let's say we want to build a tree to decide how the letter "p" will sound in various words.
- Training examples:

| p | loophole peanuts pay apple |
| f | physics telephone graph photo |
| ϕ | apple psycho pterodactyl pneumonia |

- The pronunciation of "p" depends on its context.
- Task: Using the above training data, partition the contexts into equivalence classes so as to minimize the uncertainty of the pronunciation.

Decision Trees: Letter-to-Sound Example, cont'd

- Denote the context as ... L_2 L_1 p R_1 R_2 ...
- Ask potentially useful question: $R_1 =$ "h"?
- At this point we have two equivalence classes: 1. $R_1 =$ "h" and 2. $R_1 \neq$ "h".
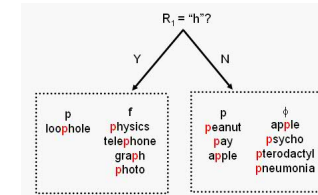
- The pronunciation of class 1 is either "p" or "f", with "f" much more likely than "p".
- The pronunciation of class 2 is either "p" or "ϕ"

Four equivalence classes. Uncertainty only remains in class 3.
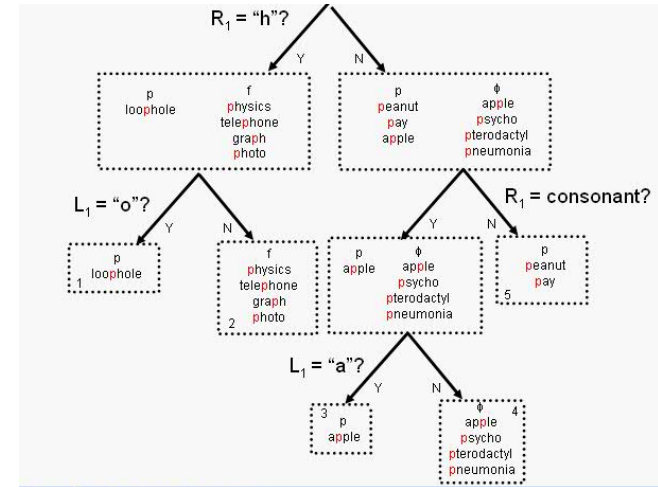
Five equivalence classes, which is much less than the number of letter contexts. No uncertainty left in the classes.

A node without children is called a leaf node. Otherwise it is called an internal node

Test Case: Paris

Test Case: gopher

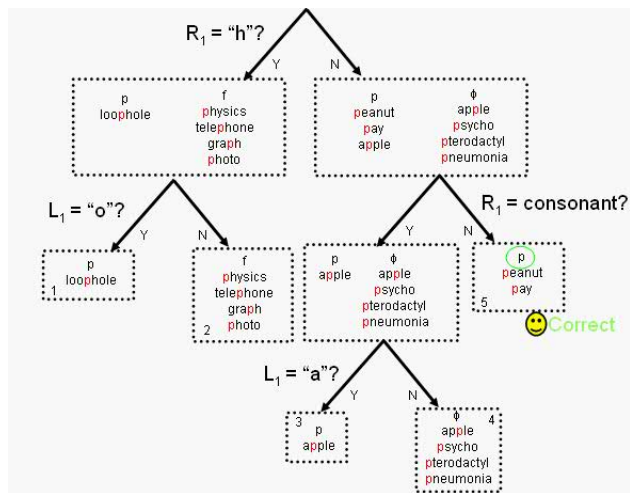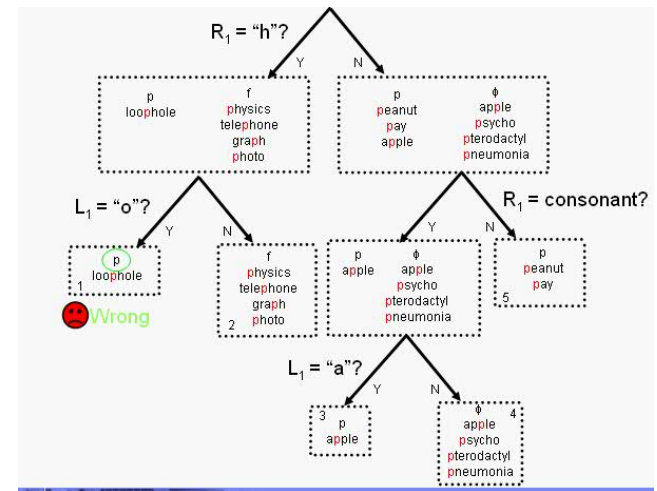Although effective on the training data, this tree does not generalize well. It was constructed from too little data.

Decision Tree Construction

1. Find the best question for partitioning the data at a given node into 2 equivalence classes.
2. Repeat step 1 recursively on each child node.
3. Stop when there is insufficient data to continue or when the best question is not sufficiently helpful.

Basic Issues to Solve

- The selection of the splits.
- The decisions when to declare a node terminal or to continue splitting.

Decision Tree Construction – Fundamental Operation

- There is only 1 fundamental operation in tree construction:
  - Find the best question for partitioning a subset of the data into two smaller subsets.
  - i.e. Take an equivalence class and split it into 2 more-specific classes.

Decision Tree Greediness

- Tree construction proceeds from the top down – from root to leaf.
- Each split is intended to be locally optimal.
- Constructing a tree in this "greedy" fashion usually leads to a good tree, but probably not globally optimal.
- Finding the globally optimal tree is an NP-complete problem: it is not practical.

Splitting

- Each internal node has an associated splitting question.
- Example questions:
  - Age <= 20 (numeric).
  - Profession in (student, teacher) (categorical).
  - 5000*Age + 3*Salary – 10000 > 0 (function of raw features).

Dynamic Questions

- The best question to ask about some discrete variable x consists of the best subset of the values taken by x.
- Search over all subsets of values taken by x at a given node. (This means generating questions on the fly during tree construction.).

$$x \in \{A, B, C\}$$

Q1:$x \in \{A\}$? Q2:$x \in \{B\}$? Q3:$x \in \{C\}$?
Q4:$x \in \{A, B\}$? Q5:$x \in \{A, C\}$? Q6:$x \in \{B, C\}$?

- Use the best question found.
- Potential problems:
  - Requires a lot of CPU. For alphabet size A there are $\sum_j \binom{A}{j}$ questions.
  - Allows a lot of freedom, making it easy to overtrain.

Pre-determined Questions

- The easiest way to construct a decision tree is to create in advance a list of possible questions for each variable.
- Finding the best question at any given node consists of subjecting all relevant variables to each of the questions, and picking the best combination of variable and question.
- In acoustic modeling, we typically ask about 2-4 variables: the 1-2 phones to the left of the current phone and the 1-2 phones to the right of the current phone. Since these variables all span the same alphabet (phone alphabet) only one list of questions.
- Each question on this list consists of a subset of the phonetic phone alphabet.

Sample Questions

| Phones	Letters
{P}	{A}
{T}	{E}
{K}	{I}
{B}	{O}
{D}	{U}
{G}	{Y}
{P,T,K}	{A,E,I,O,U}
{B,D,G}	{A,E,I,O,U,Y}
{P,T,K,B,D,G}	

Discrete Questions

- A decision tree has a question associated with every non-terminal node.
- If x is a discrete variable which takes on values in some finite alphabet A, then a question about x has the form: $x \in S$? where S is a subset of A.
- Let L denote the preceding letter in building a spelling-to-sound tree. Let S=(A,E,I,O,U). Then $L \in S$? denotes the question: Is the preceding letter a vowel?
- Let R denote the following phone in building an acoustic context tree. Let S=(P,T,K). Then $R \in S$? denotes the question: Is the following phone an unvoiced stop?

Continuous Questions

- If x is a continuous variable which takes on real values, a question about x has the form x<q? where q is some real value.
- In order to find the threshold q, we must try values which separate all training samples.
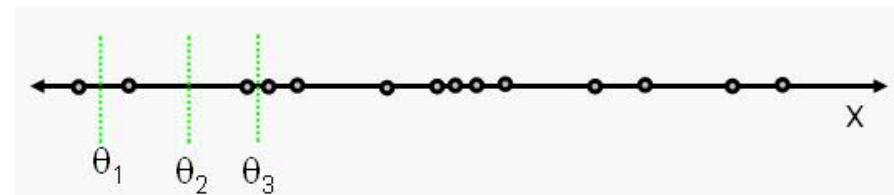
- We do not currently use continuous questions for speech recognition.

Types of Questions

- In principle, a question asked in a decision tree can have any number (greater than 1) of possible outcomes.
- Examples:
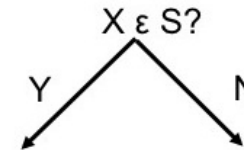 - Binary: Yes No.
  - 3 Outcomes: Yes No Don't_Know.

26 Outcomes: A B C ... Z.

- In practice, only binary questions are used to build decision trees.

Simple Binary Question

- A simple binary question consists of a single Boolean condition, and no Boolean operators.
- $X_1 \in S_1$? Is a simple question.
- $((X_1 \in S_1)\&\&(X_2 \in S_2))$? is not a simple question.
- Topologically, a simple question looks like:

Complex Binary Question

- A complex binary question has precisely 2 outcomes (yes, no) but has more than 1 Boolean condition and at least 1 Boolean operator.
- $((X_1 \in S_1)\&\&(X_2 \in S_2))$? Is a complex question.
- Topologically this question can be shown as:
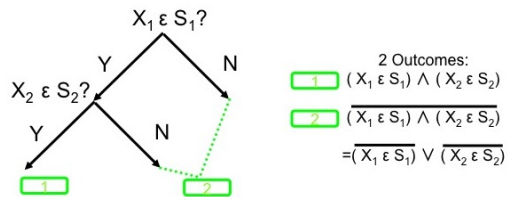
- All complex binary questions can be represented as binary trees with terminal nodes tied to produce 2 outcomes.

Configurations Currently Used

- All decision trees currently used in speech recognition use:
  - a pre-determined set
  - of simple,
  - binary questions.
  - on discrete variables.

Tree Construction Overview

- Let $x_1 \ldots x_n$ denote n discrete variables whose values may be asked about. Let Q_{ij} denote the jth pre-determined question for x_i.
- Starting at the root, try splitting each node into 2 sub-nodes:
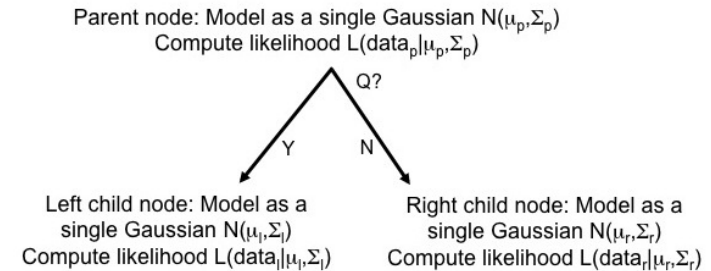
  1. For each x_i evaluate questions $Q_{i1}, Q_{i2}, \ldots$ and let Q_i' denote the best.
  2. Find the best pair x_i, Q_i' and denote it x', Q'
  3. If Q' is not sufficiently helpful, make the current node a leaf.
  4. Otherwise, split the current node into 2 new sub-nodes according to the answer of question Q' on variable x'.

- Stop when all nodes are either too small to split further or have been marked as leaves.

Question Evaluation

- The best question at a node is the question which maximizes the likelihood of the training data at that node after applying the question.

Parent node: Model as a single Gaussian N(μ_p,Σ_p)
Compute likelihood L(data$_p$|μ_p,Σ_p)

Q?

Y   N

Left child node: Model as a single Gaussian N(μ_l,Σ_l)
Compute likelihood L(data$_l$|μ_l,Σ_l)

Right child node: Model as a single Gaussian N(μ_r,Σ_r)
Compute likelihood L(data$_l$|μ_r,Σ_r)

- Goal: Find Q such that $L(data_l|\mu_l, \Sigma_l) x L(data_r|\mu_r, \Sigma_r)$ is maximized.

Question Evaluation, cont'd

- Let feature x have a set of M possible outcomes.
- Let $x^1, x^2, \ldots, x^N$ be the data samples for feature x
- Let each of the M outcomes occur $c_i(i = 1, 2, \ldots, M)$ times in the overall sample
- Let Q be a question which partitions this sample into left and right sub-samples of size n_l and n_r, respectively.
- Let c_i^l, c_i^r denote the frequency of the ith outcome in the left and right sub-samples.
- The best question Q for feature x is defined to be the one which maximizes the conditional (log) likelihood of the combined sub-samples.

log likelihood computation

- The log likelihood of the data, given that we ask question Q, is:

$$\log L(x^1, \ldots, x^n|Q) = \sum_{i=1}^{N} c_i^l \log p_i^l + \sum_{i=1}^{N} c_i^r \log p_i^r$$

- The above assumes we know the "true" probabilities p_i^l, p_i^r
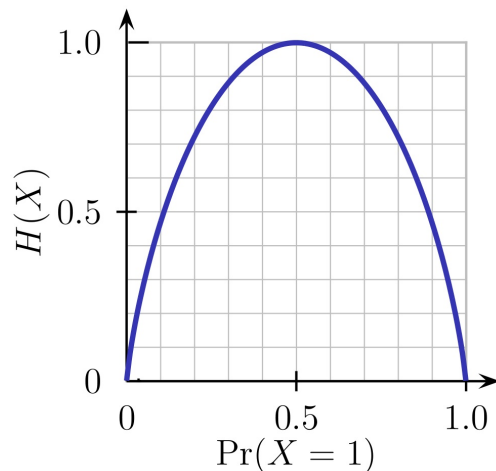
log likelihood computation (continued)

- Using the maximum likelihood estimates of p_i^l, p_i^r gives:

$$
\begin{aligned}
\log L(x^1, \ldots, x^n | Q) &= \sum_{i=1}^{N} c_i^l \log \frac{c_i^l}{n^l} + \sum_{i=1}^{N} c_i^r \log \frac{c_i^r}{n^r} \\
&= \sum_{i=1}^{N} c_i^l \log c_i^l - \log n_l \sum_{i=1}^{N} c_i^l + \sum_{i=1}^{N} c_i^r \log c_i^r - \log n_r \sum_{i=1}^{N} c_i^r \\
&= \sum_{i=1}^{N} \{ c_i^l \log c_i^l + c_i^r \log c_i^r \} - n_l \log n_l - n_r \log n_r
\end{aligned}
$$

- The best question is the one which maximizes this simple expression. c_i^l, c_i^r, n_l, n_r are all non-negative integers.
- The above expression can be computed very efficiently using a precomputed table of $n \log n$ for non-nonegative integers n

Entropy

- Let x be a discrete random variable taking values $a_1, \ldots, a_N$ in an alphabet A of size N with probabilities $p_1, \ldots, p_N$ respectively.
- The *uncertainty* about what value x will take can be measured by the entropy of the probability distribution $p = (p_1 p_2 \ldots p_N)$

$$
H = -\sum_{i=1}^{N} p_i \log_2 p_i
$$
$$
H = 0 \Leftrightarrow p_j = 1 \text{ for some } j \text{ and } p_i = 0 \text{ for } i \neq j
$$

- $H >= 0$
- Entropy is maximized when $p_i = 1/N$ for all i. Then $H = \log_2 N$
- Thus H tells us something about the sharpness of the distribution p.

What does entropy look like for a binary variable?

$\mathrm{Pr}(X = 1)$

Entropy and Likelihood

- Let x be a discrete random variable taking values $a_1, \ldots a_N$ in an alphabet A of size N with probabilities $p_1, \ldots, p_N$ respectively.
- Let $x^1, \ldots, x^n$ be a sample of x in which a_i occurs c_i times
- The sample log likelihood is: $\log L = \sum_{i=1}^{n} c_i \log p_i$
- The maximum likelihood estimate of p_i is $\hat{p}_i = c_i/n$
- Thus, an estimate of the sample log likelihood is
$\log \hat{L} = n \sum_{i=1}^{N} \hat{p}_i \log_2 \hat{p}_i \propto -\hat{H}$
- Therefore, maximizing likelihood $\Leftrightarrow$ minimizing entropy.

"p" tree, revisited

p	loophole peanuts pay apple	$c_p = 4$
f	physics telephone graph photo	$c_f = 4$
ϕ	apple psycho pterodactyl pneumonia	$c_\phi = 4, n = 12$

- Log likelihood of the data at the root node is
  - $\log_2 L(x^1, \ldots, x^{12}) = \sum_{i=1}^{3} c_i \log_2 c_i - n \log_2 n$
  - $= 4 \log_2 4 + 4 \log_2 4 + 4 \log_2 4 - 12 \log_2 12 = -19.02$
- Average entropy at the *root node* is
  - $H(x^1, \ldots, x^{12}) = -1/n \log_2 L(x^1, \ldots, x^{12})$
  - = 19.02/12 = 1.58 bits
- Let's now apply the above formula to compare three different questions.

"p" tree revisited: Question A
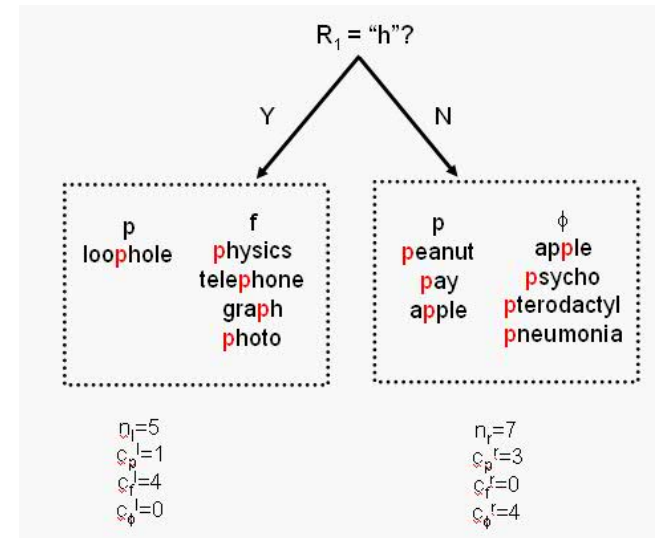
"p" tree revisited: Question A

Remember formulae for Log likelihood of data:

$$\sum_{i=1}^{N} \{c_i^l \log c_i^l + c_i^r \log c_i^r\} - n_l \log n_l - n_r \log n_r$$

Log likelihood of data after applying question A is:

$$\log_2 L(x^1, \ldots, x^{12} | Q_A) = \overbrace{1 \log_2 1}^{c_p^l} + \overbrace{4 \log_2 4}^{c_f^l} + \overbrace{3 \log_2 3}^{c_p^r} + \overbrace{4 \log_2 4}^{c_\phi^r} - \overbrace{5 \log_2 5}^{n_l} - \overbrace{7 \log_2 7}^{n_r} = -10.51$$

Average entropy of data after applying question A is

$H(x^1, \ldots, x^{12} | Q_A) = -1/n \log_2 L(x^1, \ldots, x^{12} | Q_A) = 10.51/12 = .87$ bits

Increase in log likelihood do to question A is -10.51 + 19.02 = 8.51
Decrease in entropy due to question A is 1.58-.87 = .71 bits

Knowing the answer to question A provides 0.71 bits of information about the pronunciation of p. A further 0.87 bits of information is still required to remove all the uncertainty about the pronunciation of p.

"p" tree revisited: Question B

"p" tree revisited: Question B

Log likelihood of data after applying question B is:

$\log_2 L(x^1, \ldots, x^{12} | Q_B) =$

$2\log_2 2 + 2\log_2 2 + 3\log_2 3 + 2\log_2 2 + 2\log_2 2 - 7\log_2 7 - 5\log_2 5 = -18.51$

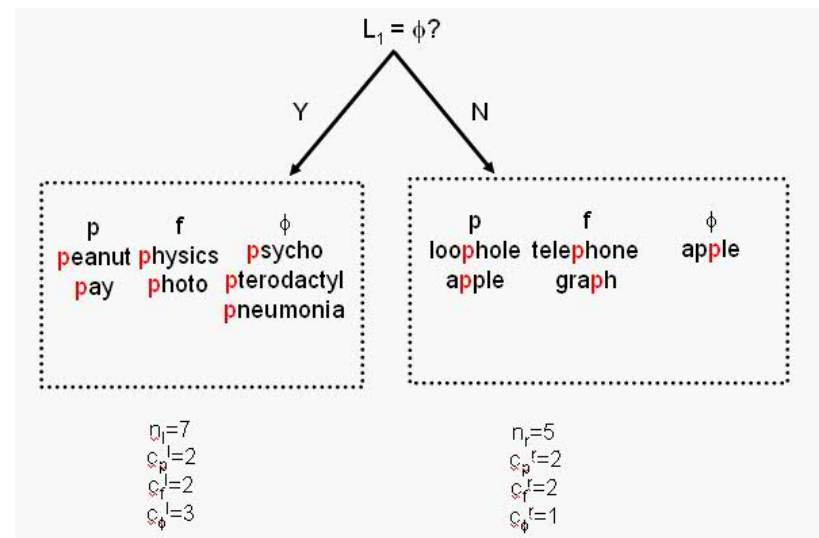Average entropy of data after applying question B is

$H(x^1, \ldots, x^{12} | Q_B) = -1/n \log_2 L(x^1, \ldots, x^{12} | Q_B) = 18.51/12 = .87$ bits
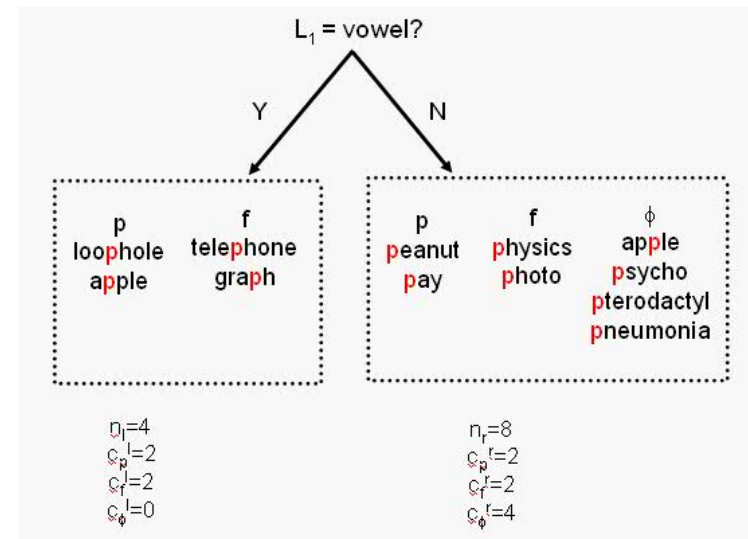
Increase in log likelihood do to question B is -18.51 + 19.02 = .51
Decrease in entropy due to question B is 1.58-1.54 = .04 bits

Knowing the answer to question B provides 0.04 bits of information (very little) about the pronunciation of p.

"p" tree revisited: Question C

"p" tree revisited: Question C

Log likelihood of data after applying question C is:

$\log_2 L(x^1, \ldots, x^{12} | Q_C) =$

$2\log_2 2 + 2\log_2 2 + 2\log_2 2 + 2\log_2 2 + 4\log_2 4 - 4\log_2 4 - 8\log_2 8 = -16.00$

Average entropy of data after applying question C is

$H(x^1, \ldots, x^{12} | Q_C) = -1/n \log_2 L(x^1, \ldots, x^{12} | Q_C) = 16/12 = 1.33$ bits

Increase in log likelihood do to question C is -16 + 19.02 = 3.02
Decrease in entropy due to question C is 1.58-1.33 = .25 bits

Knowing the answer to question C provides 0.25 bits of information about the pronunciation of p.

Comparison of Questions A, B, C

- Log likelihood of data given question:
  - A -10.51.
  - B -18.51.
  - C -16.00.
- Average entropy (bits) of data given question:
  - A 0.87.
  - B 1.54.
  - C 1.33.
- Gain in information (in bits) due to question:
  - A 0.71.
  - B 0.04.
  - C 0.25.
- These measures all say the same thing:
  - Question A is best. Question C is 2nd best. Question B is worst.

Using Decision Trees to Model Context Dependence in HMMs

- Remember that the pronunciation of a phone depends on its context.
- Enumeration of all triphones is one option but has problems
- Idea is to use decision trees to find set of equivalence classes

Using Decision Trees to Model Context Dependence in HMMs

- Align training data (feature vectors) against set of phonetic-based HMMs
- For each feature vector, tag it with ID of current phone and the phones to left and right.
- For each phone, create a decision tree by asking questions about the phones on left and right to maximize likelihood of data.
- Leaves of tree represent context dependent models for that phone.
- During training and recognition, you know the phone and its context so no problem in identifying the context-dependent models on the fly.

New Problem: dealing with real-valued data

- We grow the tree so as to maximize the likelihood of the training data (as always), but now the training data are real-valued vectors.
- Can't use the multinomial distribution we used for the spelling-to-sound example,
- instead, estimate the likelihood of the acoustic vectors during tree construction using a diagonal Gaussian model.

Diagonal Gaussian Likelihood

Let $Y = y_1, y_2 \ldots, y_n$ be a sample of independent p-dimensional acoustic vectors arising from a diagonal Gaussian distribution with mean $\vec{\mu}$ and variances $\vec{\sigma^2}$. Then

$$\log L(Y|DG(\vec{\mu}, \vec{\sigma_2})) = \tfrac{1}{2} \sum_{i=1}^{n} \{ p \log 2\pi + \sum_{j=1}^{p} \log \sigma_j^2 + \sum_{j=1}^{p} (y_{ij} - \mu_j)^2 / \sigma_j^2 \}$$

The maximum likelihood estimates of $\vec{\mu}$ and $\vec{\sigma^2}$ are

$$\hat{\mu}_j = 1/n \sum_{i=1}^{n} y_{ij}, j = 1, \ldots, p$$

$$\hat{\sigma_j^2} = 1/n \sum_{i=1}^{n} y_{ij}^2 - \mu_j^2, j = 1, \ldots p$$

Hence, an estimate of log L(Y) is:

$$\log L(Y|DG(\vec{\mu}, \vec{\sigma_2})) = 1/2 \sum_{i=1}^{n} \{ p \log 2\pi + \sum_{j=1}^{p} \log \hat{\sigma_j^2} + \sum_{j=1}^{p} (y_{ij} - \hat{\mu}_j)^2 / \hat{\sigma_j^2} \}$$

Diagonal Gaussian Likelihood

Now

$$\sum_{i=1}^{n}\sum_{j=1}^{p}(y_{ij} - \hat{\mu}_j)^2/\hat{\sigma}_j^2 = \sum_{j=1}^{p}\frac{1}{\hat{\sigma}_j^2}\sum_{i=1}^{n}(y_{ij}^2 - 2\hat{\mu}_j\sum_{i=1}^{n}y_{ij} + n\hat{\mu}_j^2)$$

$$= \sum_{j=1}^{p}\frac{1}{\hat{\sigma}_j^2}\left\{(\sum_{i=1}^{n}y_{ij}^2) - n\hat{\mu}_j^2\right\}$$

$$= \sum_{j=1}^{p}\frac{1}{\hat{\sigma}_j^2}n\hat{\sigma}_j^2 = \sum_{j=1}^{p}n$$

Hence

$$\log L(Y|DG(\hat{\mu},\hat{\sigma}^2)) = -1/2\{\sum_{i=1}^{n}p\log 2\pi + \sum_{i=1}^{n}\sum_{j=1}^{p}\hat{\sigma}_j^2 + \sum_{j=1}^{p}n\}$$

$$= -1/2\{np\log 2\pi + n\sum_{j=1}^{p}\hat{\sigma}_j^2 + np\}$$

Diagonal Gaussian Splits

- Let Q be a question which partitions Y into left and right sub-samples Y_l and Y_r, of size n_l and n_r.
- The best question is the one which maximizes
  $\log L(Y_l) + \log L(Y_r)$
- Using a diagonal Gaussian model.

Common to all splits

$$\log L(Y_l \mid DG(\hat{\mu}_l, \hat{\sigma}_l^2)) + \log L(Y_r \mid DG(\hat{\mu}_r, \hat{\sigma}_r^2))$$

$$= -\frac{1}{2}\{n_l p \log(2\pi) + n_l \sum_{j=1}^{p}\log\hat{\sigma}_{lj}^2 + n_l p$$

$$+ n_r p \log(2\pi) + n_r \sum_{j=1}^{p}\log\hat{\sigma}_{rj}^2 + n_r p\}$$

$$= -\frac{1}{2}\{np\log(2\pi) + np\} - \frac{1}{2}\{n_l \sum_{j=1}^{p}\log\hat{\sigma}_{lj}^2 + n_r \sum_{j=1}^{p}\log\hat{\sigma}_{rj}^2\}$$

Diagonal Gaussian Splits, cont'd

Thus, the best question Q minimizes:

$$D_Q = n_l \sum_{j=1}^{p}\log\hat{\sigma}_{lj}^2 + n_r \sum_{j=1}^{p}\log\hat{\sigma}_{rj}^2$$

Where

$$\hat{\sigma}_{lj}^2 = 1/n_l \sum_{y\in Y_l}y_j^2 - 1/n_l^2(\sum_{y\in Y_l}y_j^2)$$

$$\hat{\sigma}_{rj}^2 = 1/n_r \sum_{y\in Y_r}y_j^2 - 1/n_r^2(\sum_{y\in Y_r}y_j^2)$$

D_Q involves little more than summing vector elements and their squares.

How Big a Tree?

- CART suggests cross-validation.
  - Measure performance on a held-out data set.
  - Choose the tree size that maximizes the likelihood of the held-out data.
- In practice, simple heuristics seem to work well.
- A decision tree is fully grown when no terminal node can be split.
- Reasons for not splitting a node include:
  - Insufficient data for accurate question evaluation.
  - Best question was not very helpful / did not improve the likelihood significantly.
  - Cannot cope with any more nodes due to CPU/memory limitations.

Recap

- Given a word sequence, we can construct the corresponding Markov model by:
  - Re-writing word string as a sequence of phonemes.
  - Concatenating phonetic models.
  - Using the appropriate tree for each phone to determine which allophone (leaf) is to be used in that context.
- In actuality, we make models for the HMM arcs themselves
  - Follow same process as with phones - align data against the arcs
  - Tag each feature vector with its arc id and phonetic context
  - Create decision tree for each arc.

Example

The rain in Spain falls ....

Look these words up in the dictionary to get:

DH AX | R EY N | IX N | S P EY N | F AA L Z | ...

Rewrite phones as states according to phonetic model

$DH_1 DH_2 DH_3 AX_1 AX_2 AX_3 R_1 R_2 R_3 EY_1 EY_2 EY_3$ ...

Using phonetic context, descend decision tree to find leaf sequences

$DH_{1\_5} DH_{2\_27} DH_{3\_14}\ AX_{1\_53} AX_{2\_37} AX_{3\_11} R_{1\_42} R_{2\_46}$ ....

Use the Gaussian mixture model for the appropriate leaf as the observation probabilities for each state in the Hidden Markov Model.

Some Results

System	T1	T2	T3	T4
Monophone	5.7	7.3	6.0	9.7
Triphone	3.7	4.6	4.2	7.0
Arc-Based DT	3.1	3.8	3.4	6.3

- From Julian Odell's PhD Thesis (Cambridge U., 1995)
- Word error rates on 4 test sets associated with 1000 word vocabulary (Resource Management) task

Strengths & Weaknesses of Decision Trees

- **Strengths**.
  - Easy to generate; simple algorithm.
  - Relatively fast to construct.
  - Classification is very fast.
  - Can achieve good performance on many tasks.
- **Weaknesses**.
  - Not always sufficient to learn complex concepts.
  - Can be hard to interpret. Real problems can produce large trees...
  - Some problems with continuously valued attributes may not be easily discretized.
  - Data fragmentation.

Course Feedback

- Was this lecture mostly clear or unclear?
- What was the muddiest topic?
- Other feedback (pace, content, atmosphere, etc.).