Lecture 6

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

Where Are We?

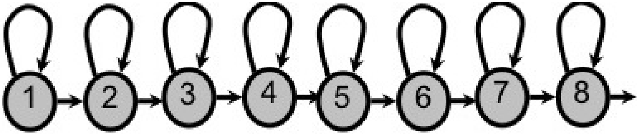
Where Are We?

In the beginning...

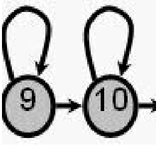
- ... . was the whole word model.
- For each word in the vocabulary, decide on a topology.
- Often the number of states in the model is chosen to be proportional to the number of phonemes in the word.
- Train the observation and transition parameters for a given word using examples of that word in the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

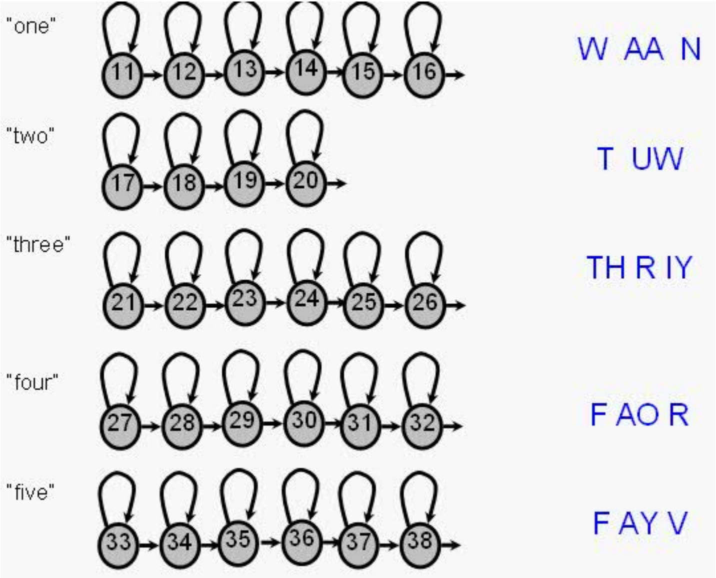
Example topologies: Digits

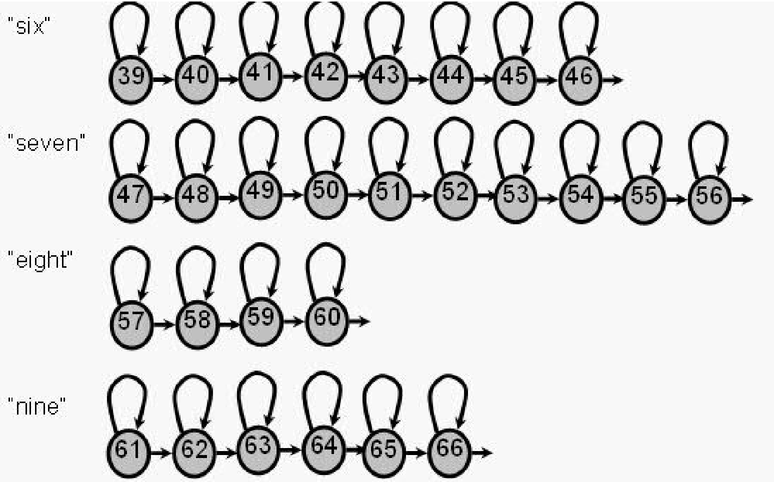
- Vocabulary consists of ("zero", "oh", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine").
- Assume we assign two states per phoneme.
- Must allow for different durations
- Models look like:
- "zero".



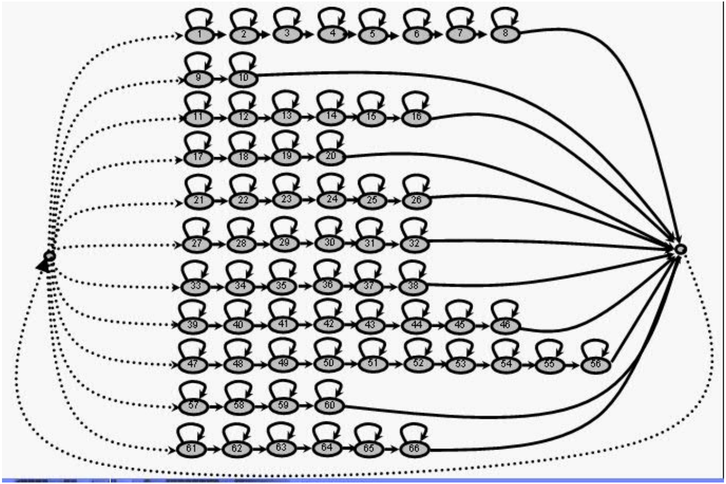
- "oh".



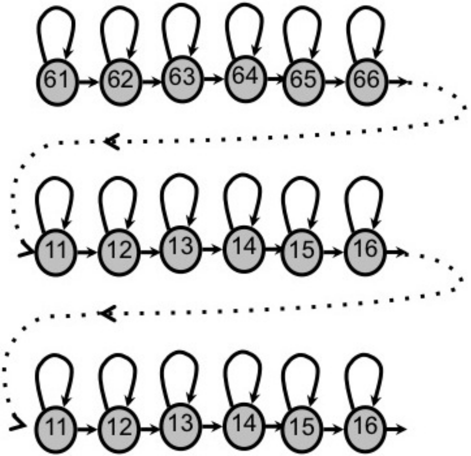

"one"
11
12
13
14
15
16
W AA N
"two"
17
18
19
20
T UW
"three"
21
22
23
24
25
26
TH R IY
"four"
27
28
29
30
31
32
F AO R
"five"
33
34
35
36
37
38
F AY V


"six"
39
40
41
42
43
44
45
46
"seven"
47
48
49
50
51
52
53
54
55
56
"eight"
57
58
59
60
"nine"
61
62
63
64
65
66

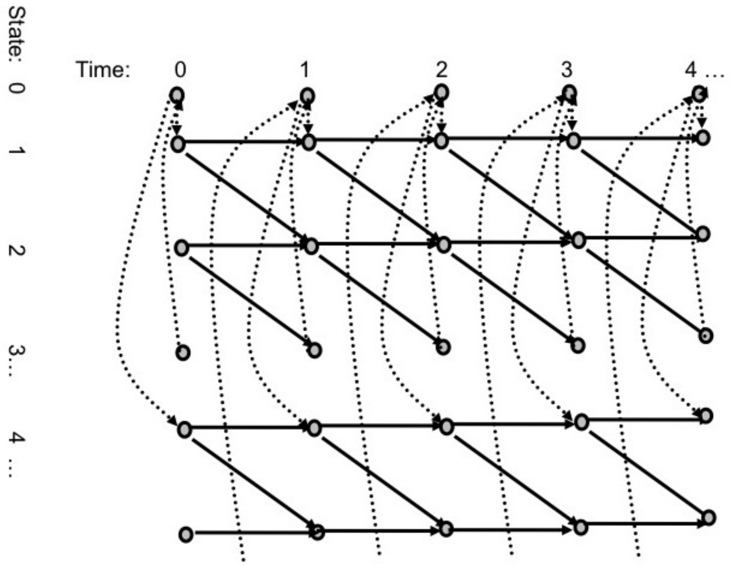
How to represent any sequence of digits?



“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.

Where Are We?

1. HMM Structures
   - Whole Word Models
   - Phonetic Models
   - Context-Dependence
   - Triphone Models

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.

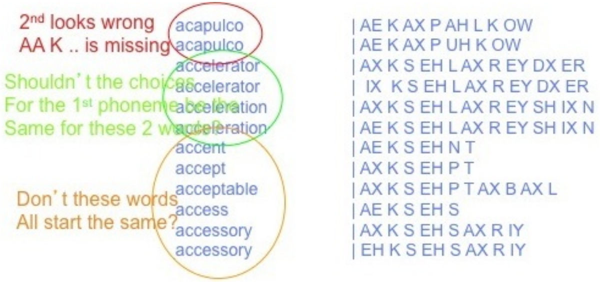
BANDS -> B AE N D Z

- Some words need more than one baseform.

THE -> DH UH
-> DH IY

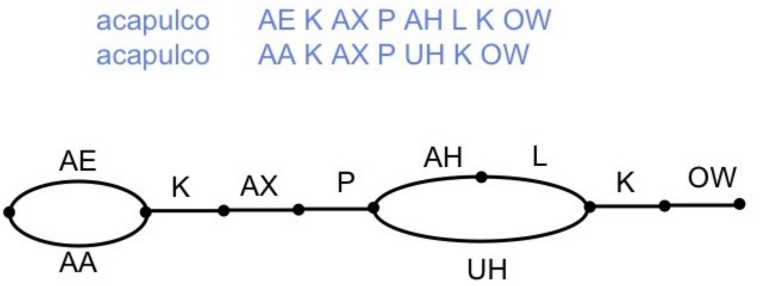
Baseform Dictionary

- 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.



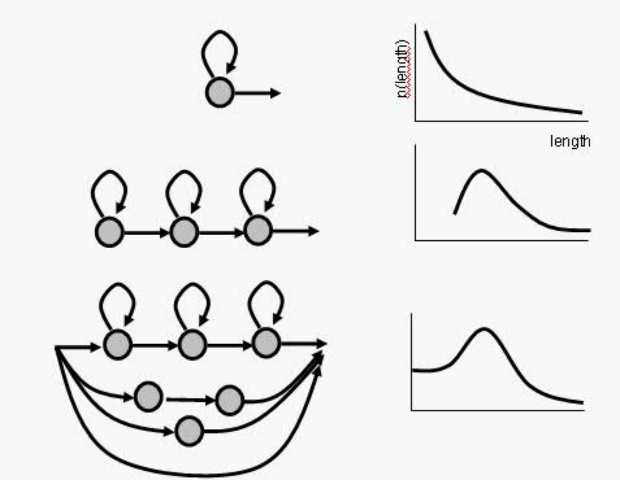
Phonetic Models, cont'd

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



Phonetic Models, cont'd

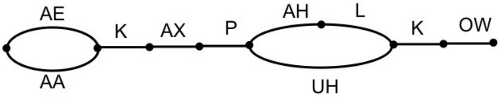
- Now, construct a Markov model for each phone.
- Examples:



Embedding

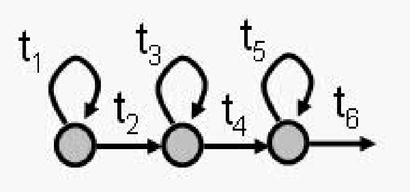
- Replace each phone by its Markov model to get a word model.
- N.b. The model for each phone will have different parameter values.

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

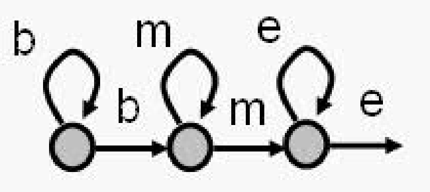
- Now consider phone-based models for this vocabulary.

0	Z IY R OW
1	W AA N
2	T UW
3	TH R IY
4	F AO R
5	F AY V
6	S IH K S
7	S EH V AX N
8	EY T
9	N AY N

- 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:



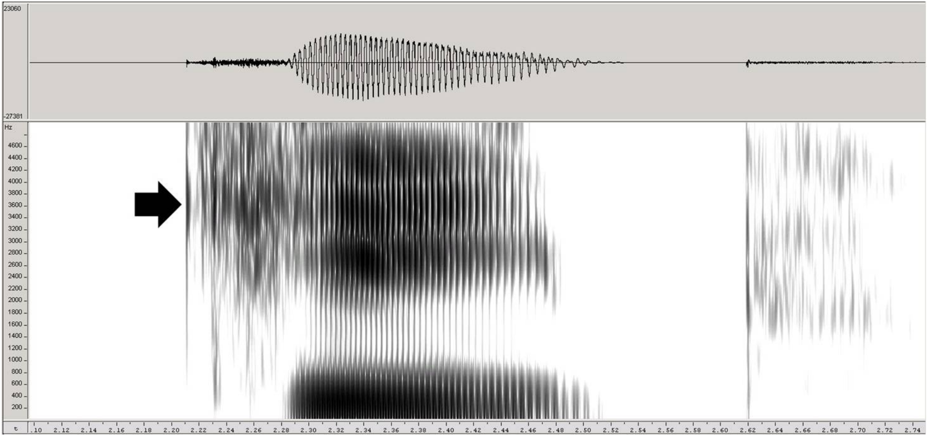
- 6 non-null arcs, but only 3 different output distributions because of tying.
- Number of model parameters is reduced.
- Tying saves storage because only one copy of each distribution is saved.
- Fewer parameters mean less training data needed.

Where Are We?

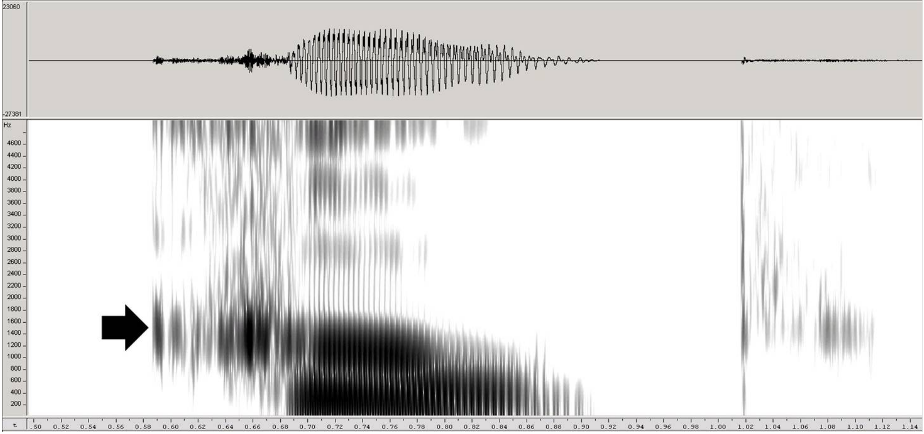
1. HMM Structures
   - Whole Word Models
   - Phonetic Models
   - Context-Dependence
   - Triphone Models

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.


23060
-27381
Hz
4600
4400
4200
4000
3800
3600
3400
3200
3000
2800
2600
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
t
.10 2.12 2.14 2.16 2.18 2.20 2.22 2.24 2.26 2.28 2.30 2.32 2.34 2.36 2.38 2.40 2.42 2.44 2.46 2.48 2.50 2.52 2.54 2.56 2.58 2.60 2.62 2.64 2.66 2.68 2.70 2.72 2.74

keep


23060
-27381
Hz
4600
4400
4200
4000
3800
3600
3400
3200
3000
2800
2600
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
t
.50 0.52 0.54 0.56 0.58 0.60 0.62 0.64 0.66 0.68 0.70 0.72 0.74 0.76 0.78 0.80 0.82 0.84 0.86 0.88 0.90 0.92 0.94 0.96 0.98 1.00 1.02 1.04 1.06 1.08 1.10 1.12 1.14

coop

Where Are We?

1. HMM Structures
   - Whole Word Models
   - Phonetic Models
   - Context-Dependence
   - Triphone Models

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.
  - K IY P $\rightarrow$ |-K+IY   K-IY+P   IY-P+|
- If we have 50 phonemes in a language, we could have as many as 50^3 triphones to model.
- Not all of these occur, or only occur a few times. Why is this bad?

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.
  - K IY P $\rightarrow$ |-K+IY K-IY+P IY-P+|
- If we have 50 phonemes in a language, we could have as many as 50^3 triphones to model.
- Not all of these occur, or only occur a few times. Why is this bad?
- Bad because of data sparsity issues. How can we solve this?

Triphone models

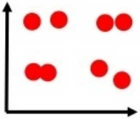
- Model each phoneme in the context of its left and right neighbor.
- E.g. K-IY+P is a model for IY when K is its left context phoneme and P is its right context phoneme.
  - K IY P $\rightarrow$ |-K+IY   K-IY+P   IY-P+|
- If we have 50 phonemes in a language, we could have as many as 50^3 triphones to model.
- Not all of these occur, or only occur a few times. Why is this bad?
- Bad because of data sparsity issues. How can we solve this?
- One Solution: Cluster triphones together in bottom-up fashion
  - For example, map K-IY+P and K-IY+F to common triphone model

"Bottom-up" (Agglomerative) 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) $\forall$ a$\in$A, b$\in$B.
    - Complete-link: dist(A,B) = max dist(a,b) $\forall$ a$\in$A, b$\in$B.
    - Centrod-based: dist(A,B) = dist($\frac{1}{N_A}\sum_{\forall a\in A} a, \frac{1}{N_B}\sum_{\forall b\in B} b$)

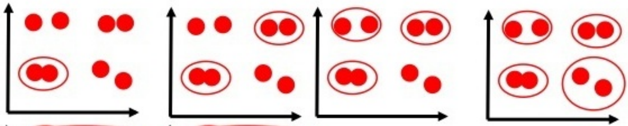
Bottom-up clustering / Single Link

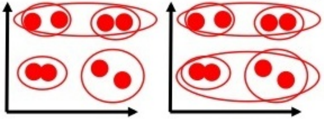
- Assume our data points look like:



Single-link: clusters are close if any of their points are:
dist(A,B) = min dist(a,b) for $a \in A$, $b \in B$

- Single-link clustering into 2 groups proceeds as:

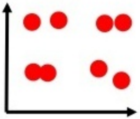




Single-link clustering tends to yield long, stringy, meandering clusters

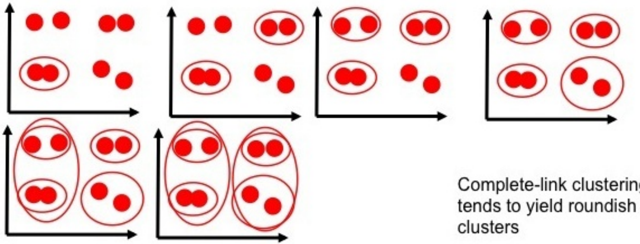
Bottom-up clustering / Complete Link

- 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

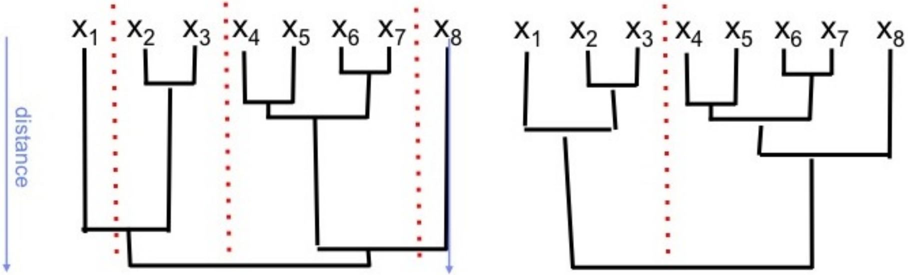
- Single-link clustering into 2 groups proceeds as:



Complete-link clustering tends to yield roundish clusters

Number of Clusters?

- 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

- How can we characterize a triphone for clustering purposes?
- Helps with data sparsity issue
- BUT still have an issue with unseen data
- To model unseen events, we can “back-off” to lower order models such as bi-phones and uni-phones. But this is still sort of ugly.

Triphone Clustering

- How can we characterize a triphone for clustering purposes?
- Helps with data sparsity issue
- BUT still have an issue with unseen data
- To model unseen events, we can “back-off” to lower order models such as bi-phones and uni-phones. But this is still sort of ugly.

Triphone Clustering

- How can we characterize a triphone for clustering purposes?
- Helps with data sparsity issue
- BUT still have an issue with unseen data
- To model unseen events, we can “back-off” to lower order models such as bi-phones and uni-phones. But this is still sort of ugly.

Where Are We?

Where Are We?

2 Context Dependence via Decision Trees

- Decision Tree Overview
- Letter-to-Sound Example
- Basics of Tree Construction
- Criterion Function
- Details of Context Dependent Modeling

Decision Trees

- Assume we have a set of data and that each data element is tagged with a set of input variables
  - In speech, the data can be acoustic feature vectors tagged with the identity of the underlying phone and the surrounding phones.
- A decision tree maps the tagged data into a set of 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 a vector of real numbers - 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

- Assume we have a set of data and that each data element is tagged with a set of input variables
  - In speech, the data can be acoustic feature vectors tagged with the identity of the underlying phone and the surrounding phones.
- A decision tree maps the tagged data into a set of 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 a vector of real numbers - 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

- Assume we have a set of data and that each data element is tagged with a set of input variables
  - In speech, the data can be acoustic feature vectors tagged with the identity of the underlying phone and the surrounding phones.
- A decision tree maps the tagged data into a set of 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 a vector of real numbers - 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

- Assume we have a set of data and that each data element is tagged with a set of input variables
  - In speech, the data can be acoustic feature vectors tagged with the identity of the underlying phone and the surrounding phones.
- A decision tree maps the tagged data into a set of 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 a vector of real numbers - criteria could be mean-square error
- The goal when constructing a decision tree is significantly improve the criterion function (relative to doing nothing)

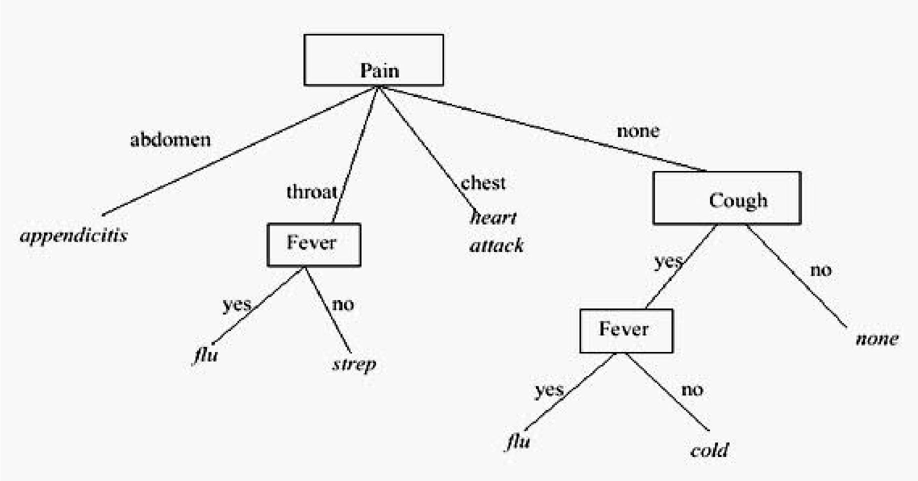
Decision Trees - A Form of Top-Down Clustering

- 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.

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 "traditional" decision tree look like?



Types of Input Attributes/Features

- 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).
- Numerical: Domain is numerically ordered (e.g., age, income).

The Classification Problem

- If the output variable is categorical, the problem is a called 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 called 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 Requirements/Properties
  - High accuracy.
  - Understandable by humans, interpretable.
  - Fast construction for very large training databases.
- Shallow trees MAY be understandable.
- For speech recognition, we built deep trees and understandibility quickly goes out the window....

Where Are We?

Decision Trees: Letter-to-Sound Example

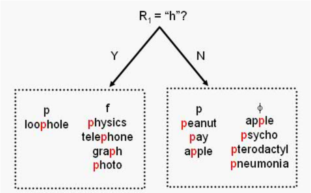
- Let's say we want to build a tree to decide how the letter "p" will sound in various words.
- Training examples:

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

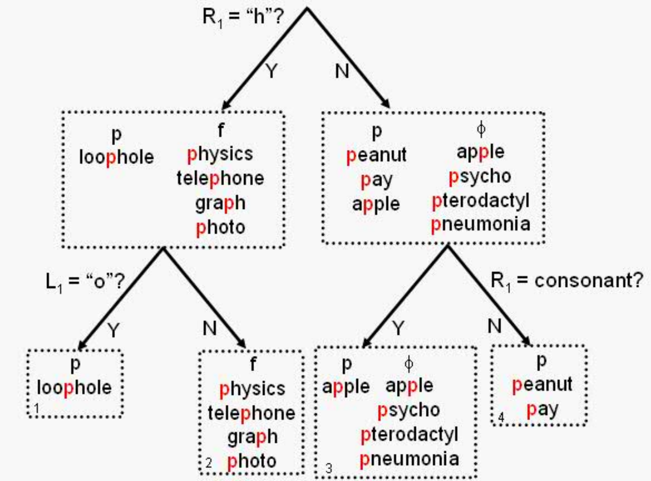
- The pronunciation of "p" depends on its letter context.
- Task: Using the above training data, partition the contexts into equivalence classes so as to minimize the uncertainty of the pronunciation.

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

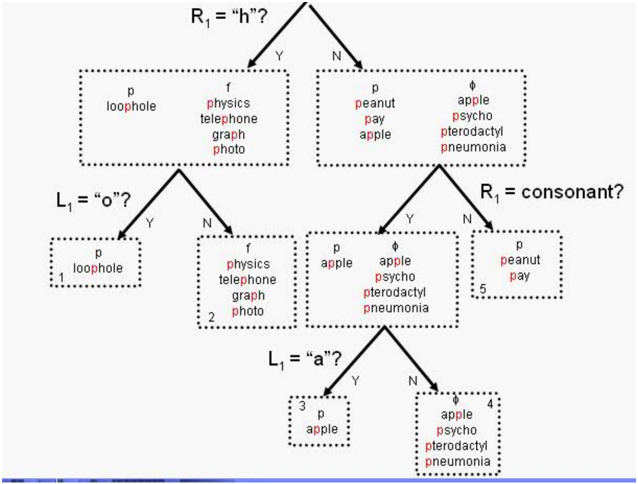
- Denote the context as $\ldots L_2\ L_1\ \ \text{p}\ \ R_1\ R_2 \ldots$
- Ask potentially useful question: $R_1 =$ "h"?
- At this point we have two equivalence classes: 1. $R_1 =$ "h" and 2. $R_1 \neq$ "h".



- 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"



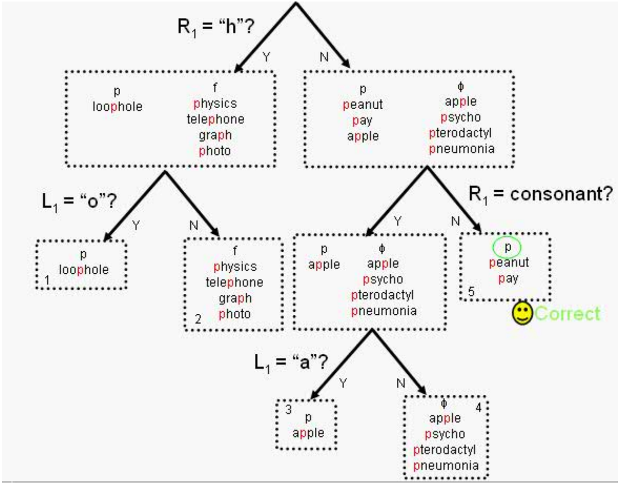
Four equivalence classes. Uncertainty only remains in class 3.



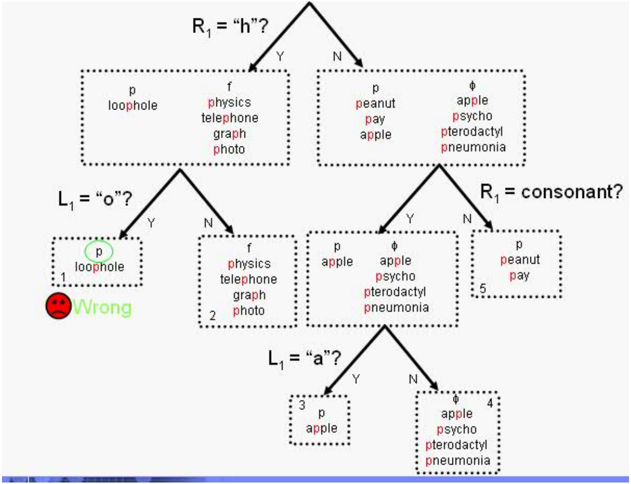
Five equivalence classes, which is much less than enumerating each of the possibilities.
No uncertainty left in the classes.

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.

Where Are We?

2. Context Dependence via Decision Trees
   - Decision Tree Overview
   - Letter-to-Sound Example
   - Basics of Tree Construction
   - Criterion Function
   - Details of Context Dependent Modeling

Decision Tree Construction

- Previous example - picked questions "out of the air"
- Need more principled way to chose questions

Decision Tree Construction

- Previous example - picked questions "out of the air"
- Need more principled way to chose questions

How to Grow a Tree

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

- Selection of the splits.
- 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 a node of the tree and split it (and the data at the node) 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

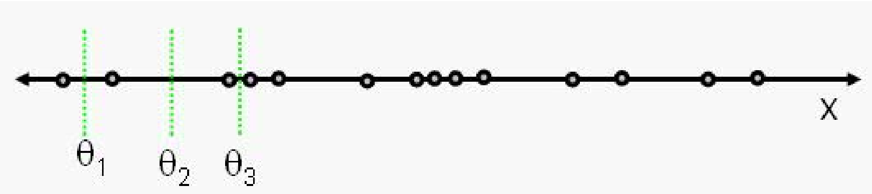
Phones	**Letters**
{P}	{A}
{T}	{E}
{K}	{I}
{B}	{O}
{D}	{U}
{G}	{Y}
{P,T,K}	{A,E,I,O,U}
{B,D,G}	{A,E,I,O,U,Y}
{P,T,K,B,D,G}	

More Formally - 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

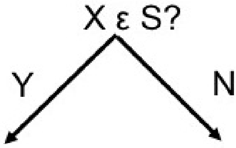
- In principle, a question asked in a decision tree can have any number (greater than 1) of possible outcomes.
- Examples:
  - Binary: Yes No.
  - 3 Outcomes: Yes No Don't_Know.
  - 26 Outcomes A B C ... Z

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.

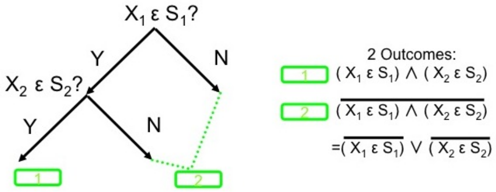
Simple Binary Question

- 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.

Where Are We?

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 - Detailed Recap

- 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.
- Goal: Find Q such that $L(\text{data}_{\text{left}}) \times L(\text{data}_{\text{right}})$ is maximized.

Question Evaluation, cont'd

- Let each discrete variable x_i have a set of M_i possible outcomes.
- Let $x_i^1, x_i^2, \ldots, x_i^N$ be the data samples for x_i
- Let each of the M_i outcomes occur c_i times in the overall sample
- Let Q_i be a question which partitions this sample into left and right sub-samples of size n_l and n_r, respectively.
- Let c_{ij}^l, c_{ij}^r denote the frequency of the jth outcome in the left and right sub-samples.
- The best question Q_i' for x_i 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 (dropping "i" for convenience)

$$\log L(x^1, \ldots, x^n | Q) = \sum_{j=1}^{N} c_j^l \log p_j^l + \sum_{j=1}^{N} c_j^r \log p_j^r$$

- The above assumes we know the "true" probabilities p_j^l, p_j^r

log likelihood computation (continued)

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

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

- The best question is the one which maximizes this simple expression. c_j^l, c_j^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

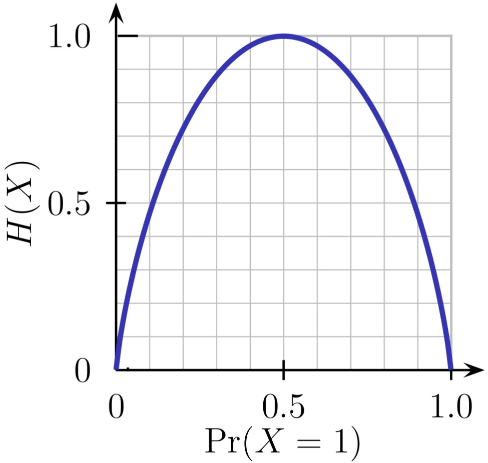
- Let x be a discrete random variable taking values $a_1, \ldots, a_N$ in an alphabet A of size N with probabilities $p_1, \ldots, p_N$ respectively.
- The *uncertainty* about what value x will take can be measured by the entropy of the probability distribution $p = (p_1 p_2 \ldots p_N)$

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i$$

$$H = 0 \Leftrightarrow p_j = 1 \text{ for some } j \text{ and } p_i = 0 \text{ for } i \neq j$$

- $H >= 0$
- 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?



Entropy and Likelihood

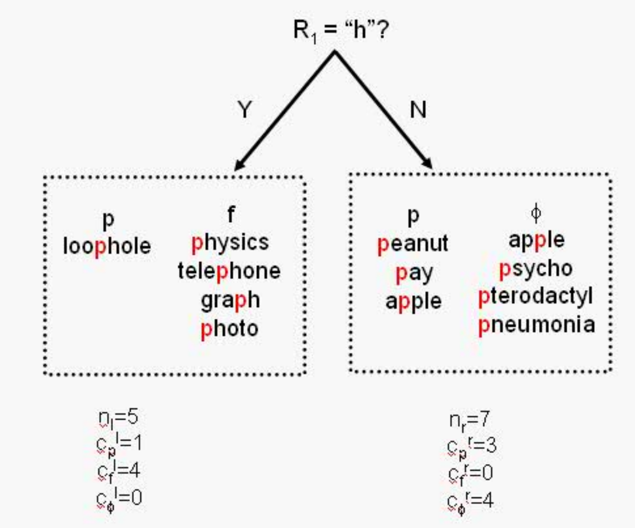
- Let x be a discrete random variable taking values $a_1, \ldots\ a_N$ in an alphabet A of size N with probabilities $p_1, \ldots, p_N$ respectively.
- Let $x^1, \ldots, x^n$ be a sample of x in which a_i occurs c_i times
- The sample log likelihood is: $\log L = \sum_{i=1}^{n} c_i \log p_i$
- The maximum 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:

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

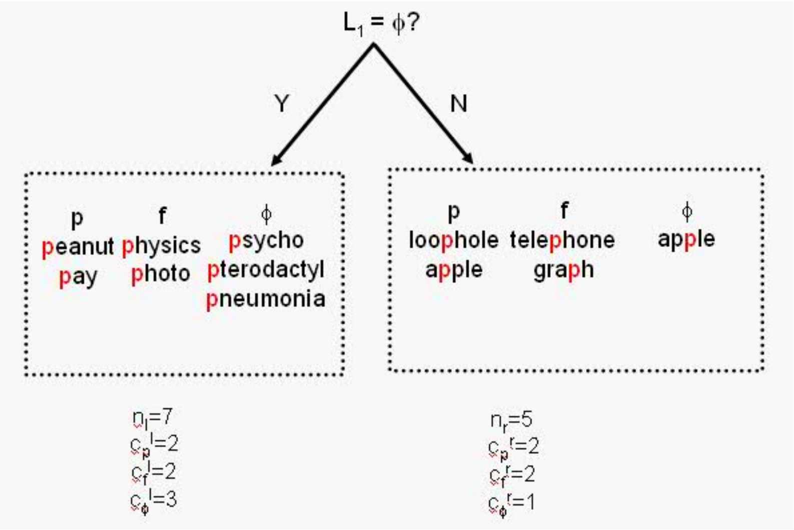
Average entropy of data after applying question A is

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

Increase in log likelihood due 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:

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

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

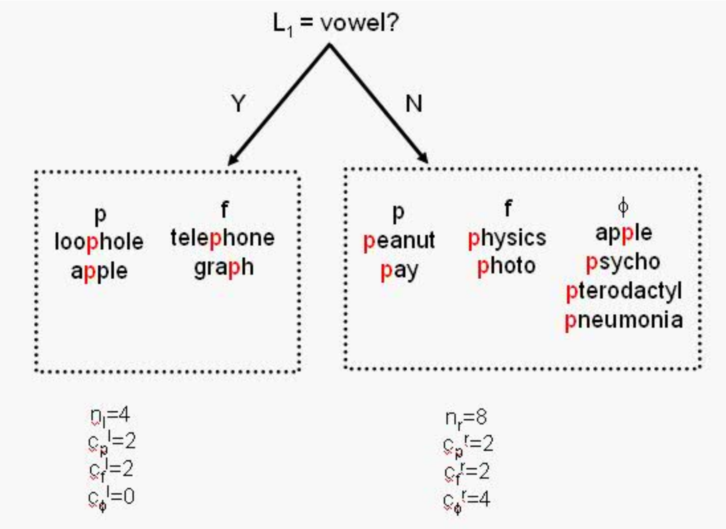
Average entropy of data after applying question B is

$$H(x^1, \ldots, x^{12}|Q_B) = -1/n \log_2 L(x^1, \ldots, x^{12}|Q_B) = 18.51/12 = .87 \text{ bits}$$

Increase in log likelihood due 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

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

Increase in log likelihood due 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.

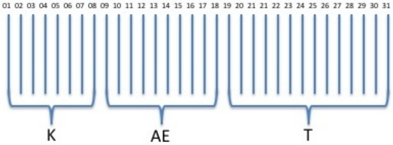
Where Are We?

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.



	L_1	C	R_1
07:	|	K	AE
08:	|	K	AE
09:	K	AE	T
10:	K	AE	T
18:	K	AE	T
19:	AE	T	|

Using Decision Trees to Model Context Dependence in HMMs

- 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

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

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 $\log L(Y_l) + \ logL(Y_r)$
- Using a diagonal Gaussian model.

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

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

Common to all splits

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

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

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.

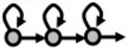
Example

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.).