Pronunciation Modeling for Large Vocabulary Continuous Speech Recognition

Stanley Chen, Michael Picheny, and Ellen Eide
ELEN E6885 Fall 2005 Lecture 6

Word Error Rate

- How do we measure the performance of an ASR system?
- Define WER = (substitutions + deletions+ insertions) / (number of words in reference script)
- Example:

  deletion

  ref: The dog is here now

  hyp: The uh bog is now

  insertion substitution
- Compute WER efficiently using dynamic programming (DTW)
- Can WER be above 100% ?

Model Order

- Should we use big or small models?
  e.g. 3-gram or 5-gram?
  With smaller models, less sparse data issues → better probability estimates?
  - Empirically, bigger is better
  - With best smoothing, little or no performance degradation if model is too large
  - With lots of data (100M words +) significant gain from 5-gram

  Limiting resource: disk/memory
  Count cutoffs can be used to reduce the size of the LM
  Discard all n-grams with count less than threshold

Administrivia

- Main feedback from last lecture
  - more class discussion would be good
  - trellis for held-out estimation unclear
- Labs
  - mixed feedback on difficulty
  - want more documentation
- Lab 1 handed back today
- Lab 2 extension; now due 10/24 at 12:01am

Evaluating Language Models

- Best way: plug into ASR system, see how LM affects WER

  Expensive to compute

- Is there something cheaper that predicts WER well?

  “perplexity” (PP) of test data (only needs text)

  Doesn’t always predict WER well, but has theoretical significance

  Predicts best when 2 LM’s being compared are trained on same data

Perplexity

- Perplexity is average branching factor, i.e. how many alternatives the LM believes there are following each word
- Another interpretation: $\log_2 PP$ is the average number of bits per word needed to encode the test data using the model P( )

- Ask a speech recognizer to recognize digits: 0,1,2,3,4,5,6,7,8,9 simple task (?) perplexity = 10
- Ask a speech recognizer to recognize alphabet: a,b,c,d,e,…z
  more complex task … perplexity = 26
  alpha, bravo, charlie … yankee, zulu
  perplexity = 26

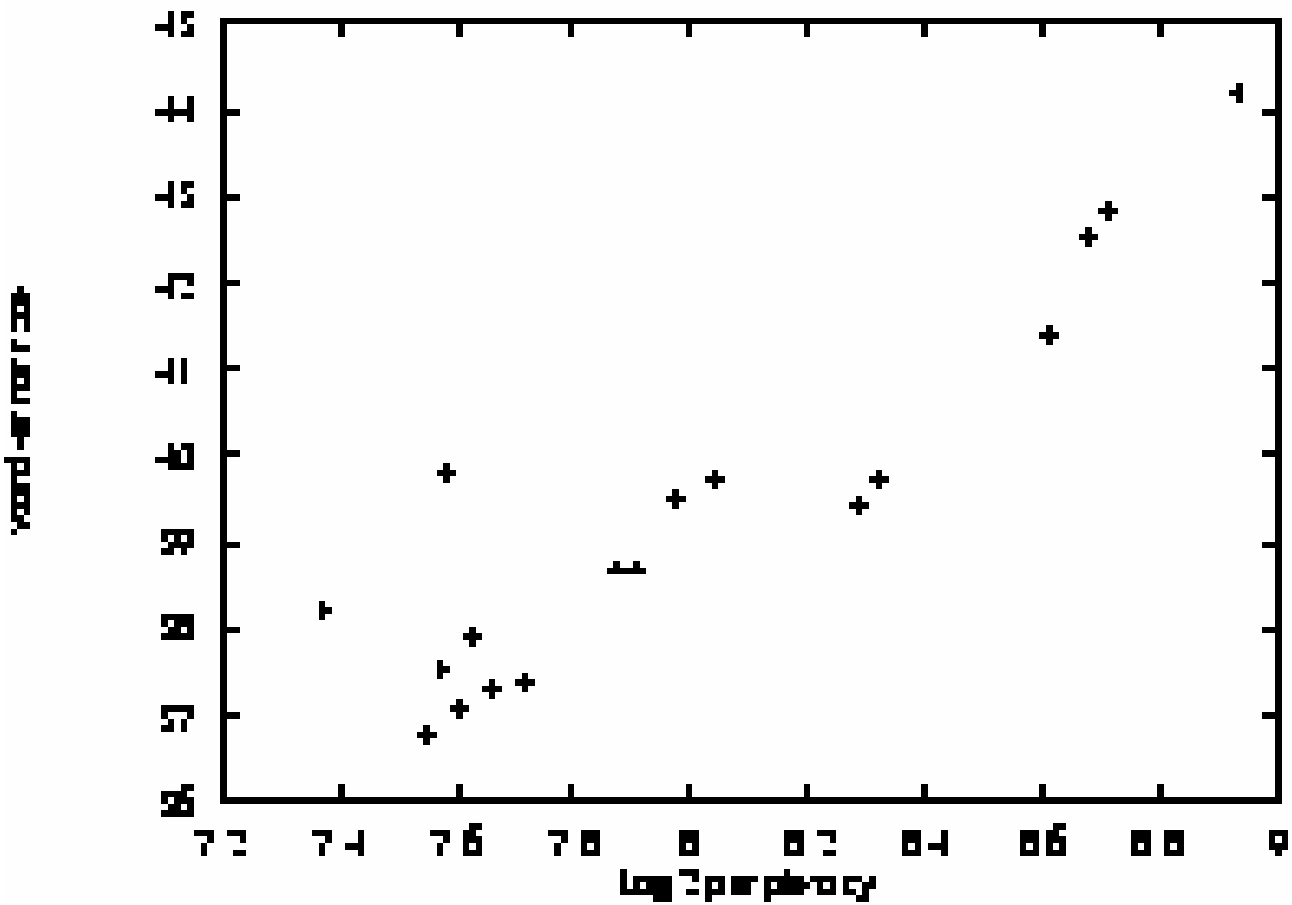
Perplexity measures LM difficulty, not acoustic difficulty

Computing Perplexity

1. Compute the geometric average probability assigned to each word in test data $w_1..w_n$ by model P( )

$$p_{avg} = [\prod_{i=1}^{n} P(w_i \mid w_1 ... w_{i-1})]^{\frac{1}{n}}$$

2. Invert it: PP = $1/p_{avg}$

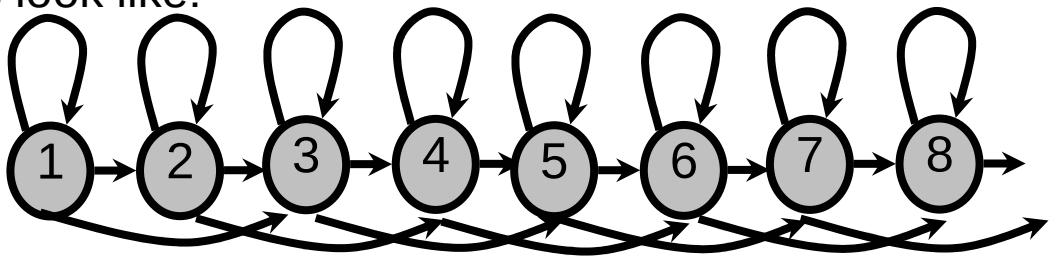


In the beginning…

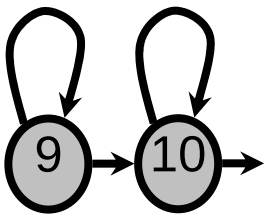
- …. was the whole word model
- For each word in the vocabulary, decide on a topology
- Often the number of states in the model is chosen to be proportional to the number of phonemes in the word
- Train the observation and transition parameters for a given word using examples of that word in the training data
  (Recall problem 3 associated with Hidden Markov Models)
- Good domain for this approach: digits

Example topologies: Digits

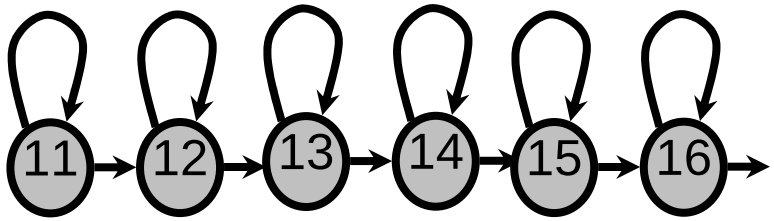
- Vocabulary consists of {“zero”, “oh”, “one”, “two”, “three”, “four”, “five”, “six”, “seven”, “eight”, “nine”}
- Assume we assign two states per phoneme
- Must allow for different durations – use self loops and skip arcs
- Models look like:
- “zero”



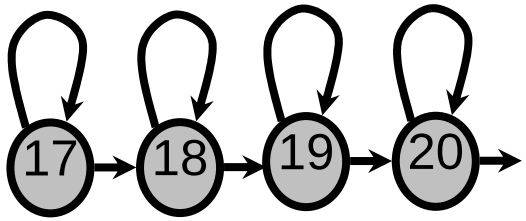
- “oh”



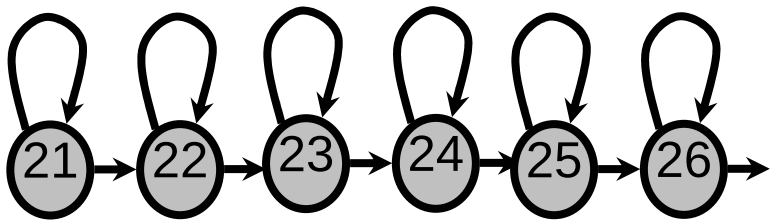
- “one”



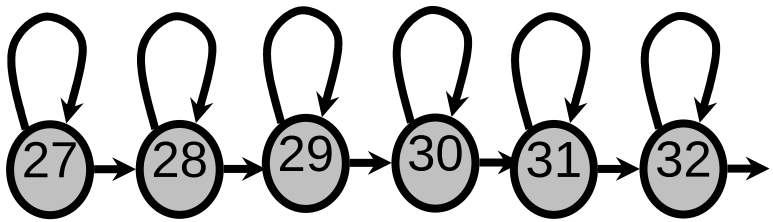
- “two”



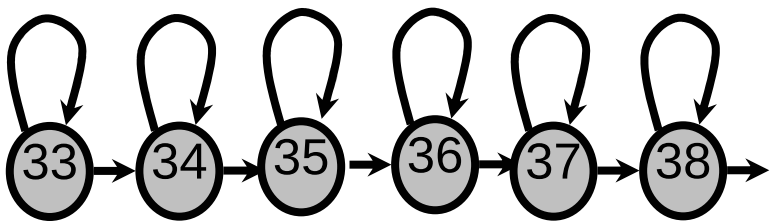
- “three”



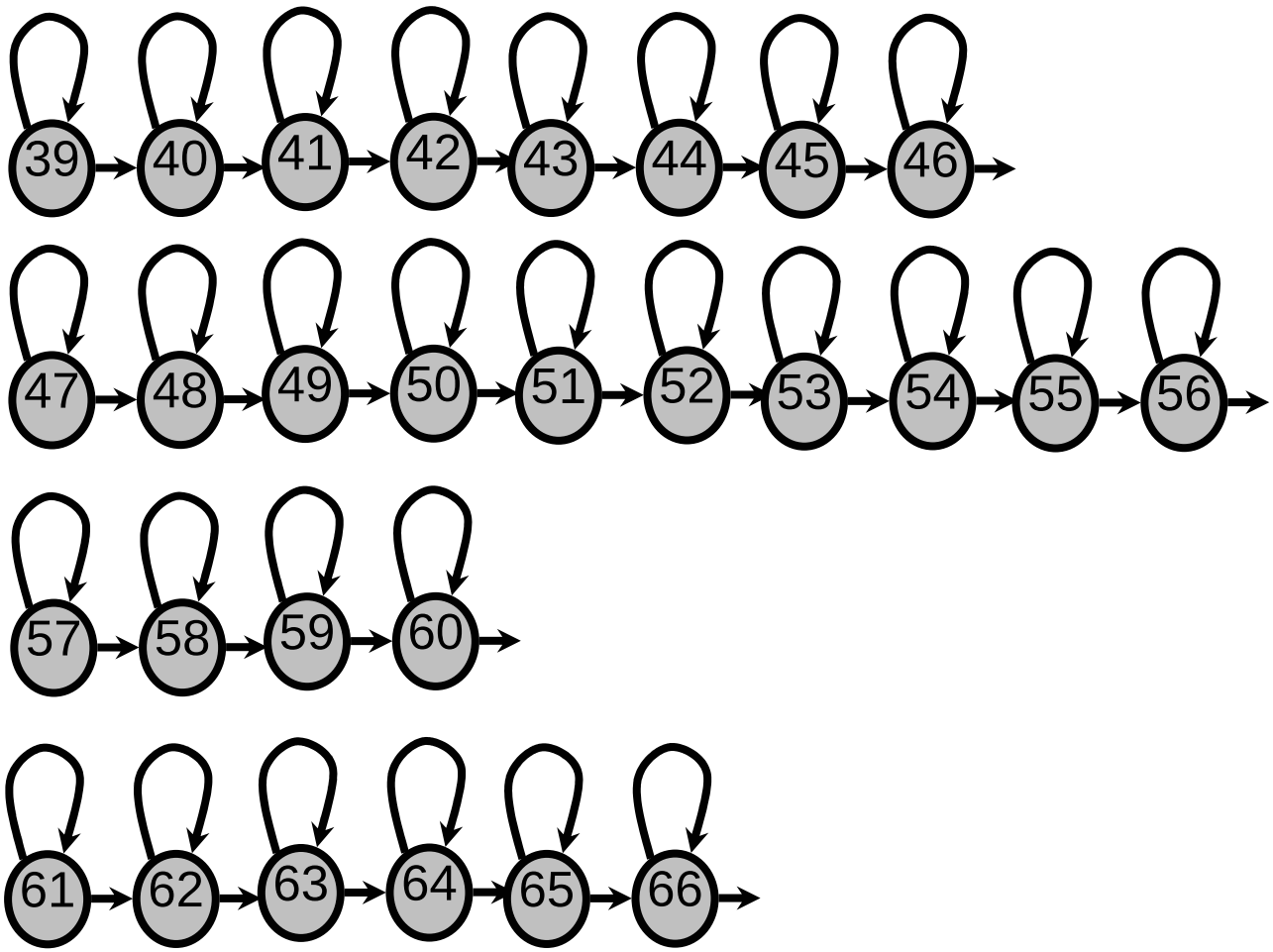
- “four”



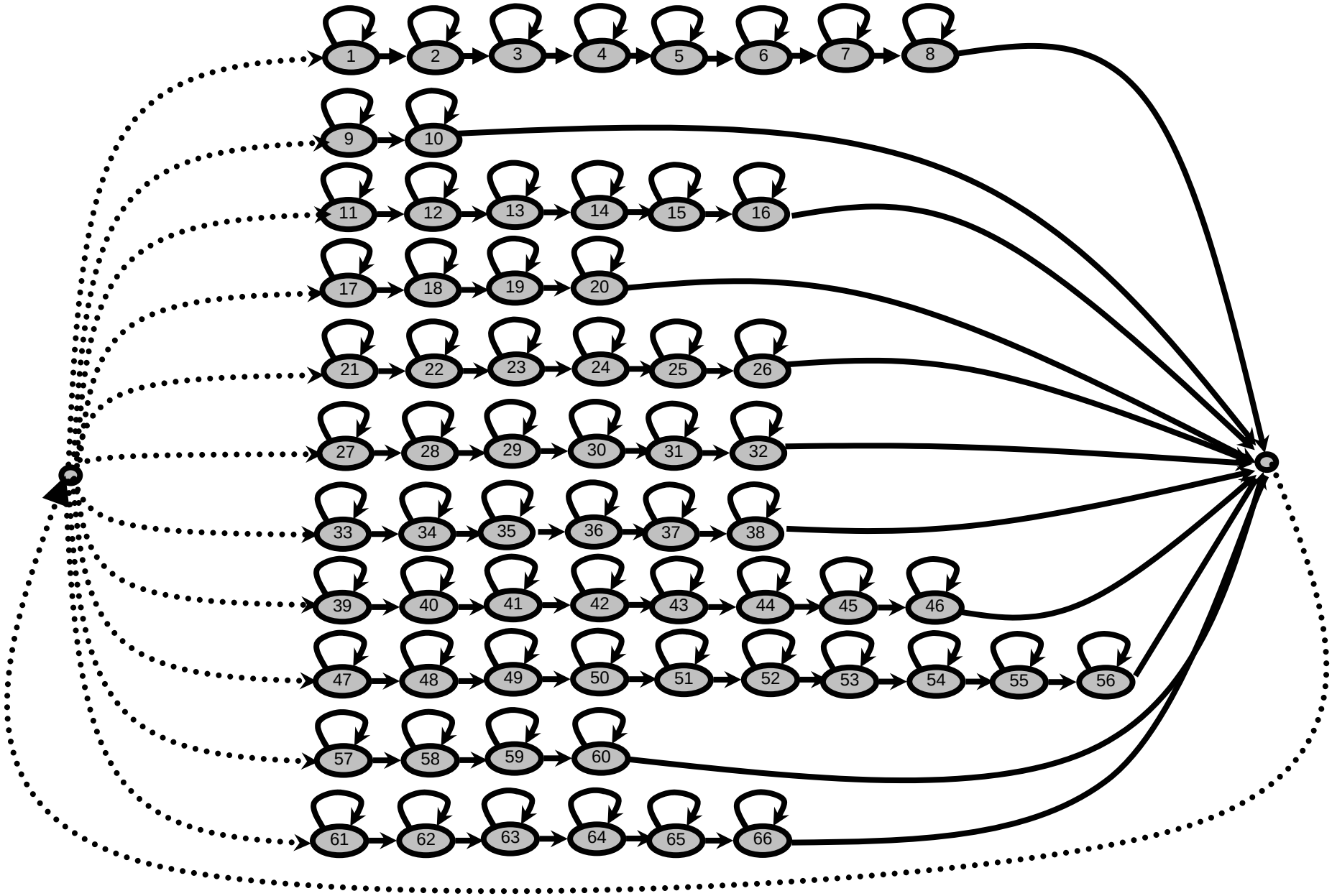
- “five”



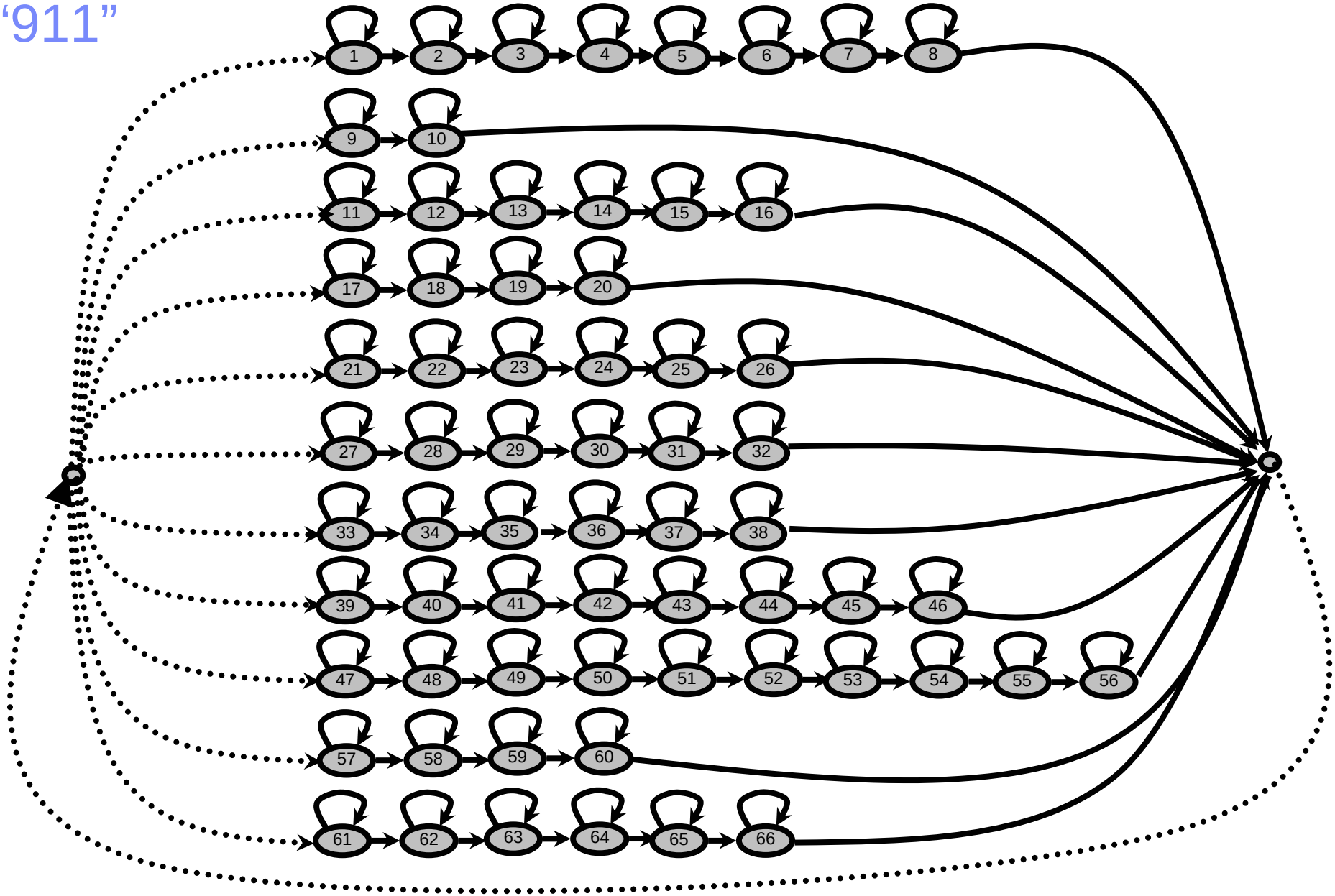
- “six”
- “seven”
- “eight”
- “nine”



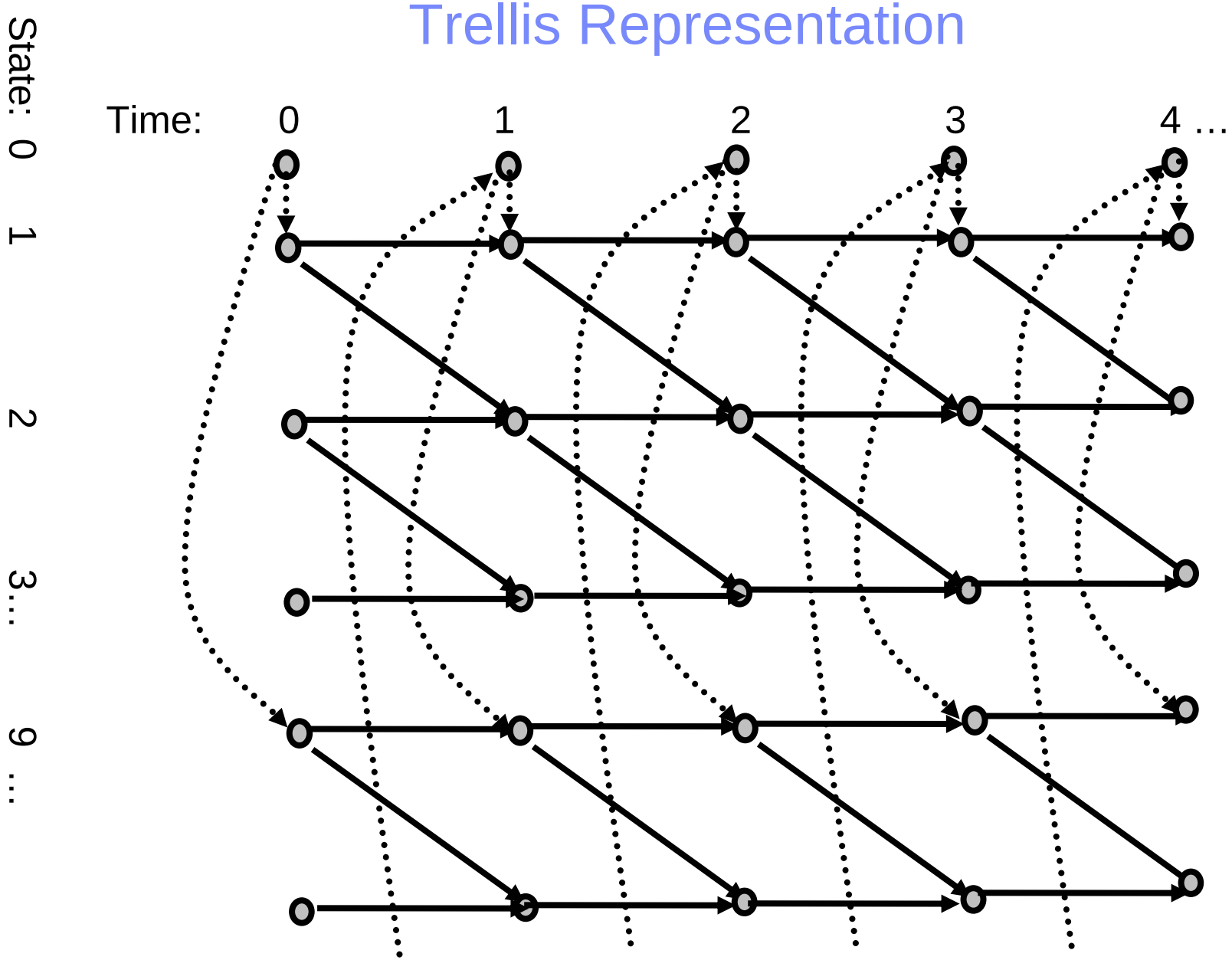
How to represent any sequence of digits?



“911”


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

Trellis Representation



Whole-word model limitations

- The whole-word model suffers from two main problems

    1. 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.

    2. 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
Triphones	10,000
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.

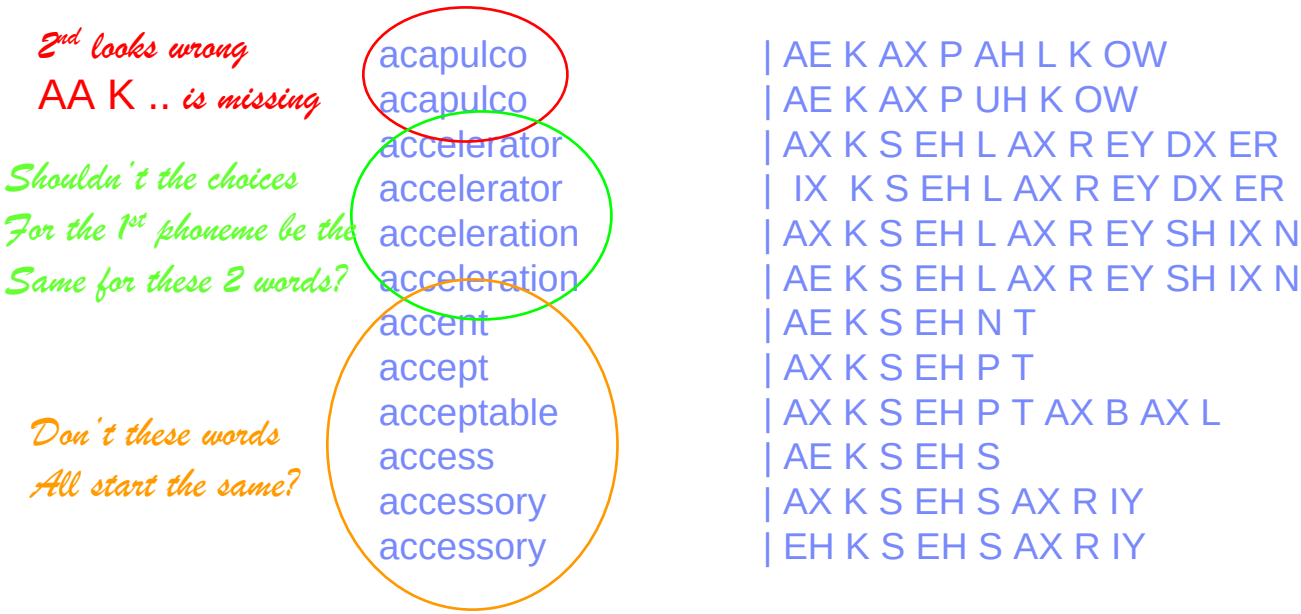
  BANDS → B AE N D Z

- Some words need more than one baseform

  THE → DH UH
  DH IY

Baseform Dictionary

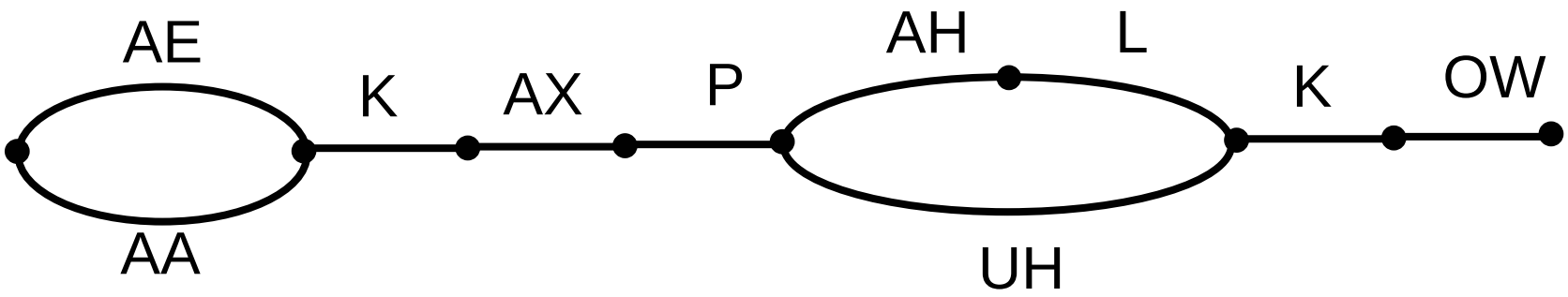
- To determine the pronunciation of each word, we look it up in a dictionary
- Each word may have several possible pronunciations
- Every word in our training script and test vocabulary must be in the dictionary
- The dictionary is generally written by hand
- Prone to errors and inconsistencies



Phonetic Models, cont'd

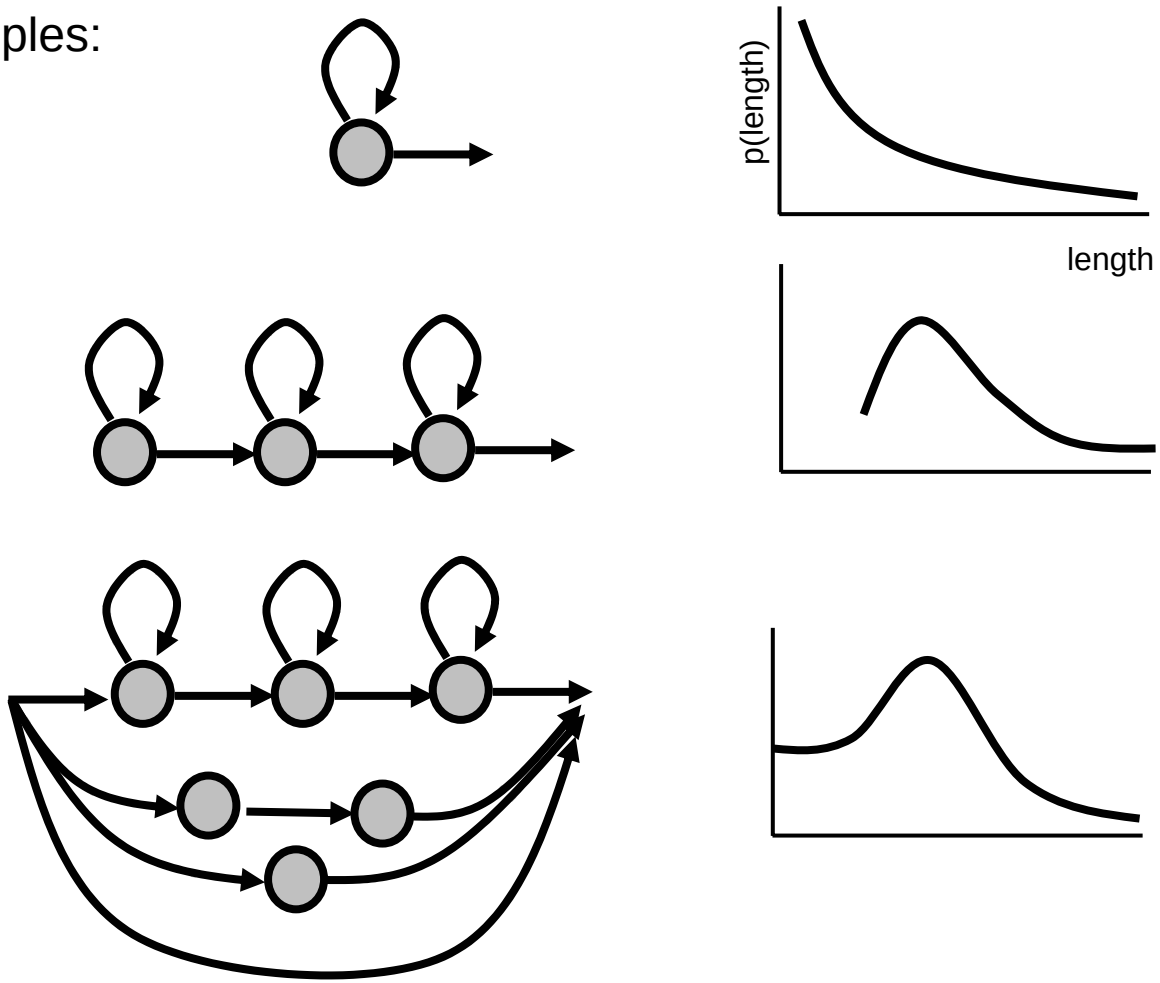
- We can allow for phonological variation by representing baseforms as graphs

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



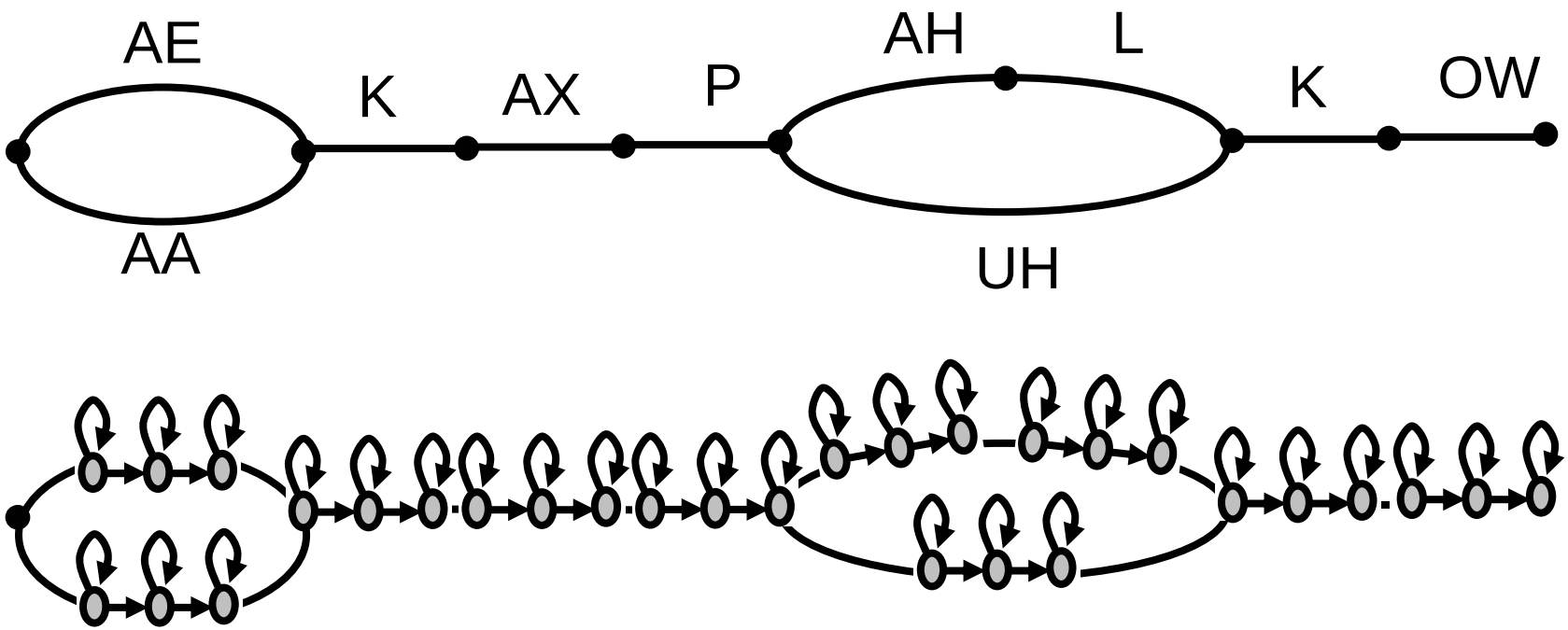
Phonetic Models, cont'd

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



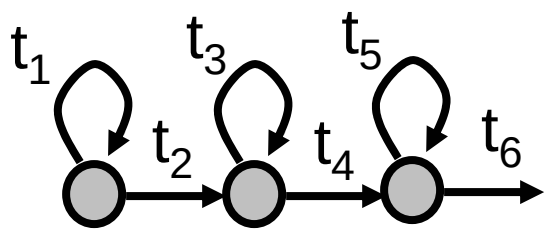
Embedding

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



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

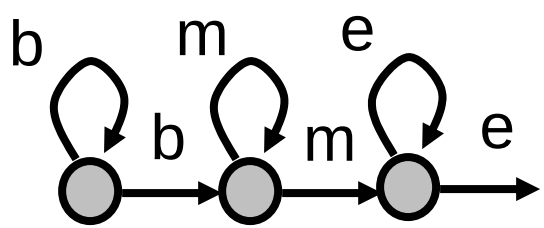
- Now consider phone-based models for this vocabulary

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

Variations in realizations of phonemes

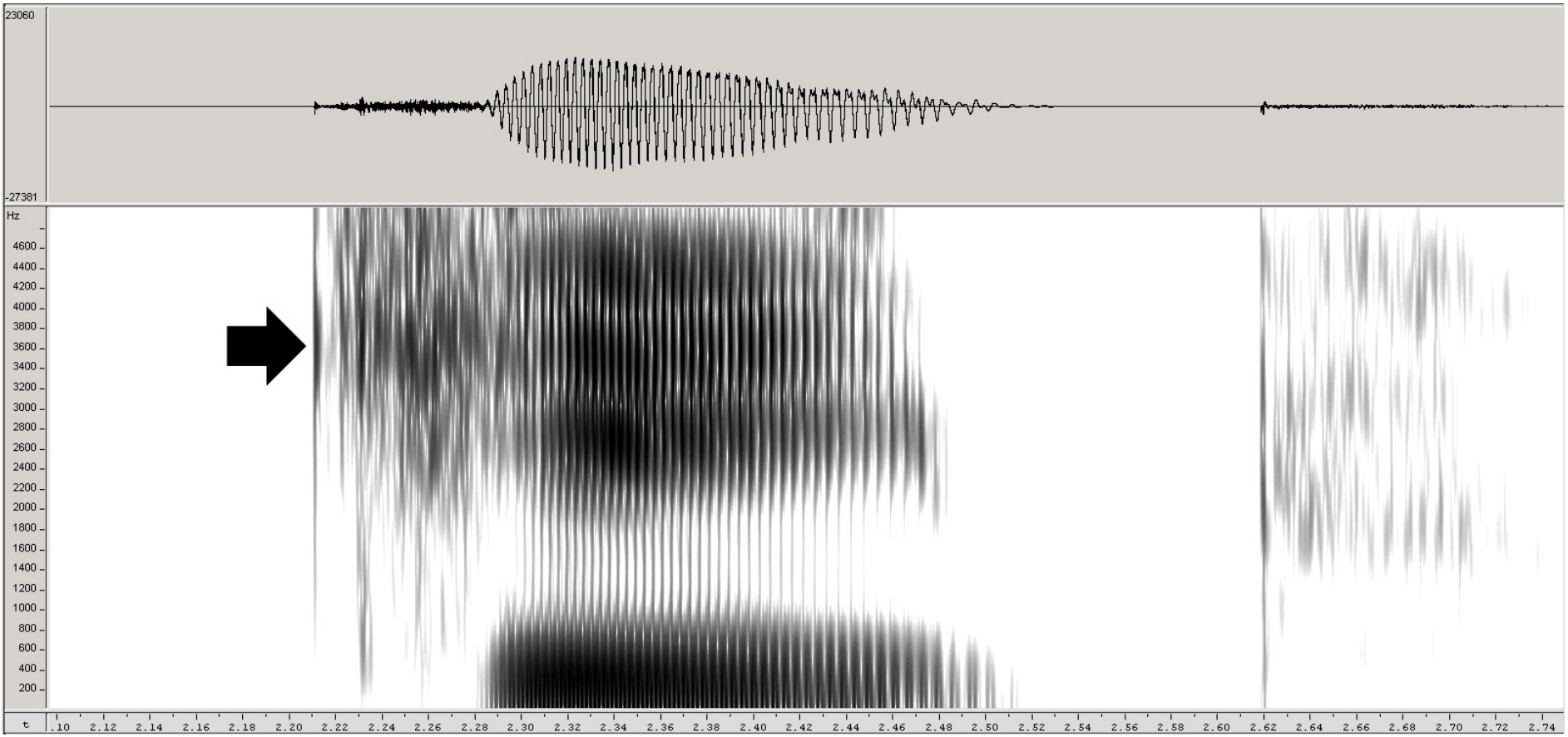
- The broad units, phonemes, have variants known as allophones

  Example: In English 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.

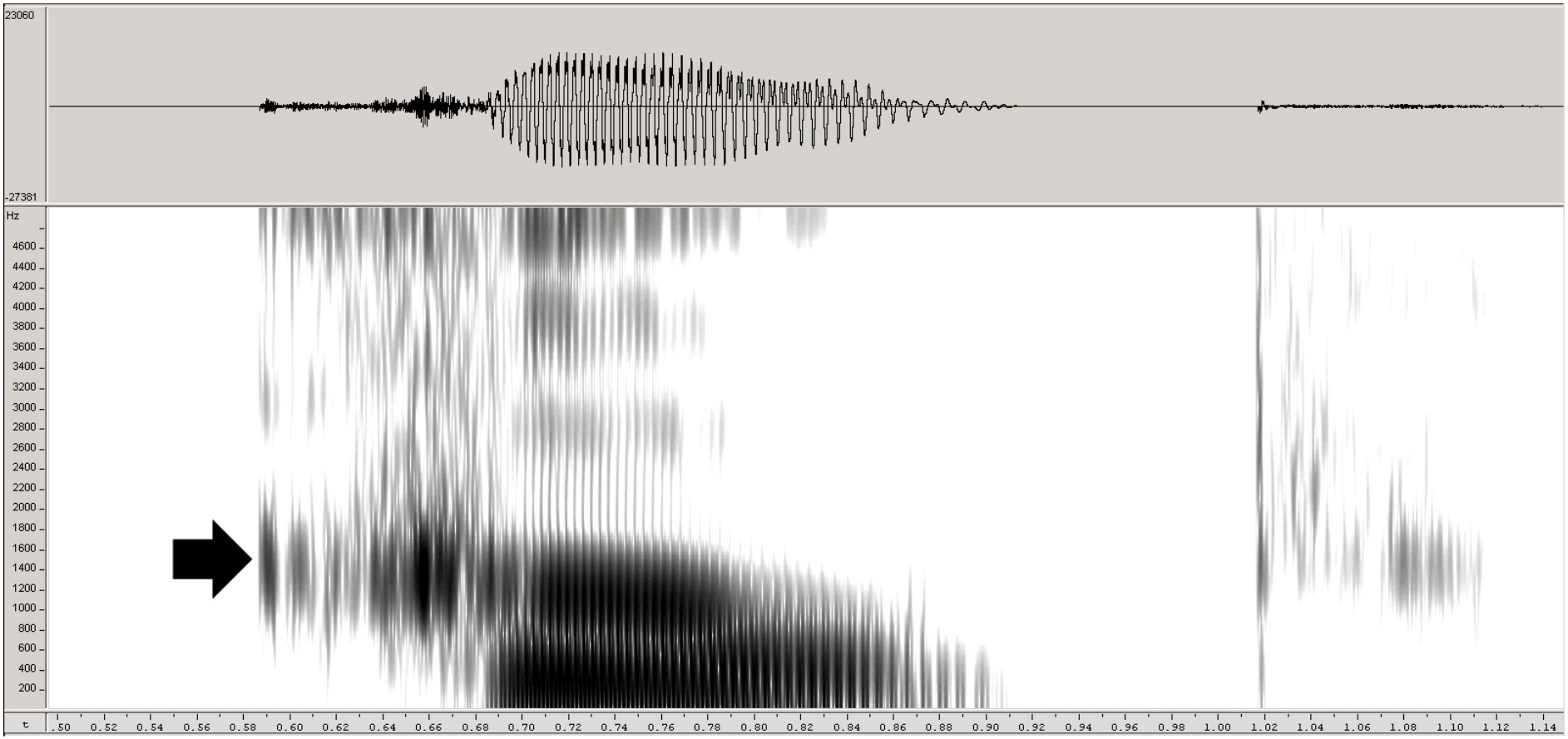
- Articulators have inertia, thus the pronunciation of a phoneme is influenced by surrounding phonemes. This is known as co-articulation

  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
2.10 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”

Context-dependent models

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

Tri-phone 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

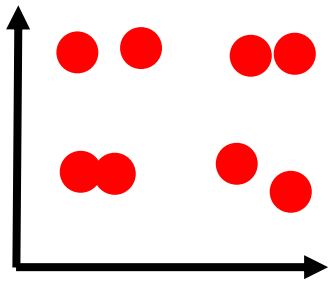
- Start with each item in a cluster by itself
- Find “closest” pair of items
- Merge them into a single cluster
- Iterate
- Different results based on distance measure used

  Single-link: dist(A,B) = min dist(a,b) for $a \in A$, $b \in B$

  Complete-link: dist(A,B) = max dist(a,b) for $a \in A$, $b \in B$

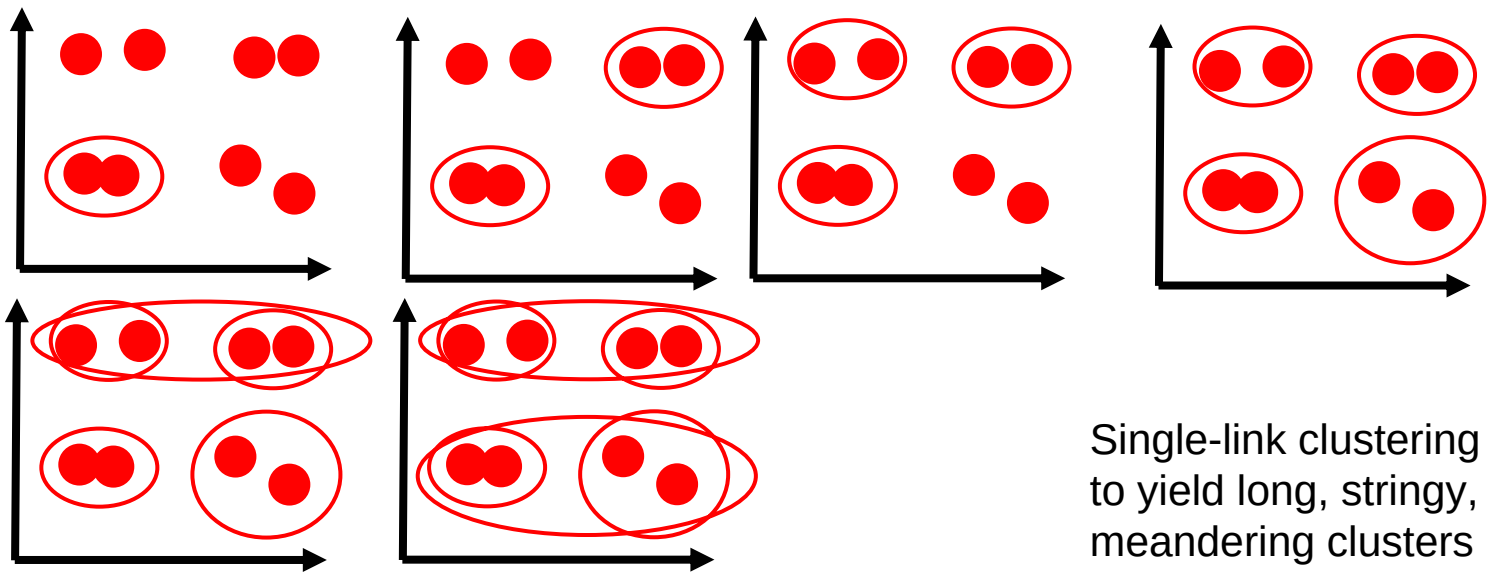
Bottom-up clustering / Single Link

- 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∈A, b∈B

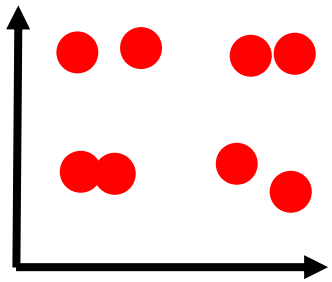
- Single-link clustering into 2 groups proceeds as:



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

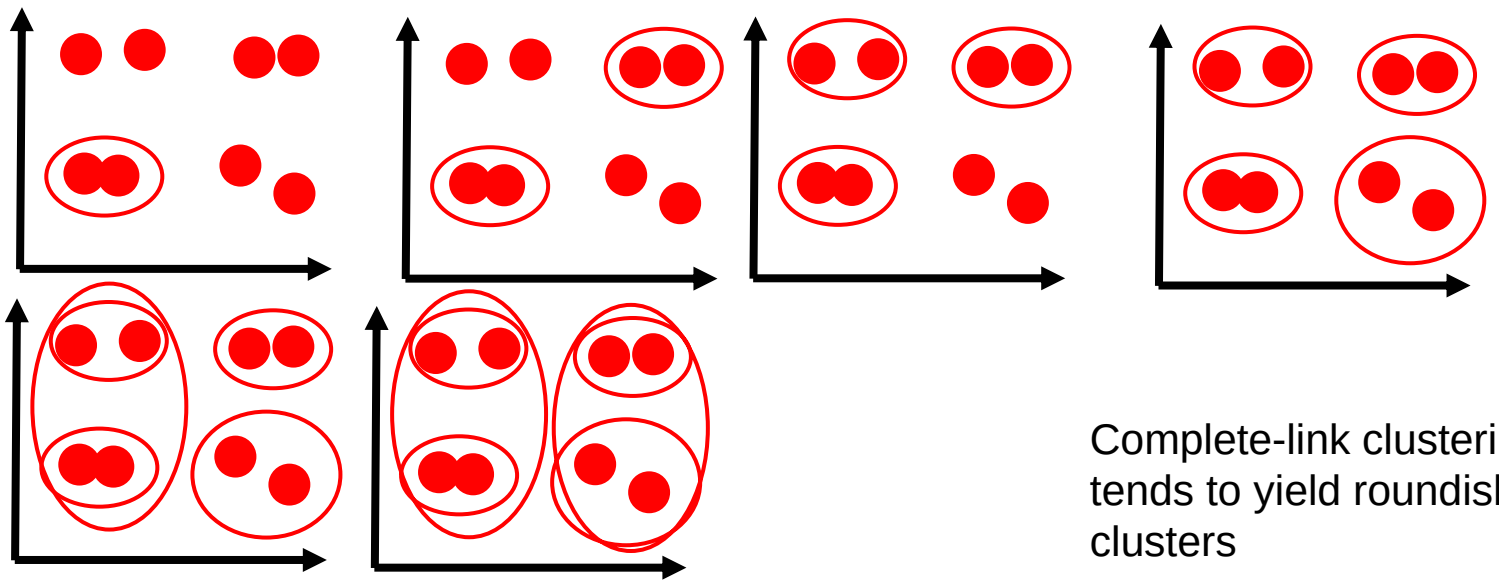
Bottom-up clustering / Complete Link

- 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

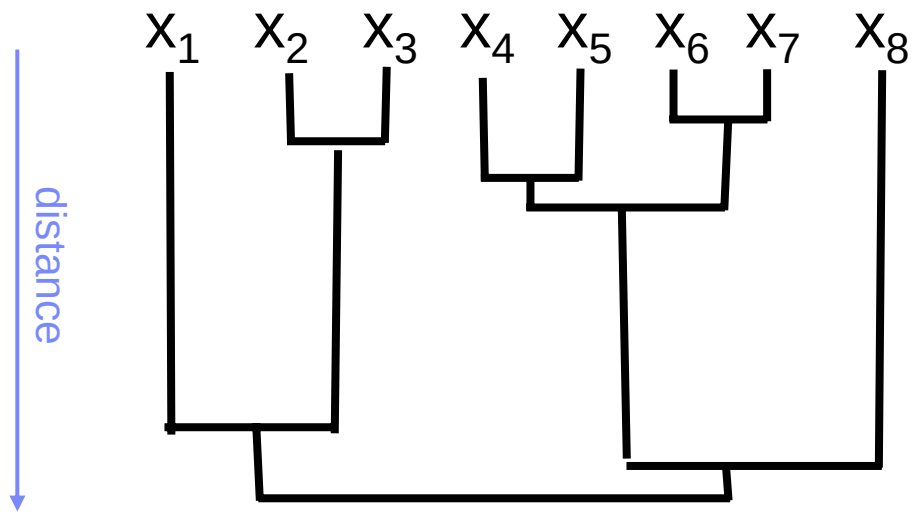
- Single-link clustering into 2 groups proceeds as:



Complete-link clustering tends to yield roundish clusters

Dendogram

- A natural way to display clusters is through a “dendogram”
- 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

Allophones

- Pronunciation variations of phonemes
- Less rigidly defined than triphones
- In our speech recognition system, we typically have about 50 allophones per phoneme.
- Older techniques (“tri-phones”) tried to enumerate by hand the set of allophones for each phoneme.
- We currently use automatic methods to identify the allophones of a phoneme.
- Boils down to conditional modeling.

Conditional Modeling

- In speech recognition, we use conditional models and conditional statistics
- Acoustic model
  - Conditioned on the phone context
- Spelling-to-sound rules
  - Describe the pronunciation of a letter in terms of a probability distribution that depends on neighboring letters and their pronunciations.
  - The distribution is conditioned on the letter context and pronunciation context.
- Language model
  - Probability distribution for the next word is conditioned on the preceding words

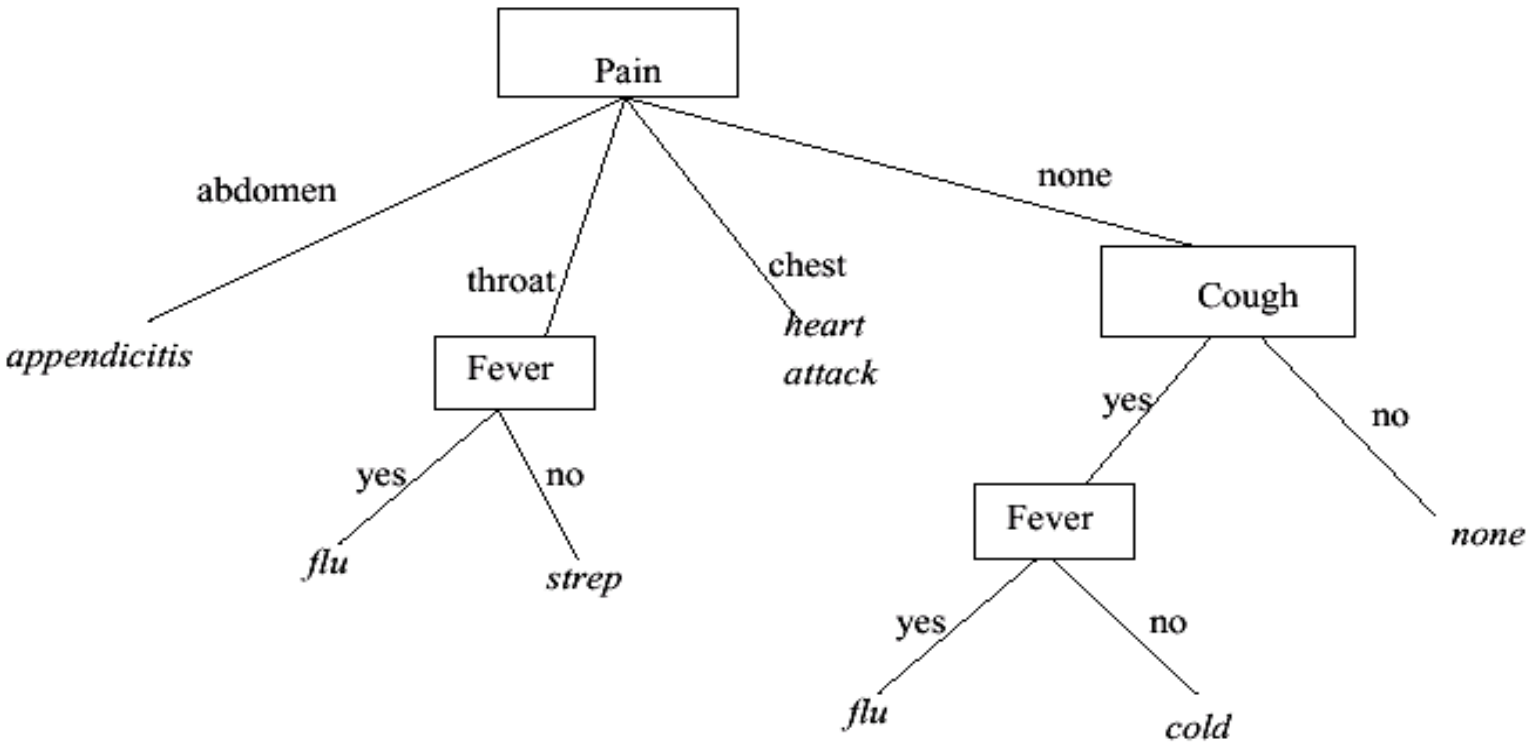
- One way to define the set of conditions is a class of statistical models known as decision trees

- Classic text: L. Breiman et al. Classification and Regression Trees. Wadsworth & Brooks. Monterey, California. 1984.

Decision Trees

- We would like to find equivalence classes among our training samples… DTs categorize data into multiple disjoint categories
- The purpose of a decision tree is to map conditions (such as phonetic contexts) into equivalence classes
- The goal in constructing a decision tree is to create good equivalence classes
- DTs are examples of divisive / “top-down” clustering
- Each internal node specifies an attribute that an object must be tested on
- Each leaf node represents a classification

What does a decision tree look like?



Types of 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:

  $$\text{Prob}\,(d(X_1, \ldots, X_k) \mathrel{!}= C$$

- 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 rate of d as:

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

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

Goals & Requirements

- Goals:
  - To produce an accurate classifier/regression function
  - To understand the structure of the problem
- Requirements on the model:
  - High accuracy
  - Understandable by humans, interpretable
  - Fast construction for very large training databases

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:

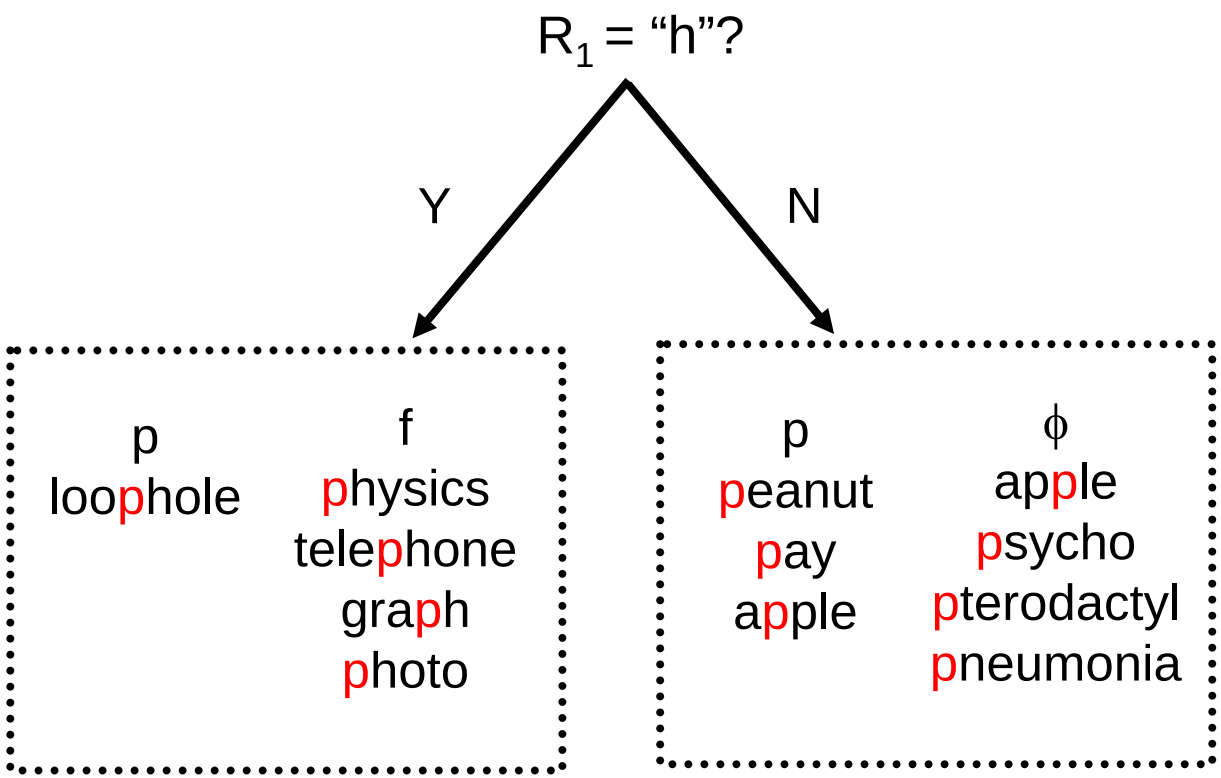
- "p" loophole peanuts pay apple
- "f" physics telephone graph photo
- ϕ apple psycho pterodactyl pneumonia

- The pronunciation of "p" depends on its context.

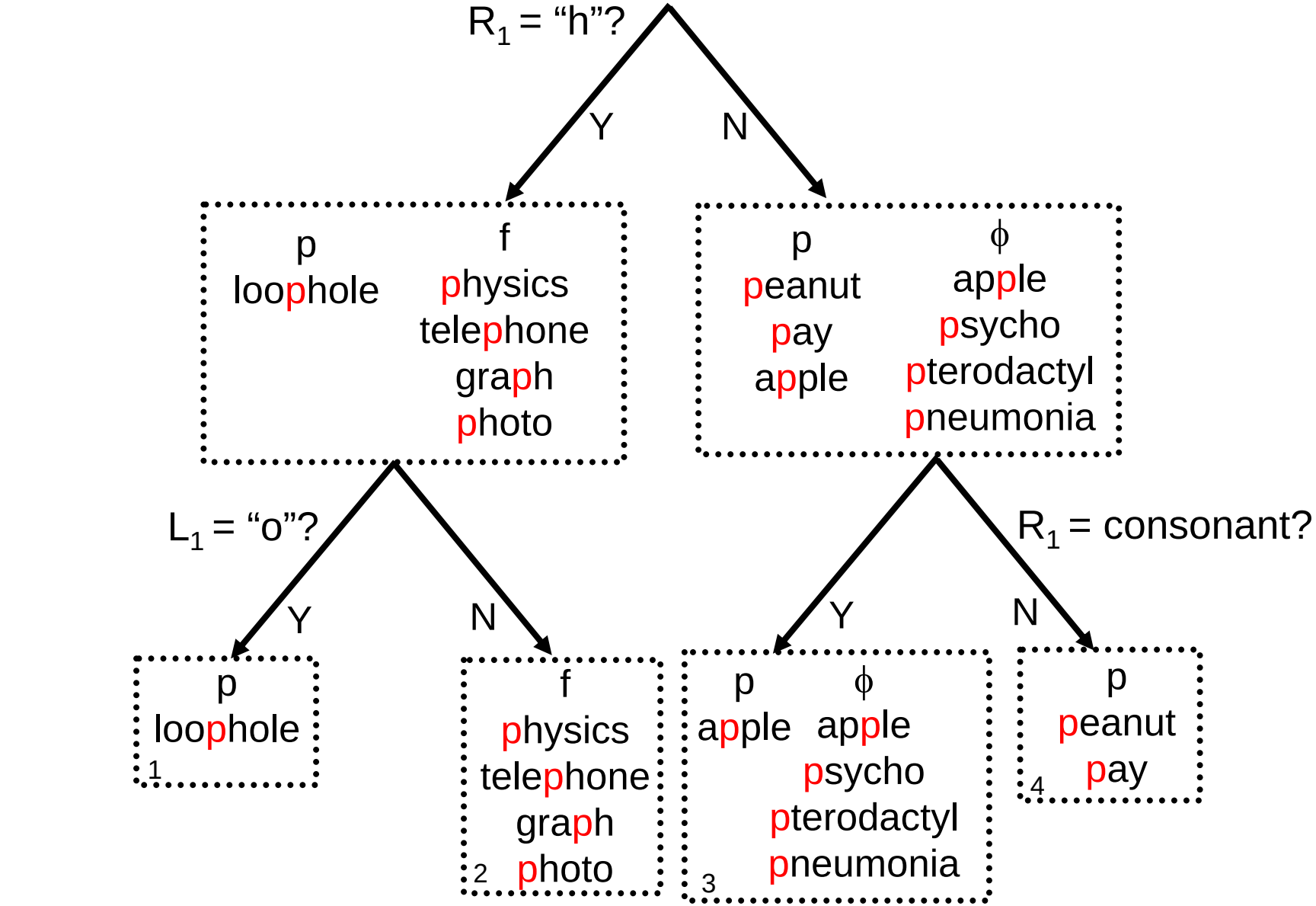
- Task: Using the above training data, partition the contexts into equivalence classes so as to minimize the uncertainty of the pronunciation.

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

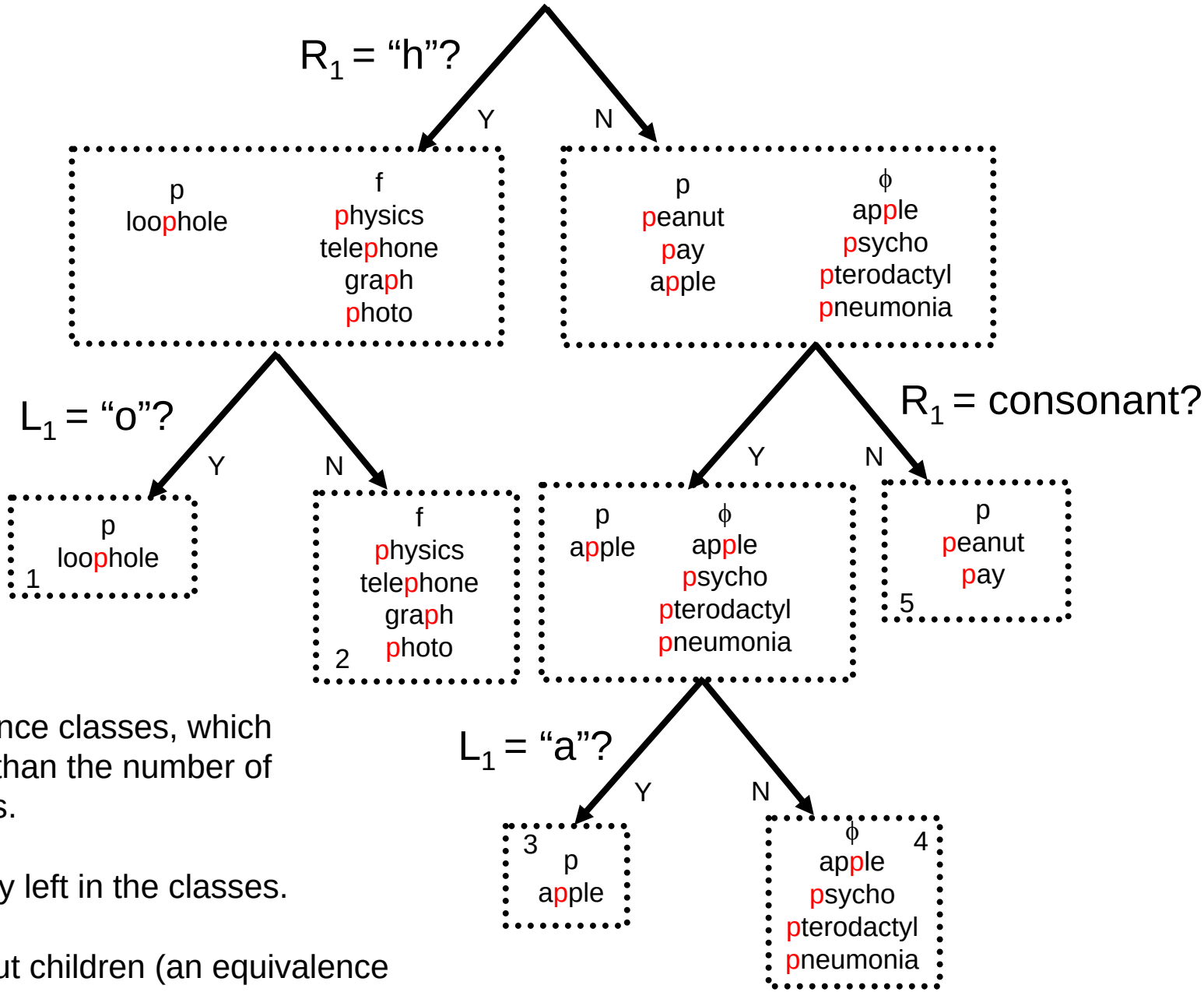
- Denote the context as … L_2 L_1 p R_1 R_2 …



- 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 remains only in class 3.

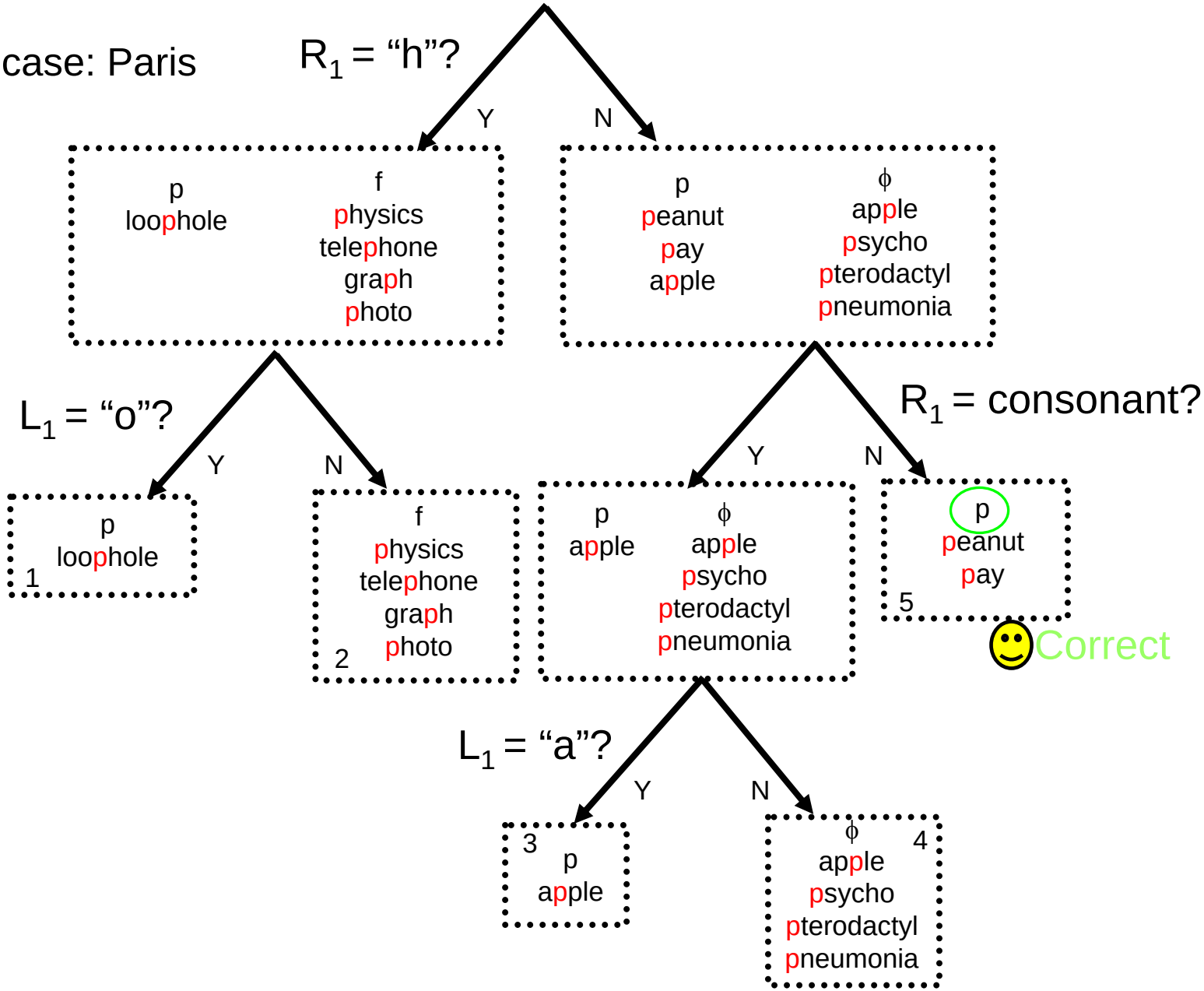


Five equivalence classes, which is much less than the number of letter contexts.

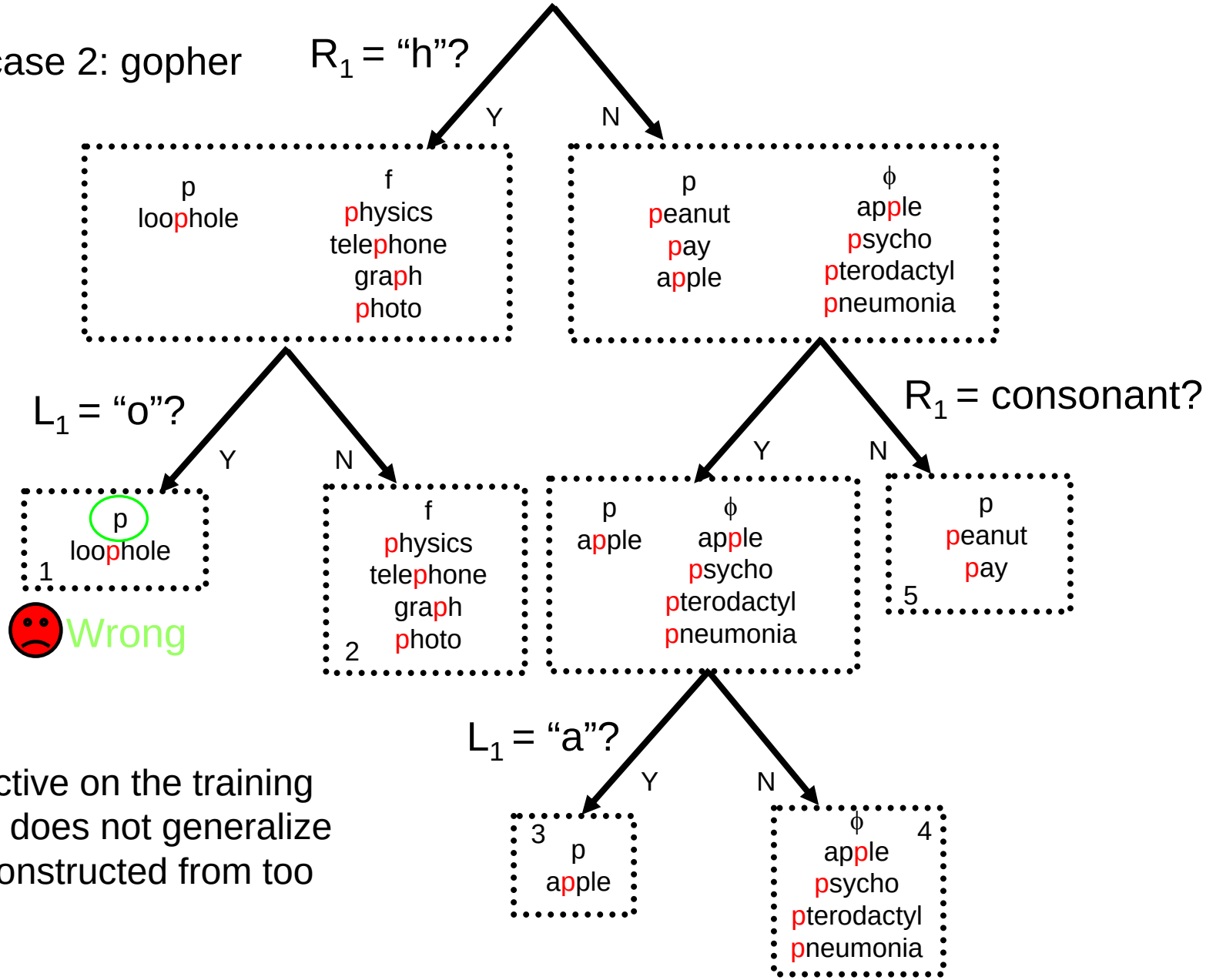
No uncertainty left in the classes.

A node without children (an equivalence class) is called a leaf node. Otherwise it is called an internal node.

Consider test case: Paris



Consider test case 2: 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
- The assignment of each node to a class

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 is 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: $x \in \{B,C\}$?
Q7: $x \in \{A,B,C\}$?

- Use the best question found.
- Potential problems:
  1. Requires a lot of CPU. For alphabet size A there are $\sum_j \binom{A}{j}$ questions.
  2. 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 10 variables: the 5 phones to the left of the current phone and the 5 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

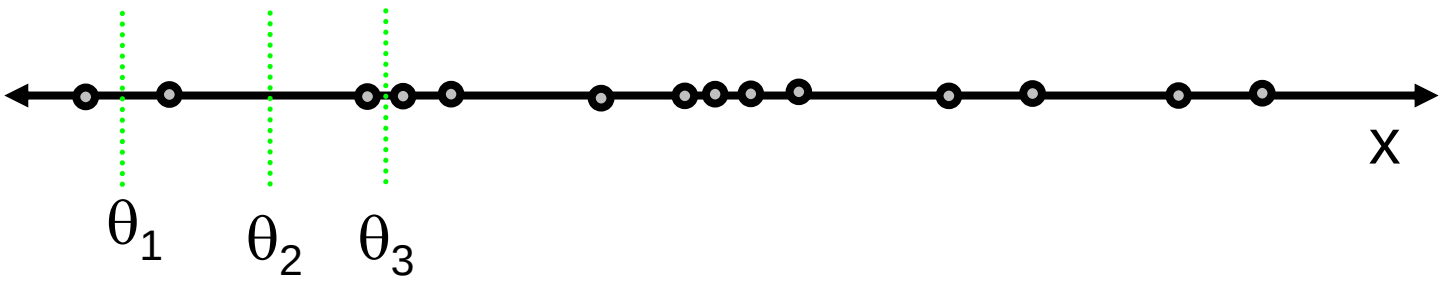
Phones	Letters
{P}	{A}
{T}	{E}
{K}	{I}
{B}	{O}
{D}	{U}
{G}	{Y}
{P,T,K}	{A,E,I,O,U}
{B,D,G}	{A,E,I,O,U,Y}
{P,T,K,B,D,G}	

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<θ? where θ is some real value.
- In order to find the threshold θ, we must try values which separate all training samples.

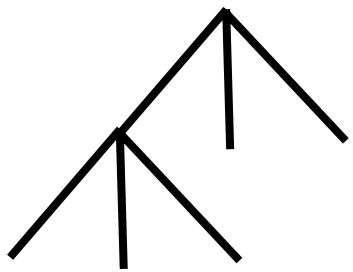


- We do not currently use continuous questions for speech recognition.

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

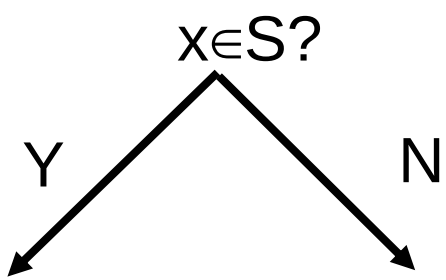


  26 Outcomes: A B C … Z

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

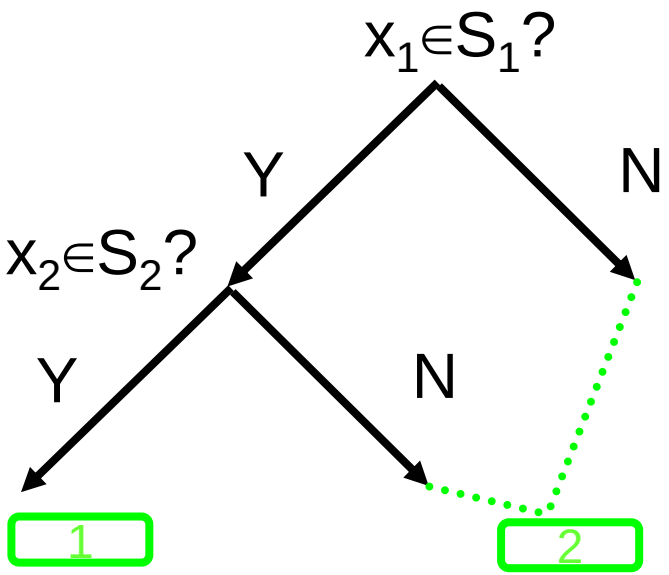
Simple Binary Question

- 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:



2 Outcomes:

1 $(X_1 \in S_1) \cap (X_2 \in S_2)$

2 $\overline{(X_1 \in S_1) \cap (X_2 \in S_2)}$

$= \overline{(X_1 \in S_1)} \cup \overline{(X_2 \in S_2)}$

- All complex 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.

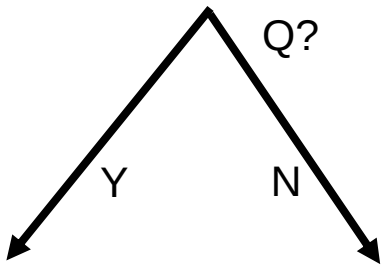
Tree Construction Overview

- Let $x_1 \ldots x_n$ denote n discrete variables whose values may be asked about. Let Q_{ij} denote the j^{th} pre-determined question for x_i.
- Starting at the root, try splitting each node into 2 sub-nodes:
- 1. For each variable x_i evaluate questions Q_{i1}, Q_{i2}, … 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 $N(\mu_p,\Sigma_p)$
Compute likelihood $\mathcal{L}(data_p|\mu_p,\Sigma_p)$



Left child node: Model as a single Gaussian $N(\mu_\ell,\Sigma_\ell)$
Compute likelihood $\mathcal{L}(data_\ell|\mu_\ell,\Sigma_\ell)$

Right child node: Model as a single Gaussian $N(\mu_r,\Sigma_r)$
Compute likelihood $\mathcal{L}(data_r|\mu_r,\Sigma_r)$

- Goal: Find Q such that $\mathcal{L}(data_\ell|\mu_\ell,\Sigma_\ell) \times \mathcal{L}(data_r|\mu_r,\Sigma_r)$ is maximized.

Question Evaluation, cont'd

- Let $x^1, x^2, \ldots x^N$ be a sample of feature x, in which outcome a_i occurs c_i times.
- 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 a_i in the left and right sub-samples.
- The best question Q for feature x is the one which maximizes the conditional likelihood of the sample given Q, or, equivalently, maximizes the conditional log likelihood of the sample.

log likelihood computation

- The log likelihood of the data, given that we ask question Q, is: $\log L(x^1..x^n \mid Q) = \sum_{i=1}^{N} c_i^l \log p_i^l + \sum_{i=1}^{N} c_i^r \log p_i^r$

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

$$\log L(x^1..x^n \mid Q) = \sum_{i=1}^{N} c_i^l \log(c_i^l / n_l) + \sum_{i=1}^{N} c_i^r \log(c_i^r / n_r)$$

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

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

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

Entropy

- Entropy is a measure of uncertainty, measured in bits.
- Let x be a discrete random variable taking values $a_1..a_N$ 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 $p=(p_1\ p_2\ \ldots\ p_N)$.

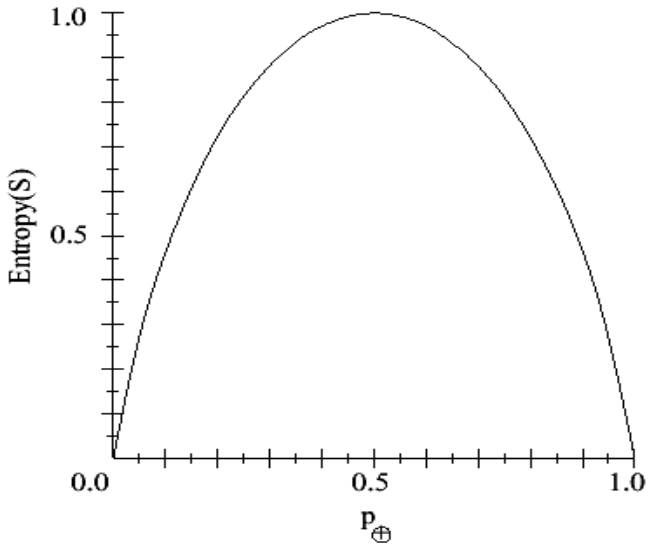
$$H = -\sum_{i=1}^{N} p_i \log_2 p_i$$

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

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.

H can be interpreted as the theoretical **minimum average number of bits** that are required to encode/transmit the distribution p.

What does entropy look like for a binary variable?



- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity, or disorder, of S

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Entropy and Likelihood

Let x be a discrete random variable taking values $a_1..a_N$ in an alphabet A of size N with probabilities $p_1 \ldots p_N$ respectively.

Let $x^1 .. x^n$ be a sample of x in which a_i occurs c_i times, i=1..N.

The sample likelihood is: $L = \prod_{i=1}^{N} p_i^{c_i}$

The maximum likelihood estimate of p_i is $\hat{p}_i = c_i / n$

Thus, an estimate of the sample log likelihood is: $\log \hat{L} = \sum_{i=1}^{N} c_i \log \hat{p}_i$

and $-\frac{1}{n} \log_2 \hat{L} = -\sum_{i=1}^{N} \hat{p}_i \log_2 \hat{p}_i = \hat{H}$

Since n is a constant, and log is a monotonic function, H is monotonically related to L.

Therefore, maximizing likelihood ⇔ minimizing entropy.

“p” tree, revisited

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

Log likelihood of data at the root node is:

$$\log_2 L(x^1..x^{12}) = \sum_{i=1}^{3} c_i \log_2 c_i - n \log_2 n$$

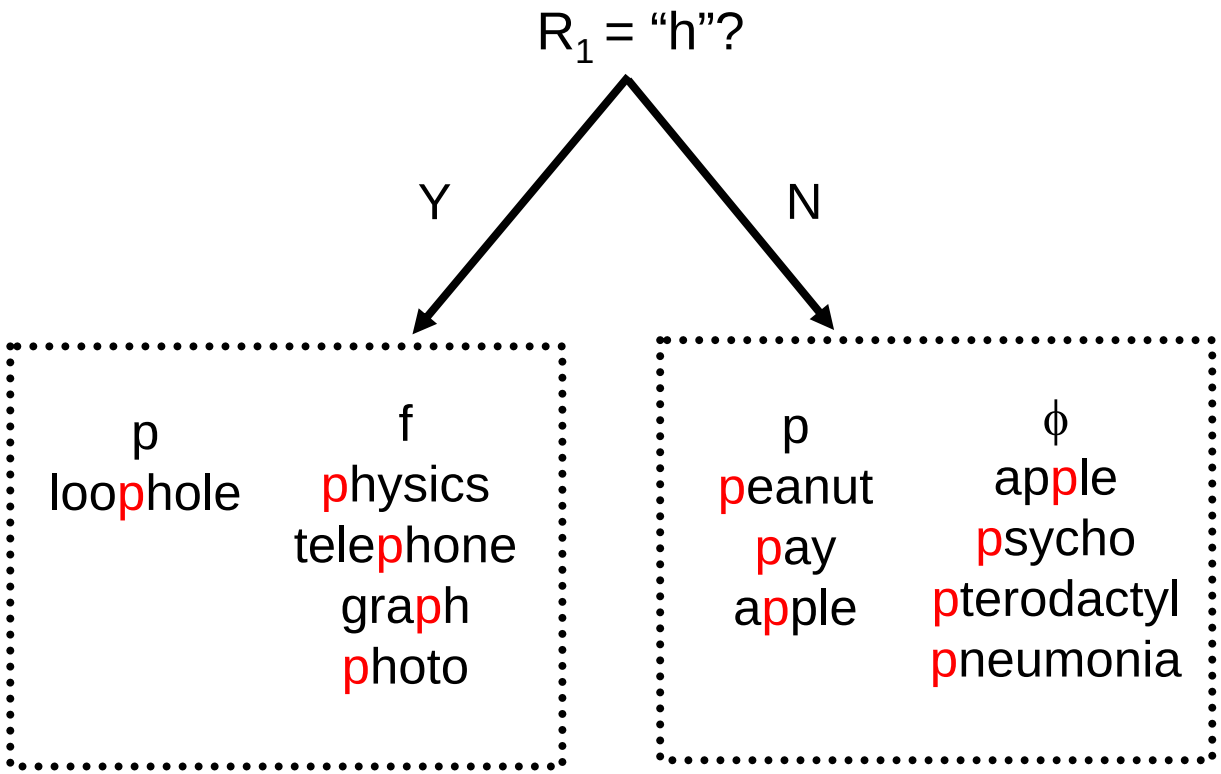
$$= 4\log_2 4 + 4\log_2 4 + 4\log_2 4 - 12\log_2 12 = -19.02$$

Average entropy at the root node is:

$$H(x^1..x^{12}) = -\frac{1}{n}\log_2 L(x^1..x^{12})$$

$$= 19.02/12 = 1.58 \text{ bits}$$

“p” tree revisited: Question A



n_ℓ=5
c_p^ℓ=1
c_f^ℓ=4
c_ϕ^ℓ=0

n_r=7
c_p^r=3
c_f^r=0
c_ϕ^r=4

“p” tree revisited: Question A

n_ℓ=5 c_p^ℓ=1 c_f^ℓ=4 c_ϕ^ℓ=0

n_r=7 c_p^r=3 c_f^r=0 c_ϕ^r=4

- Log likelihood of data after applying question A is:

$$\log_2 L(x^1..x^{12} \mid Q_A) = 4\log_2 4 - 5\log_2 5 + 3\log_2 3 + 4\log_2 4 - 7\log_2 7 = -10.51$$

- Average entropy of data after applying question A is:

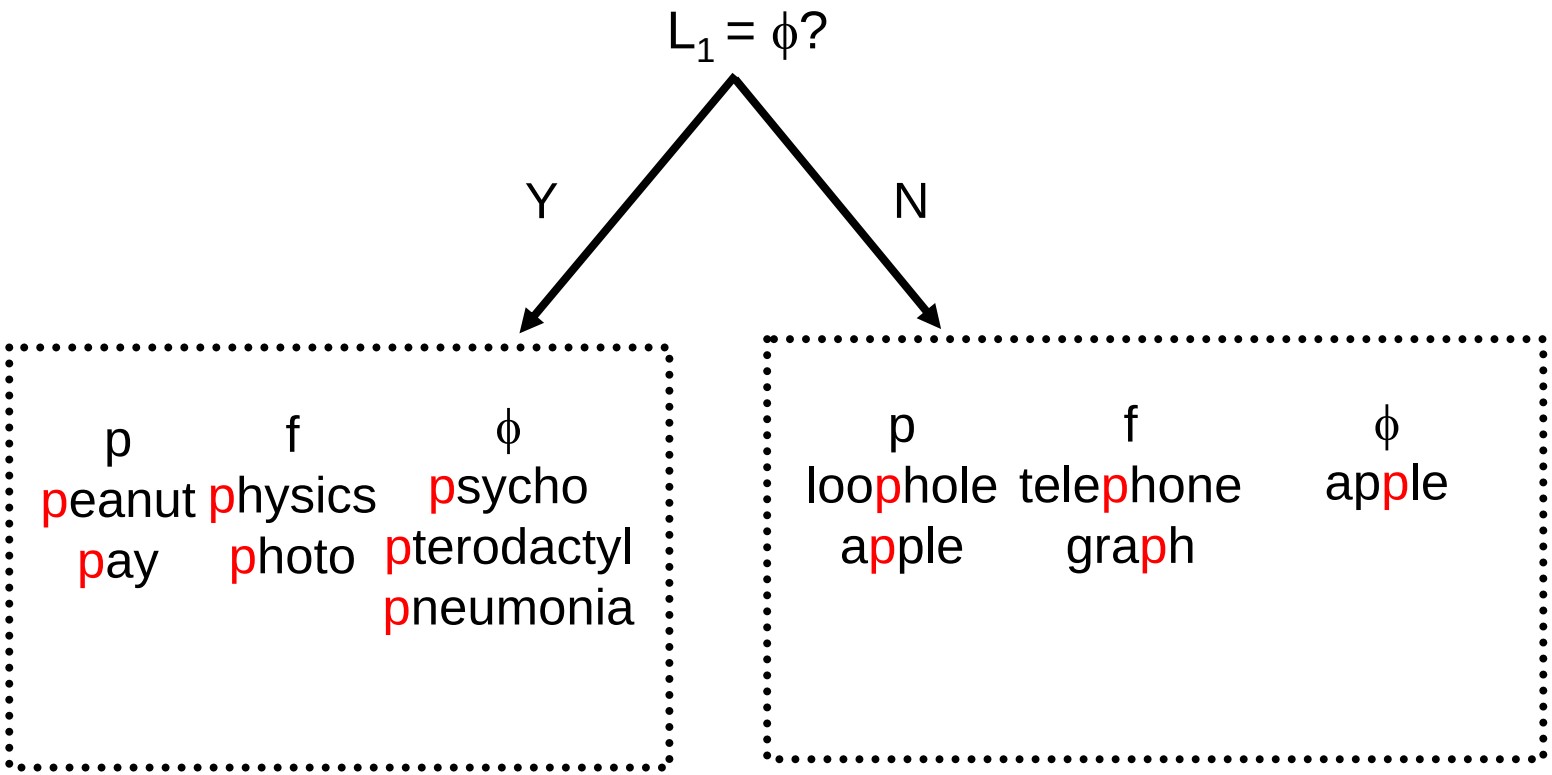
$$H(x^1..x^{12} \mid Q_A) = \frac{-1}{12}\log_2 L(x^1..x^{12} \mid Q_A) = 10.51/12 = 0.87 \text{ bits}$$

- Increase in log likelihood due to question A is -10.51+19.02=8.51

- Decrease in entropy due to question A is 1.58-0.87=0.71 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



$n_\ell = 7$
$c_p^\ell = 2$
$c_f^\ell = 2$
$c_\phi^\ell = 3$

$n_r = 5$
$c_p^r = 2$
$c_f^r = 2$
$c_\phi^r = 1$

“p” tree revisited: Question B

n_ℓ=7 c_p^ℓ=2 c_f^ℓ=2 c_ϕ^ℓ=3

n_r=5 c_p^r=2 c_f^r=2 c_ϕ^r=1

- Log likelihood of data after applying question B is:

$$\log_2 L(x^1..x^{12} \mid Q_B) = 2\log_2 2 + 2\log_2 2 + 3\log_2 3 - 7\log_2 7 + 2\log_2 2 + 2\log_2 2 - 5\log_2 5 = -18.51$$

- Average entropy of data after applying question B is:

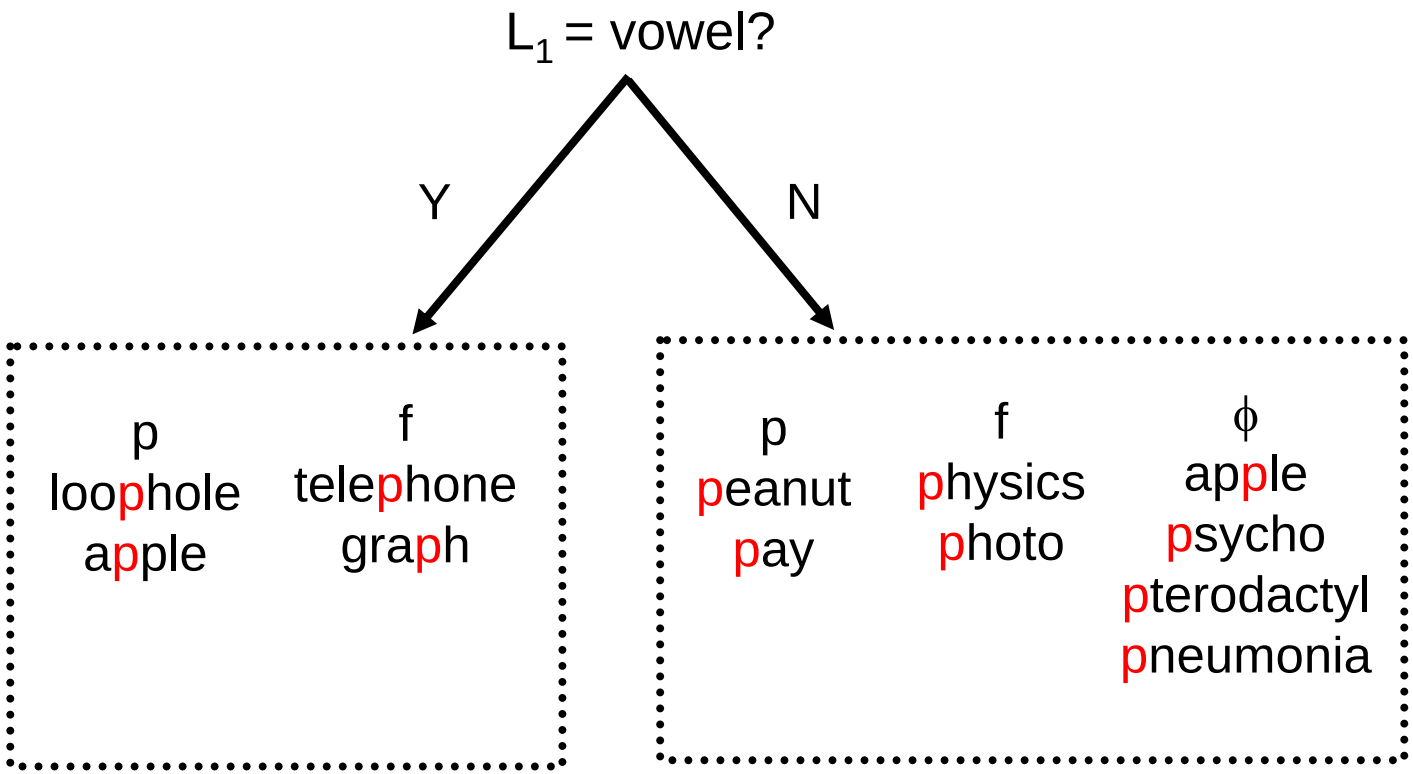
$$H(x^1..x^{12} \mid Q_B) = \frac{-1}{12}\log_2 L(x^1..x^{12} \mid Q_B) = 18.51/12 = 1.54 \text{ bits}$$

- Increase in log likelihood due to question B is -18.51+19.02=0.51

- Decrease in entropy due to question B is 1.58-1.54=0.04 bits

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

“p” tree revisited: Question C



$n_\ell = 4$
$c_p^\ell = 2$
$c_f^\ell = 2$
$c_\phi^\ell = 0$

$n_r = 8$
$c_p^r = 2$
$c_f^r = 2$
$c_\phi^r = 4$

“p” tree revisited: Question C

n_ℓ=4 c_p^ℓ=2 c_f^ℓ=2 c_ϕ^ℓ=0

n_r=8 c_p^r=2 c_f^r=2 c_ϕ^r=4

- Log likelihood of data after applying question C is:

$$\log_2 L(x^1..x^{12} \mid Q_C) = 2\log_2 2 + 2\log_2 2 - 4\log_2 4$$
$$+ 2\log_2 2 + 2\log_2 2 + 4\log_2 4 - 8\log_2 8 = -16.00$$

- Average entropy of data after applying question C is:

$$H(x^1..x^{12} \mid Q_C) = \frac{-1}{12}\log_2 L(x^1..x^{12} \mid Q_C) = 16/12 = 1.33 \text{ bits}$$

- 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=0.25 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.

Best Question

- In general, we seek questions which maximize the likelihood of the training data given some model.
- The expression to be maximized depends on the model.
- In the previous example the model is a multinomial distribution.
- Another example: context-dependent prototypes use a continuous distribution.

Best Question: Context-Dependent Prototypes

- For context-dependent prototypes, we use a decision tree for each arc in the Markov model.
- At each leaf of the tree is a continuous distribution which serves as a context-dependent model of the acoustic vectors associated with the arc.
- The distributions and the tree are created from acoustic feature vectors aligned with the arc.
- 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.
- We estimate the likelihood of the acoustic vectors during tree construction using a diagonal Gaussian model.
- When tree construction is complete, we replace the diagonal Gaussian models at the leaves by more accurate models to serve as the final prototypes. Typically we use mixtures of diagonal Gaussians.

Diagonal Gaussian Likelihood

- Let Y=$\underline{y}_1$ $\underline{y}_2$ $\underline{y}_n$ be a sample of independent p-dimensional acoustic vectors arising from a diagonal Gaussian distribution with mean $\underline{\mu}$ and variances $\underline{\sigma}^2$. Then

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

- The maximum likelihood estimates of $\underline{\mu}$ and $\underline{\sigma}^2$ are:

$$\hat{u}_j = \frac{1}{n}\sum_{i=1}^{n} y_{ij} \qquad j = 1,2,\ldots p$$

$$\hat{\sigma}_j^2 = \frac{1}{n}\sum_{i=1}^{n} y_{ij}^2 - \hat{\mu}_j^2 \qquad j = 1,2,\ldots p$$

- Hence, an estimate of log $\mathcal{L}$(Y) is

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

Diagonal Gaussian Likelihood

- Now,

$$\sum_{i=1}^{n}(y_{ij}-\hat{\mu}_j)^2=\sum_{i=1}^{n}y_{ij}^2-2\hat{\mu}_j\sum_{i=1}^{n}y_{ij}+n\hat{\mu}_j^2 \qquad j=1,2,..p$$

$$=\sum_{i=1}^{n}y_{ij}^2-n\hat{\mu}_j^2=n\hat{\sigma}_j^2 \qquad j=1,2,...p$$

- Hence

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

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

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 $\log\mathcal{L}(Y_\ell)+\log\mathcal{L}(Y_r)$
- 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

$$\hat{\sigma}_{lj}^2 = \frac{1}{n_l} \sum_{y \in Y_l} y_j^2 - \frac{1}{n_l^2} \left(\sum_{y \in Y_l} y_j \right)^2 \qquad j = 1,2,..p$$

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

How Big a Tree?

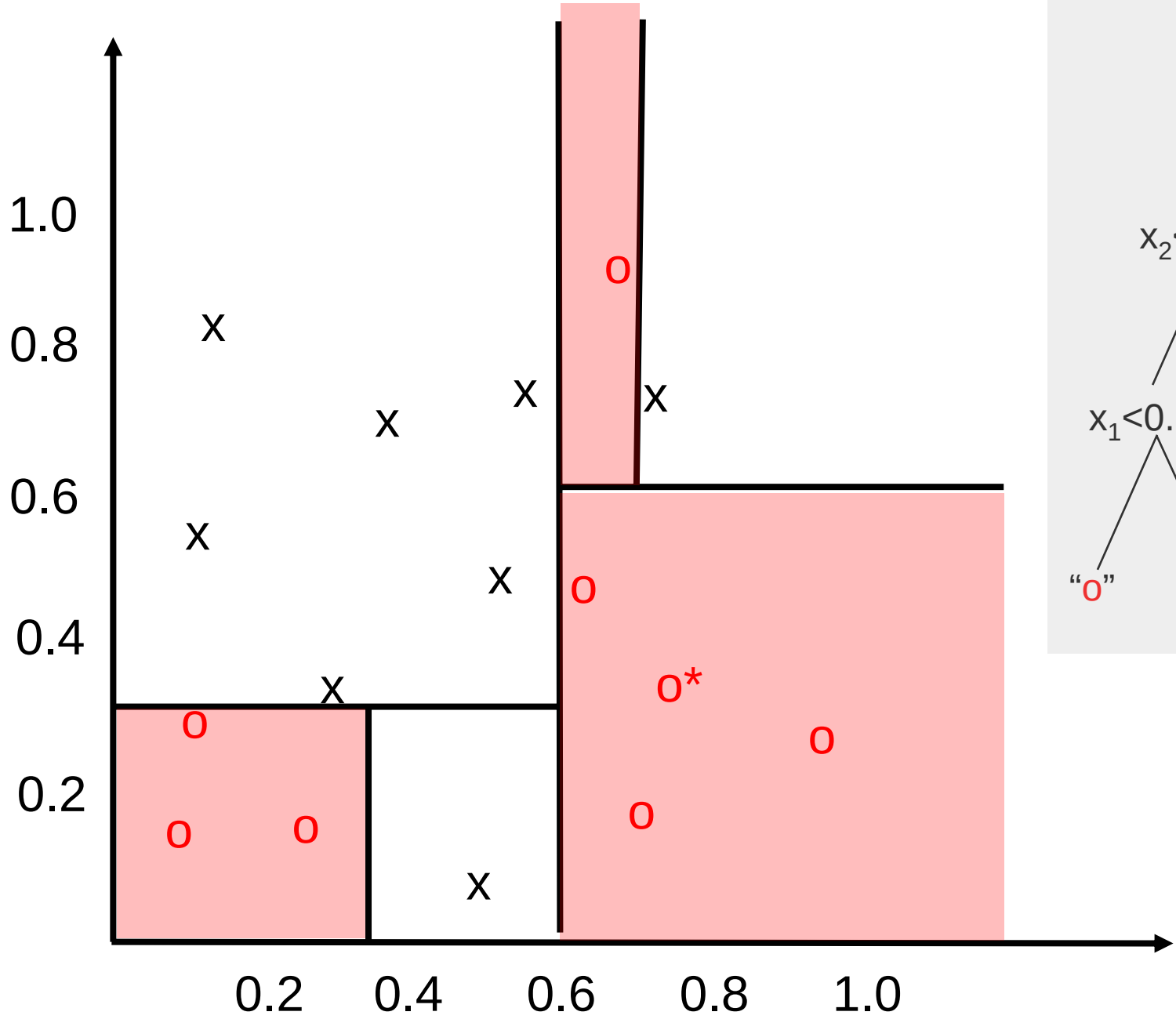
- CART suggests cross-validation
  - Measure performance on a held-out data set
  - Choose the tree size that maximizes the likelihood of the held-out data
- In practice, simple heuristics seem to work well
- A decision tree is fully grown when no terminal node can be split
- Reasons for not splitting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

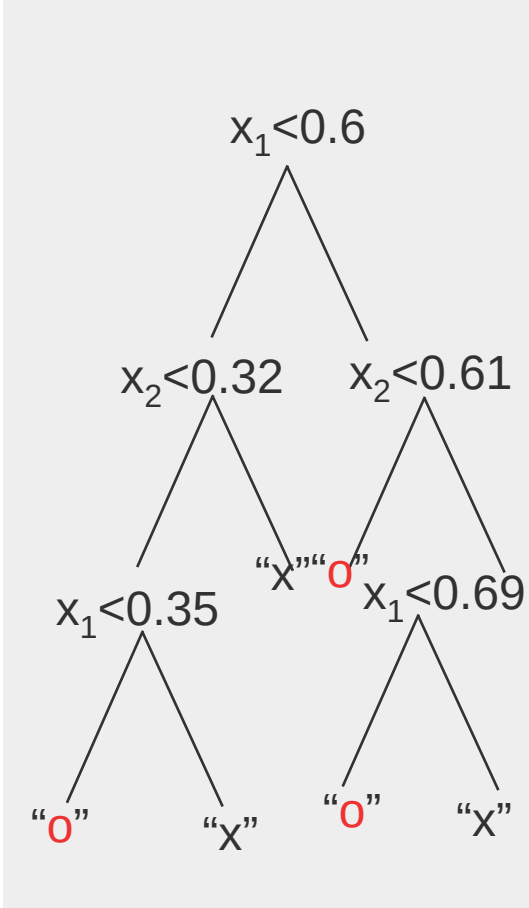
Instability

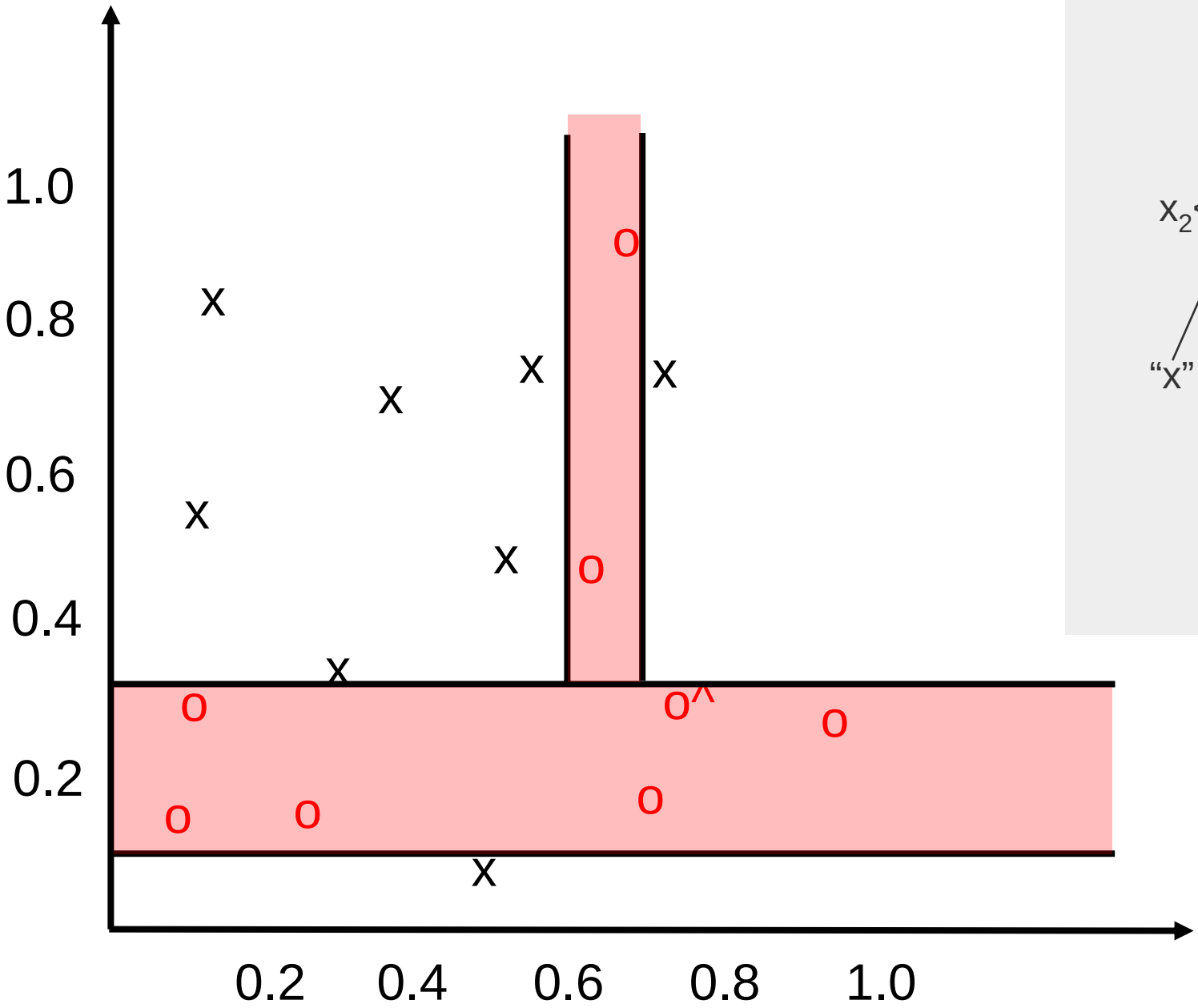
- Decision trees have the undesirable property that a small change in the data can result in a large difference in the final tree
- Consider a 2-class problem, w_1 and w_2
- Observations for each class are 2-dimensional

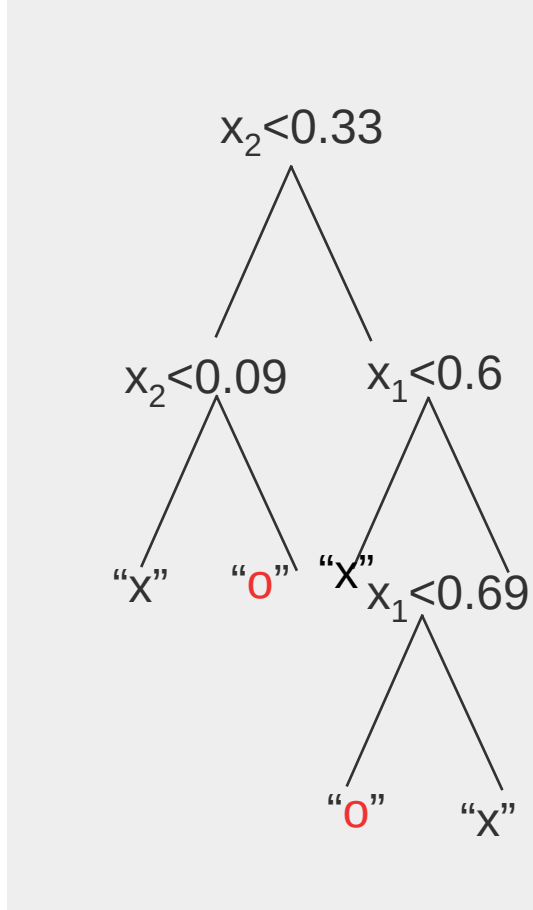
w_1 "x"	
x_1	x_2
.15	.83
.09	.55
.29	.35
.38	.70
.52	.48
.57	.73
.73	.75
.47	.06

w_2 "o"	
x_1	x_2
.10	.29
.08	.15
.23	.16
.70	.19
.62	.47
.91	.27
.65	.90
.75	.36* (.32^)


1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0
x
x
x
x
x
x
x
o
o
o
o
o
o
o*
o


x1<0.6
x2<0.32
x2<0.61
"x"
"o"
x1<0.35
x1<0.69
"o"
"x"
"o"
"x"


1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0


x2<0.33
x2<0.09
x1<0.6
“x”
“o”
“x”
x1<0.69
“o”
“x”

Bagging

Create multiple models by training the same learner on different samples of the training data

Given a training set of size n, create m different training sets of size n by sampling with replacement

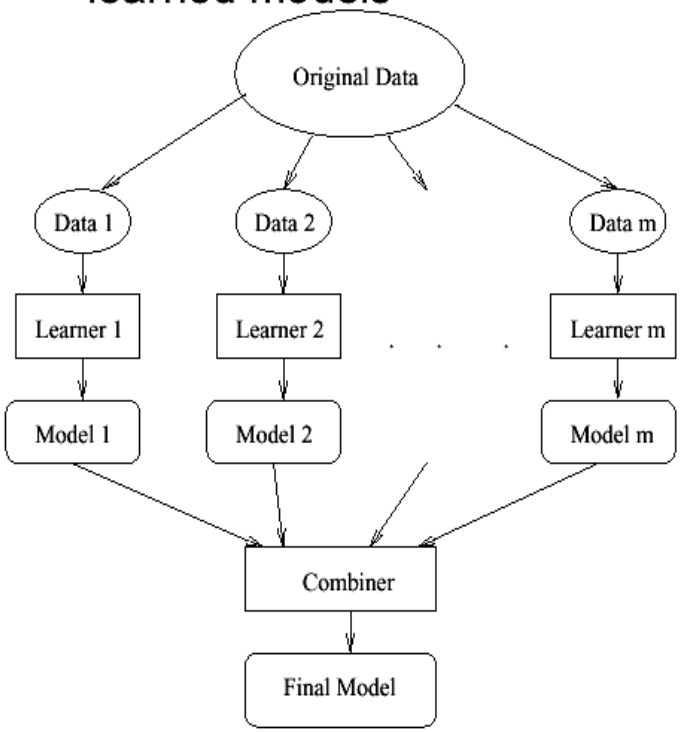
Combine the m models using a simple majority vote.

Can be applied to any learning method, including decision trees.

Decreases the generalization error by reducing variance in the results for unstable learners (i.e. when the hypothesis can change dramatically if training data is slightly altered.)

Multiple Models

- Learn multiple, alternative definitions of the concept
- Make final decisions based on a (weighted) voting of the multiple learned models



Boosting

- Another method for producing multiple models by repeatedly altering the data given to a learner.
- Examples are given weights, and at each iteration
  – a new hypothesis is learned and
  – the examples are reweighted to focus on those that the latest hypothesis got wrong
- During testing, each of the hypotheses gets a weighted voted proportional to its training set accuracy.

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

Putting it all together

Given a word sequence, we can construct the corresponding Markov model by:

1. Re-writing word string as a sequence of phonemes
2. Concatenating phonetic models
3. Using the appropriate tree for each arc to determine which alloarc (leaf) is to be used in that context

Example

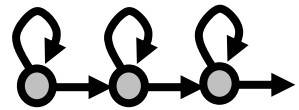
The rain in Spain falls ….

Look these words up in the dictionary to get:

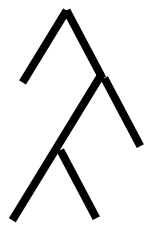
DH AX | R EY N | IX N | S P EY N | F AA L Z | …

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.

Course Feedback

Was this lecture mostly clear or unclear?

What was the muddiest topic?

Other feedback (pace, content, atmosphere, etc.)