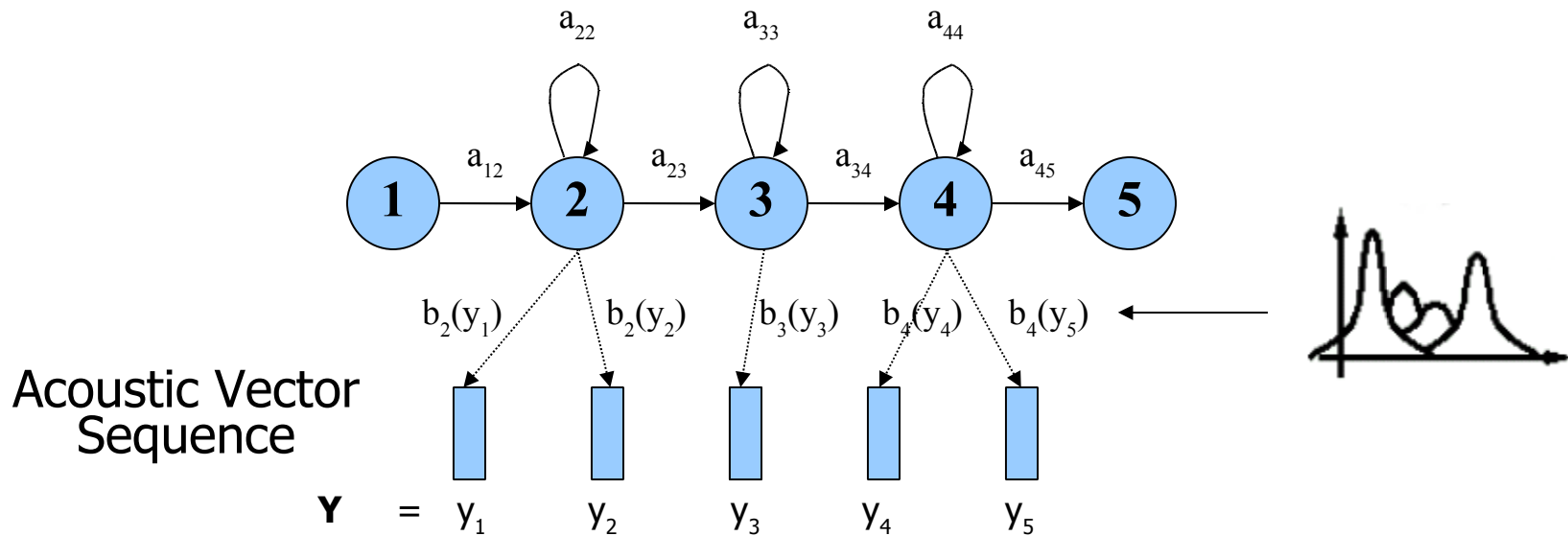


Acoustic Modeling I

11-751 Speech Recognition

10-08-2008

HMM Acoustic Modeling



- Hidden Markov Model for each phone or context dependent phone
- Transition probabilities: \mathbf{a}_{ij} model durational variability in speech
- Output distribution: $\mathbf{b}_i(\mathbf{y}_k)$ model spectral variability
- **Modeling emission probabilities most difficult aspect of HMM modeling and training**

HMMs for Acoustic Modeling

- Modeling emission probabilities most difficult aspect of HMM modeling and training
 - **Continuous HMMs:** Mixture Gaussians
 - Very large number of parameters to estimate
 - **Discrete HMMs:** via Vector Quantization
 - emission probabilities replaced with explicit token
 - **Semi-Continuous:** trade-off between above approaches
 - Share Gaussians across all models
 - Model dependent mixture weights

HMMs for Acoustic Modeling

- Discrete HMMs

no Gaussians at all
discrete feature space

- Fully continuous HMMs

each model has it's
own codebook of Gaussians

- Semi-continuous HMMs

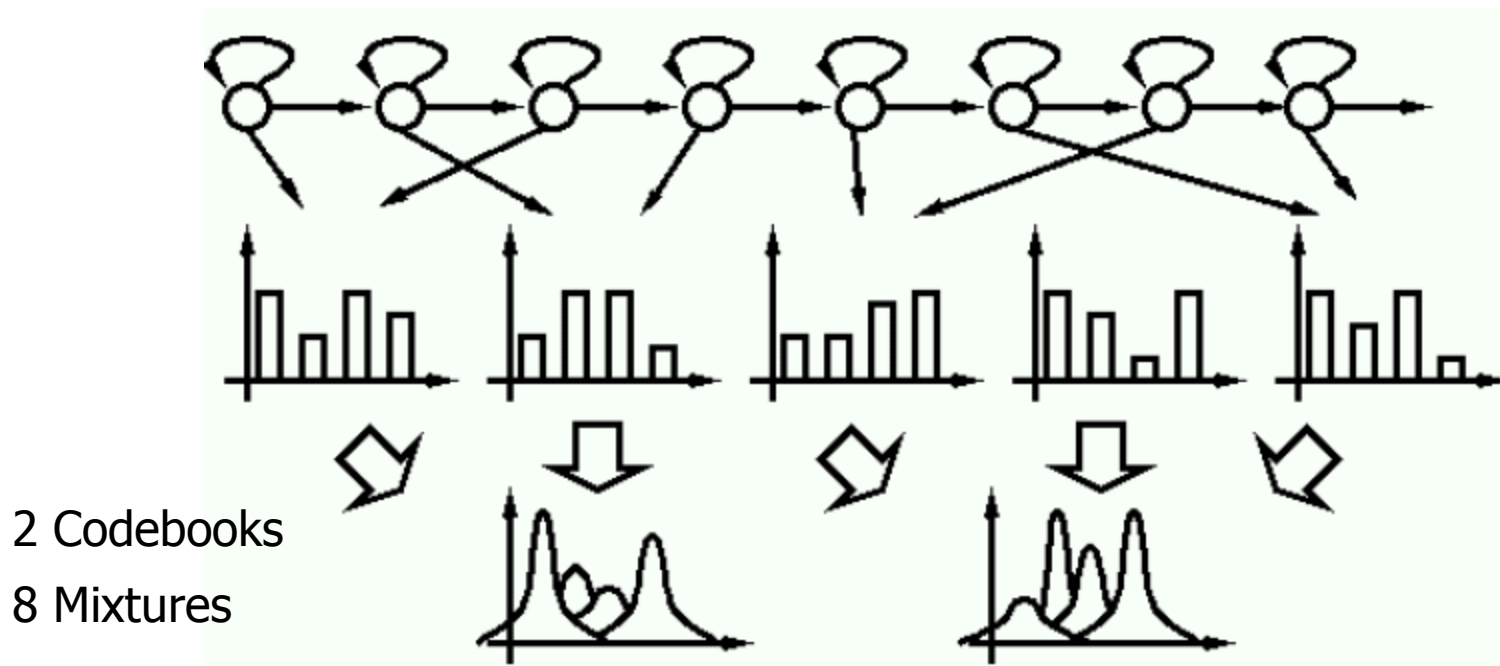
one codebook of Gaussians
shared by all models



- SCHMMs can provide a good compromise between modeling and trainability
- Often variable degree of tying: any number of Gaussian codebooks, any number of mixture weight sets

Arbitrary Degrees of Continuity

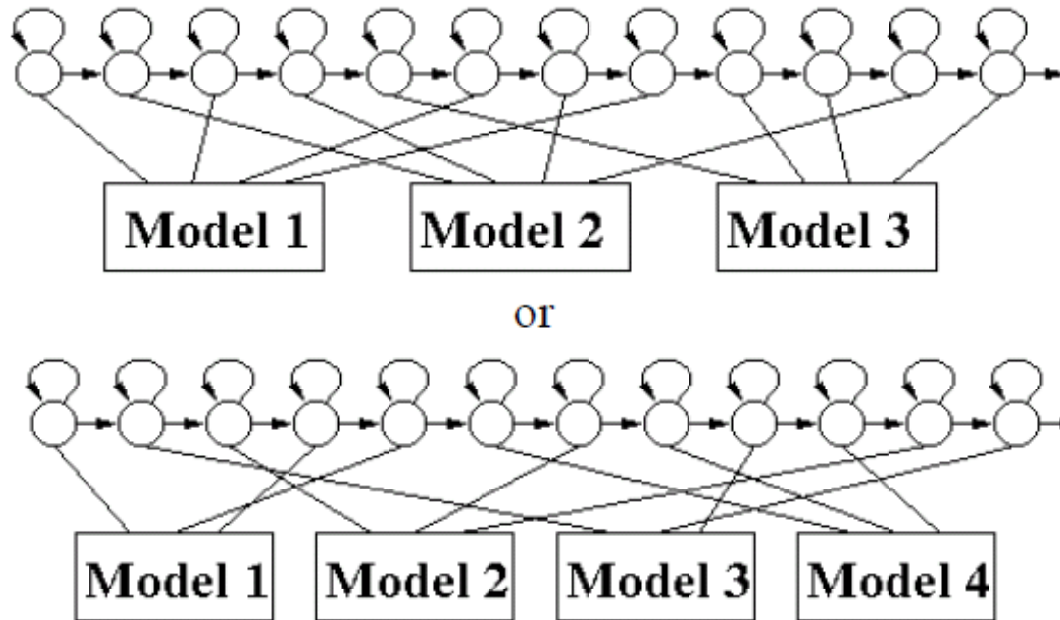
- The most flexible way of parameter tying is to use an arbitrary number of mixture weight sets and an arbitrary number of Gaussian codebooks:



- Typically: Codebooks: 1,000, Mixtures: 50,000 (1996, WSJ)
- 2006 Gale: Codebooks: 5,000, Mixtures: 300,000
- What number to choose? How to decide which model is used by which state?

Parameter Tying – Detail vs Trainability

- Parameter tying means using the same parameters for different HMM-states



- Goal 1:** Why waste two models (parameter subsets) on two HMM-states that are very similar or equivalent
- Goal 2:** The more tying we use, the fewer models we have. Thus, every model gets more training data assigned and can be trained more robustly

Parameter Tying – not just Gaussians

- Two generally different approaches to parameter tying:
 - Knowledge driven:
 - have expert sit down and decide which parameters to tie
 - Data driven:
 - tie parameters with "similar" training data together
- Which parameters can be tied:
 - Sets of Gaussians: semi continuous vs. fully continuous HMMs
 - Within a Mixture: have different Gaussians use the same covariance matrix
 - Transitions: different HMM states share the same transition probabilities
- **And most important:** HMM state-models: different HMM-states share the same acoustic atom

Acoustic Modeling

- Pronunciation Variants
- Context Dependent Acoustic Modeling
- Clustering of Context
- Bottom-Up vs. Top-Down
- Clustering Distances Between Model Clusters
- Problems with Vocabulary Dependencies
- Clustering with Decision Trees

Pronunciation Variants

Problem:

The same word can be pronounced differently in different situations due to:

- Context (coarticulation effects)
- Various correct pronunciations (the + w/vowel, a = AE or A = EY, ...)
- Dialects (something vs. somephin, thang vs. thing, ...)
- Correct pronunciation differs from most commonly used ones (February vs. Febyury, Monday vs. Mondy, ...)

Solution:

- Add multiple entries into the pronunciation lexicon
- When building HMMs for a word, build multiple, state sequences

Example:

(excerpt from real dictionary)

THE(1)	TH AH	WHEN(1)	W EH N
THE(2)	TH AX	WHEN(2)	HH W EH N
THE(3)	TH IY	WHEN(3)	W IH N
		WHEN(4)	HH W IH N

How to find Pronunciation Variants?

- Defined by linguist or phonetician

Problem: acoustic sounds do not necessarily correspond to linguistic units

- ASR-expert adds variants to the dictionary

Problem: most successful approach but also very expensive

- In both cases we might face as lot of inconsistencies

→ Define rules and apply to existing pronunciations

e.g. "...ing" → ... **IX NG**, ... **AX NG**, ... **AE NG**

Automatic Dictionary Learning

(Tilo Sloboda, 1996)

- 1)** Train a phone based recognizer (vocab consists of phonemes)
 - 2)** Run recognition on word-based segmented training data
 - 3)** Single out the top-N most frequent phoneme sequences per word
 - 4)** Add these sequences as pronunciation to the dictionary
- **Problems with this approach**
 - Accuracy of phone recognizer
 - Requires word-based segmentation (how to get without dictionary)
 - Requires enough examples per word to find reliable candidates

Context Dependent Acoustic Modeling

Consider the Pronunciations of **TRUE**, **TRAIN**, **TABLE**, and **TELL**

Most common lexicon entries are:

TRUE	T R UW
TRAIN	T R EY N
TABLE	T EY B L
TELL	T EH L

The actual pronunciation sounds a bit like:

TRUE	CH R UW
TRAIN	CH R EY N
TABLE	T HH EY B L
TELL	T HH EH L

- The phoneme **T** sounds different depending on whether the following phoneme is an **R** or a vowel

Context Dependent Acoustic Modeling

First idea:

use actual pronunciations in the lexicon:

i.e.: "CH R UW" instead of "T R UW"

Problem: The **CH** in TRUE does sound different from the **CH** in CHURCH

Second idea:

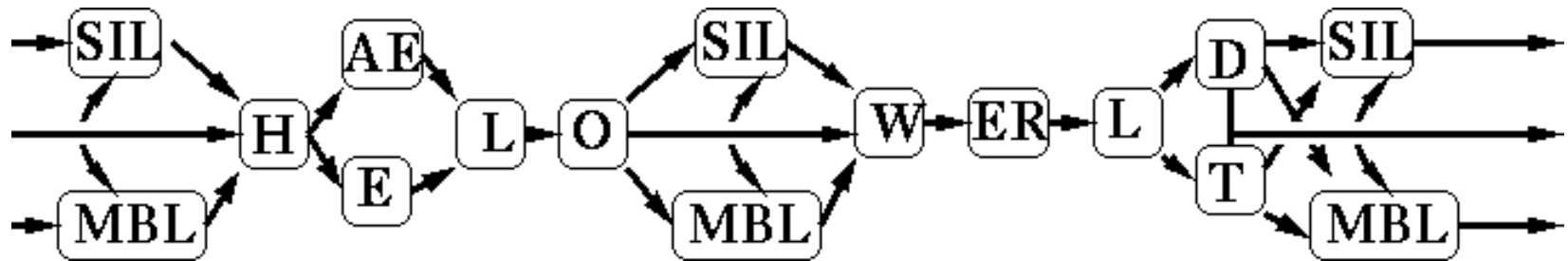
Introduce new acoustic units such that the lexicon looks like:

TRUE	T(R) R UW
TRAIN	T(R) R EY N
TABLE	T(vowel) EY B L
TELL	T(vowel) EH L

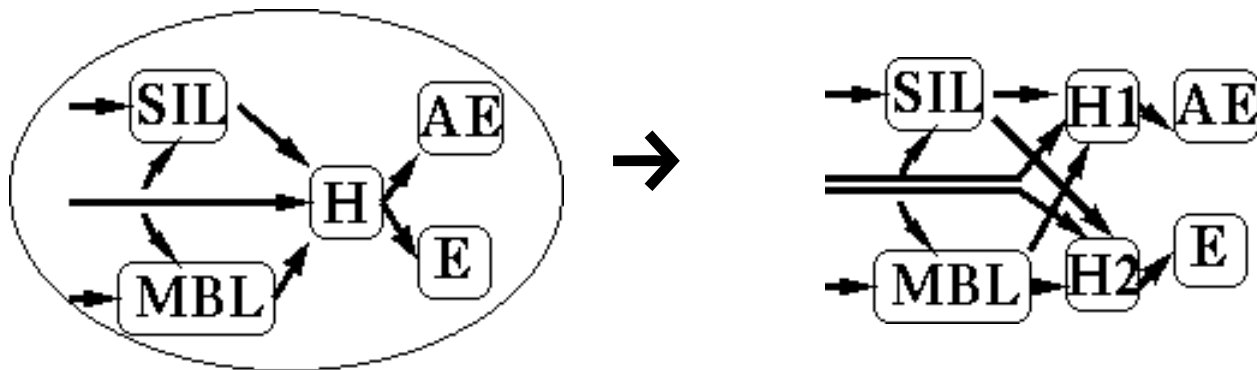
→ use **context dependent** models of the phoneme **T**

Context Dependent HMMs

A context independent HMM for the sentence "HELLO WORLD" could look like this:



Making the phoneme **H** dependent on its successor, becomes:



- Introducing context dependencies reduce recognition errors by 30% to 50%

Speech Units

Speech units: words, syllables, monophones, biphones, triphones, quintphones, polyphones, sub-*n*-phones

Note: These units are not the same as acoustic atoms! Every unit can consist of several atoms

The pronunciation lexicon entry for the word *HELLO* could be:

words	HELLO
syllables	HE LOW
monophones	H EH L OW
left biphones	H(sil) EH(H) L(EH) OW(L)
right biphones	H(EH) EH(L) L(OW) OW(sil)
triphones	H(sil,EH) EH(H,L) L(EH,OW) OW(L,sil)
quintphones	H(sil,sil EH,L) EH(sil,H L,OW) L(H,EH OW,sil) OW(EH,L sil,sil)
polyphones	H(sil EH,L,OW,sil) EH(sil,H L,OW,sil) L(sil,H,EH OW,sil) ...
subtriphones	H-b(sil,EH) H-e(sil,EH) EH-b(H,L) EH-e(H,L) L-b(EH,OW) ...
generalized	H-b(17) H-e(33) EH-b(2) EH-e(71) L-b(40) L-e(55) ...

Crossword Context Modeling

Observation:

Not only the sound of phonemes in the middle of a word depends on the context, but also at the beginning and at the end

from the pronunciation dictionary:

triphones	H(sil,EH) EH(H,L) L(EH,OW) OW(L,sil)
-----------	--------------------------------------

Should not assume that the left context of the word is always "silence"

Solution:

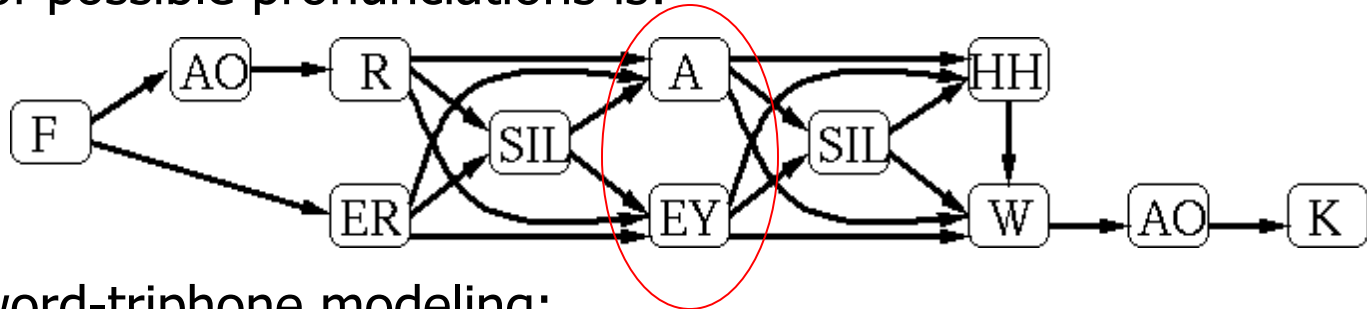
Extend context modeling **across** word boundaries while building a sentence HMM

Crossword Context Modeling

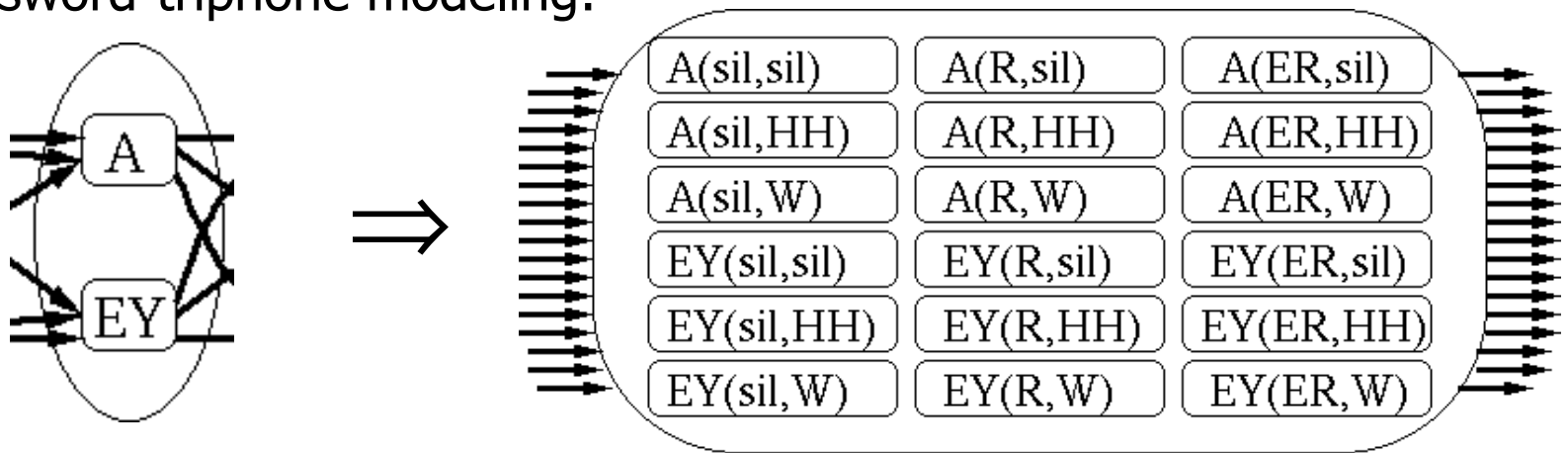
Given utterance "**for a walk**"
with the pronunciation dictionary:

FOR	F AO R	F ER
A	EY	A
WALK	W AO K	HH W AO K

The graph of possible pronunciations is:



With crossword-triphone modeling:



Now imaging Crossword Quintphone modeling!!!

Position Dependent Modeling

Observation:

The same phoneme (with the same context) is pronounced differently depending on its position in the word

e.g.: the D-phoneme in IDAHO compared to the same triphone in the word sequence MY DAD

Solution:

Use position dependent models: ***wb*** indicates a word boundary

MY DAD	M(wb)(sil,AY)	AY(we)(M,D)	D(wb)(AY,AE)	AE(D,D)	...
IDAHO		AY(wb)(sil,D)	D(AY,AE)	AE(D,HH)	...

Problems with Context Dependent Modeling

Some figures for the Wall Street Journal Task (Set 0 + 1):

Training Utterances	45,000
Training Vocabulary Size	15,000
Monophones	50
Position Dependent Triphones	70,000
Position Dependent Subquintphones	800,000

Typical parameter space: 64 48-variate Gaussians per codebook:
800,000 continuous densities => 10 GBytes!

With only 80 hours of training speech, this means:
on average less than one training vector (10ms) per Gaussian

→ Combine multiple contexts into same context-class (parameter tying)

Tying of Contexts

Knowledge based:

let linguist define classes C_1, C_2, \dots of phonemes (e.g. vowels, consonants, fricatives)

then define generalized triphones: **Monophone**($C_{\text{left}}, C_{\text{right}}$)

Backoff procedure based on amount of training data:

```
if "enough" training data for triphones
    → use the triphone model PhoneM(PhoneL, PhoneR)
else:
    if "enough" training data for right/left context biphones
        → use PhoneM(PhoneL) or PhoneM(PhoneR) or both
    else
        → use the context independent model PhoneM
```

Data driven:

use some algorithm together with an optimization criterion to decide which contexts should be tied

Clustering of Context

Unsupervised Clustering (bottom up):

1. start with classes $C_i = \{ \text{Phone}_i \}$
2. compare all class pairs: C_i with $C_j (j > i)$
3. if we find that C_i and C_j are "**similar enough**"
 replace C_i with $C_i + C_j$
 remove C_j
4. continue until satisfied

Clustering of Context

Unsupervised Clustering (top down):

- 1.** start with class $C_0 = \{ \text{context}_1, \text{context}_2, \dots, \text{context}_n \}$
- 2.** anticipate all possible splits of every class C_i into two subclasses
- 3.** if we find that it is good to split C_i then replace C_i with its two subclasses
- 4.** continue with step 2 until satisfied

Clustering of Context

Unsupervised Clustering (top down):

Problem:

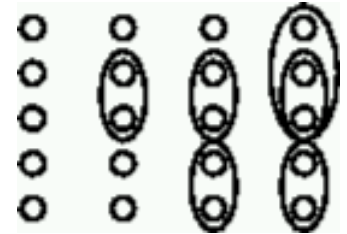
- If we start with n different contexts of the same phoneme then there are 2^n possible separations!
- Most real-world cases have hundreds of contexts
- This makes the algorithm not applicable

Bottom-Up vs. Top-Down Clustering

There are two different approaches to clustering:

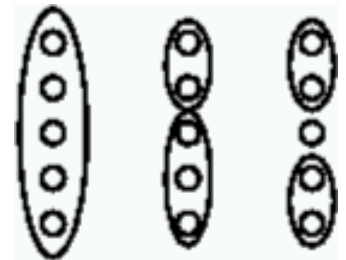
bottom-up clustering (agglomerative)

look for good combination of two classes into one

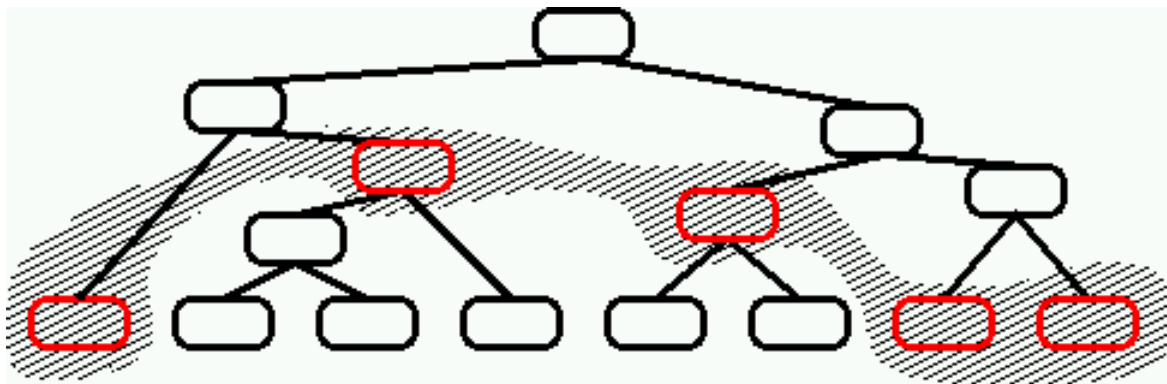


top-down clustering (divisive)

look for good separation of a class into two subclasses



Both approaches result in a clustering tree:



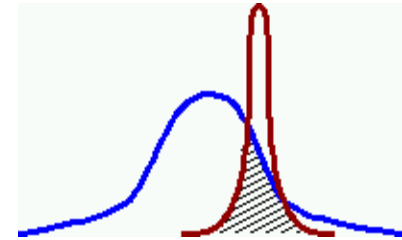
Distances between Model Clusters

Use distance measure between model clusters to

- decide whether a class separation or combination is "good"
- find out which separation/combination is best

Continuous parametric models:

e.g. $d(f,g) = \int \min(f(x),g(x))$



or Kullback-Leibler distances, and others: $KL(f,g) = \sum f(x_i) \log f(x_i) / g(x_i)$

In general (but typically for nonparametric models):

Entropy-distance: $d(f,g) = H(f+g) - 1/2 \cdot H(f) - 1/2 \cdot H(g)$,
where $H(f)$ is the entropy of the function f , and $f+g$ is the combined model

Combining two models \rightarrow lose parameters \rightarrow lose information

- The entropy distance measures the loss of information
- Our goal is to lose as little information as possible

Discrete Entropy Distance

Review:

Semicontinuous and discrete HMMs are represented by discrete distributions

For a discrete distribution $f[i]$ the entropy:

$$H(f) = - \sum_i f[i] \log_2 f[i] = \sum_i f[i] \log_2 (1/f[i])$$

$$d(f,g) = H(f+g) - 1/2 \cdot H(f) - 1/2 \cdot H(g)$$



Obvious:

if $f=g$ then $H(f) = H(g) = H(f+g)$, thus $d(f,g) = 0.0$

if $f = \{ 1 \ 0 \}$, $g = \{ 0 \ 1 \}$ then $H(f) = H(g) = 0$, $H(f+g) = 1$, $d(f,g) = 1.0$

Example: $f = \{ 1/2 \ 1/2 \}$, $g = \{ 3/4 \ 1/4 \}$, $f+g = \{ 5/8 \ 3/8 \}$

$$H(f) = 1/2 \cdot \log_2(2) + 1/2 \cdot \log_2(2) = 1.0$$

$$H(g) = 3/4 \cdot \log_2(4/3) + 1/4 \cdot \log_2(4/1) = 0.811$$

$$H(f+g) = 5/8 \cdot \log_2(8/5) + 3/8 \cdot \log_2(8/3) = 0.954$$

$$\Rightarrow d(f,g) = 0.049$$

Weighted Discrete Entropy Distance

Problem: should also consider

given three models $\{M_1, M_f, M_m\}$

M_1 trained with many examples \rightarrow robust

M_f trained with few examples \rightarrow unreliable

M_m trained with more examples than $M_f \rightarrow$ more reliable than M_f

if the distributions of M_f and M_m are the same then;

$d(M_1, M_f)$ should be less than $d(M_1, M_m)$

Solution:

Weight the model entropy by number of training samples, so the commonly used entropy distance is:

$$d(f,g) = (n_f+n_g) \cdot H(f+g) - n_f \cdot H(f) - n_g \cdot H(g)$$

Context Clustering (Kai-Fu Lee)

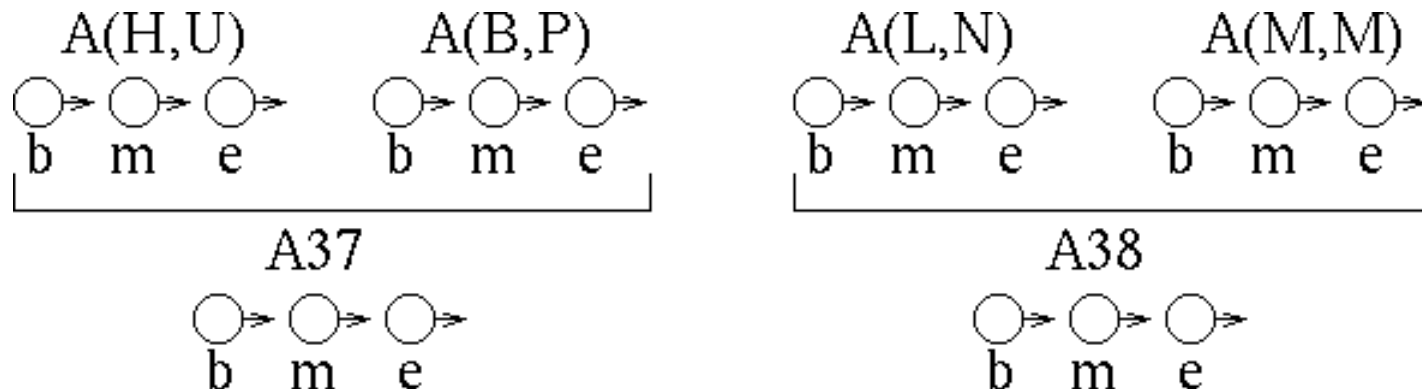
- 1) Train semicontinuous models for all three states of each triphone
(e.g. triphone = T(AE,K)-b T(AE,K)-m T(AE,K)-e)
- 2) Initialize a context class for every triphone
(a class is defined by three distributions: e.g. T_{17} -b T_{17} -m T_{17} -e)
- 3) Compute all distances between different context classes of same phone:
$$d(C_i, C_j) = E(C_i\text{-b}, C_j\text{-b}) + E(C_i\text{-m}, C_j\text{-m}) + E(C_i\text{-e}, C_j\text{-e}),$$
where E is the weighted entropy distance
- 4) Replace the two classes with the smallest distance by their combination
- 5) Try to improve distance by moving any element from any class into any other class

Continue with step 3 while end criterion is not met

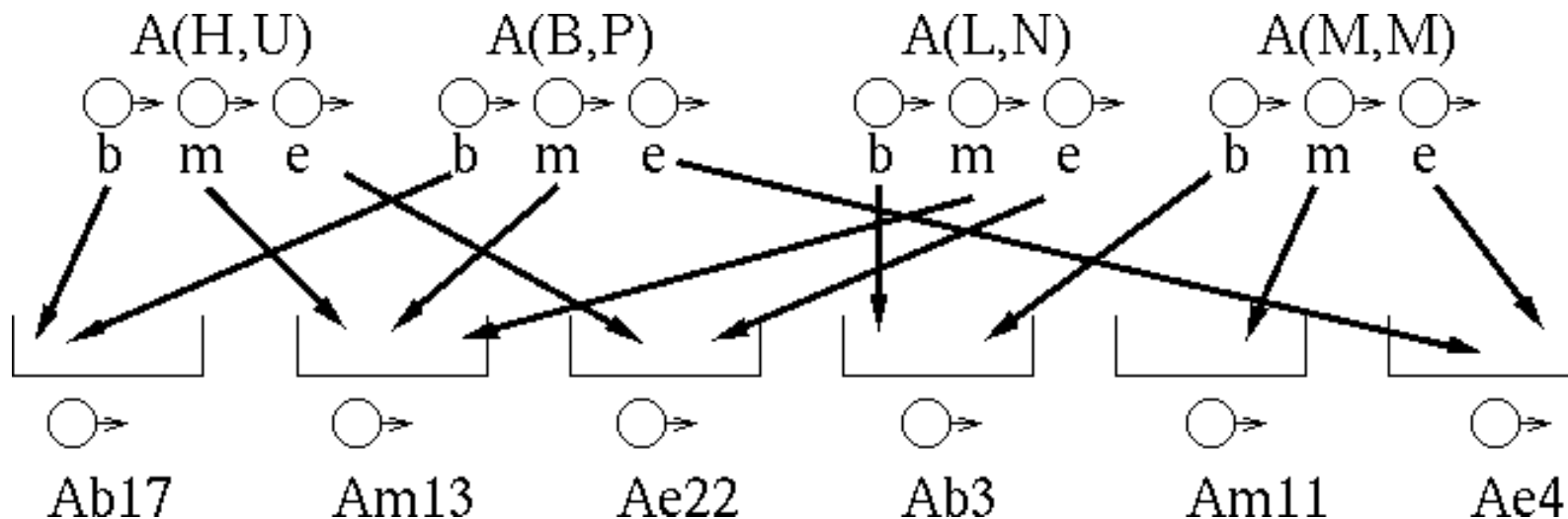
- This algorithm is completely data driven
- Step 5 is expensive but important

Generalized Triphones vs. Senones

Kai-Fu Lee's algorithm produces generalized triphones:



A better approach (M.Hwang) produces generalized subtriphones (**senones**):



Problems with Vocabulary Dependencies

Example Scenario:

During training we have seen the phoneme P_1 in the contexts

$$P_1(P_2, P_3),$$

$$P_1(P_4, P_5),$$

$$P_1(P_6, P_7),$$

$$P_1(P_8, P_9).$$

After clustering we have found the classes:

$$C_1 = \{ P_1(P_2, P_3), P_1(P_4, P_5) \} \quad \text{and} \quad C_2 = \{ P_1(P_6, P_7), P_1(P_8, P_9) \}$$

During testing we would like to recognize the word with phoneme sequence:

$$P_3 P_1 P_7$$

Problem: Do we use C_1 or C_2 to model $P_1(P_3, P_7)$?

Clustering with Decision Trees

Approaches to achieve vocabulary independency:

- 1)** if recognition vocabulary contains untrained context $m(l,r)$ then use context independent model m that was trained on all contexts
- 2)** use the model of a context class that is somehow "*similar*" to the unseen context

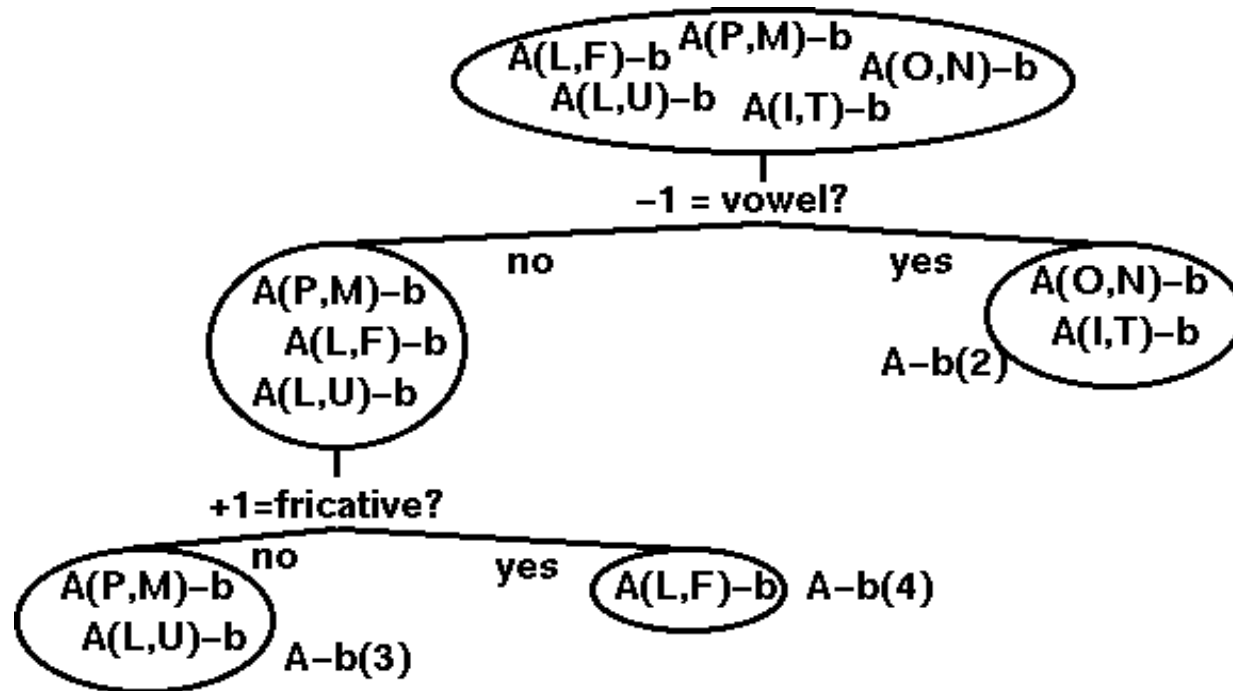
In general:

- if a context has not been seen during training then use some class further up in the hierarchy that was trained
- To make a system independent of the vocabulary, we have to be able to find out in which context class it would have been clustered

Solution:

→ Build a **decision tree** that asks phonetic questions about the context

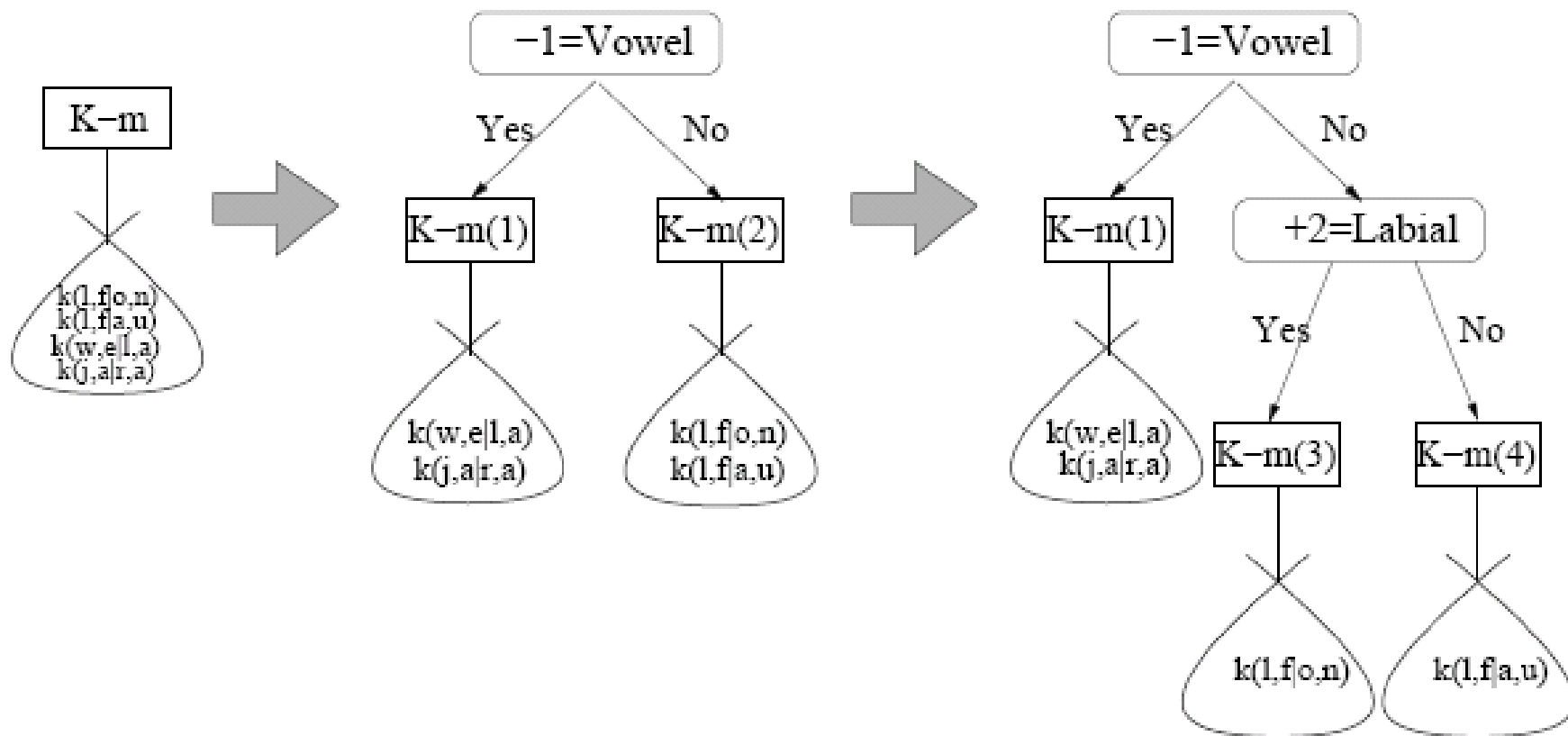
Clustering with Decision Trees



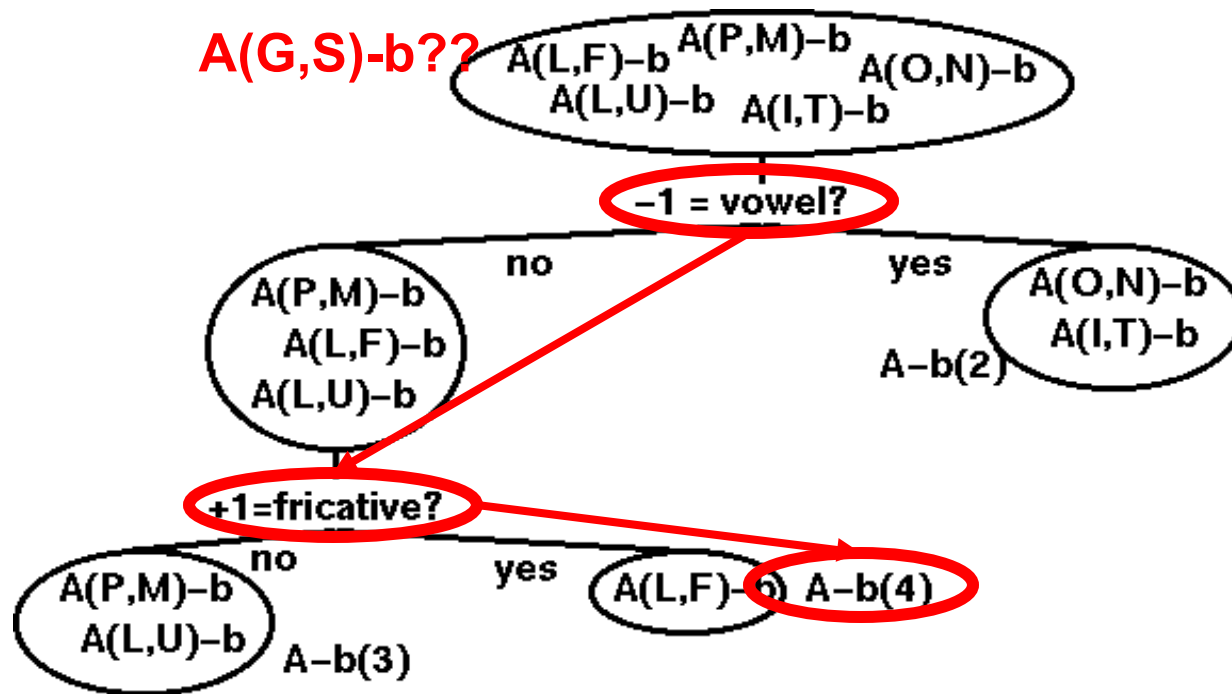
Top-Down Clustering

1. initialize one cluster containing all contexts
2. for all clusters / questions: compute distance of subclusters
3. perform the split that get the largest distance (information gain)
4. continue with step 2 until satisfied (number of clusters)

Growing the Decision Tree



Find the best matching Model



During training five contexts have been seen. These were clustered into three clusters. If we need to model the context **A(G,S)** we will use:

Left context of A is a vowel? (-1 = vowel?) NO, G is not a vowel

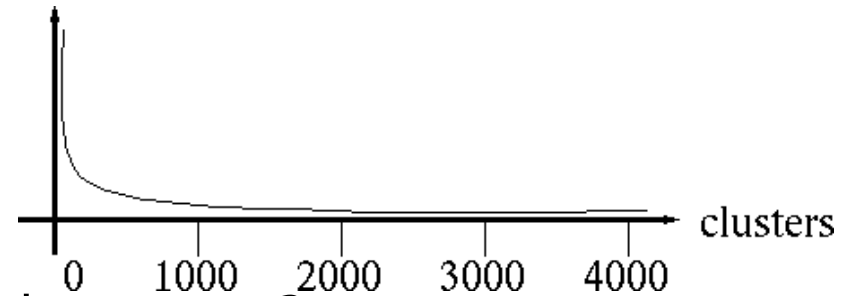
Right context of A is a fricative (+1 = fricative?) YES, S is a fricative

→ use model A-b(4).

- **Disadvantage:** Need expert to define useful questions
- Selection of questions is not critical

Stopping Criterion for Building Decision Trees

Typical optimal entropy distance during clustering:



Question: How many clusters (models) do we want?

- standard answer to many question about the amount of parameters: "As many as our CPU / memory can stand."
- some kind of intelligent guess (based on experience)
- number of samples per cluster does not fall below a certain threshold
- use cross validation set:
 - separate training data into two (or more) subsets A, B , train models from A
 - when computing the distance between clusters C_1 and C_2
 - compute the likelihood P_1 of all data from B that belong to C_1 ,
 - compute the likelihood P_2 of all data from B that belong to C_2 ,
 - and P_{1+2} of all data from B that belong to C_1 or C_2 using the combined class C_1+C_2 , define the distance as $(P_1 \cdot P_2) / P_{1+2}$
 - the likelihood gain of the split will not always be positive

Phonetic Questions

- Knowledge-based
 - Expert in ASR defines “natural classes” based on IPA classification (Example list: Table 9.3 of [Huang, Acero, Hon])
 - nasal: m n ng
 - velar: k g ng
 - labial: w m b p v
- Automatically learned classes (e.g. Rita Singh, CMU)
 - provide phone names and feature properties
 - use acoustic distance to cluster features
 - these become questions for context clustering
- Random Selection of questions (IBM)

Summary

- Pronunciation Variants
- Context Dependent Acoustic Modeling
- From Sentence to Context Dependent HMM
- Speech Units
- Crossword Context Modeling, Problems
- Tying of Contexts
- Clustering of Context
- Bottom-Up vs. Top-Down Clustering
- Distances Between Model Clusters
- Problems with Vocabulary Dependencies
- Clustering with Decision Trees