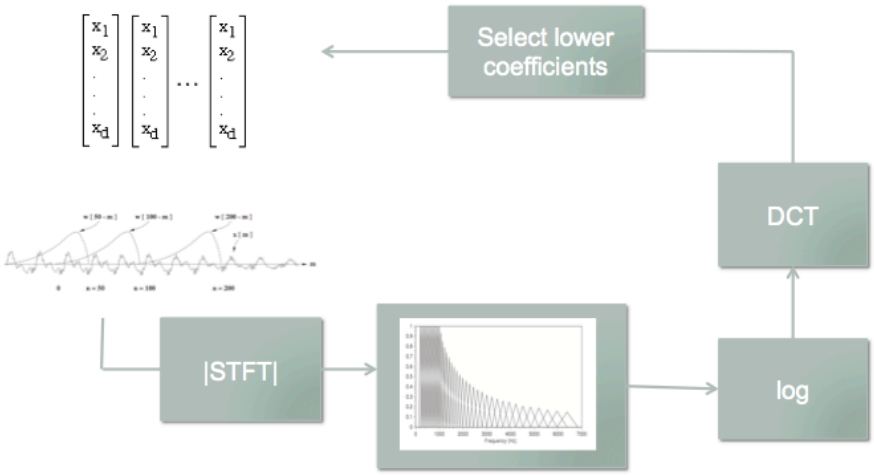
ACOUSTIC MODELING PART II

HMM and Search

Many illustrations provided by courtesy of James Glass

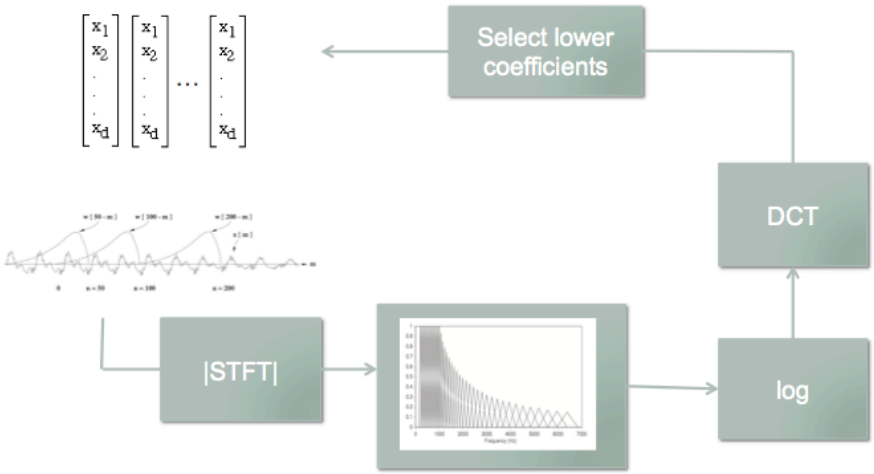
Recap++

- Speech features – MFCC



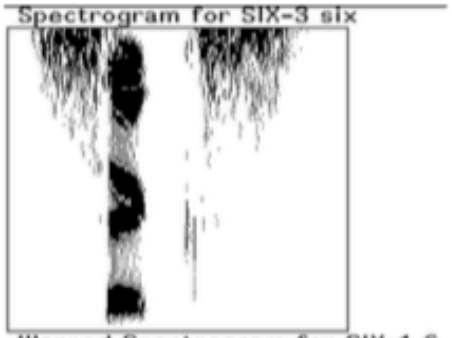
Recap++

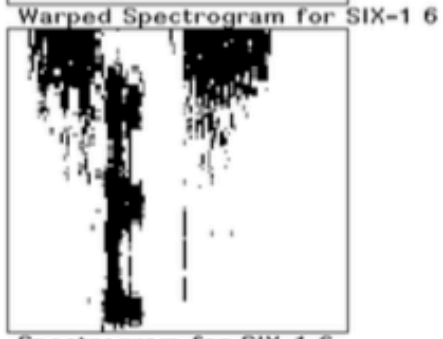
- Speech features – MFCC

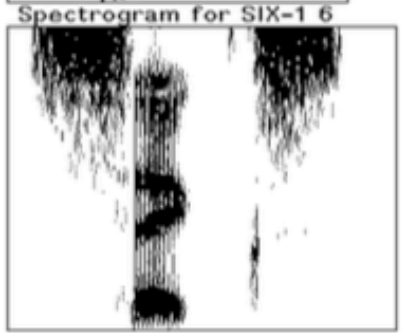


- Word-based recognition
  - DTW – dynamic programming
  - Need template for every word
  - Gowajee detector

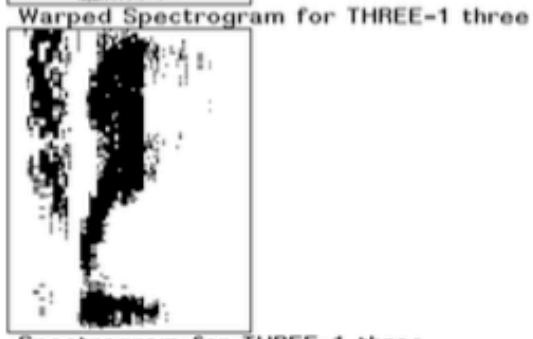
Dynamic Time Warping

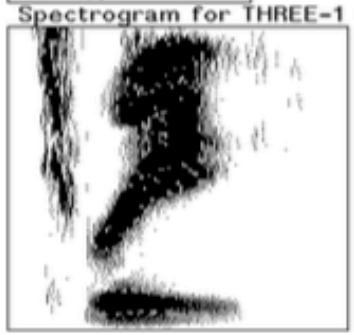






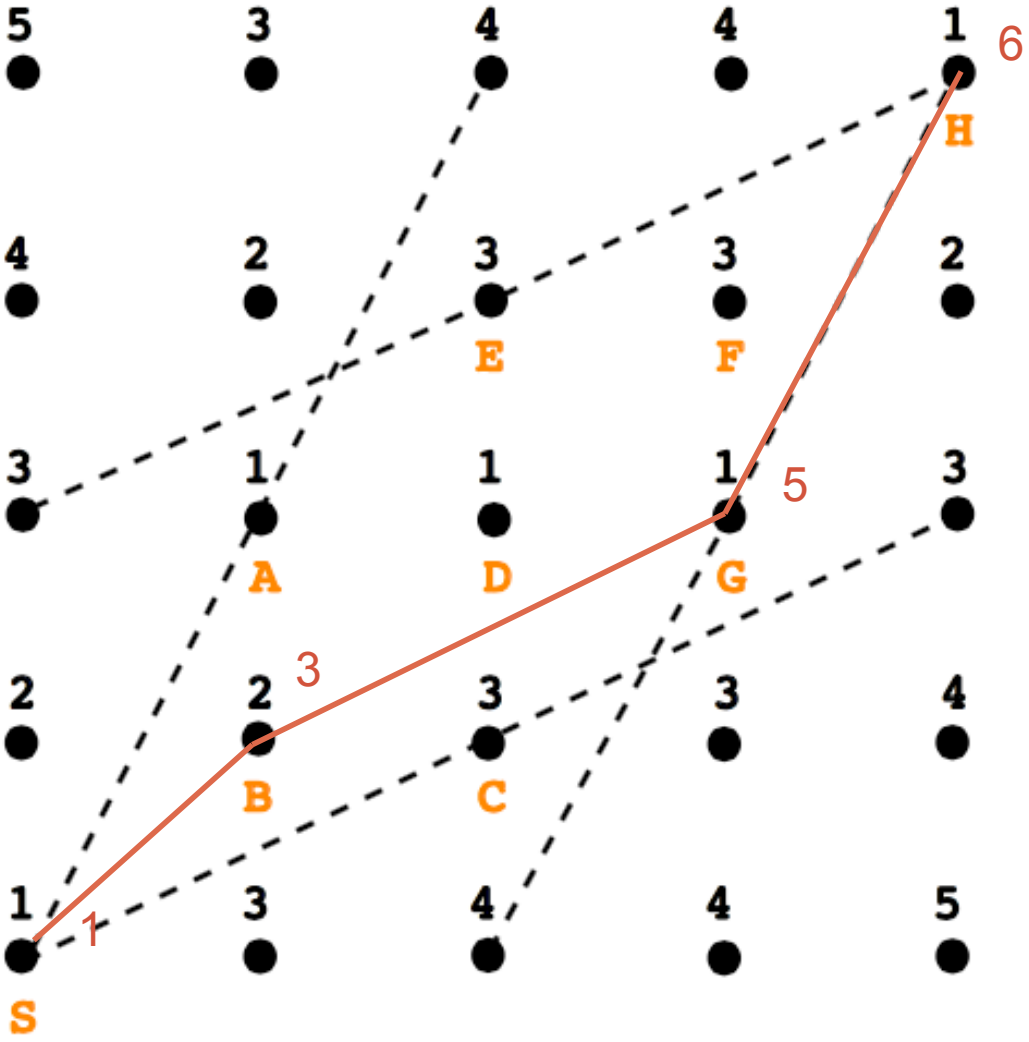
Match

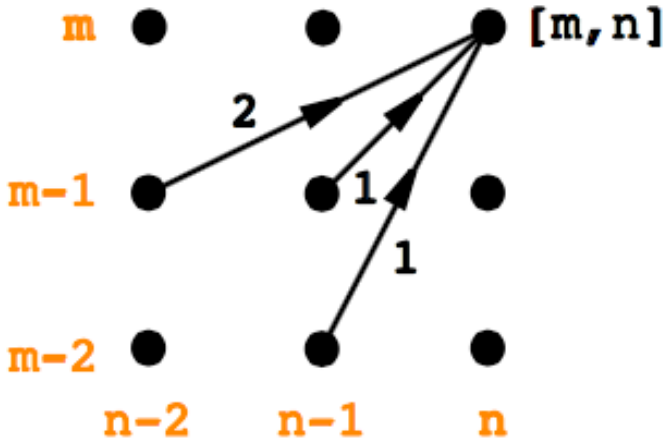




Mismatch

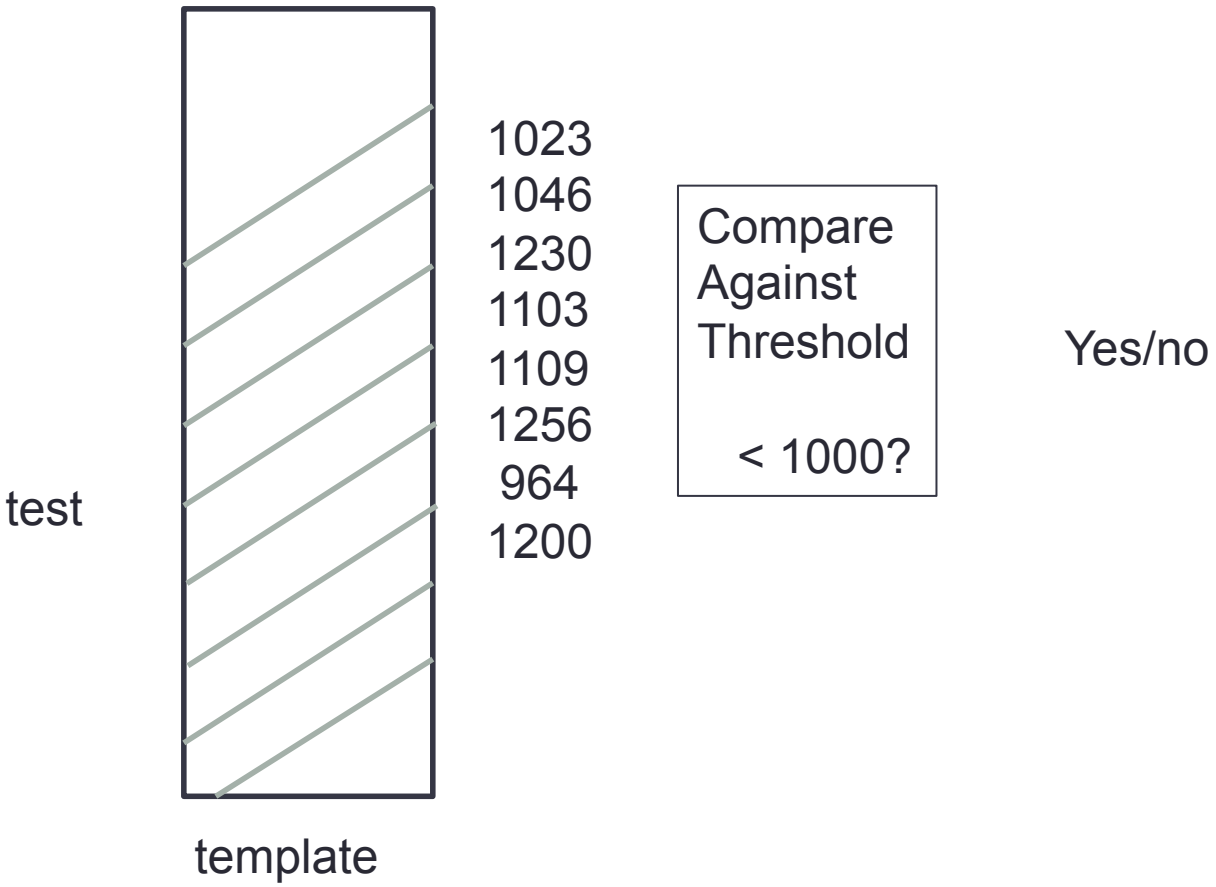
DTW example


5 3 4 4 1
6
H
4 2 3 3 2
E F
3 1 1 1 3
5
A D G
3
2 2 3 3 4
B C
1 3 4 4 5
1
S


m
[m,n]
2
m-1
1
1
m-2
n-2
n-1
n

Gowajee detector

Sliding DTW



3 Questions

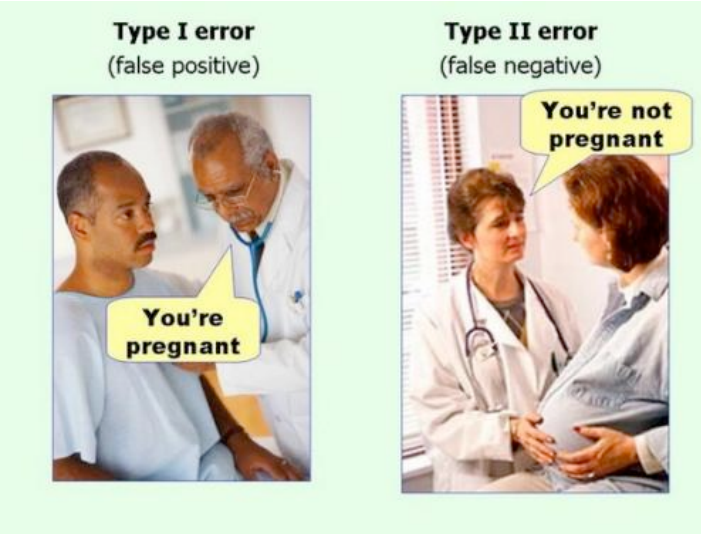
- How to evaluate performance of a detector?
- How to pick a threshold?
- How to compare one detector with another?

Evaluating a detection problem

- Pick a threshold evaluate performance
- Classify one DTW distance, what possible scenarios

		Detector	
		Yes	No
Actual	Yes	True positive	False negative
	No	False Alarm	True negative

True positive + False negative = # of actual yes
False alarm + True negative = # of actual no

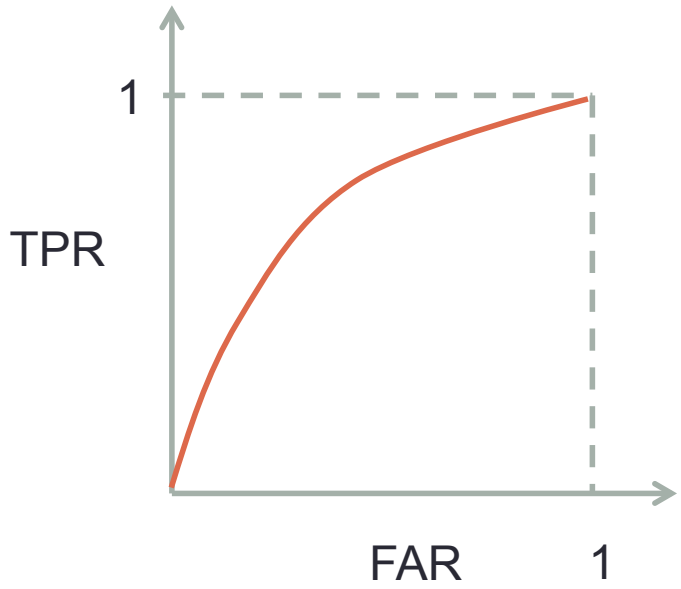


- False alarm and True positive carries all the information of the performance.

https://effectsizefaq.com/2010/05/31/i-always-get-confused-about-type-i-and-ii-errors-can-you-show-me-something-to-help-me-remember-the-difference/

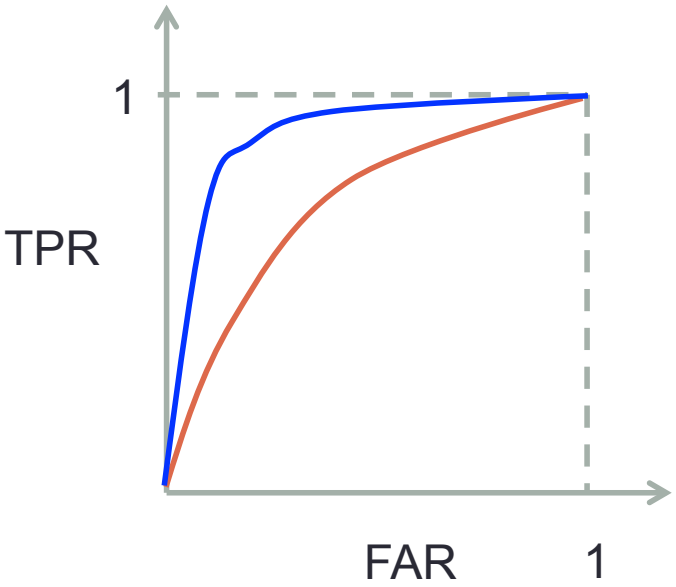
Receiver operation Characteristic (RoC) curve

- What if we change the threshold
- FA TP is a tradeoff
- Plot FA rate and TP rate as threshold changes



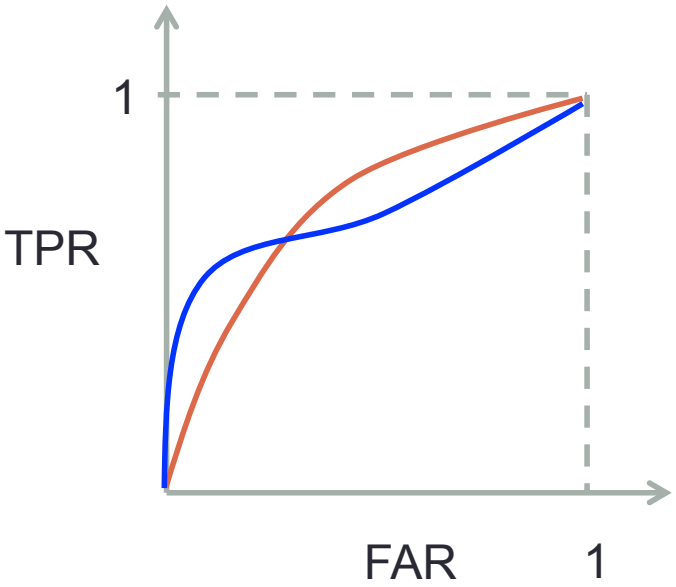
Comparing detectors

- Which is better?



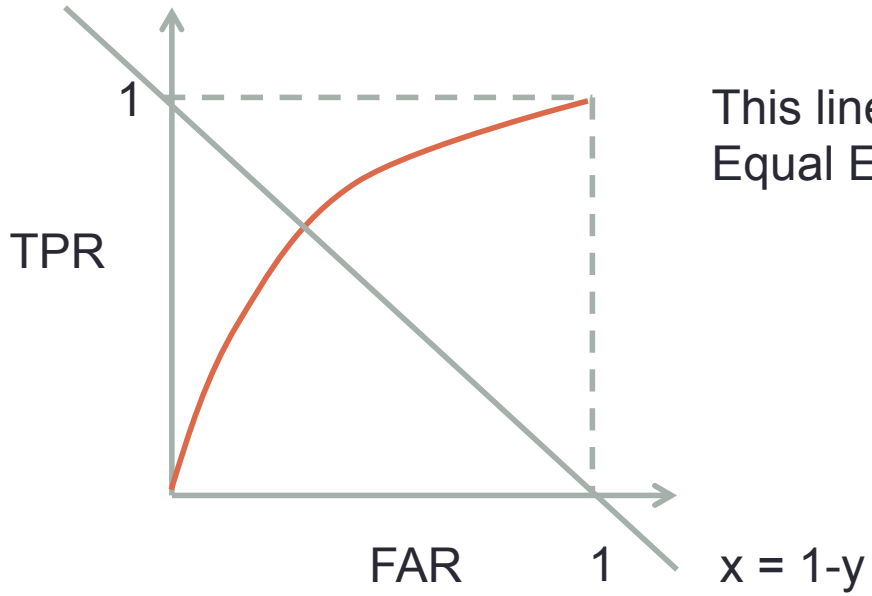
Comparing detectors

- Which is better?



Selecting the threshold

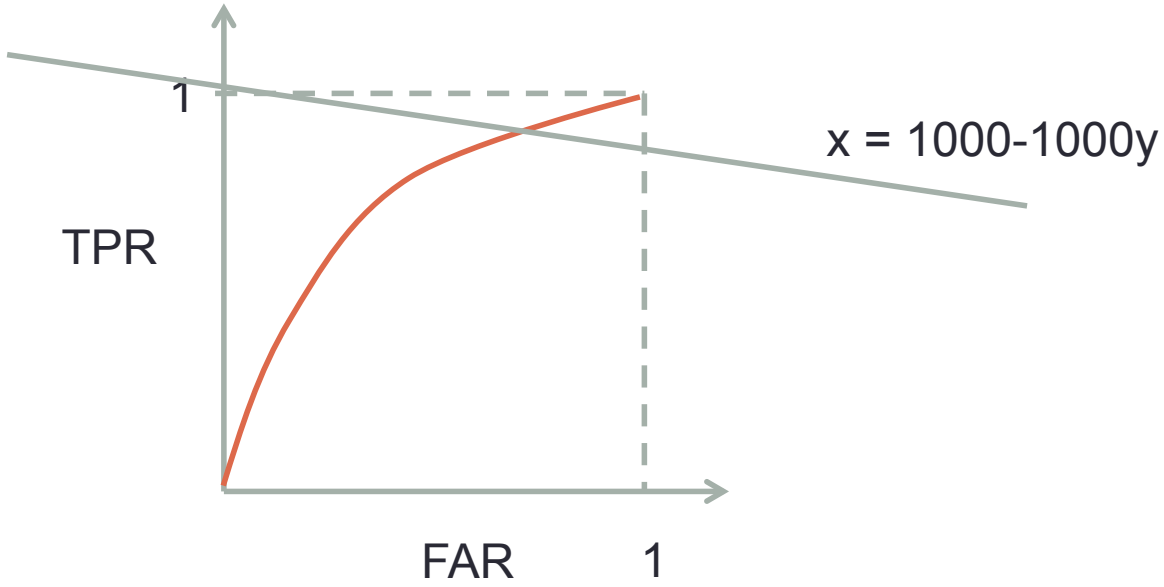
- Select based on the application
- Trade off between TP and FA. Know your application, know your users.
  - A miss is as bad as a false alarm FAR = 1-TPR => x = 1-y



This line has a special name
Equal Error Rate (EER)

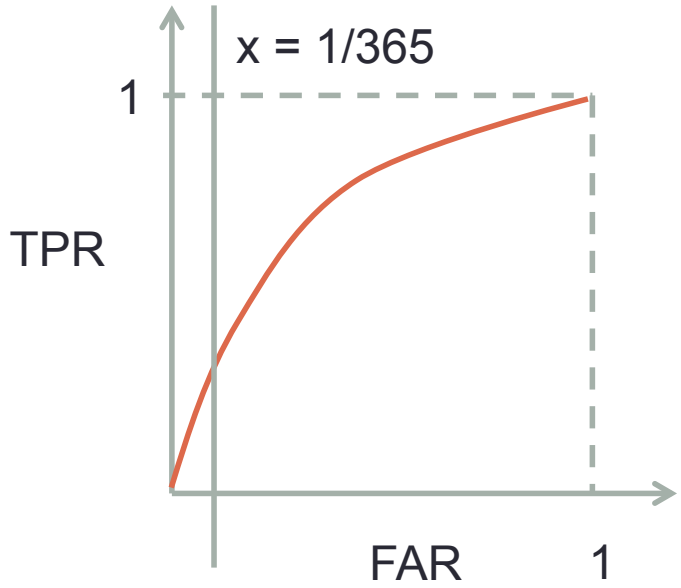
Selecting the threshold

- Select based on the application
- Trade off between TP and FA. Know your application, know your users. Is the application about safety?
  - A miss is 1000 times more costly than false alarm.
    - FAR = 1000(1-TPR) => x = 1000-1000y

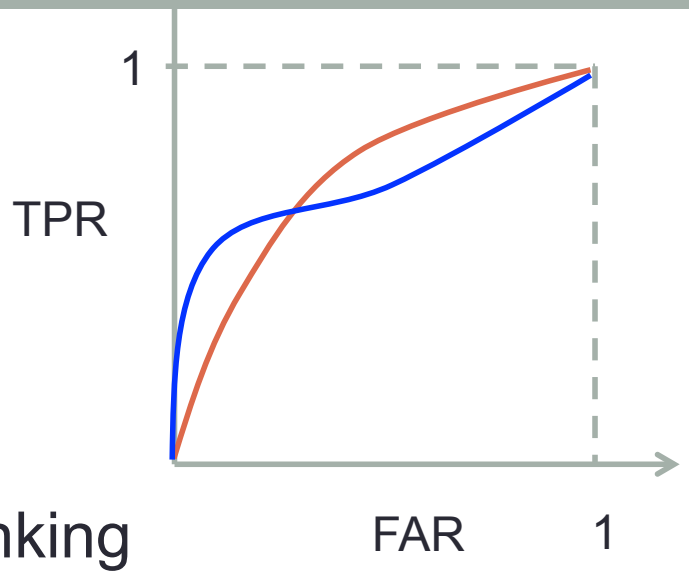


Selecting the threshold

- Select based on the application
- Trade off between TP and FA.
  - Regulation or hard threshold
  - Cannot exceed 1 False alarm per year
    - If 1 decision is made everyday, FAR = 1/365



Comparing detectors

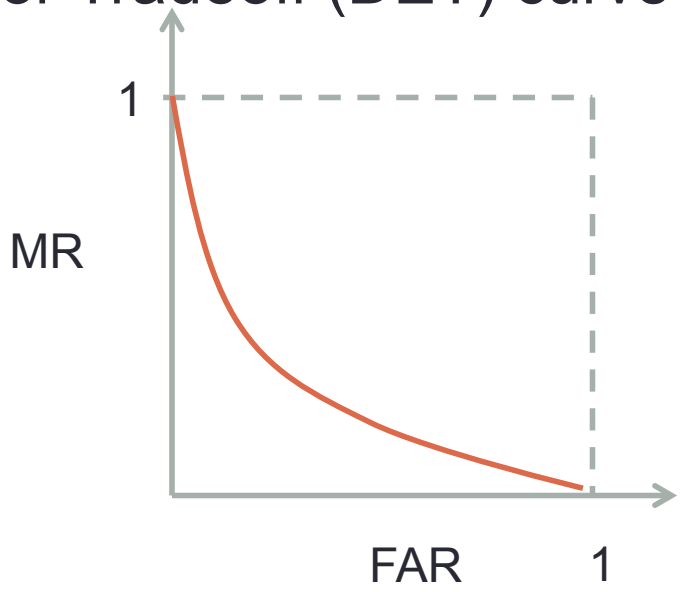


- Which is better?

- Speaker verification for phone banking
- Gowajee detector for home
- Gowajee detector with speaker verification for unlocking phones
- A detector for the command “stop” for the robotic forklift.

Misc. about RoC

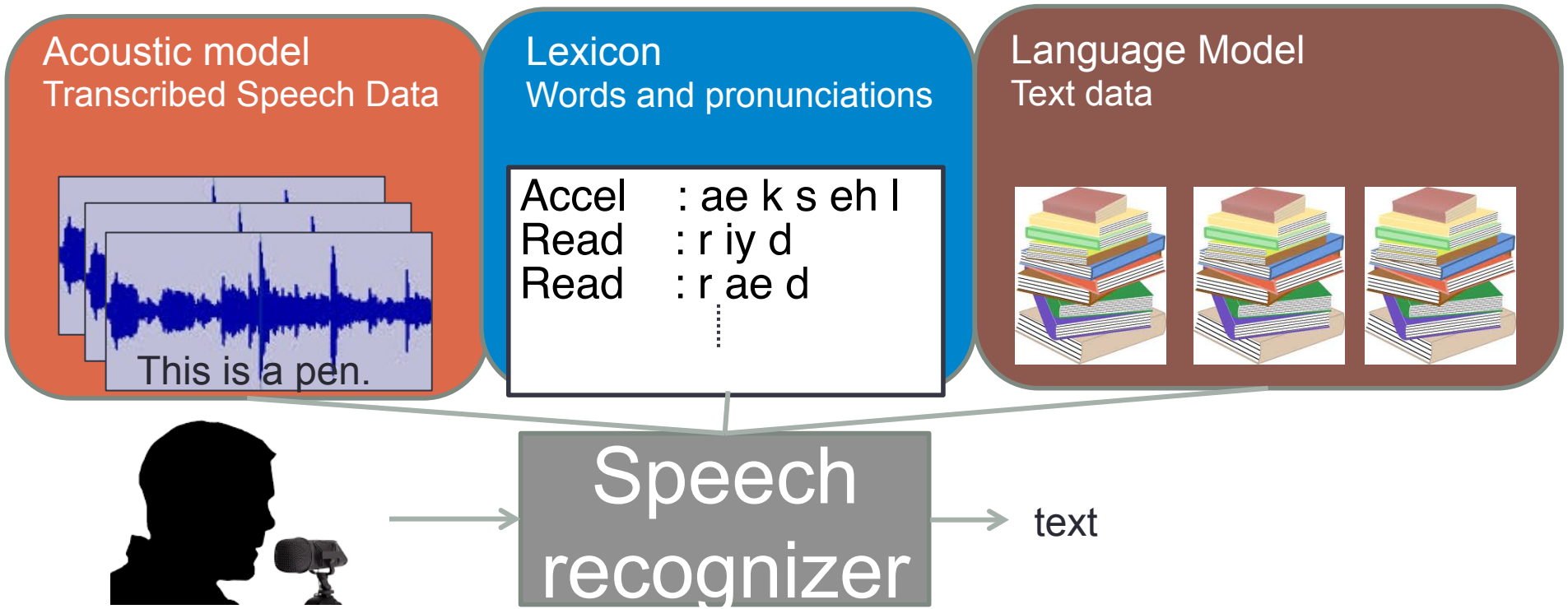
- Ways to compress RoC to just a number for easier comparison -- use with care!!
  - EER
  - Area under the curve
  - F score
- Other similar curve - Detection Error Tradeoff (DET) curve
  - Plot False alarm vs Miss rate
- Can plot on log scale for clarity



The ASR equation

$$= argmax_{W,L} P(X \mid L) P(L \mid W) P(W)$$

X - waveform, L - pronunciation, W - words



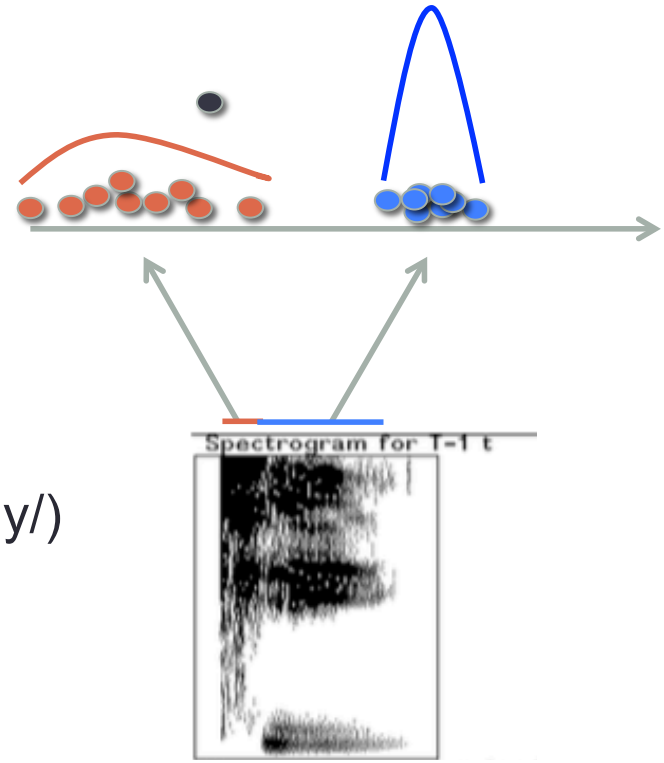
Acoustic Model

- Instead of modeling words, model subwords
  - Syllable
  - Phoneme
  - Sub-phoneme
- Can construct new words using parts
- Less class, more data per class, less overfitting problem

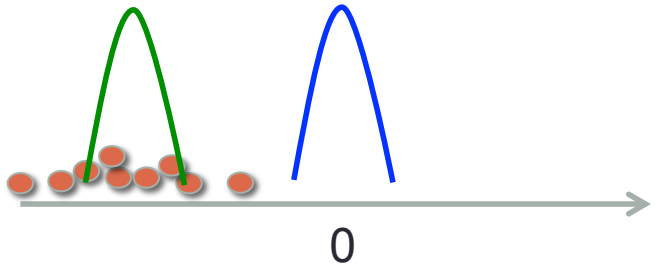
Overfit is when a model considered irrelevant information as important

Acoustic Model big picture

- Training
  - Find all frames with phoneme /iy/
  - Train a P(x | /iy/)
  - Repeat for all phonemes
- Testing
  - For each frame give probability of P(x | /iy/)
  - Repeat for all phonemes
  - Get most likely phoneme
    - Need to consider other constraint



How do we know which model is a good fit for the data



- Model A N(0,2) Model B N(-3,2)
- Maximum likelihood
  - Best model is the model that maximize the probability (likelihood) of the data
  - If each data point is independently drawn from the model, what is the probability
  - Find M that maximizes $P(x_1 \mid M)\, P(x_2 \mid M)\, P(x_3 \mid M) \ldots P(x_n \mid M)$

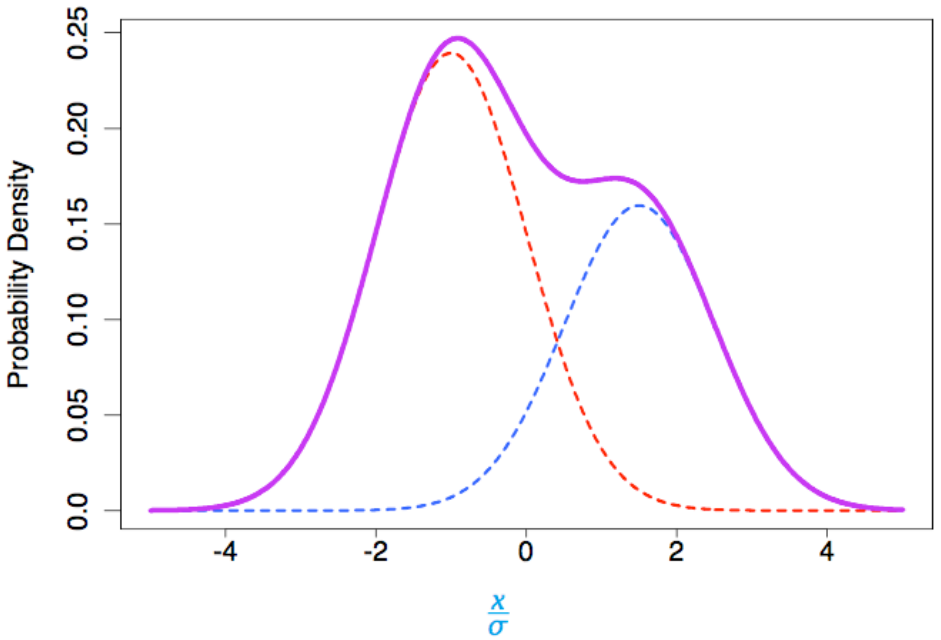
Likelihood of the data

- The best Guassian fit has the same mean and variance as the data

Gaussian Mixture Models (GMMs)

- Model each subword using GMMs
  - Gaussians cannot handle multi-modal data well
  - Consider a class can be further divided into additional factors
    - gender
    - microphone
- Mixing weight makes sure the overall probability sums to 1

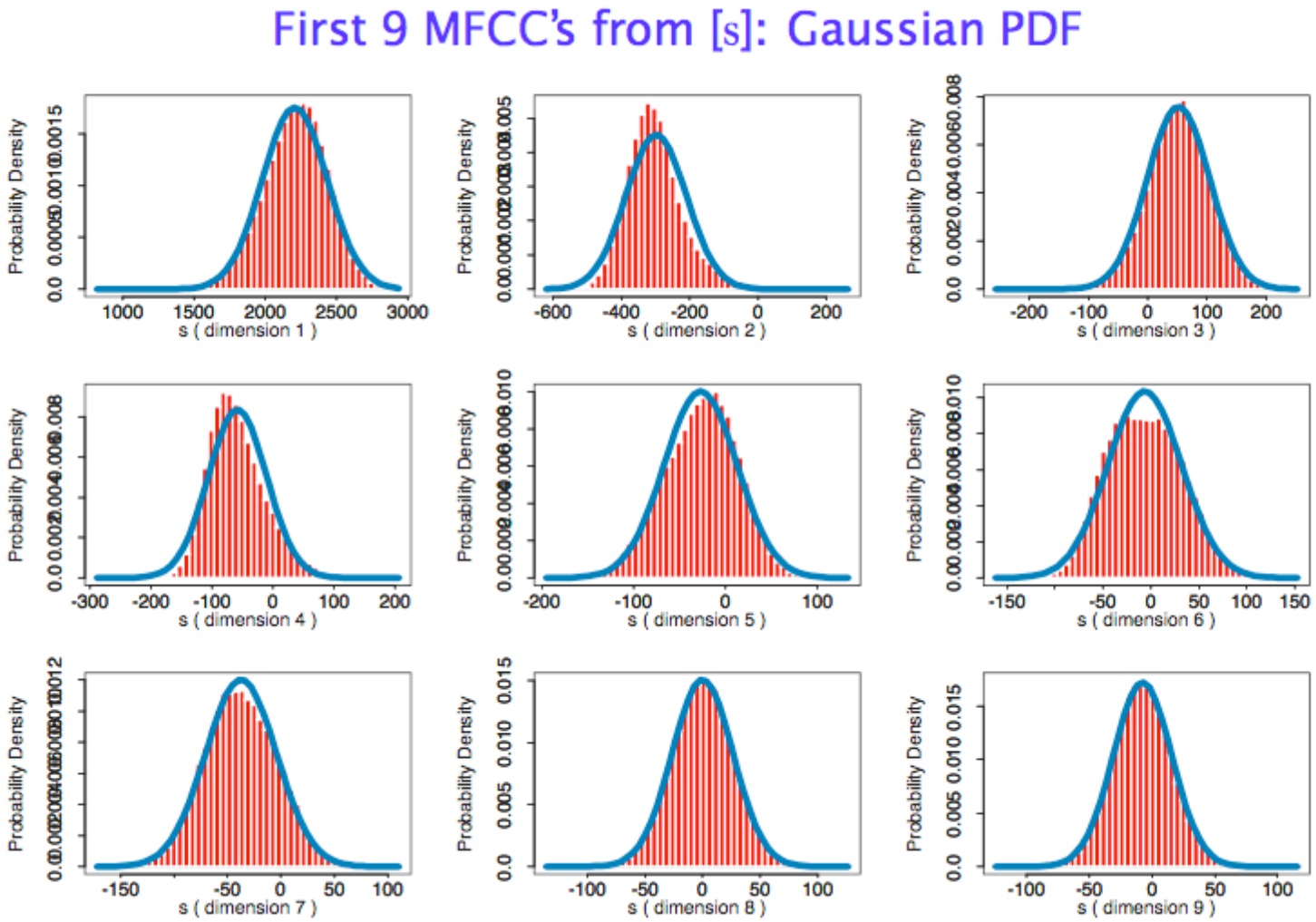
$$P(x) \sim \sum_{k=1}^{K} w_k N(\mu_k, \sigma_k)$$



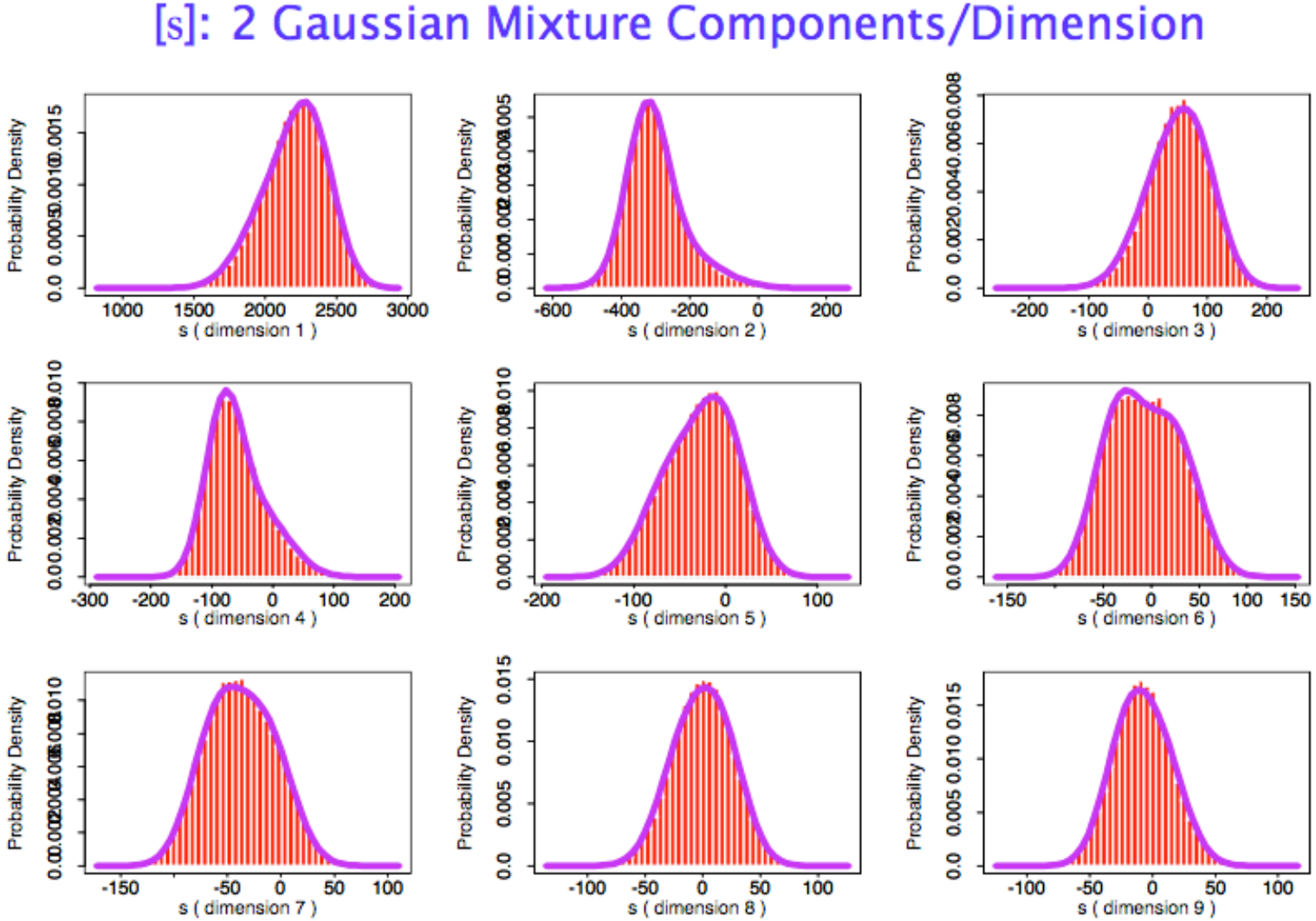
$p(x) = 0.6p_1(x) + 0.4p_2(x)$

$p_1(x) \sim N(-\sigma, \sigma^2)$ $p_2(x) \sim N(1.5\sigma, \sigma^2)$

Model of one Gaussian



Mixture of two Gaussians



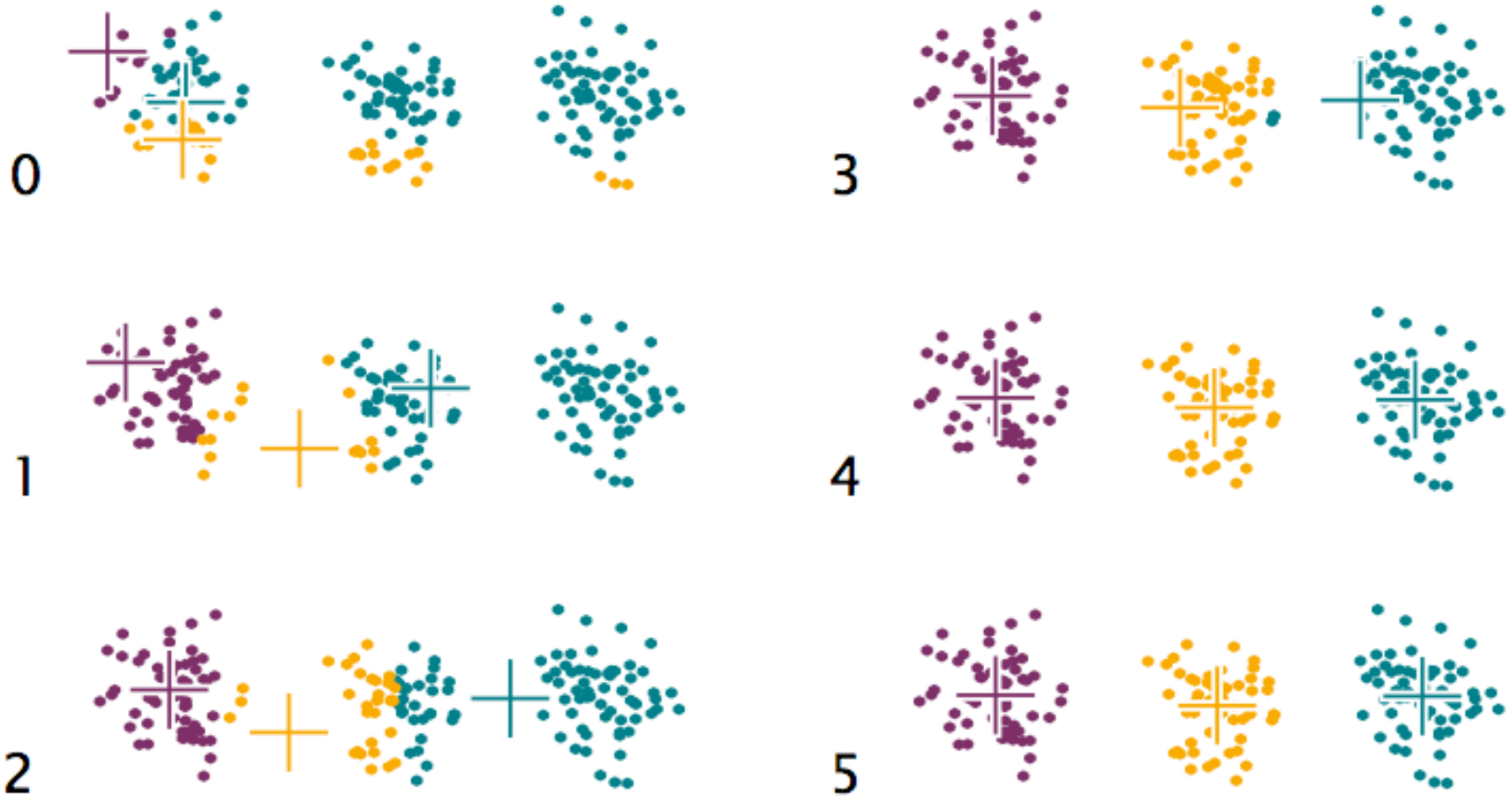
How to estimate GMM parameters

- For a Gaussian, we need to know the mean and variance of data
- For GMM, if we know which data belong to which Gaussian, we can treat it as a regular Gaussian
  - Hard to do
- Let's look at a simpler problem, K-mean clustering

K-mean clustering

- Task: cluster data into groups
- K-mean algorithm
  - Initialization: Pick K data points as cluster centers
  - Assign: Assign data points to the closest centers
  - Update: Re-compute cluster center
  - Repeat: Assign and Update

K-mean clustering



EM algorithm for GMM

- Task: cluster data into Gaussians
- EM algorithm
  - Initialization: Randomly initialize parameters Gaussians
  - Expectation: Assign data points to the closest Gaussians
  - Maximization: Re-compute Gaussians parameters according to assigned data points (find mean and variance)
  - Repeat: Expectation and Maximization

- Note: assigning data points is actually a soft assignment (with probability)

Co-articulation effect

- Phoneme /iy/ in followed by /ng/ is different than followed by /n/
- Need to model phoneme sequences
  - We call this context ***dependent*** model
  - Model sequence of two phonemes – diphone /x-a/
  - 3 phonemes – triphone /a-x-b/
  - 5 phonemes – quintphone /a-b-x-c-d/
  - /x/ is call the center phoneme
- Modeling triphones has N^3 classes!
  - Not enough data
  - Need to cluster some triphones to the same class
  - Some triphones we never seen

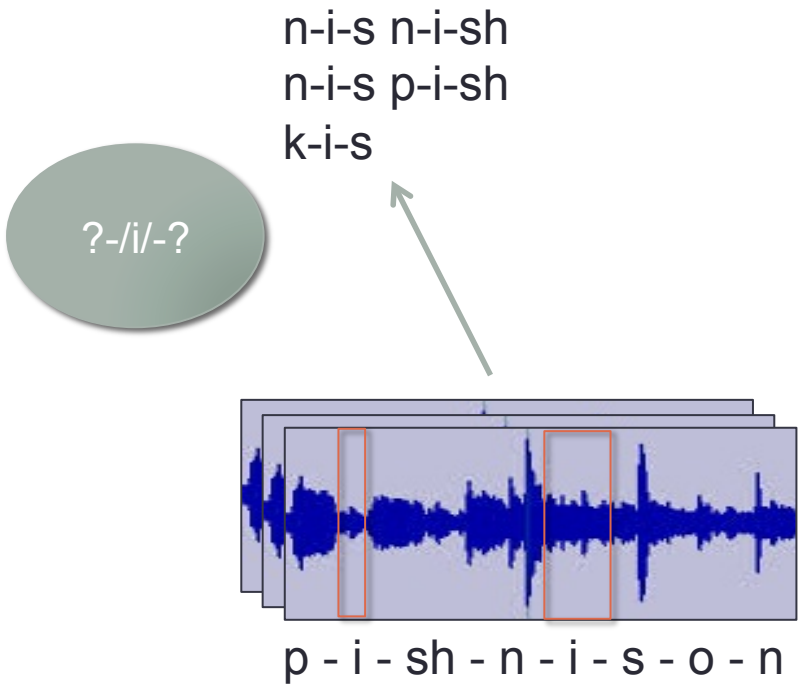
Tree clustering

- Goal: reduce the amount of context dependent classes
- Cluster many triphones into senones
- Cluster according to the center phones
- The center phone is split according to questions
- Select the best question according to maximum likelihood.
- Stop when not enough data to form a cluster

Likelihood $P(x_1 \mid M)\ P(x_2 \mid M)\ P(x_3 \mid M) \ldots P(x_n \mid M)$

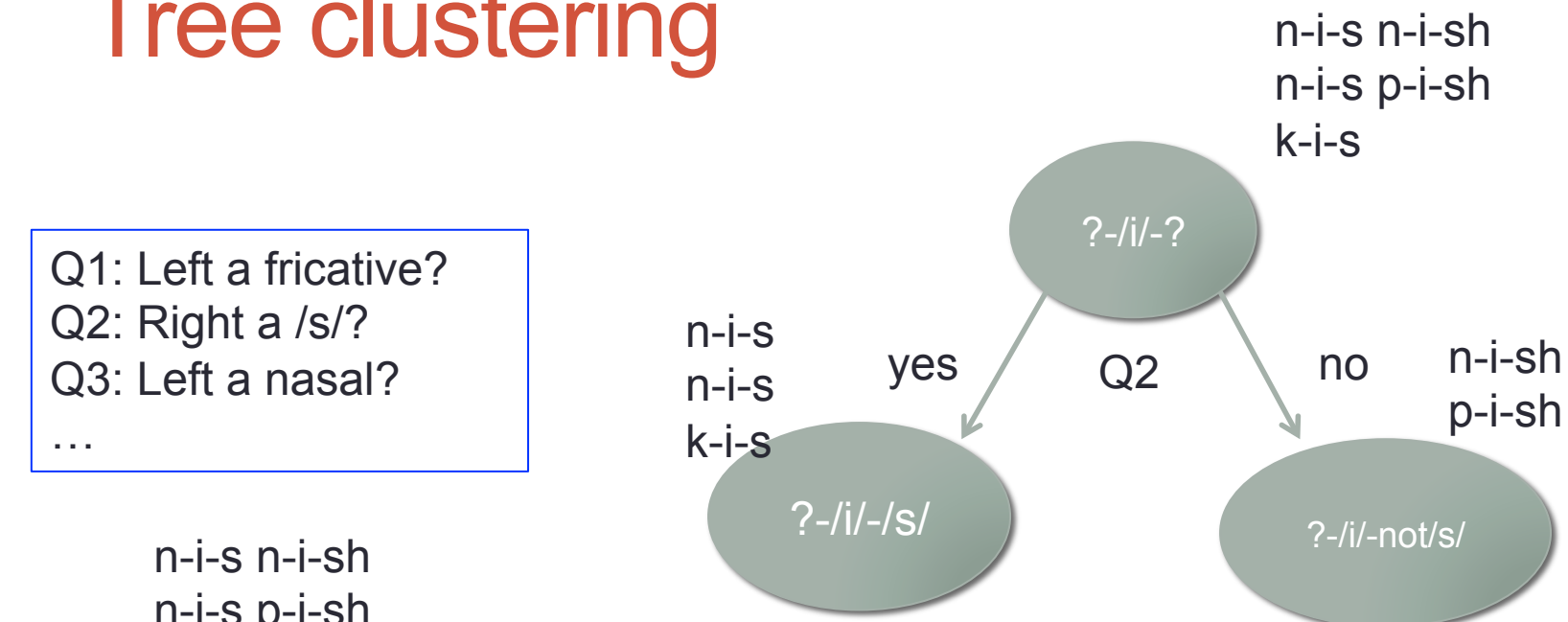
Tree clustering

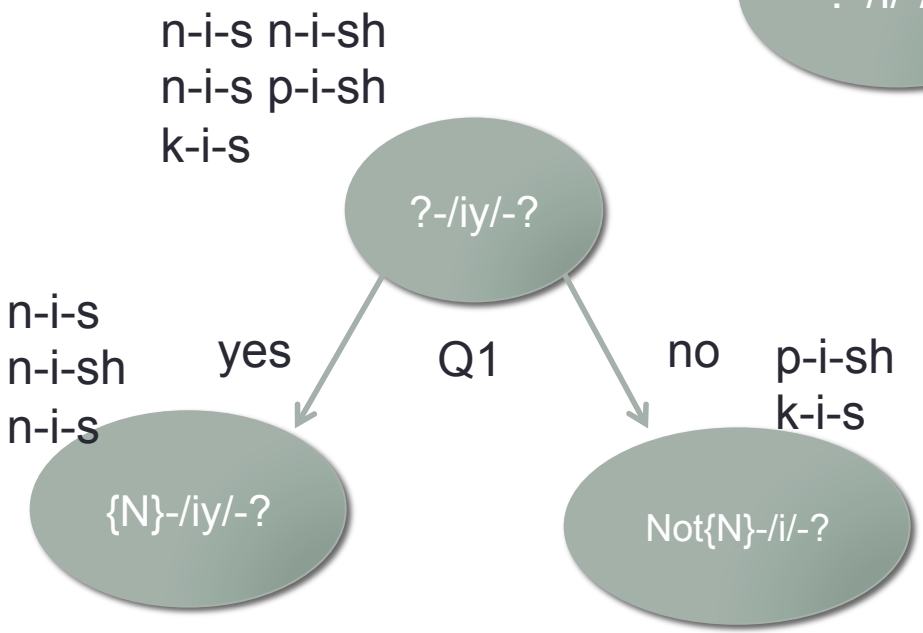
Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



Tree clustering

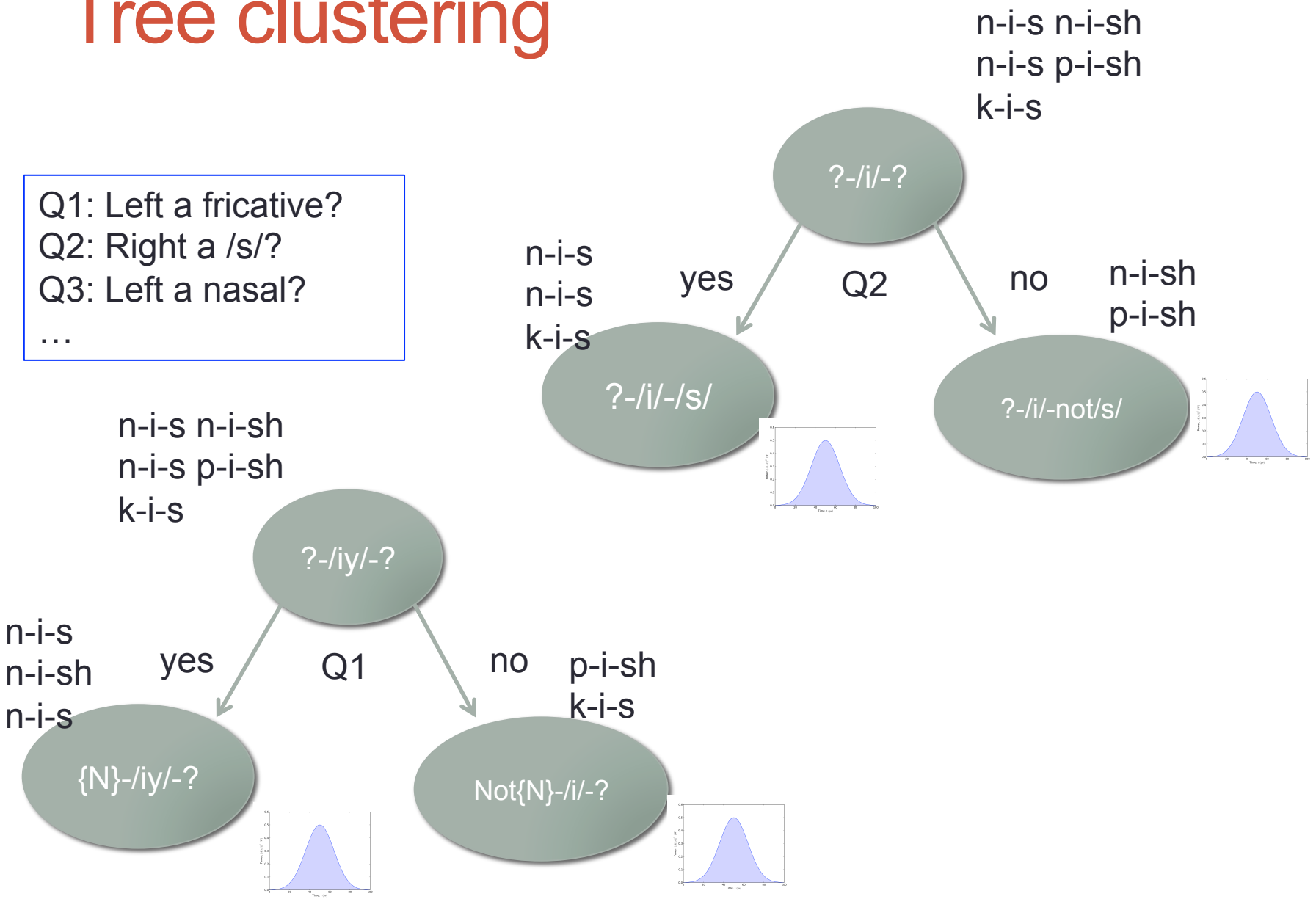
Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…





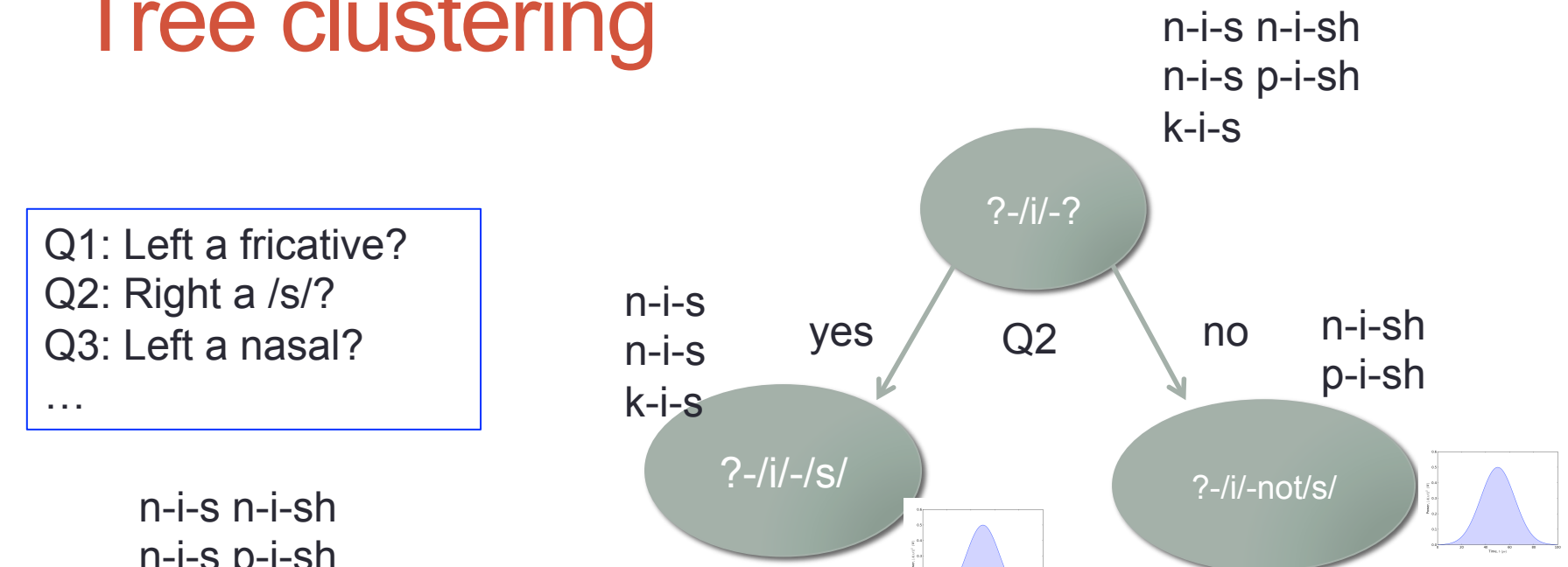
Tree clustering

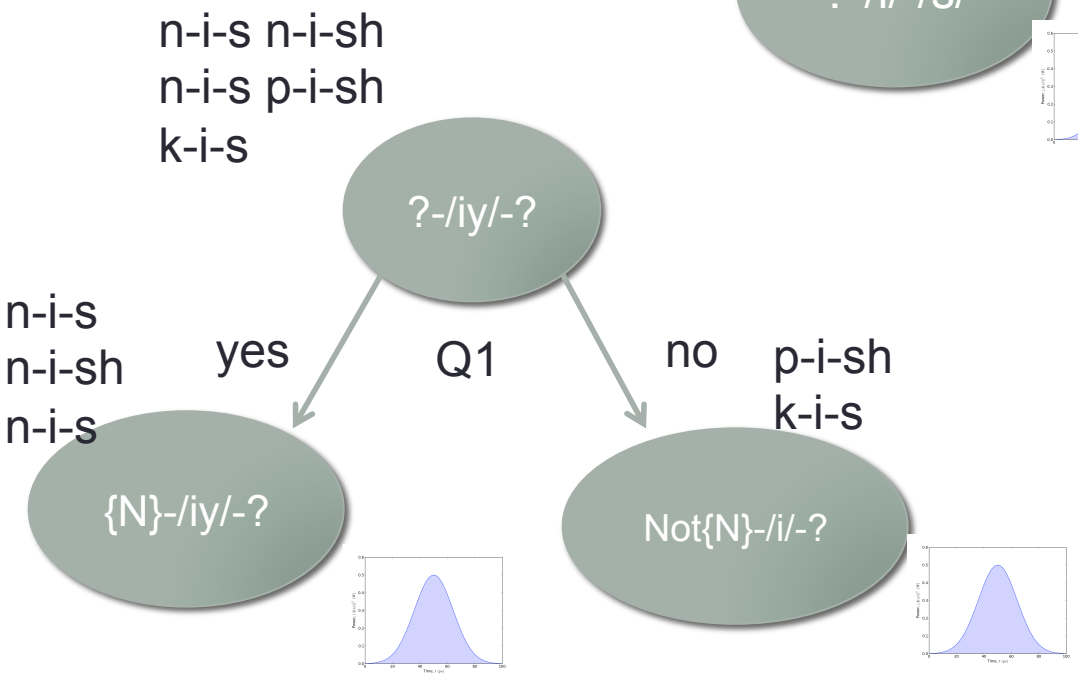
Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



Tree clustering

Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…

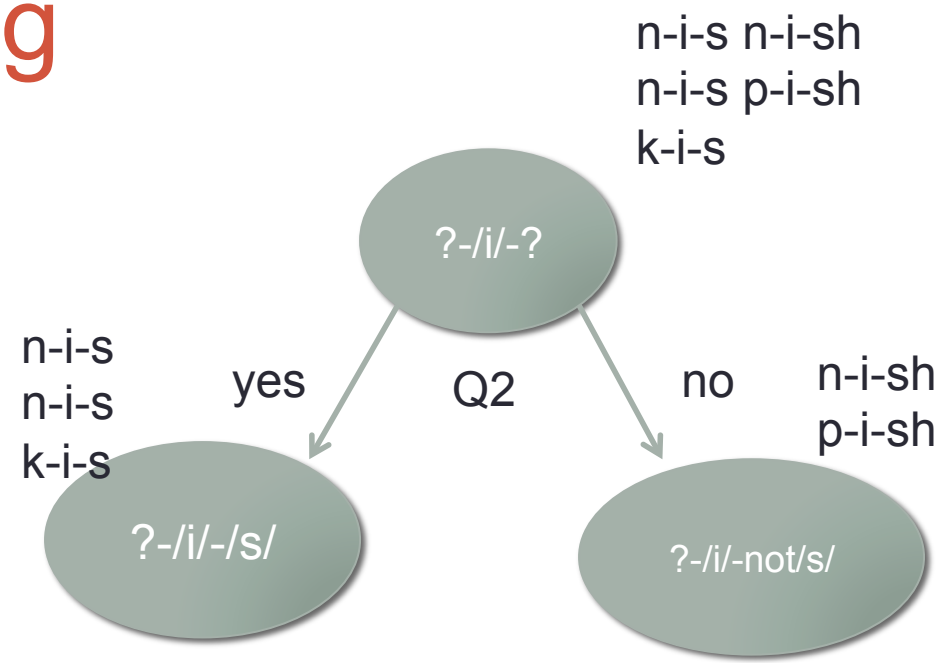




Likelihood Q2 > Likelihood Q1

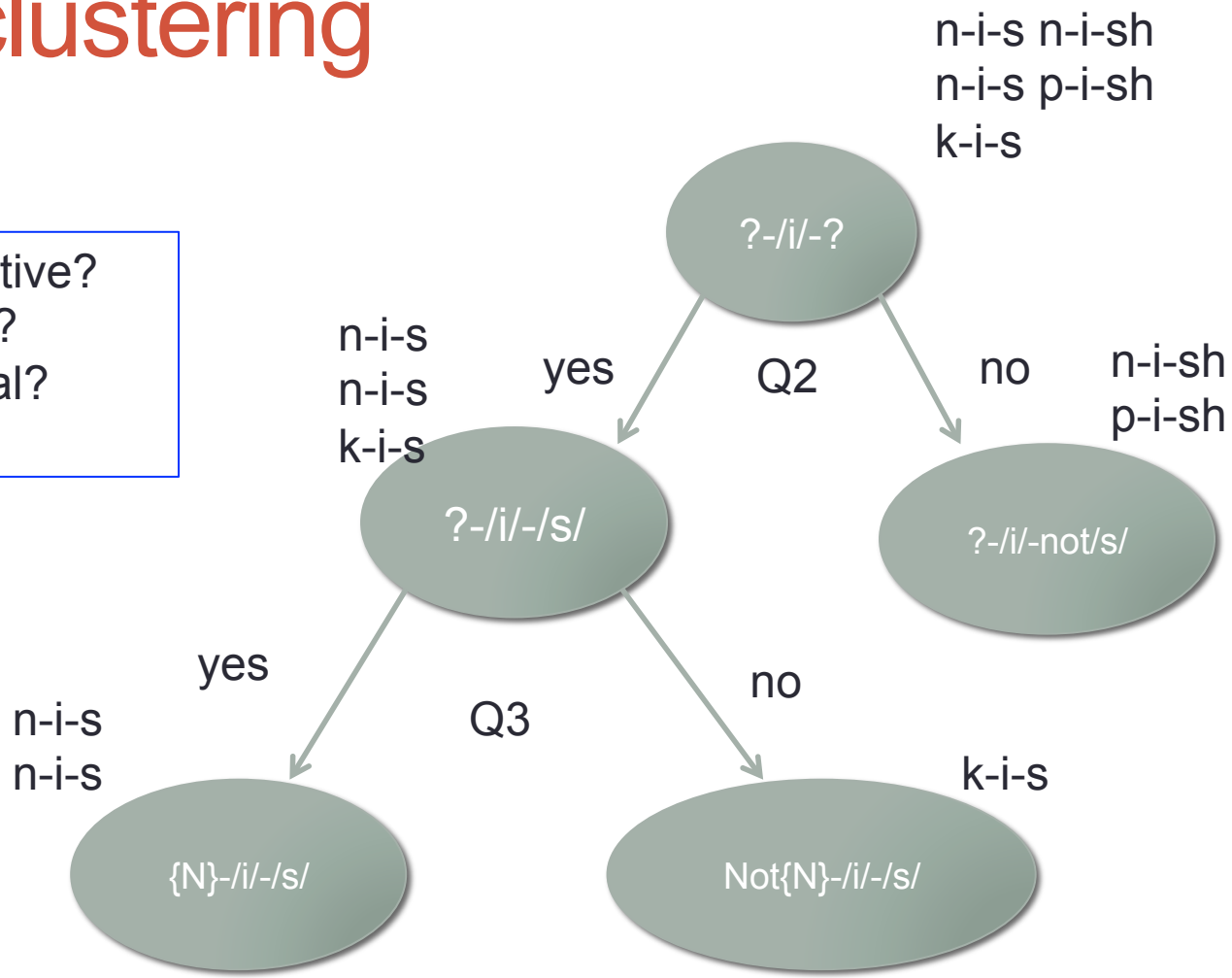
Tree clustering

Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



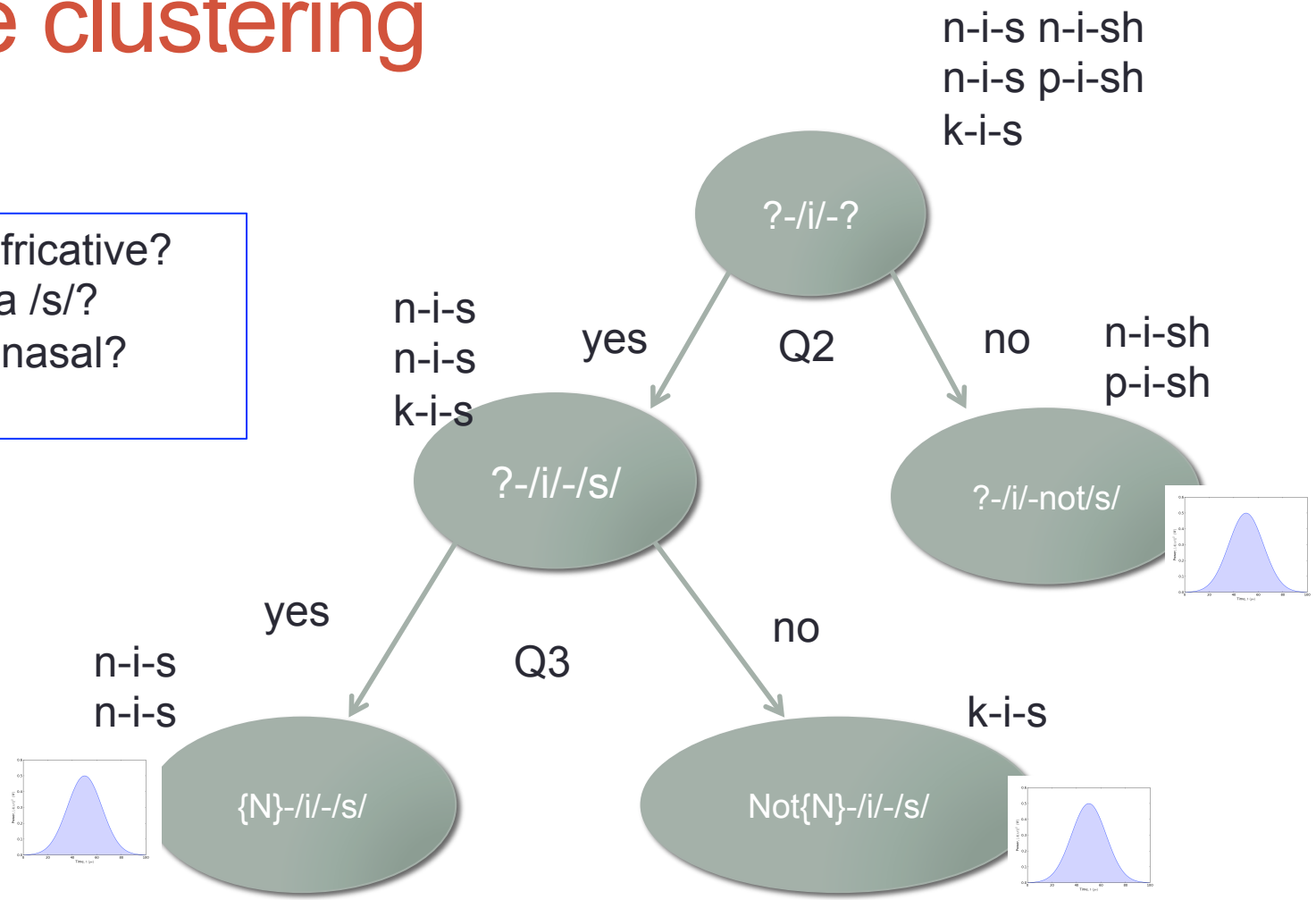
Tree clustering

Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



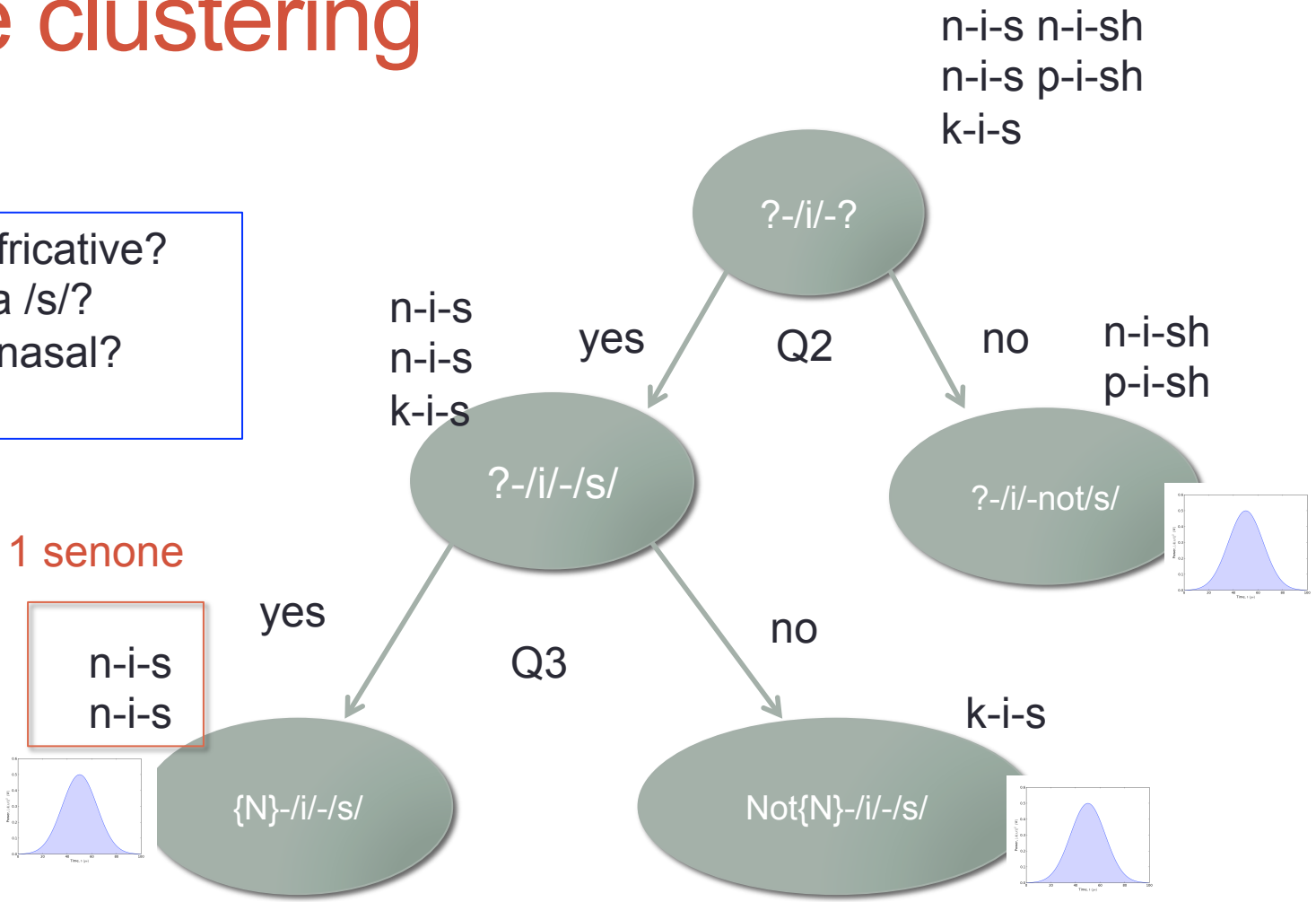
Tree clustering

Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



Tree clustering

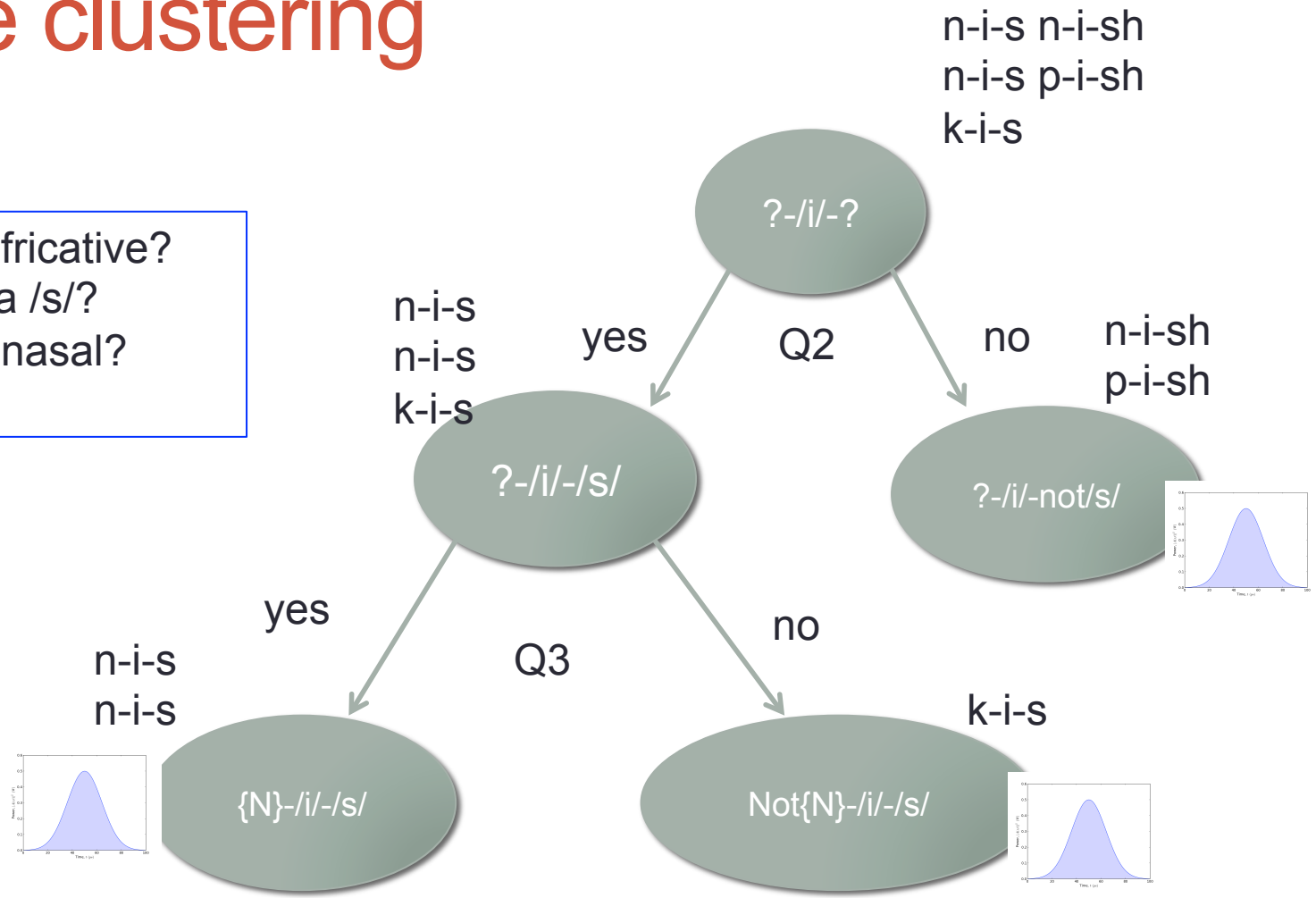
Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



You now have a new word that needs b-i-s. Which model?

Tree clustering

Q1: Left a fricative?
Q2: Right a /s/?
Q3: Left a nasal?
…



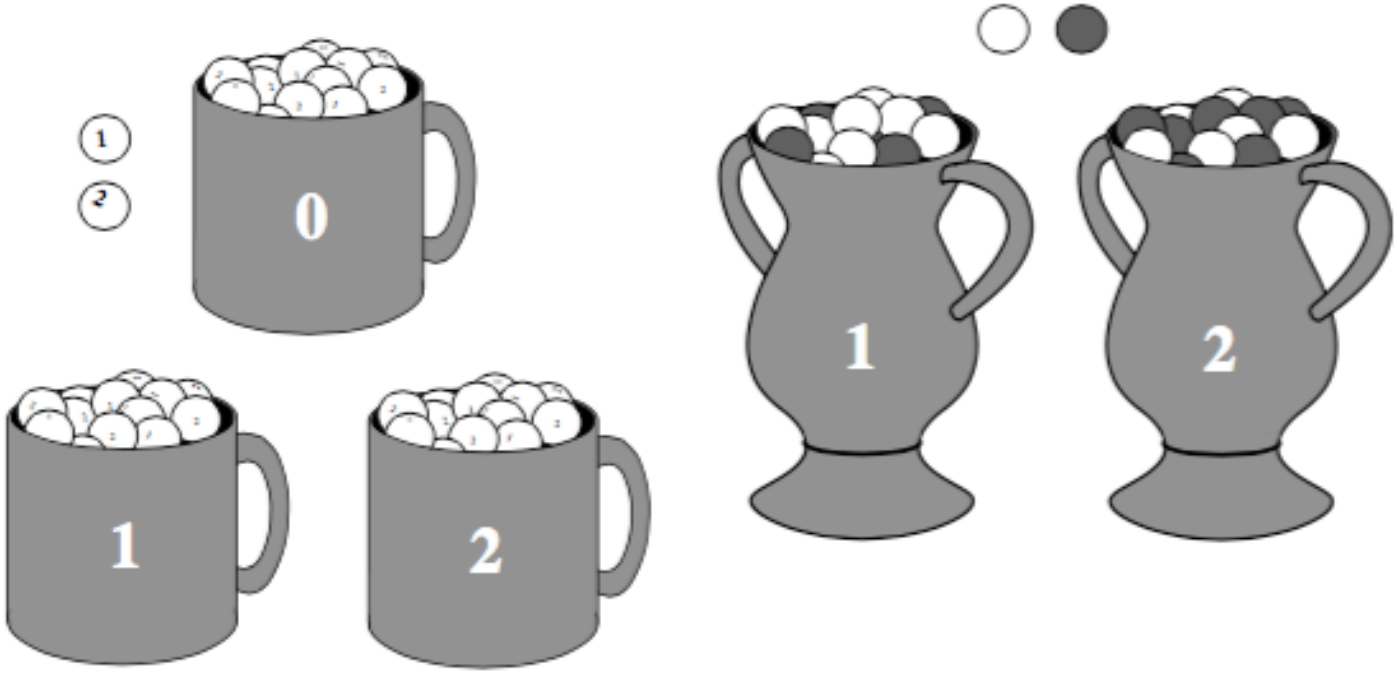
We model P(x | {N}-/i/-/s/), P(x | Not{N}-/i/-/s/) P (x | ?-/i/-not/s/)

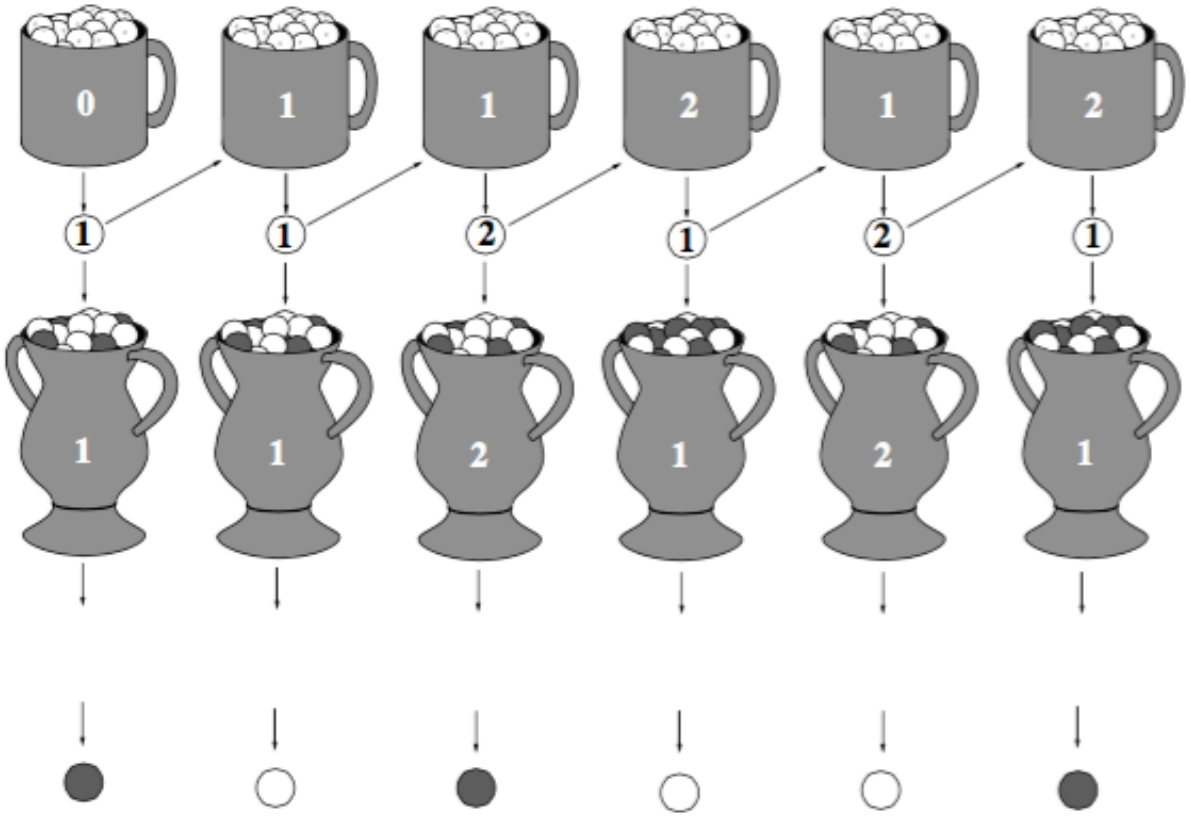
The ASR equation P(X|L)P(L|W)P(W)

- She sell vs Seashell
- [x1 x2 x3 x4 x5 x6]
- P([x1 x2 x3 x4 x5 x6] | /sh/ /iy/ /s/ /e/ /l/) P(/sh/ /iy/ /s/ /e/ /l/ | She sells) P(She sell) = ?
- P([x1 x2 x3 x4 x5 x6] | /s/ /iy/ /sh/ /e/ /l/) P(/s/ /iy/ /sh/ /e/ /l/ | She sells) P(Sea shell) = ?

- How to get probably P([x1 x2 x3 x4 x5 x6] | /sh/ /iy/ /s/ /e/ /l/)
- We have P(x | /sh/) P(x | /e/)
- P(x | /iy/) P(x | /l/)
- P(x | /s/)
- We know the overall sequence, but don't know which frame comes from which

Hidden Markov Model: an example



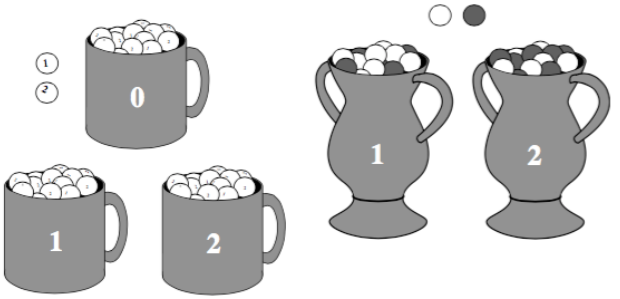


Observation Sequence: $\boldsymbol{O} = \{B, W, B, W, W, B\}$

State Sequence: $\boldsymbol{Q} = \{1, 1, 2, 1, 2, 1\}$

Or given MFCC frames, determine phoneme sequence

Hidden Markov Model



- Defining HMM requires
- 1. Starting state probability p_0 = [0.7 0.3]
- 2. Transition probability, A_{ij}

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

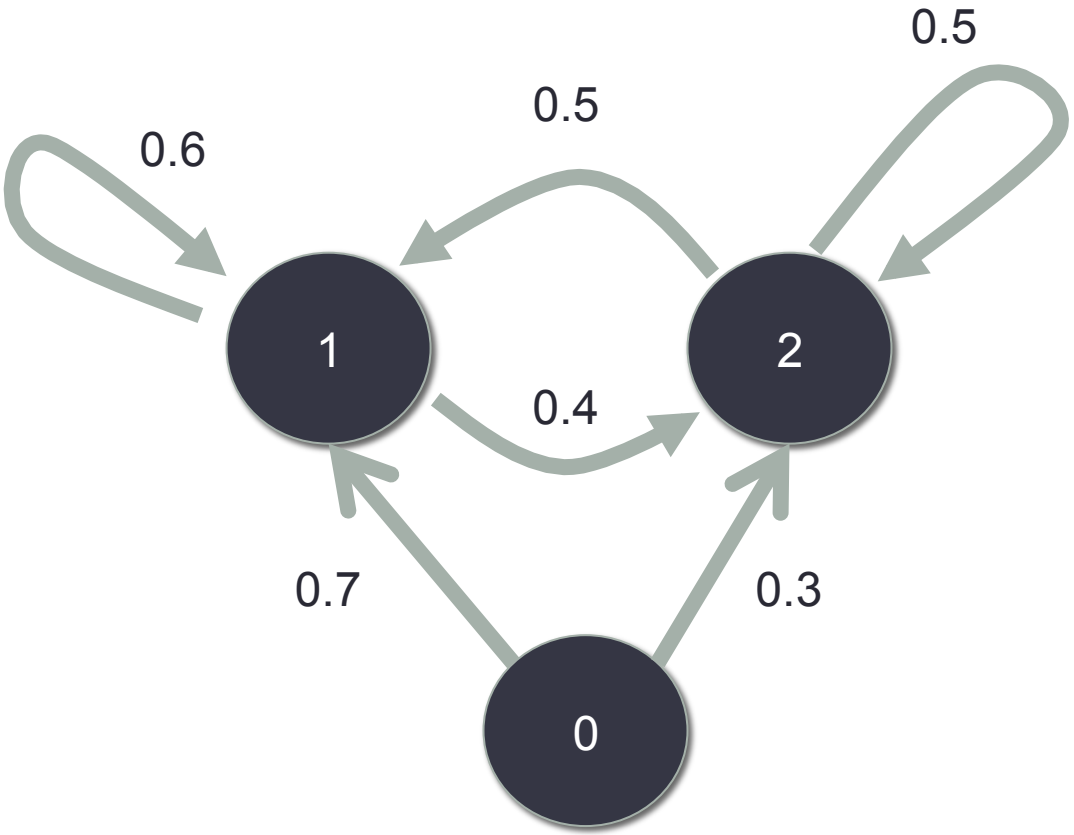
- 3. Emission probably, B_{ik}
- If emits discrete values
  - Discrete HMM

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

- If emits continuous values (GMM emitting MFCC values)
  - Continuous HMM

Hidden Markov Model

- The states and the transition can be written as a graph (finite state machine)



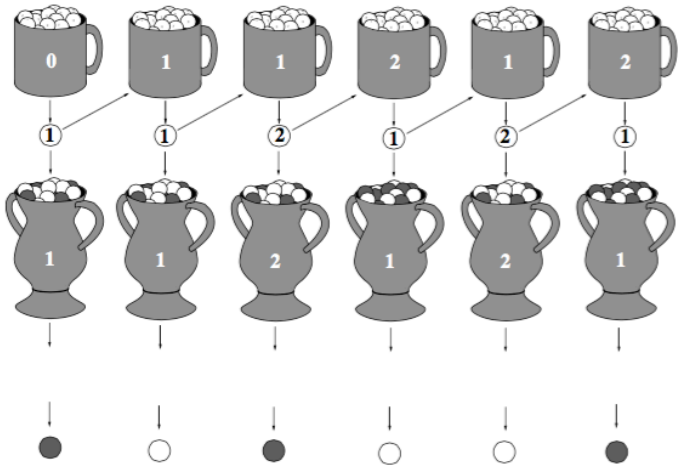
p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

Hidden Markov Models 3 questions

- Evaluation – Given model parameters, what is the probability of a observation sequence?
- Decoding – Given model parameters, what is the most likely hidden state sequence?
- Training – How do we learn the model parameters?



Observation Sequence: $\boldsymbol{O} = \{B, W, B, W, W, B\}$
State Sequence: $\boldsymbol{Q} = \{1, 1, 2, 1, 2, 1\}$

For more info, L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, 1989.

Evaluation

- If we know the state sequence, we can easily compute the probability
- P([B W W] | [1 1 2])
  = 0.8 * 0.2 * 0.7

- If we don't know the state sequence
- P([B W W]) = ?

p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

Evaluation

- If we don't know the state sequence
- P([B W W]) = ?

  Σ P([B W W]| [S]) P([S])
- Brute force

  Add all state sequence possibilities
- P([B W W] | (1 1 1)) P(1 1 1)
  +P([B W W] | (1 1 2)) P(1 1 2)
  +P([B W W] | (1 2 1)) P(1 2 1)
  +P([B W W] | (1 2 2)) P(1 2 2)
  +P([B W W] | (2 1 1)) P(2 1 1)
  +P([B W W] | (2 1 2)) P(2 1 2)
  +P([B W W] | (2 2 2)) P(2 2 2)

p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

Evaluation

- If we don't know the state sequence
- P([B W W]) = ?

  Σ P([B W W]| [S]) P([S])
- An efficient algorithm
  - Forward-Backward Algorithm
  - Not really used in speech recognition

p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

Best underlying sequence (Decoding)

- If we observed [B W W] and we know the model parameters, what's the most likely state sequence

p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

- Brute force, find maximum of
- P([B W W] | (1 1 1)) P(1 1 1)
  P([B W W] | (1 1 2)) P(1 1 2)
  P([B W W] | (1 2 1)) P(1 2 1)
  P([B W W] | (1 2 2)) P(1 2 2)
  P([B W W] | (2 1 1)) P(2 1 1)
  P([B W W] | (2 1 2)) P(2 1 2)
  P([B W W] | (2 2 2)) P(2 2 2)
- Or $\underset{S}{\text{argmax}}[\ P(O|S)P(S)\ ]$

State sequence

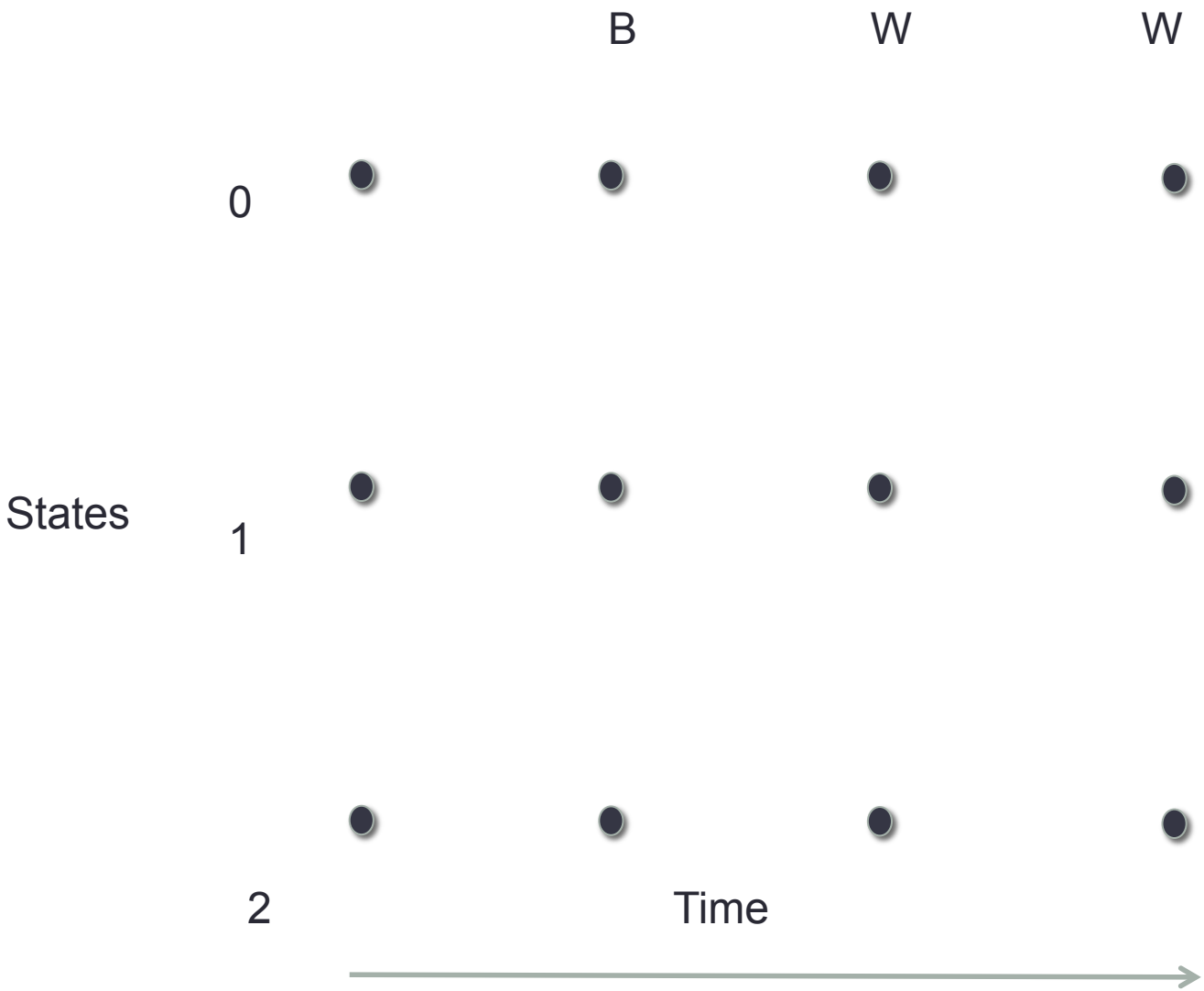
The Viterbi Algorithm

- In DTW, we align frames from template to frames of test
- In Decoding, we align frames from observation to states

- In DTW we reduce computation by saving previous answers (dynamic programing)
  - Best path to (x,y) = argmin[d(x'y') + distance from (x',y') to (x,y)]
- In Decoding can do the same
- Best state sequence till n ending at s_n = argmax[P(best sequence till n-1)P(getting o_n from state s_n)]

- We call this algorithm the Viterbi Algorithm.

The Viterbi Algorithm



Setup

- Best state sequence till n ending at s_n = argmax[P(best sequence till n-1)P(getting o_n from state s_n)]
- Q(n , s_n)
- And R(n, s_n) remembering best previous state

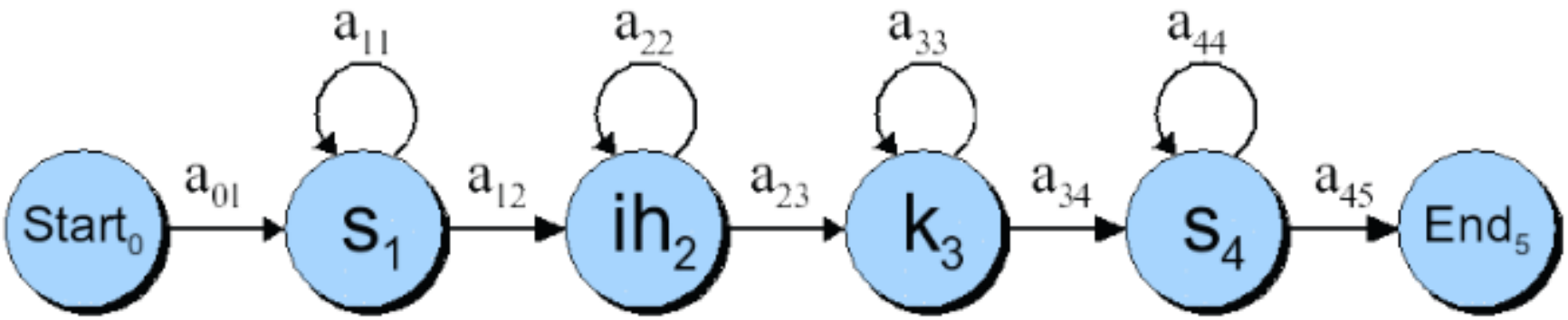
p_0 = [0.7 0.3]

A_{ij}	To 1	To 2
From 1	0.6	0.4
From 2	0.5	0.5

B_{ik}	B	W
state 1	0.8	0.2
state 2	0.3	0.7

HMM in speech

- A word is a sequence of phonemes
- Each phoneme has different emission probabilities (GMMs)



https://web.stanford.edu/class/cs224s/lec/224s.14.lec3.pdf

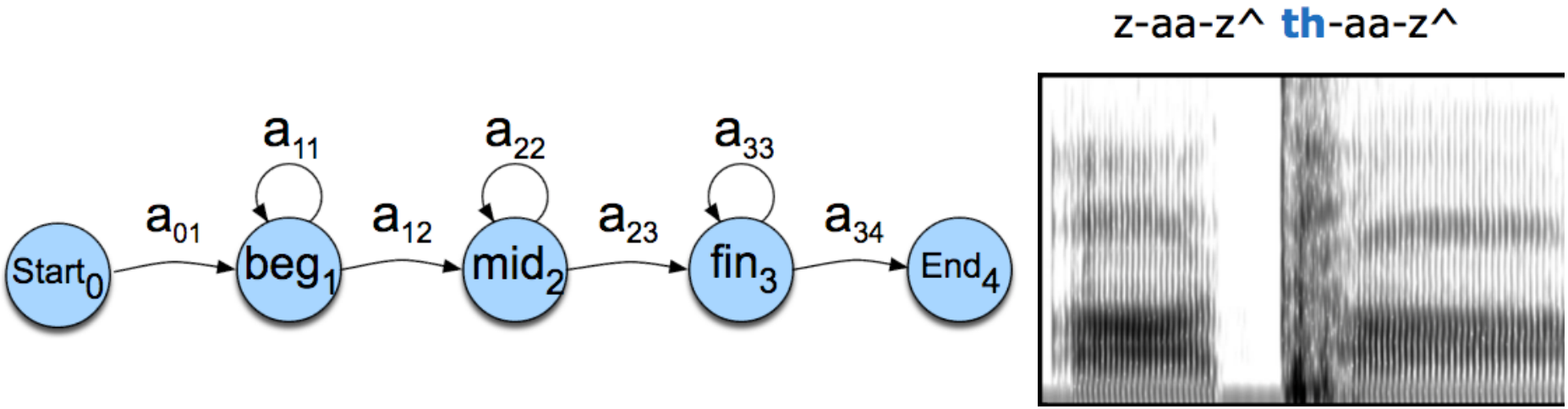
The ASR equation P(X|L)P(L|W)P(W)

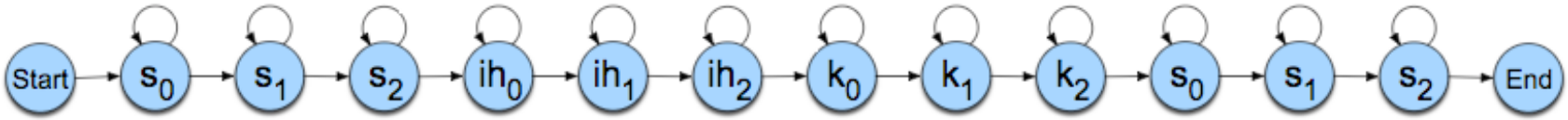
- She sell vs Seashell
- [x1 x2 x3 x4 x5 x6]
- P([x1 x2 x3 x4 x5 x6] | /sh/ /iy/ /s/ /e/ /l/) P(/sh/ /iy/ /s/ /e/ /l/ | She sells) P(She sell) = ?
- P([x1 x2 x3 x4 x5 x6] | /s/ /iy/ /sh/ /e/ /l/) P(/s/ /iy/ /sh/ /e/ /l/ | She sells) P(Sea shell) = ?

- How to get probably P([x1 x2 x3 x4 x5 x6] | /sh/ /iy/ /s/ /e/ /l/)
  - Compute Viterbi of HMM /sh/ - /iy/ - /s/ - /e/ - /l/
  - Note : this is not really P([x1 x2 x3 x4 x5 x6] | /sh/ /iy/ /s/ /e/ /l/), but close. Real answer requires Forward-Backward algorithm

HMM in speech – 3 state hmm

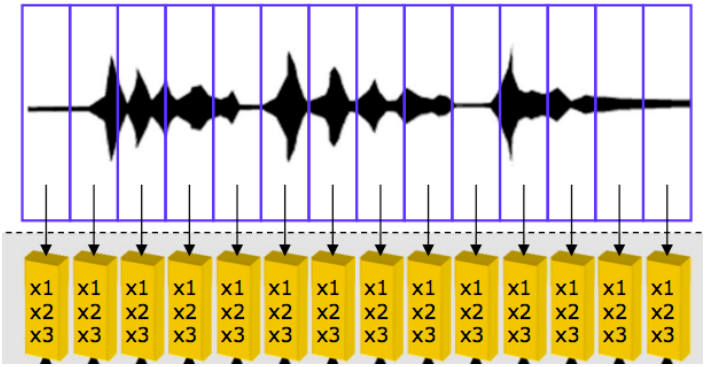
- Each phoneme is divided into multiple states
  - Phonemes are not uniform over time
  - 3-state hmm : start – middle – end for each phoneme
  - Assume a minimum phoneme length of 3 frames

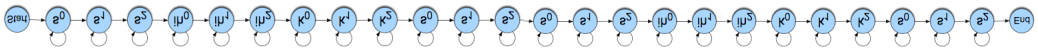




Training HMM-GMM using Viterbi

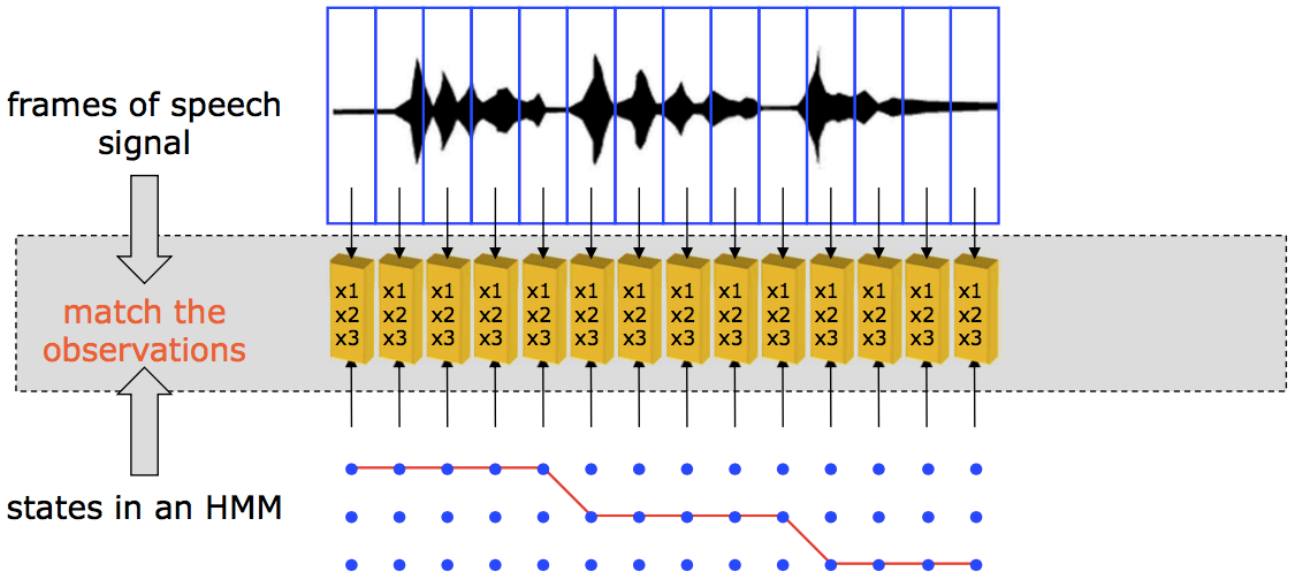
- We need to know transmission probability and emission probability.
- Our data : Sentences – with sentence-level transcription
- Goal : Assign each frame to a phoneme (or part of phoneme)
- Key idea: If we have a model of all phonemes, build a HMM with the phonemes of that sentence, then do viterbi alignment





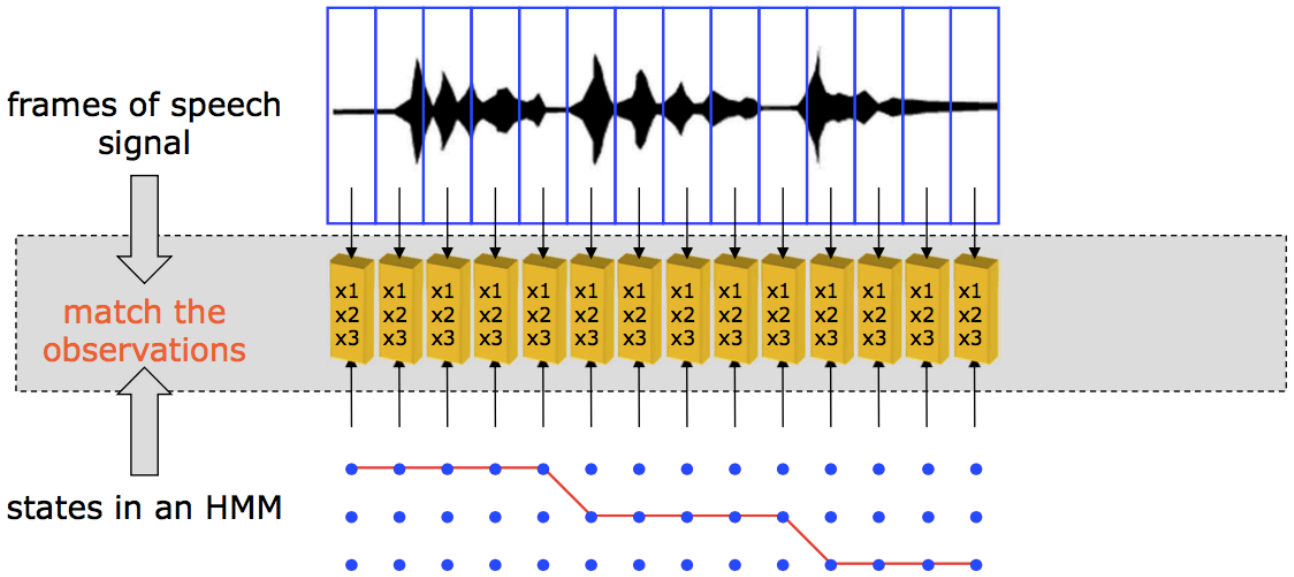
EM with Viterbi

- Expectation Maximization
  - Initialize : Uniformly segment the whole sentence. Assign to each state in that sentence. Train the model.
  - Assign : Compute Viterbi alignment using current model
  - Update : Update model



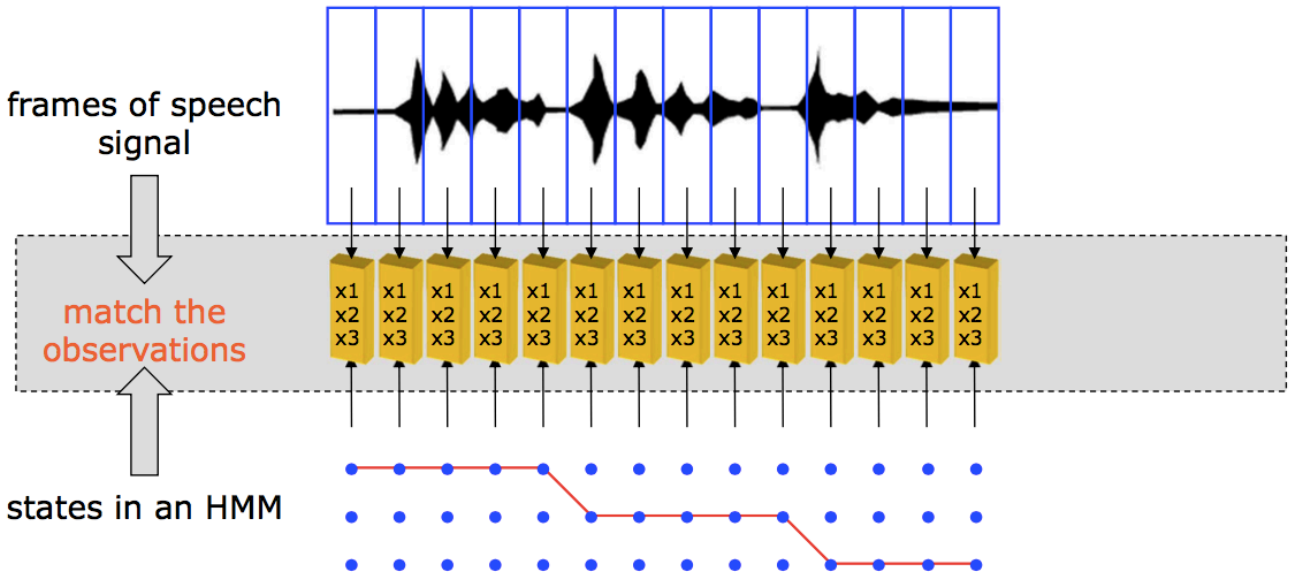
Transition Probability Estimation

- Use Viterbi alignment and count
  - $a_{i,j}$ = # of frames transition from state i to state j / # of frames in state i



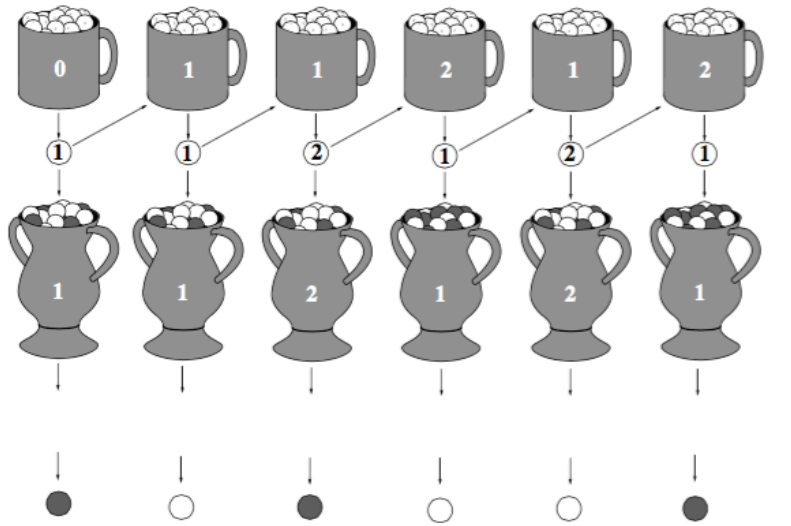
Emission probability Estimation

- Use Viterbi alignment and do another EM for GMM parameters
  - $B_i \sim \sum_{k=1}^{K} w_k N(\mu_k, \sigma_k)$



Sequence models

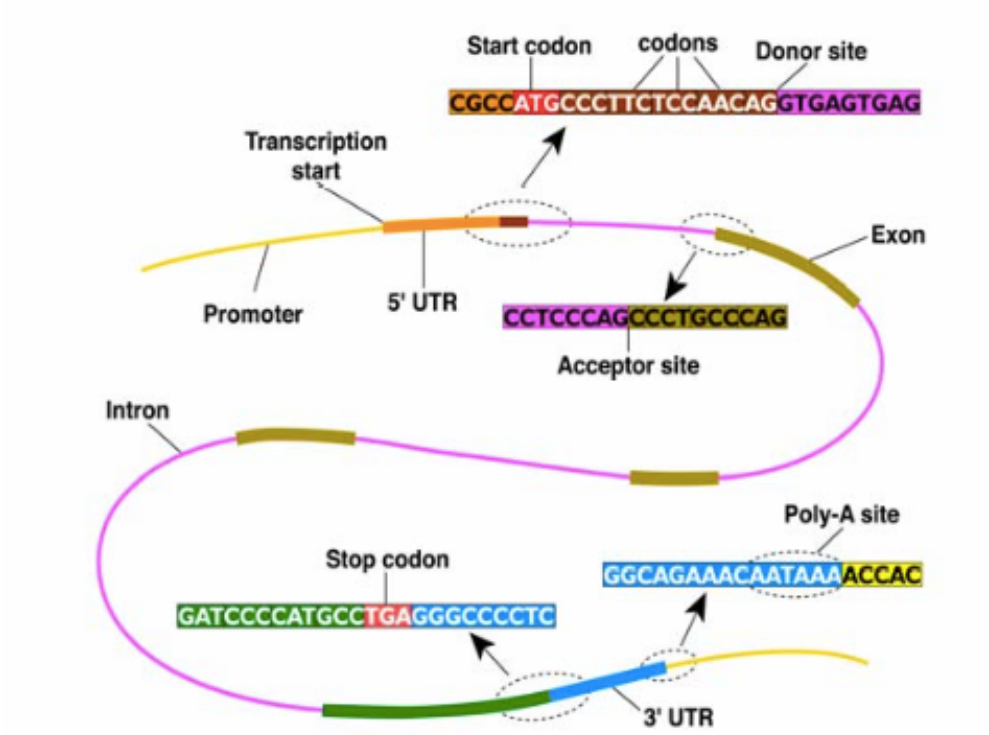
- Models that take into account the information regarding the order of occurrence
  - HMM, CRF, Recurrent neural networks



Observation Sequence: $O = \{B, W, B, W, W, B\}$
State Sequence: $Q = \{1, 1, 2, 1, 2, 1\}$

Other applications for sequence modeling

- Finance
- NLP
- Genetics and molecular biology



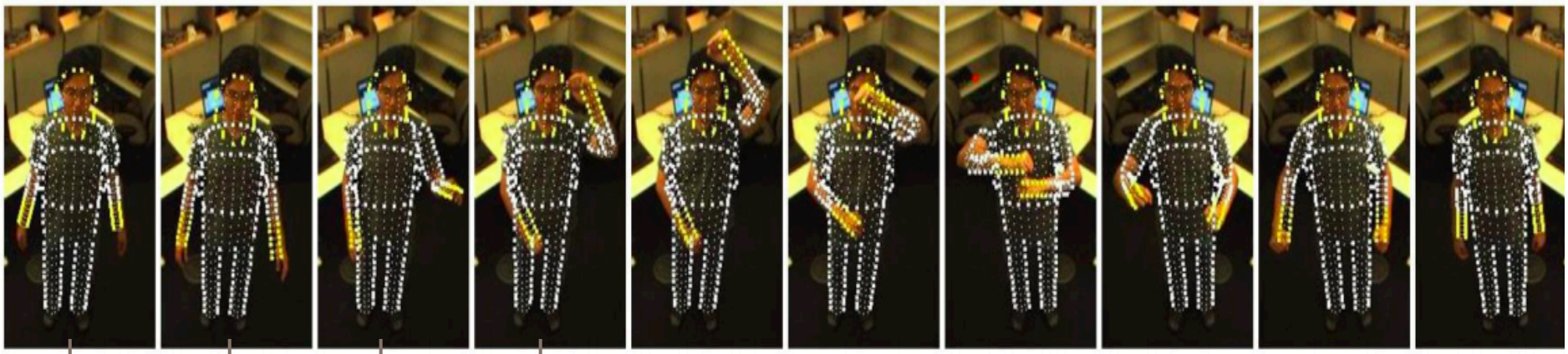
http://www.cs.cmu.edu/~epxing/Class/10708-07/Slides/lecture12-CRF-HMM-annotation.pdf

Sequence models in videos

Gesture recognition

possible gestures

sequence of photos



X

Y ? ? ? ? ...

http://people.csail.mit.edu/sybor/cvpr06_wang.pdf

AM model summary

- MFCC
- Modeling sub-phonemes
- HMM
- GMM
- Context dependent modeling

- Speech can have different lengths
- Different gender sounds different
- /iy/ before /ng/ is different from /iy/ before /m/
- Source-filter separation
- Diphthong is a combination of two vowels

AM summary

- Modeling phonemes and sub-phonemes
- Context dependent modeling with tree clustering
- GMM
- HMM
- Viterbi
- EM training

- “We used a 3-state CD-HMM with 2500 states, and 18 Gaussians components per state. For features, we used 39 dimensional MFCCs with delta, and delta-delta features.”

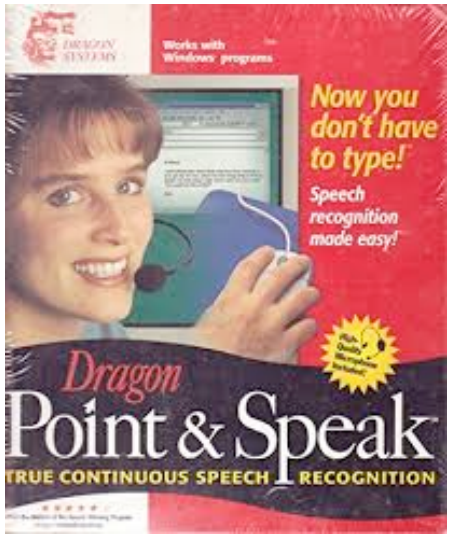
Delta and Delta-Delta features

- Some phonemes are distinguished by changes in the spectrum over time
  - Stop consonants – closure -> burst -> affricate
- Use difference between MFCCs of different frames = delta features

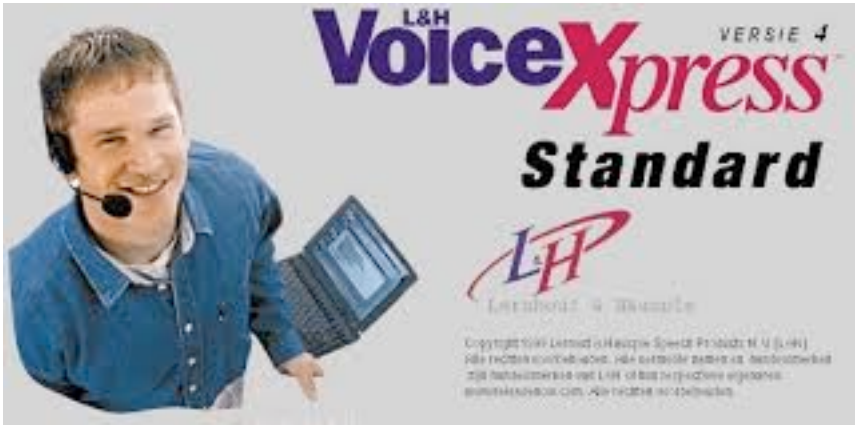
  $v_t = x_t - x_{t-1}$
- Difference between delta features = delta-delta features

  $a_t = v_t - v_{t-1}$
- Final feature are concatenated $f_t = [x_t\ v_t\ a_t]$
- Sometimes called velocity and acceleration features

A story about how research become Siri



Goldman Sachs



NUANCE®

More info http://www.nytimes.com/2012/07/15/business/goldman-sachs-and-a-sale-gone-horribly-awry.html

LANGUAGE MODELING

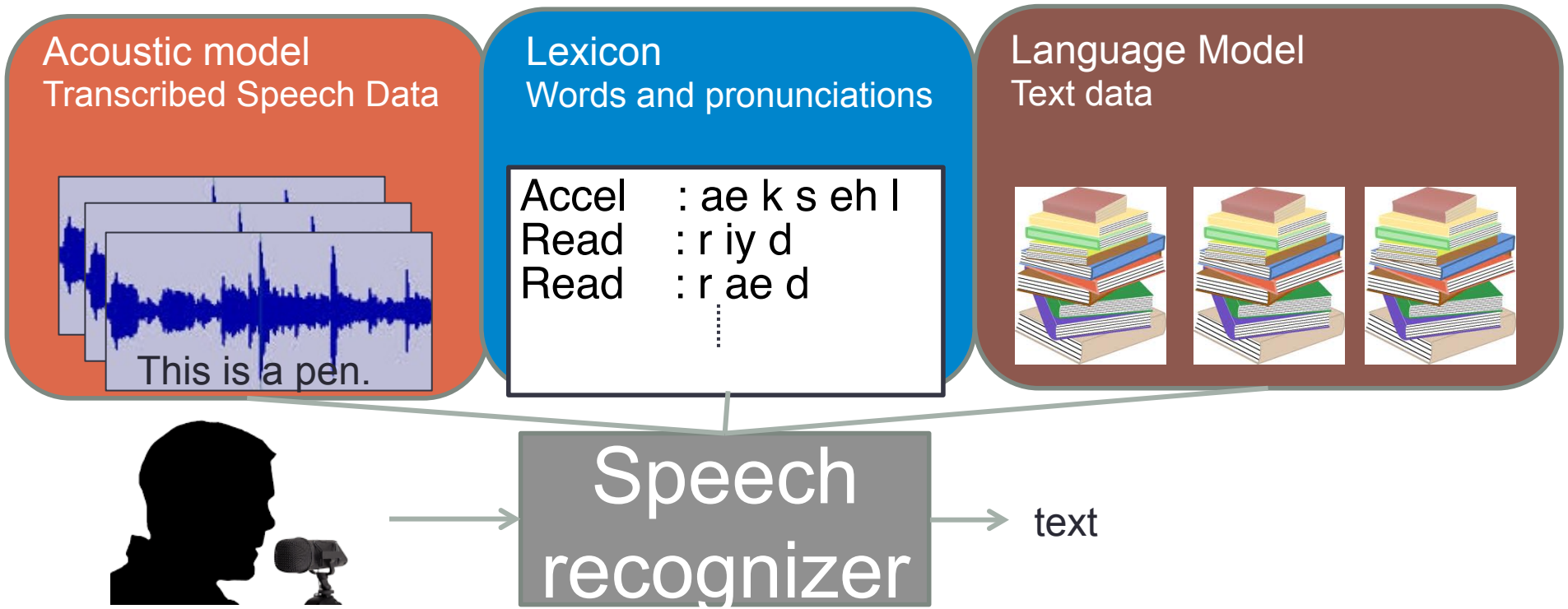
N-grams, discounting, and back-off

Many illustrations provided by courtesy of James Glass

The ASR equation

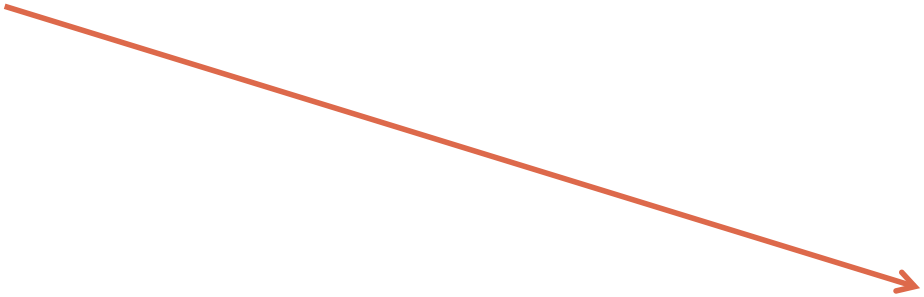
$$= argmax_{W,L} P(X \mid L) P(L \mid W) P(W)$$

X - waveform, L - pronunciation, W - words



What is language modeling

- How to evaluate the probability of a sentence
  - P(“This is a pen”)
  - P(“This is pineapple”)
  - P(“This is a pens”)



$$= argmax_{W,L} P(X \mid L) P(L \mid W) P(W)$$

Why language modeling

- Distinguish between acoustically similar utterances using prior knowledge
  - “Please write a letter right now to Mrs. Wright. Tell her that two is too many to buy.”
  - “How to wreck a nice beach” vs “How to recognize speech”
- Can help distinguish Thai tones.

Types of LM

- N-grams
- Rule based grammar (Finite State Networks) <= Next lectures
- Context Free Grammar (CFGs) <= Next lectures

History-based statistical LMs

- Statistical language models (LMs) assign a probability estimate $P(W)$ to each word sequence $W = \{w_1, \ldots, w_K\}$ subject to

$$\sum_W P(W) = 1$$

- $P(W)$ can be expanded using the chain rule:

$$P(W) = \prod_{i=1}^{K} P(w_i \mid w_1, \ldots, w_{i-1}) = \prod_{i=1}^{K} P(w_i \mid h_i)$$

  where $h_i = \{w_1 \ldots, w_{i-1}\}$ is the word history for w_i

  - Note: Initial and final words are typically taken to be the sentence boundary marker, $w_1 = w_K = <>$

N-gram language model

- Remembering the full history is hard
  - Too many model parameters to estimate
- Remember the last n-1 words = n-gram
  - Remember no history $P(w_n)$ = unigram
  - Remember the last 1 word $P(w_n|w_{n-1})$ = bigram
  - Remember the last 2 words $P(w_n|w_{n-1}w_{n-2})$ = trigram

$$P(W) = \prod_{i=1}^{K} P(w_i \mid w_1, \ldots, w_{i-1}) = \prod_{i=1}^{K} P(w_i \mid h_i)$$

where $h_i = \{w_1 \ldots, w_{i-1}\}$ is the word history for w_i

P(w5| w1 w2 w3 w4) = P(w5 | w4) for bigram (Markov assumption)

N-gram language model

- Larger n-gram requires more parameters
  - Vocabulary size V
  - Unigram => V
  - Bigram => V^2
  - Trigram => V^3

P(I eat rice) = P(I) P(eat) P(rice)
P(I eat rice) = P(I| ??) P(eat| I) P(rice | eat)

- Larger n-gram requires more data to not overfit

Unigram	I	eat	rice
P	0.3	0.1	0.6

Next word

Previous word	Bigram	I	eat	rice
	I	0.1	0.7	0.1
	eat	0.1	0.2	0.7
	rice	0.0	0.5	0.6

N-gram language model

- How to use n-gram to give score for a sentence? P(W)
- How to estimate parameters of n-gram?
- How to compare the performance between n-grams?

N-gram example

- Trigram

- P(เช้าฟาดผัดฟัก เย็นฟาดฟักผัด) = ?

- What is the probability of a sequence we never seen before?
- What is the probability of a word not in the dictionary?

Estimating N-gram parameters

- Based on counts of the training data

$$P(w_i \mid w_{i-2}w_{i-1}) \approx f(w_i \mid w_{i-2}w_{i-1}) = \frac{c(w_{i-2}w_{i-1}w_i)}{c(w_{i-2}w_{i-1})}$$

- c(W) is the count of how many times the sequence W appear in the training data.

End and start token

- We need to be able to end a sentence
  - P(I eat rice eat I ……..) = P(I) P(eat) P(rice) P(eat) ….
  - P(I eat rice </s>) = P(I) P(eat) P(rice) P(and) P(</s>)

</s> is the end token.
We now need some probability weights for it

Unigram	I	eat	rice
P	0.3	0.1	0.6

Next word

Previous word	Bigram	I	eat	rice
	I	0.1	0.7	0.1
	eat	0.1	0.2	0.7
	rice	0.0	0.5	0.6

End and start token

- We need to be able to end a sentence
  - P(I eat rice eat I ……..) = P(I) P(eat) P(rice) P(eat) ….
  - P(I eat rice </s>) = P(I) P(eat) P(rice) P(and) P(</s>)

</s> is the end token.
We now need some probability weights for it

Unigram	I	eat	rice	</s>
P	0.25	0.05	0.5	0.2

Next word

Previous word	Bigram	I	eat	rice	</s>
	I	0.1	0.65	0.1	0.05
	eat	0.1	0.15	0.55	0.2
	rice	0.0	0.3	0.4	0.3

End and start token

- We need to a token to represent the start of a sentence to indicate what word can start a sentence
  - P(I eat rice </s>) = P(I) P(eat) P(rice) P(and) P(</s>)
  - P(I eat rice </s>) = P(I | <s>) P(eat | I) P(rice | eat) P(</s> | rice)
- For unigrams, we ignore the start token (same number of terms)

Unigram	I	eat	rice	</s>
P	0.25	0.05	0.5	0.2

Next word

Previous word	Bigram	I	eat	rice	</s>
	I	0.1	0.65	0.1	0.05
	eat	0.1	0.15	0.55	0.2
	rice	0.0	0.3	0.4	0.3
	<s>	0.4	0.3	0.2	0.1

What does this mean?

N-gram estimation

- Unigram and bigram
- เช้าฟาดผัดฟัก เย็นฟาดฟักผัด
- Vocab size = ?

Perplexity

- Compare n-grams by measuring how well they fit our test data. The better n-gram should give higher probability.
- For large datasets, usually measure the log ("logprob") of the probability instead because of numerical issues

$$LP = -\frac{1}{n}\log_2 \hat{P}(W) = -\frac{1}{n}\sum_i \log_2 \hat{P}(w_i|\phi(h_i))$$

- LP is a measure of uncertainty or entropy
  - Uniform distribution -> high entropy.
- Perplexity is calculated from entropy $PP = 2^{LP}$
- Intuition: measure branch factor for the each word
  - How many choices are there for the word given previous history?

Perplexity example

Domain	Size	Type	Perplexity
Digits	11	All word	11
Resource Management	1,000	Word-pair	60
		Bigram	20
Air Travel Understanding	2,500	Bigram	29
		4-gram	22
WSJ Dictation	5,000	Bigram	80
		Trigram	45
	20,000	Bigram	190
		Trigram	120
Switchboard Human-Human	23,000	Bigram	109
		Trigram	93
NYT Characters	63	Unigram	20
		Bigram	11
Shannon Letters	27	Human	~ 2

Perplexity is related to vocabulary size.
Comparing perplexity between different vocabulary size is unfair!

Shanon letters

- Guess the next letter until correct.
- How many guesses determine the branching factor
  - Interpret as how many bits to encode letter according to the LM

T	H	E	R	E		I	S		N	O		R	E	V	E	R	S	E
1	1	1	5	1	1	2	1	1	2	1	1	15	1	17	1	1	1	2
	O	N		A		M	O	T	O	R	C	Y	C	L	E		A	...
1	3	2	1	2	2	7	1	1	1	1	4	1	1	1	1	1	3	...

Notes on N-grams

- Language model data does not need to be the same as acoustic model data
- Can be much larger! Text is cheap (web scraping)
- Talking style does not match Writing style
  - Twitter style? Facebook style?
  - How to make use of different styles?

N-gram interpolation

- Mixture of n-grams

$$P(w_i|h_i) = \sum_j \lambda_j f(w_i|\phi_j(h_i)) \qquad \sum_j \lambda_j = 1$$

$$P(w_i|w_{i-1}) = \lambda_2 f(w_i|w_{i-1}) + \lambda_1 f(w_i) + \lambda_0 \frac{1}{|V|}$$

- Can use to combine different histories, or different data.
- Find best weights that minimizes the perplexity on development data.

Smoothing and discounting

- Zero counts yields zero probability
  - P(eat rice) = P(eat|<s>) P(rice| eat) P(</s>| rice)

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0	2	15
rice	3	6	0	20
<s>	1	0	30	60

Smoothing and discounting

- Smoothing and discounting are ways to deal with 0 counts

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0	2	15
rice	3	6	0	20
<s>	1	0	30	60

Smoothing and discounting

- Smoothing and discounting are ways to deal with 0 counts
- We can smooth with interpolation
  - P(eat rice) = P(eat|<s>) P(rice| eat) P(</s>| rice)

$$P(w_i|h_i) = \sum_j \lambda_j f(w_i|\phi_j(h_i)) \qquad \sum_j \lambda_j = 1$$

$$P(w_i|w_{i-1}) = \lambda_2 f(w_i|w_{i-1}) + \lambda_1 f(w_i) + \lambda_0 \frac{1}{|V|}$$

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0	2	15
rice	3	6	0	20
<s>	1	0	30	60

Discounting

- Take from high counts and give to low counts

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0	2	15
rice	3	6	0	20
<s>	1	0	30	60

Discounting

- Take from high counts and give to low counts
- Which to take? How much to take?

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0+x	2-x	15
rice	3	6-y	0+y	20
<s>	1-z	0+z	30	60

Good Turing discounting

- GT estimate for an item occurring c times is

$$c^* = (c+1)\frac{n_{c+1}}{n_c} \qquad P_c = \frac{c^*}{N}$$

- where n_c is the number of items occurring c times.
- Example bigram counts from AP news (273,000 word vocabulary) (Church and Gale, 1991)

c	n_c	c^*
0	74,671,100,000	0.0000270
1	2,018,046	0.446
2	449,721	1.26
3	188,933	2.24
4	105,668	3.24
5	68,379	4.22

Back-off n-grams

- Back-off is a way to discount based on lower n-grams rather than fixed contexts
- Take counts from low-count sequences. (discounting)

$$P(w_i|w_{i-1}) = \begin{cases} f(w_i|w_{i-1}) & c(w_{i-1}w_i) \geq \alpha \\ f_d(w_i|w_{i-1}) & \alpha > c(w_{i-1}w_i) > 0 \\ q(w_{i-1})P(w_i) & c(w_{i-1}w_i) = 0 \end{cases}$$

- Find f_d and q such that $\Sigma P(w|w_{i-1}) = 1$

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0	2	15
rice	3	6	0	20
<s>	1	0	30	60

Back-off n-grams

- Back-off to lower n-gram when there is not enough counts for higher n-gram
- Take counts from low-count sequences. (discounting)

$$P(w_i|w_{i-1}) = \begin{cases} f(w_i|w_{i-1}) & c(w_{i-1}w_i) \geq \alpha \\ f_d(w_i|w_{i-1}) & \alpha > c(w_{i-1}w_i) > 0 \\ q(w_{i-1})P(w_i) & c(w_{i-1}w_i) = 0 \end{cases}$$

- Find f_d and q such that $\Sigma P(w|w_{i-1}) = 1$

Counts	I	eat	rice	</s>
I	10	5	7	8
eat	5	0+P(eat)	2-X	15
rice	3	6	0+0	20
<s>	1-X	$0+q_1$*P(eat)	30	$0+q_2$*P(</s>)

Take from small counts
Give to zero counts

Kneser-Ney (KN) smoothing

- Problem with back-offs

$$P(w_i|w_{i-1}) = \begin{cases} f(w_i|w_{i-1}) & c(w_{i-1}w_i) \geq \alpha \\ f_d(w_i|w_{i-1}) & \alpha > c(w_{i-1}w_i) > 0 \\ q(w_{i-1})P(w_i) & c(w_{i-1}w_i) = 0 \end{cases}$$

- Some words appear very often but not very useful in many contexts

San Francisco is the largest city. San Francisco is in California…

The rice ___ very good
P(grows| rice) vs P(Francisco | rice)
q(rice) P(grows) vs q(rice) P(Francisco)

Kneser-Ney (KN) smoothing

- Back off according to how useful the word is for other context.

P(Francisco) = # of words (word type) that appear before Francisco / # of bigrams

$$P(w_i) = \frac{|\{w_{i-1} : c(w_{i-1}w_i) > 0\}|}{\sum_{w_i} |\{w_{i-1} : c(w_{i-1}w_i) > 0\}|}$$

$$|\{w_{i-1} : c(w_{i-1}w_i) > 0\}|$$

The rice ___ very good
P(grows| rice) vs P(Francisco | rice)
q(rice) P(grows) vs q(rice) P(Francisco)

How many different words, w_{i-1}, is seen before w_i

- Discount using a fixed discount D
  - calculate q() that sums to 1.

$$P_D(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i) - D}{c(w_{i-1})} \qquad c(w_{i-1}w_i) > 0$$

Notes about N-gram

- Very hard to beat N-gram performance (until DNN)
- Cannot use long term history very well

กรุงเทพ ซึ่ง เป็น เมือง หลวง ของ ประเทศ ไทย มี ปัญหา การ

- Cannot accommodate new word, Out-of-vocabulary (OOV)

Unigram	I	eat	rice	</s>
P	0.25	0.05	0.5	0.2

P(I eat banana) = ?

- No real understanding, just statistics
- Still required for current ASR technologies (augmented with DNN)

Summary

- N-grams
  - Perplexity
  - Interpolation
  - Back-off, smoothing, and discounting