Speech Processing and Understanding

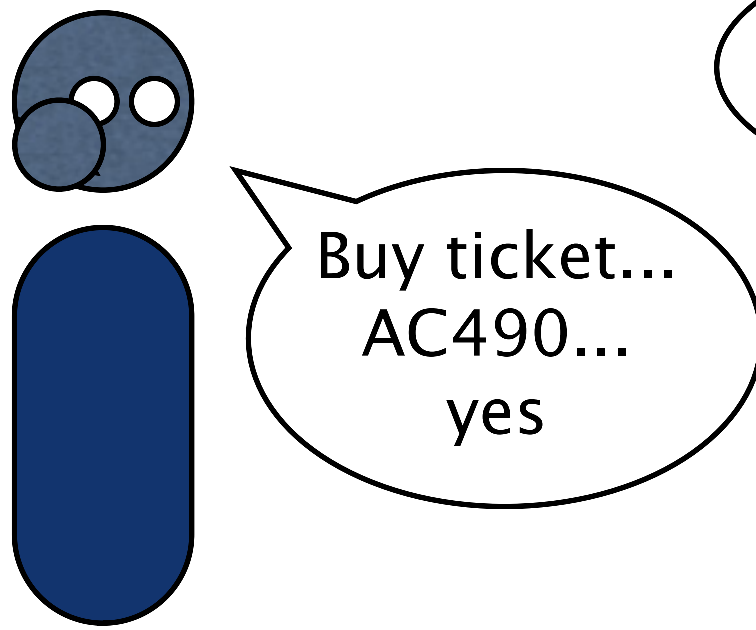
CSC401 Assignment 3

Agenda

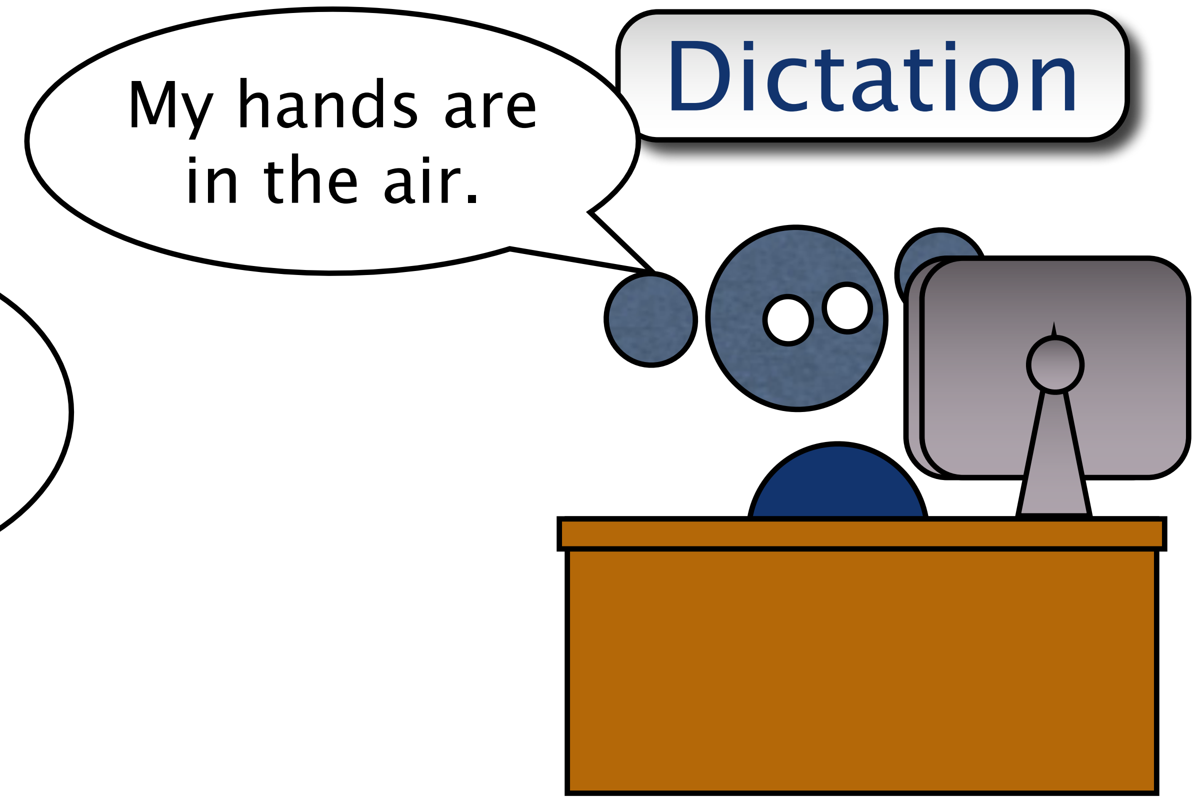
- Background
  - Speech technology, in general
  - Acoustic phonetics
- Assignment 3
  - Speaker Recognition: Gaussian mixture models
  - Speech Recognition:
    - Continuous hidden Markov models
    - Transcription
    - Word-error rates with Levenshtein distance.

Applications of Speech Technology

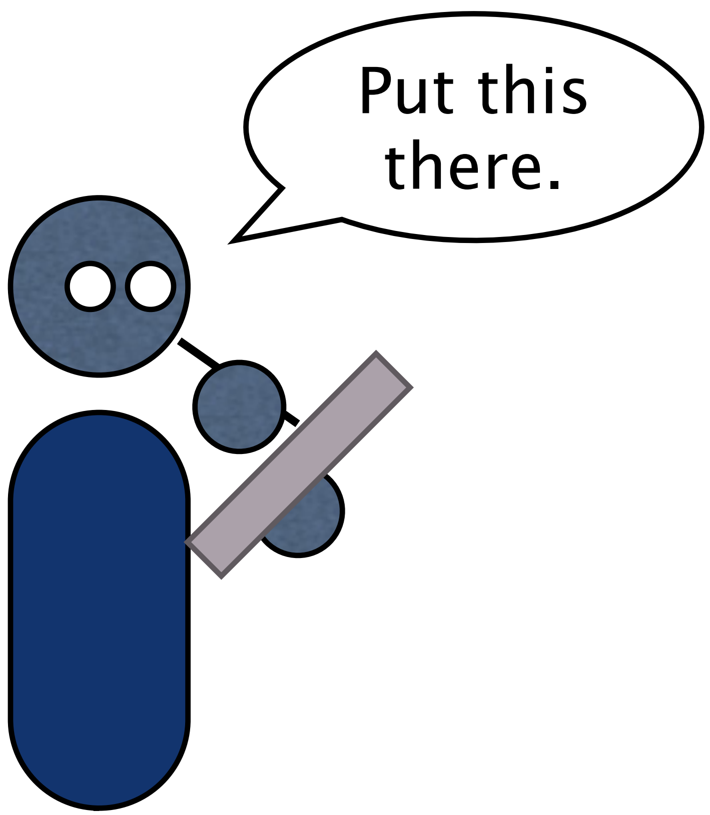
Telephony



Dictation



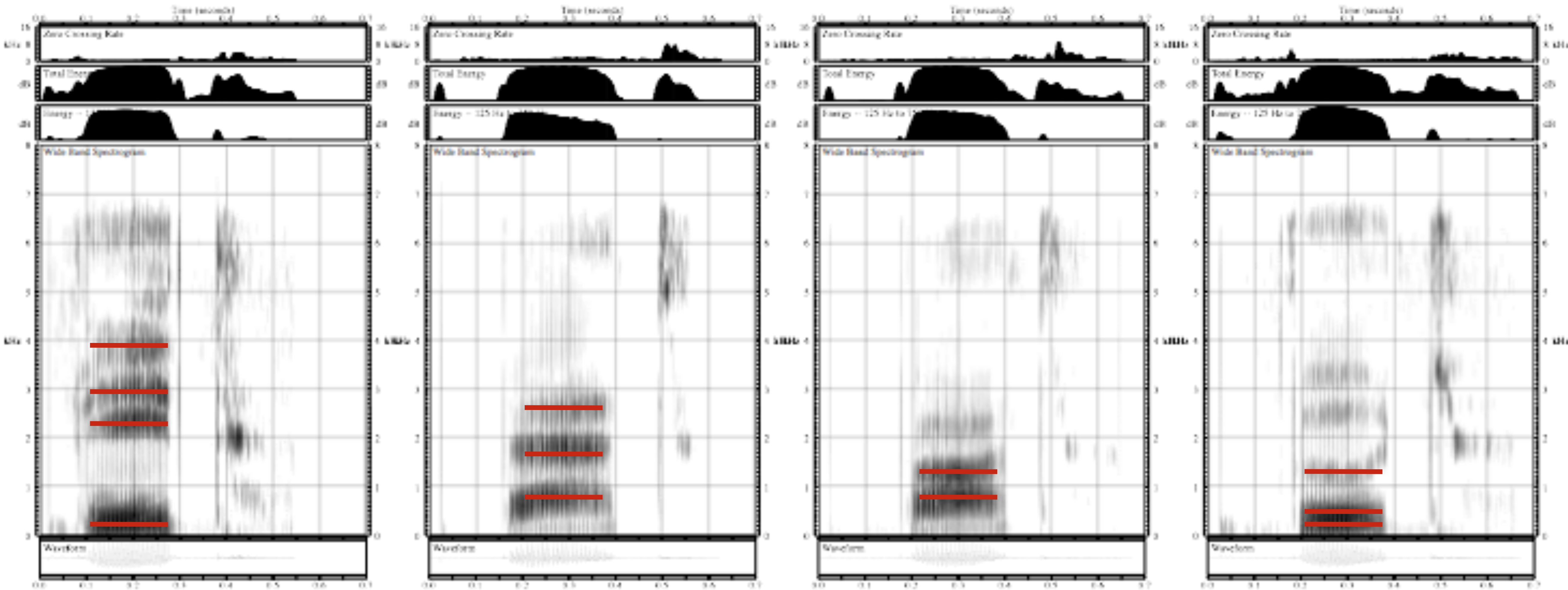
Multimodality & HCI



Emerging...

- Data mining/indexing.
- Assistive technology.
- Conversation.

Formants in sonorants

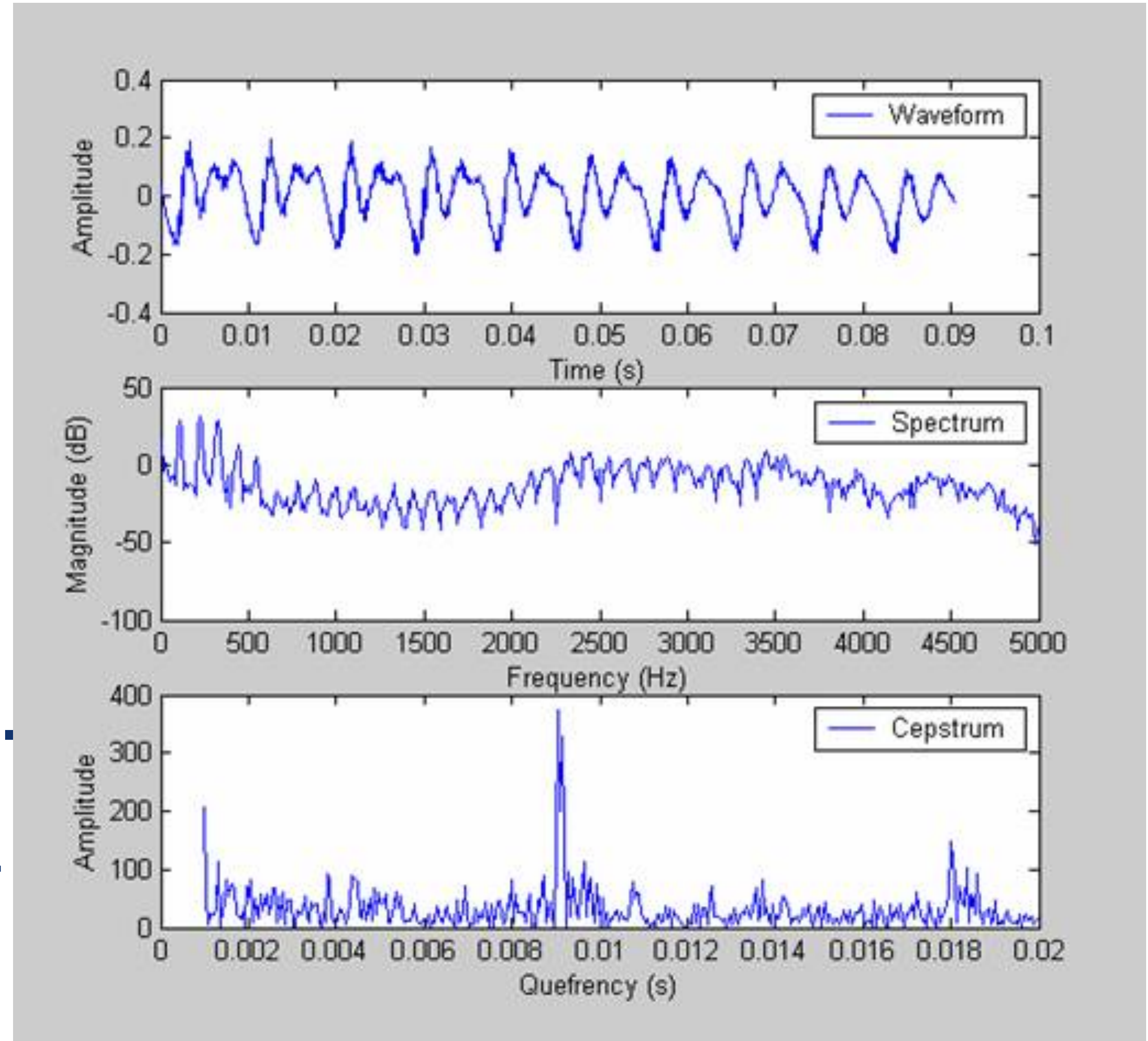


beet /bi^{y}t/ | bat /bæt/ | bott /bat/ | boot /but/

- However, formants are insufficient features for use in speech recognition generally…

Mel-frequency cepstral coefficients

- In real speech data, the spectrogram is often transformed to a representation that more closely represents human auditory response and is more amenable to accurate classification.

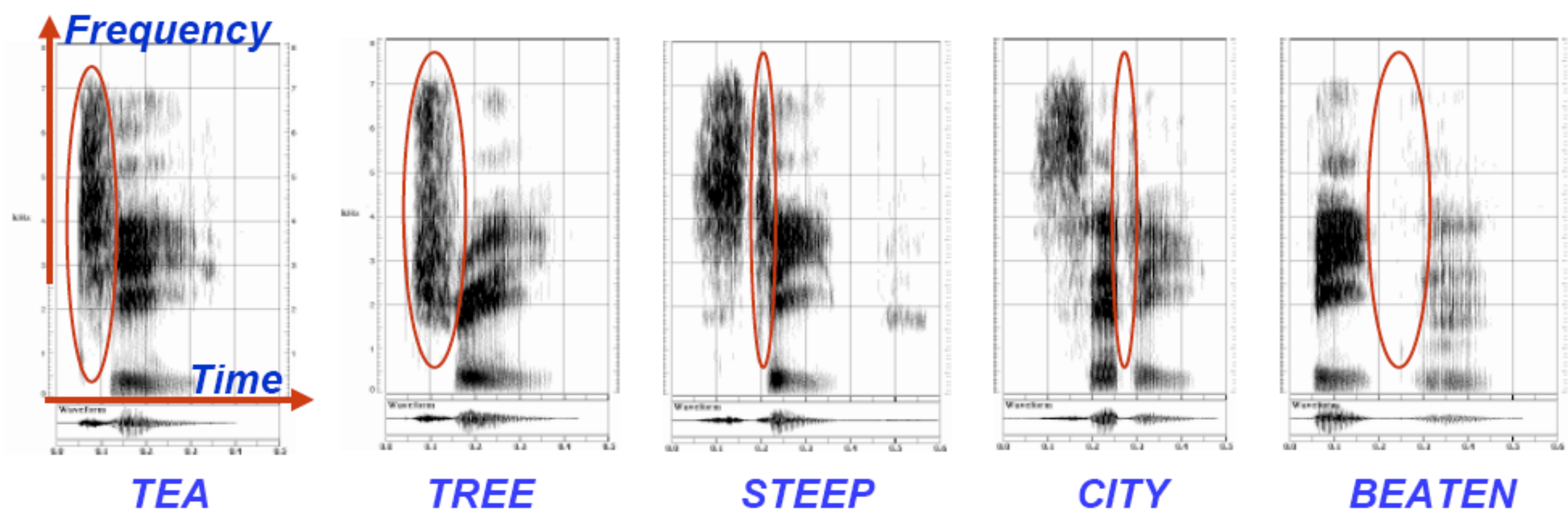


- MFCCs are 'spectra of spectra'. They are the discrete cosine transform of the logarithms of the nonlinearly Mel-scaled powers of the Fourier transform of windows of the original waveform.

UNIVERSITY OF TORONTO

Challenges in speech data

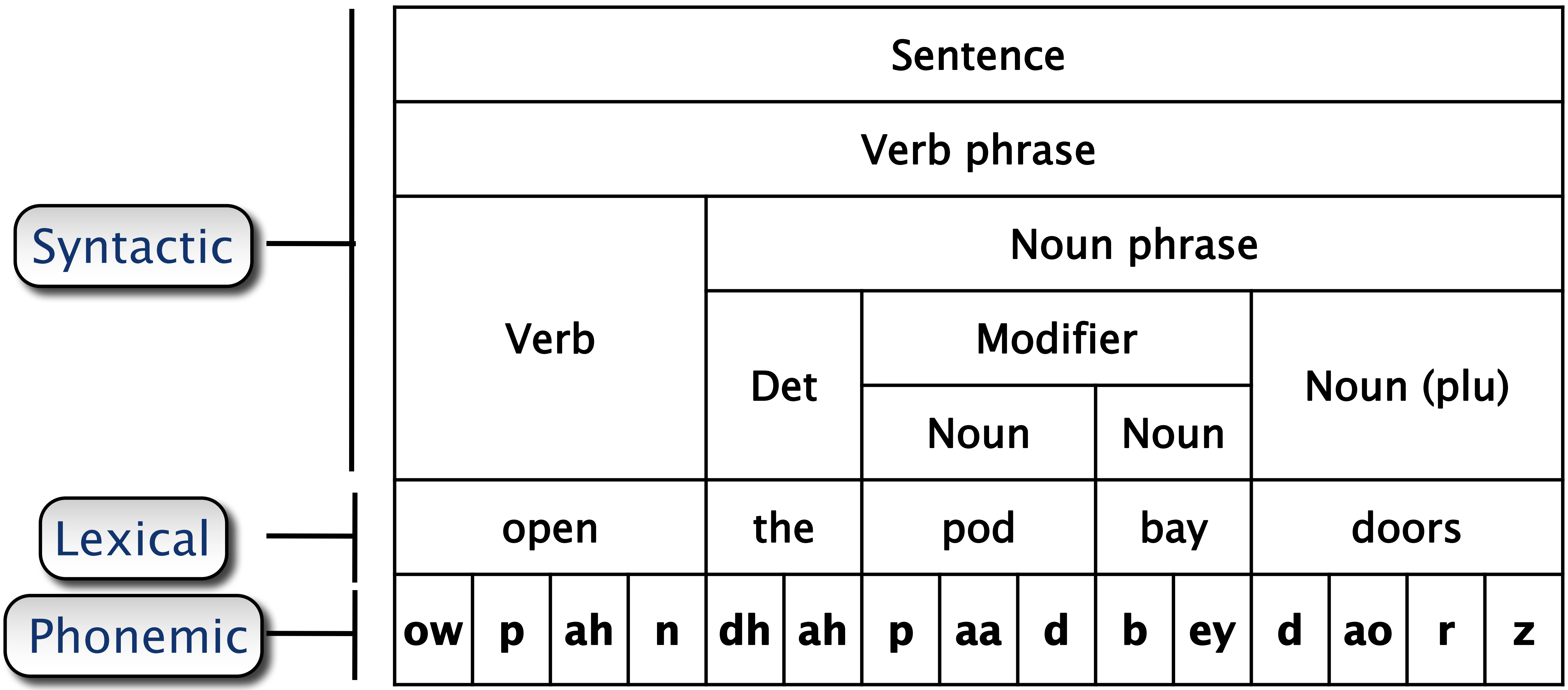
- Co-articulation and dropped phonemes.
- (Intra-and-Inter-) Speaker variability.
- No word boundaries.
- Slurring, disfluency (e.g., ‘um’).
- Signal Noise.
- Highly dimensional.



Phonemes

- Words are formed by **phonemes** (aka ‘phones’),
  e.g., ‘pod’ = /p aa d/

- Words have different pronunciations. and in practice we can never be certain of which phones were uttered, nor their start/stop points.



UNIVERSITY OF TORONTO

Phonetic alphabets

- International Phonetic Association (IPA)
  - Can represent sounds in all languages
  - Contains non-ASCII characters
- ARPAbet
  - One of the earliest attempts at encoding English for early speech recognition.
- **TIMIT/CMU**
  - Very popular among modern databases for speech recognition.
  - **Used in assignment 3**

Example phonetic alphabets

IPA	CMU	TIMIT	Example	IPA symbol name
[ɑ]	AA	aa	father, hot	script a
[æ]	AE	ae	had	digraph
[ə]	AH0	ax	sofa	schwa (common in unstressed syllables)
[ʌ]	AH1	ah	but	turned v
[ɔ:]	AO	ao	caught	open o – Note, many speakers of Am. Eng. do not distinguish between [ɔ:] and [ɑ]. If your "caught" and "cot" sound the same, you do not.
[ɛ]	EH	eh	head	epsilon
[ɪ]	IH	ih	hid	small capital I
[i:]	IY	iy	heed	lowercase i
[ʊ]	UH	uh	hood, book	upsilon
[u:]	UW	uw	boot	lowercase u
[aɪ]	AY	ay	hide	
[aʊ]	AW	aw	how	
[eɪ]	EY	ey	today	
[oʊ]	OW	ow	hoed	
[ɔɪ]	OY	oy	joy, ahoy	
[ɚ]	ER0	axr	herself	schwar (schwa changed by following r)
[ɝ]	ER1	er	bird	reverse epsilon right hook

IPA	CMU	TIMIT	Example	IPA symbol name
[ŋ]	NG	ng	sing song	eng or angma
[ʃ]	SH	sh	sheet, wish	esh or long s
[tʃ]	CH	ch	cheese	
[j]	Y	y	yellow	lowercase j
[ʒ]	ZJ	zh	vision	long z or yogh
[dʒ]	JH	jh	judge	
[ð]	DH	dh	thee, this	eth

- The other consonants are transcribed as you would expect
  - i.e., p, b, m, t, d, n, k, g, s, z, f, v, w, h

Agenda

- Background
  - Speech technology, in general
  - Acoustic phonetics
- Assignment 3
  - Speaker Recognition: Gaussian mixture models
  - Speech Recognition:
    - Continuous hidden Markov models
    - Transcription
    - Word-error rates with Levenshtein distance.

Assignment 3

- Two parts:
  - Speaker identification: Determine which of 30 speakers an unknown test sample of speech comes from, given Gaussian mixture models you will train for each speaker.
  - Speech recognition: Learn about phonetic annotation of speech data, training continuous hidden Markov models, and using these to identify phonemes with probabilities produced by the Forward algorithm. You will also learn to compute word-error rates with the Levenshtein distance.

Speaker Data

- 30 speakers (e.g., FCJF0, MDPK0).
- Each speaker has 9 training utterances.
    - e.g., `Training/FCJF0/SA1.*`, `Training/FCJF0/SI1028.*`
- Each utterance has 5 files:
    - `*.wav` : The original wave file.
    - `*.mfcc` : The relevant acoustic features.
    - `*.txt` : Sentence-level transcription.
    - `*.wrd` : Word-level transcription.
    - `*.phn` : Phoneme-level transcription.

Speaker Data (cont.)

- All you need to know: A speech utterance is an Nxd matrix
  - Each row represents the features of a d-dimensional point in time.
  - There are N rows in a sequence of N frames.
  - These data are in space-delimited text files `*mfcc`

data dimension

time

frames

	1	2		d
1	$X_1[1]$	$X_1[2]$	...	$X_1[d]$
2	$X_2[1]$	$X_2[2]$	...	$X_2[d]$
	...	...	...	...
N	$X_N[1]$	$X_N[2]$	...	$X_N[d]$

Speaker Data (cont.)

- You are given ‘transcription files’ (`*.wrd`, `*.txt`, `*.phn`) in which each line tells the start and end frames for a unit of speech.
- For example, if a `*.wrd` file has the line: `501 540 pod`, then frames 501 to 540 inclusive represent the word ‘pod’

data dimension

time ↓ frames	1	2		d
...	...	...	...	...
501	$X_{501}[1]$	$X_{501}[2]$	...	$X_{501}[d]$
...	...	...	...	...
540	$X_{540}[1]$	$X_{540}[2]$	...	$X_{540}[d]$
...	...	...	...	...

Agenda

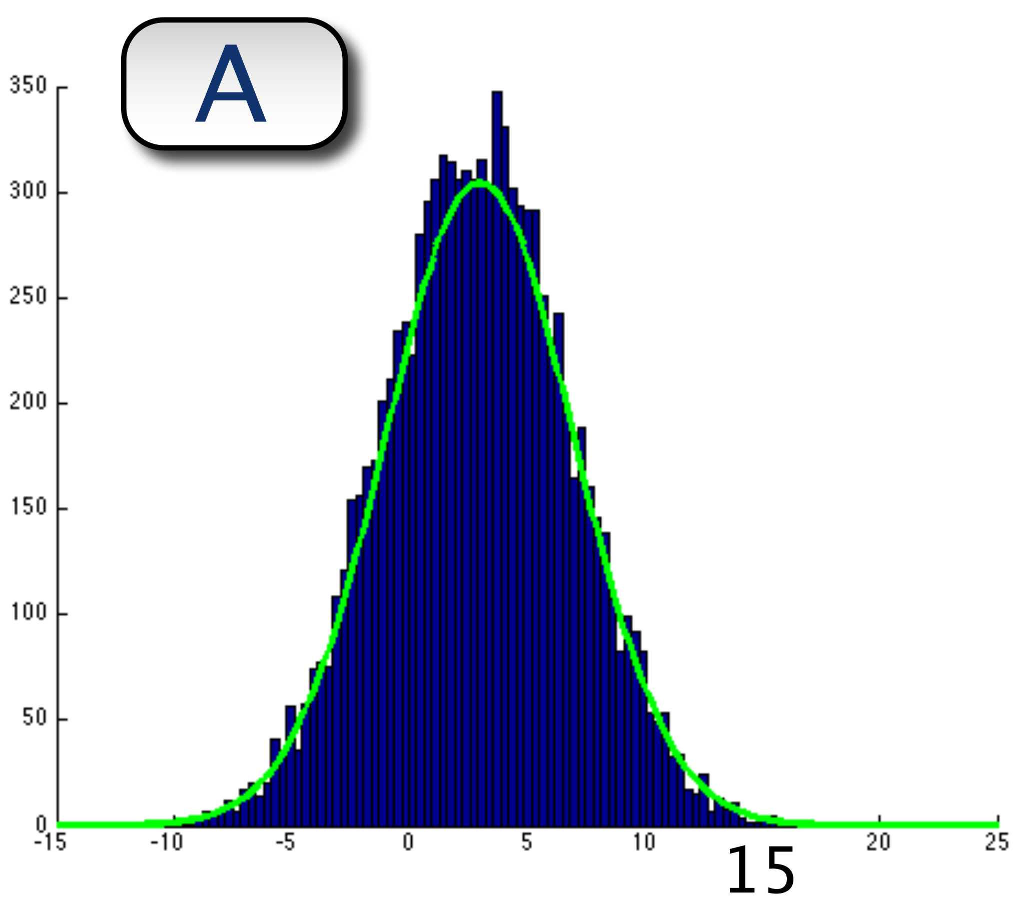
- Background
  - Speech technology, in general
  - Acoustic phonetics
- Assignment 3
  - Speaker Recognition: Gaussian mixture models
  - Speech Recognition:
    - Continuous hidden Markov models
    - Transcription
    - Word-error rates with Levenshtein distance.

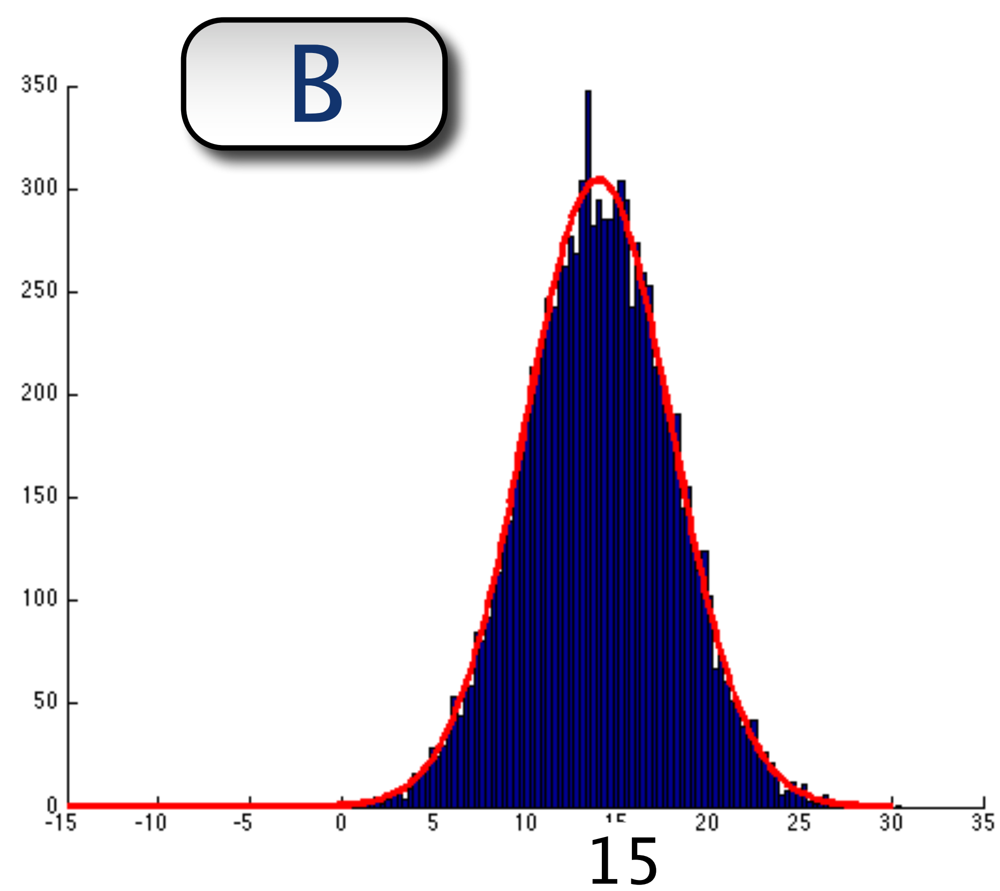
Speaker Recognition

- There are 30 testing utterances.
    - e.g., `Testing/unkn_1.*`, `Testing/unkn_2.*`
    - Each speaker produced 1 of these testing utterances.
    - We don't know which speaker produced which test utterance.
- Every speaker occupies a characteristic part of the acoustic space.
- We want to learn a probability distribution for each speaker that describes their acoustic behaviour.
    - Use those distributions to identify the speaker-dependant features of some unknown sample of speech data.

Some background: fitting to data

- Given a set of observations X of some random variable, we wish to know how X was generated.
- Here, we assume that the data was sampled from a Gaussian Distribution (validated by data).
- Given a new data point (x=15), It is more likely that x was generated by B.



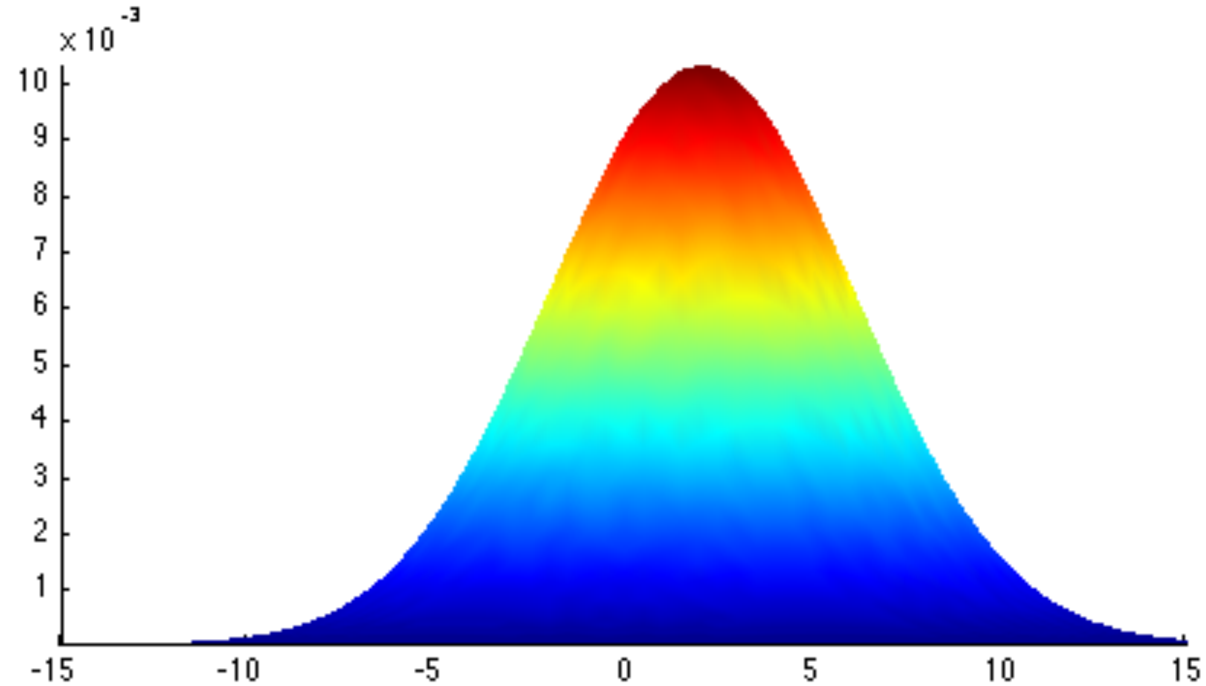


Finding parameters: 1D Gaussians

- Often called *Normal* distributions

$$p(x) = \frac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$N(\mu, \sigma)$$



$$\mu = E(x) = \int xp(x)dx$$

$$\sigma^2 = E((x-\mu)^2) = \int (x-\mu)^2 p(x)dx$$

- The parameters we can adjust to fit the data are μ and σ^2:

$$\theta = \langle \mu, \sigma \rangle$$

Maximum likelihood estimation

- Given data: $X = \{x_1, x_2, \ldots, x_n\}$
- and Parameter set: θ
- Maximum likelihood attempts to find the parameter set that maximizes the likelihood of the data.

$$L(X, \theta) = p(X \mid \theta) = p(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} p(x_i \mid \theta)$$

- The likelihood function $L(X, \theta)$ provides a surface over all possible parameterizations. In order to find the Maximum Likelihood, we set the derivative to zero:

$$\frac{\partial}{\partial \theta} L(X, \theta) = 0$$

MLE – 1D Gaussian

- Estimate $\hat{\mu}$:

$$L(X, \mu) = p(X \mid \mu) = \prod_{i=1}^{n} p(x_i \mid \mu) = \prod_{i=1}^{n} \frac{\exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$$

$$\log L(X, \mu) = -\frac{\sum_i (x_i - \mu)^2}{2\sigma^2} - n \log \sqrt{2\pi}\sigma$$

$$\frac{\partial}{\partial \mu} \log L(X, \mu) = \frac{\sum_i (x_i - \mu)}{\sigma^2} = 0$$

$$\hat{\mu} = \frac{\sum_i x_i}{n}$$

- A similar approach gives the MLE estimate of $\hat{\sigma}^2$:

$$\hat{\sigma}^2 = \frac{\sum_i (x_i - \hat{\mu})^2}{n}$$

UNIVERSITY OF TORONTO

Multidimensional Gaussians

- When your data is d-dimensional, the input variable is

$$\vec{x} = \langle x[1], x[2], \ldots, x[d] \rangle$$

the mean vector is

$$\vec{\mu} = E(\vec{x}) = \langle \mu[1], \mu[2], \ldots, \mu[d] \rangle$$

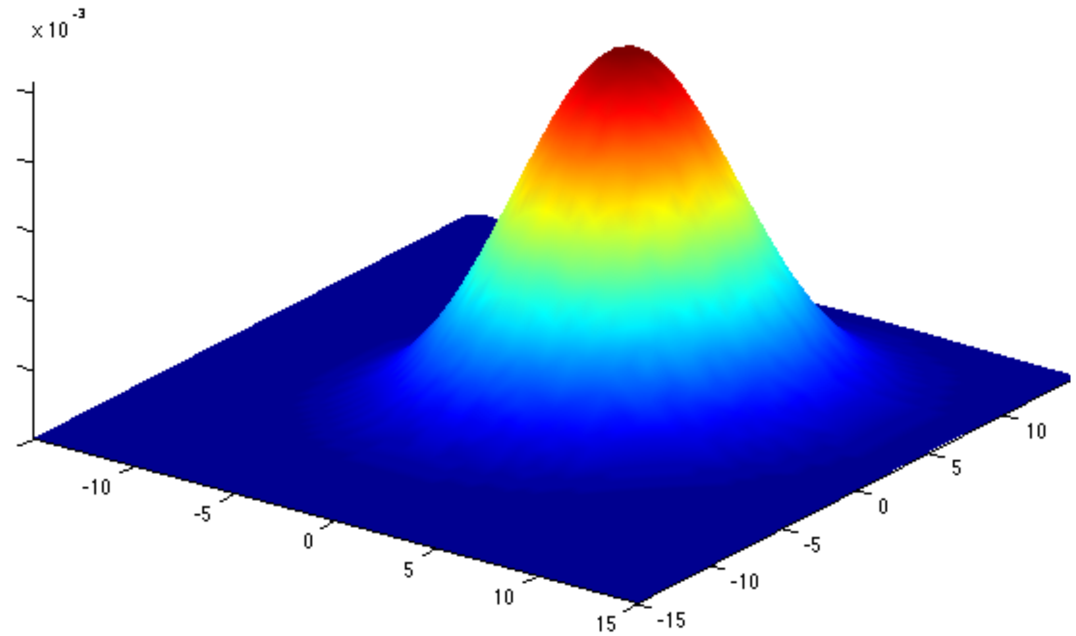
the covariance matrix is

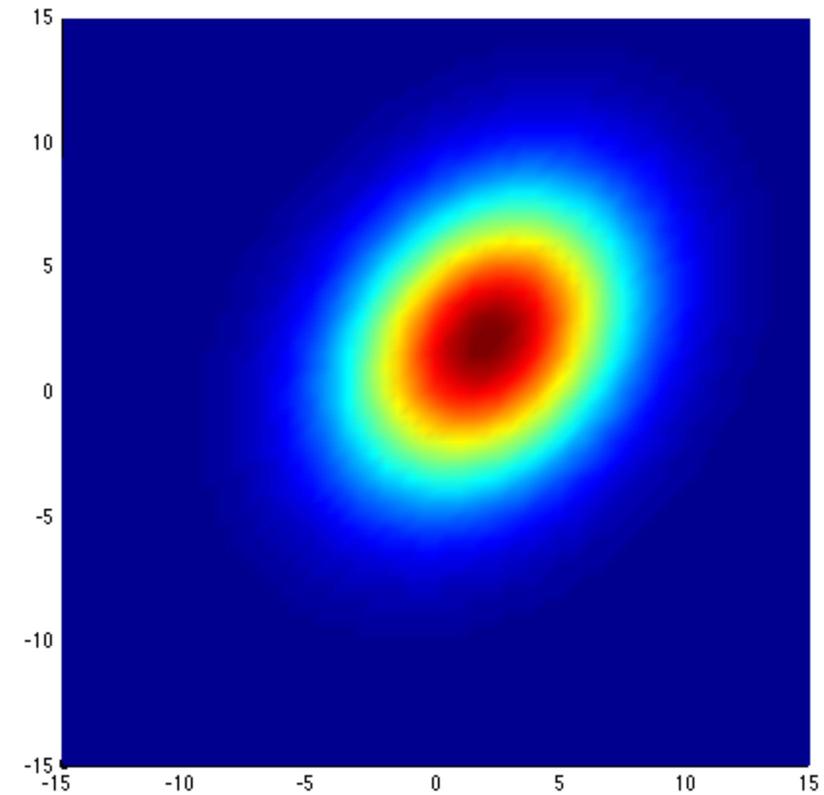
$$\Sigma = E((\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T)$$

with $\Sigma[i,j] = E(x[i]x[j]) - \mu[i]\mu[j]$

and

$$p(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{d/2} |\Sigma|^{1/2}}$$

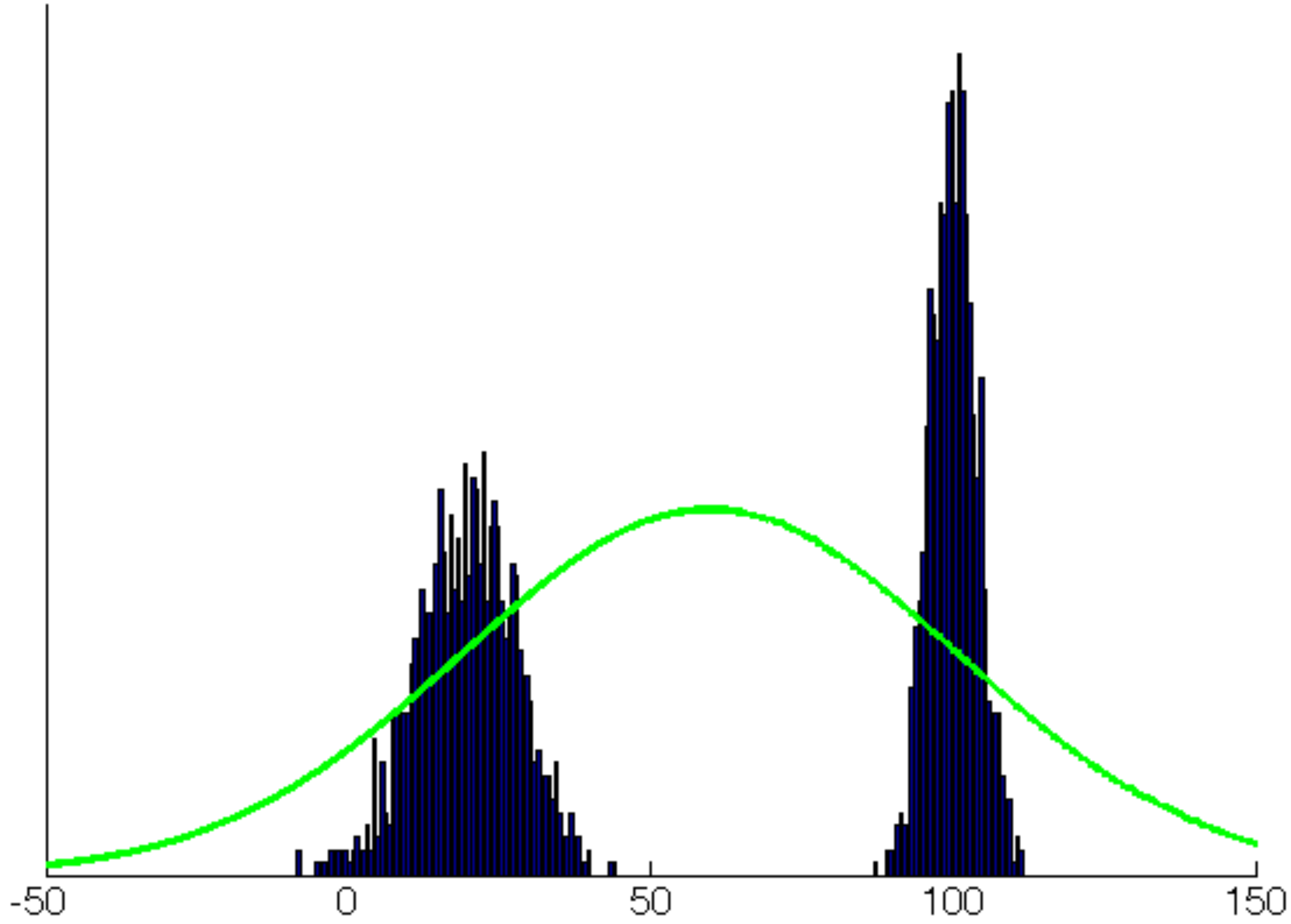




UNIVERSITY OF TORONTO

Non-Gaussian data

- Our speaker data does not behave unimodally.
  - i.e., we can't use just 1 Gaussian per speaker.
- E.g., observations below occur mostly bimodally, so fitting 1 Gaussian would not be representative.



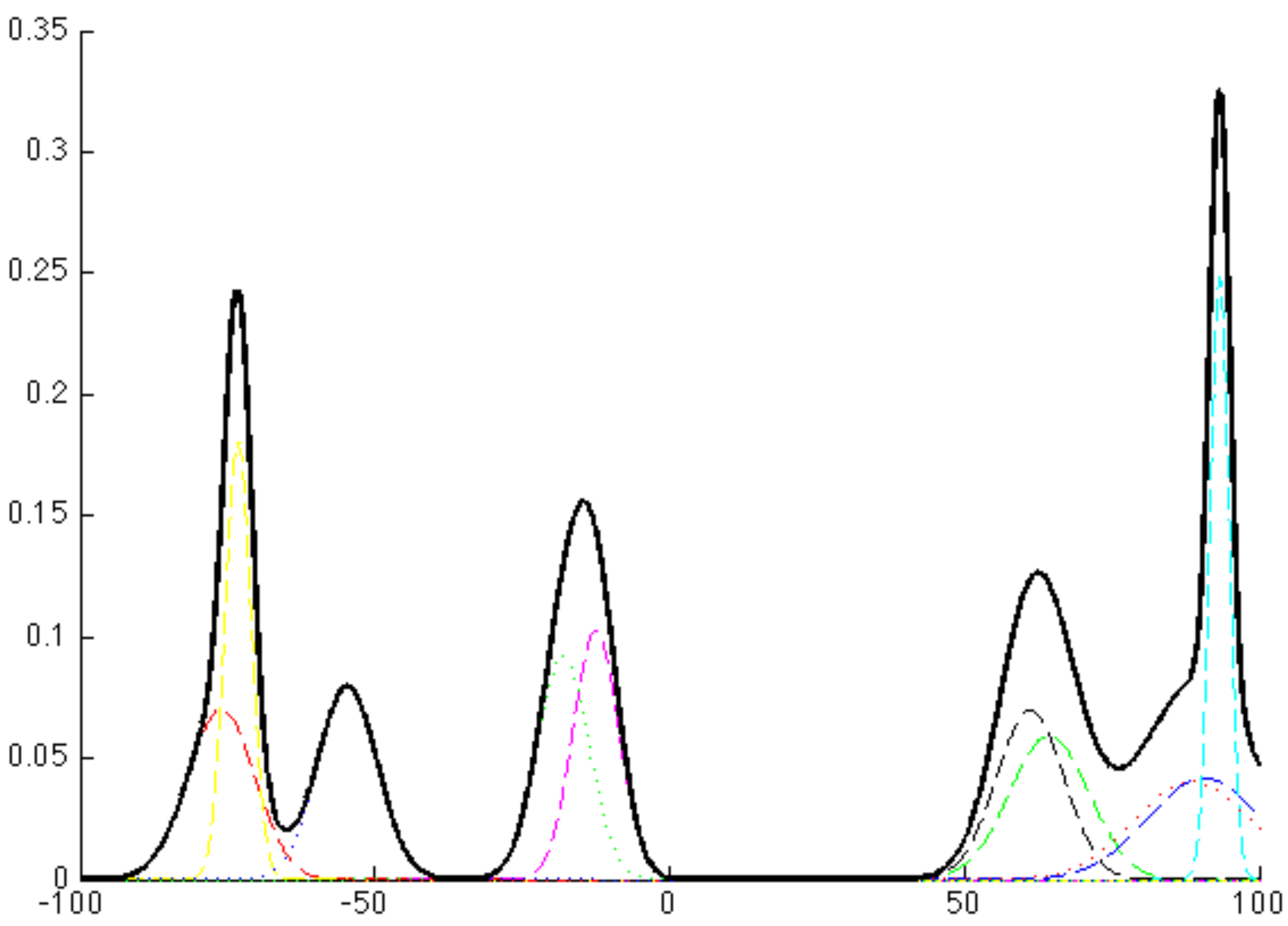
UNIVERSITY OF TORONTO

Gaussian mixtures

- Gaussian mixtures are a weighted linear combination of M component gaussians.

$$\langle \Gamma_1, \ldots, \Gamma_M \rangle$$

$$p(\vec{x}) = \sum_{j=1}^{M} p(\Gamma_j) p(\vec{x} \mid \Gamma_j)$$



MLE for Gaussian mixtures

- For notational convenience, $\omega_m = p(\Gamma_m),\ b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$
- So

$$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t}),\ \Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle,\ m = 1, \ldots, M$$

$$\boxed{b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d} \frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d} \sigma_m^2[i]\right)^{1/2}}}$$

- To find $\hat{\Theta}$, we solve $\nabla_\Theta \log L(X, \Theta) = 0$ where

$$\log L(X, \Theta) = \sum_{t=1}^{N} \log p_\Theta(\vec{x_t}) = \sum_{t=1}^{N} \log\left(\sum_{m=1}^{M} \omega_m b_m(\vec{x_t})\right)$$

...see Appendix for more

MLE for Gaussian mixtures (pt. 2)

- Given $$\frac{\partial \log L(X,\Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{1}{p_\Theta(\vec{x_t})} \left[\frac{\partial}{\partial \mu_m[n]} \omega_m b_m(\vec{x_t}) \right]$$
- Since $$\frac{\partial}{\partial \mu_m[n]} b_m(\vec{x_t}) = b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]}$$
- We obtain $\hat{\mu_m}[n]$ by solving for $\mu_m[n]$ in :

$$\frac{\partial \log L(X,\Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t}) \frac{x_t[n] - \mu_m[n]}{\sigma_m^2[n]} = 0$$

$$b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$$
$$p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t})$$

and:

$$\hat{\mu_m}[n] = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta) x_t[n]}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}$$

Recipe for GMM ML estimation

- Do the following for each speaker individually. Use all the frames available in their respective **Training** directories

1. **Initialize**: Guess $\Theta = \langle \omega_m, \vec{\mu_m}, \Sigma_m \rangle,\ m = 1, \ldots, M$ with M random vectors in the data, or by performing M-means clustering.
2. **Compute likelihood**: Compute $b_m(\vec{x_t})$ and $P(\Gamma_m \mid \vec{x_t}, \Theta)$
3. **Update parameters**:

$$\hat{\omega_m} = \frac{1}{T}\sum_{t=1}^{T} p(\Gamma_m \mid \vec{x_t}, \Theta)$$

$$\hat{\vec{\mu_m}} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)\vec{x_t}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)}$$

$$\hat{\vec{\sigma}}_m^{\,2} = \frac{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)\vec{x_t}^{\,2}}{\sum_t p(\Gamma_m \mid \vec{x_t}, \Theta)} - \hat{\vec{\mu}}_m^{\,2}$$

Repeat 2&3 until converges

$$\log p(X \mid \hat{\Theta}_{i+1}) - \log p(X \mid \hat{\Theta}_i) < \epsilon$$

Cheat sheet

$$b_m(\vec{x_t}) = p(\vec{x_t} \mid \Gamma_m)$$

$$b_m(\vec{x_t}) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x_t[i]-\mu_m[i])^2}{\sigma_m^2[i]}\right)}{(2\pi)^{d/2}\left(\prod_{i=1}^{d}\sigma_m^2[i]\right)^{1/2}}$$

Probability of observing x_t in the m^{th} Gaussian

$$\omega_m = p(\Gamma_m)$$

Prior probability of the m^{th} Gaussian

$$p(\Gamma_m \mid \vec{x_t}, \Theta) = \frac{\omega_m}{p_\Theta(\vec{x_t})} b_m(\vec{x_t})$$

Probability of the m^{th} Gaussian, given x_t

$$p_\Theta(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t})$$

Probability of x_t in the GMM

Initializing theta

$$\Theta = \langle \omega_1, \mu_1, \Sigma_1, \omega_2, \mu_2, \Sigma_2, \ldots, \omega_M, \mu_M, \Sigma_M \rangle$$

- Initialize each mu_m to a random vector from the data.
- Initialize $Sigma_m$ to a `random' diagonal matrix (or identity matrix).
- Initialize $omega_m$ randomly, with these constraints:

$$0 \leq \omega_m \leq 1$$

$$\sum_m \omega_m = 1$$

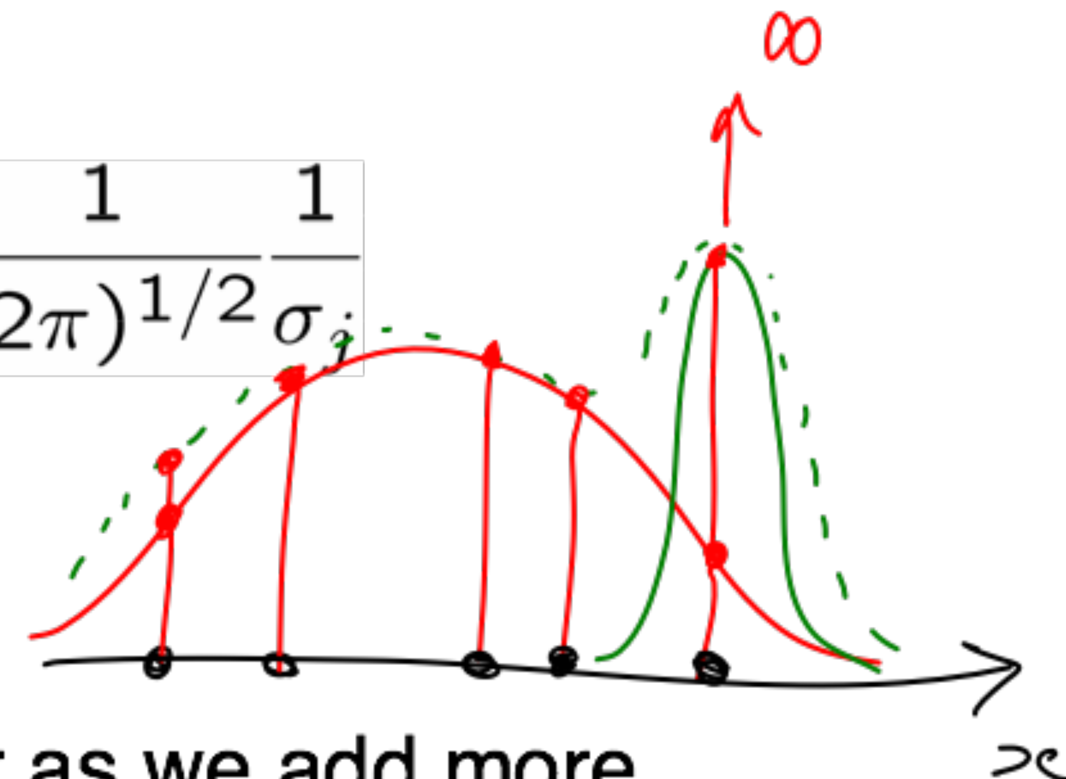
- A good choice would be to set $omega_m$ to 1/M

Over-fitting in Gaussian Mixture Models

- Singularities in likelihood function when a component 'collapses' onto a data point:

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma_j^2\mathbf{I}) = \frac{1}{(2\pi)^{1/2}}\frac{1}{\sigma_j}$$

then consider $\sigma_j \to 0$



- Likelihood function gets larger as we add more components (and hence parameters) to the model
  - not clear how to choose the number *K* of components

Solutions:

- Ensure that the variances don't get too small.
- Bayesian GMMs

* Slide borrowed from Chris Bishop's presentation

Your Task

- For each speaker, train a GMM, gmmTrain, using the EM algorithm, assuming diagonal covariance.
- Identify the speaker of each test utterance: gmmClassify.
- Experiment with the number of mixture elements in the models, the improvement threshold, number of possible speakers, etc.

Practical tips for MLE of GMMs

- We assume diagonal covariance matrices. This reduces the number of parameters and can be sufficient in practice given enough components.
- Numerical Stability: Compute likelihoods in the log domain (especially when calculating the likelihood of a sequence of frames).

$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d} \frac{(\vec{x_t}[n] - \vec{\mu_m}[n])^2}{2\vec{\sigma_m}^2[n]} - \frac{d}{2}\log 2\pi - \frac{1}{2}\log \prod_{n=1}^{d} \vec{\sigma_m}^2[n]$$

- Here, $\vec{x_t},\ \vec{\mu_m}$ and $\vec{\sigma_m}^2$ are d-dimensional vectors.

Practical tips (pt. 2)

- Efficiency: Pre-compute terms not dependent on $\vec{x_t}$

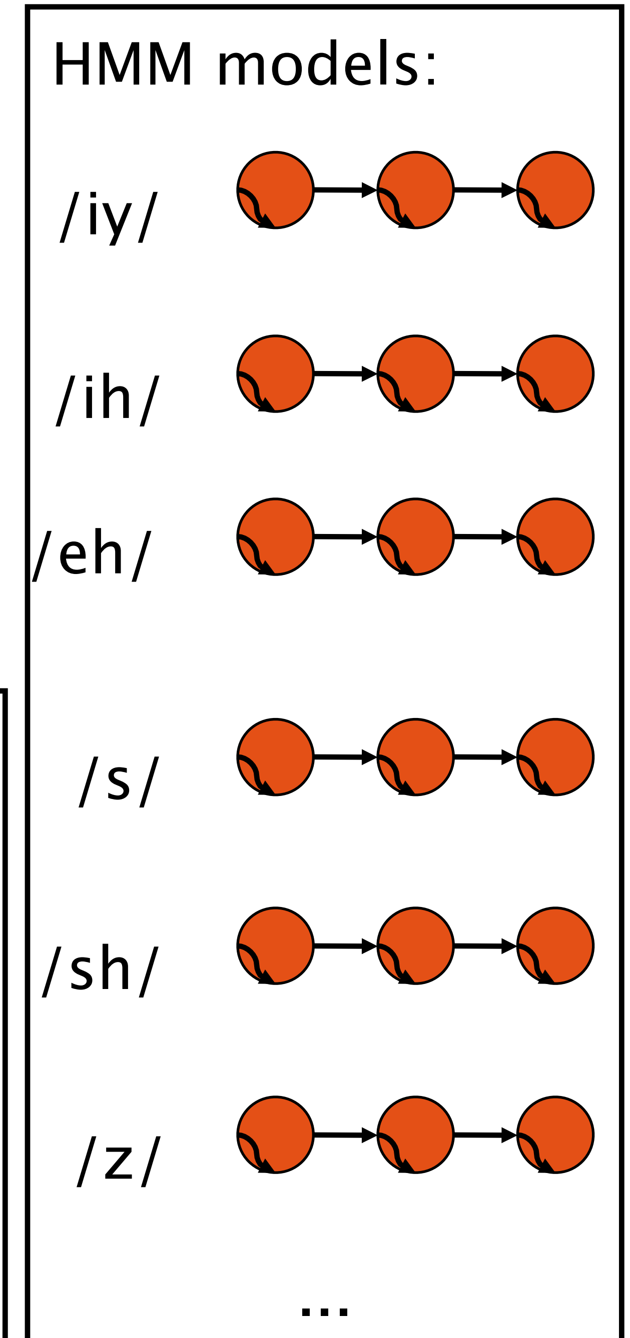
$$\log b_m(\vec{x_t}) = -\sum_{n=1}^{d}\left(\frac{1}{2}\vec{x_t}[n]^2\vec{\sigma_m}^{-2}[n] - \vec{\mu_m}[n]\vec{x_t}[n]\vec{\sigma_m}^{-2}[n]\right) - \left(\sum_{n=1}^{d}\frac{\vec{\mu_m}[n]^2}{2\vec{\sigma_m}^{2}[n]} + \frac{d}{2}\log 2\pi + \frac{1}{2}\log\prod_{n=1}^{d}\vec{\sigma_m}^{2}[n]\right)$$

Agenda

- Background
  - Speech technology, in general
  - Acoustic phonetics
- Assignment 3
  - Speaker Recognition: Gaussian mixture models
  - Speech Recognition:
    - Continuous hidden Markov models
    - Transcription
    - Word-error rates with Levenshtein distance.

Speech recognition task

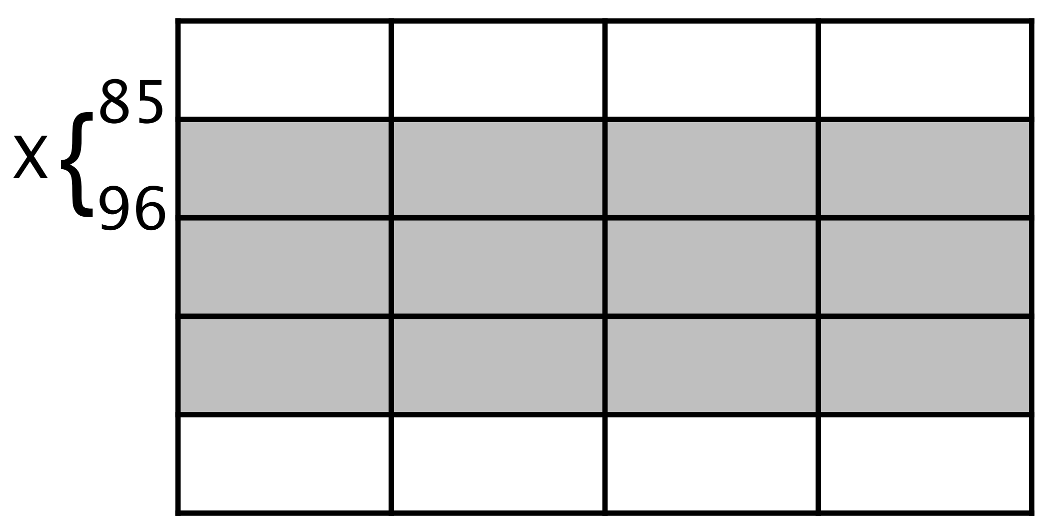
- You will train one HMM for each phoneme in the database.
- Given phonetically annotated test utterances, determine if the correct HMM gives the highest probability for each phone segment.



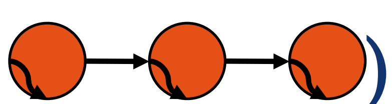
e.g.,

...
64 85 ae
85 96 s
96 102 epi
102 106 m
...

unkn_24.phn



unkn_24.mfcc

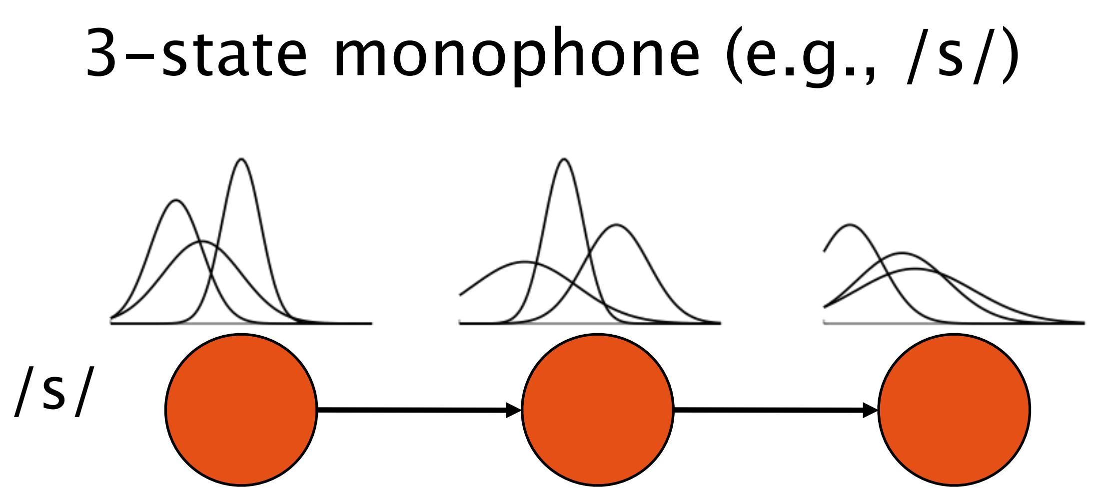
Is P(X| /s/ ) > P(X| all other HMMs)?

UNIVERSITY OF TORONTO

Continuous multivariate HMMs

- Multivariate HMMs with continuous GMM emission likelihoods are very similar to discrete HMMs, except:
  - Instead of reading letters or words from a discrete alphabet, we're reading d-dimensional feature vectors consisting of real numbers.
  - Observation likelihoods are computed with Gaussian mixture models.
- For each phoneme, you should have a three-state HMM:

e.g.,



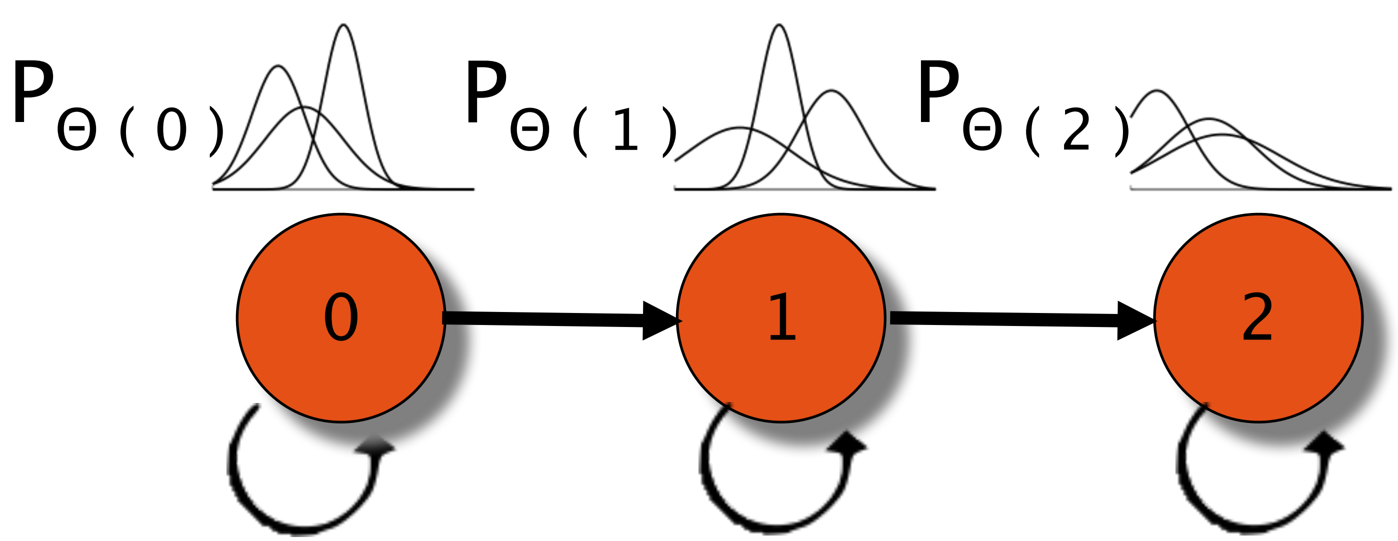
UNIVERSITY OF TORONTO

Continuous, multivariate HMMs

Given a d-dimensional transformation, $\vec{x_t}$, which represents ~16ms of speech and given that you're in state s at time t,

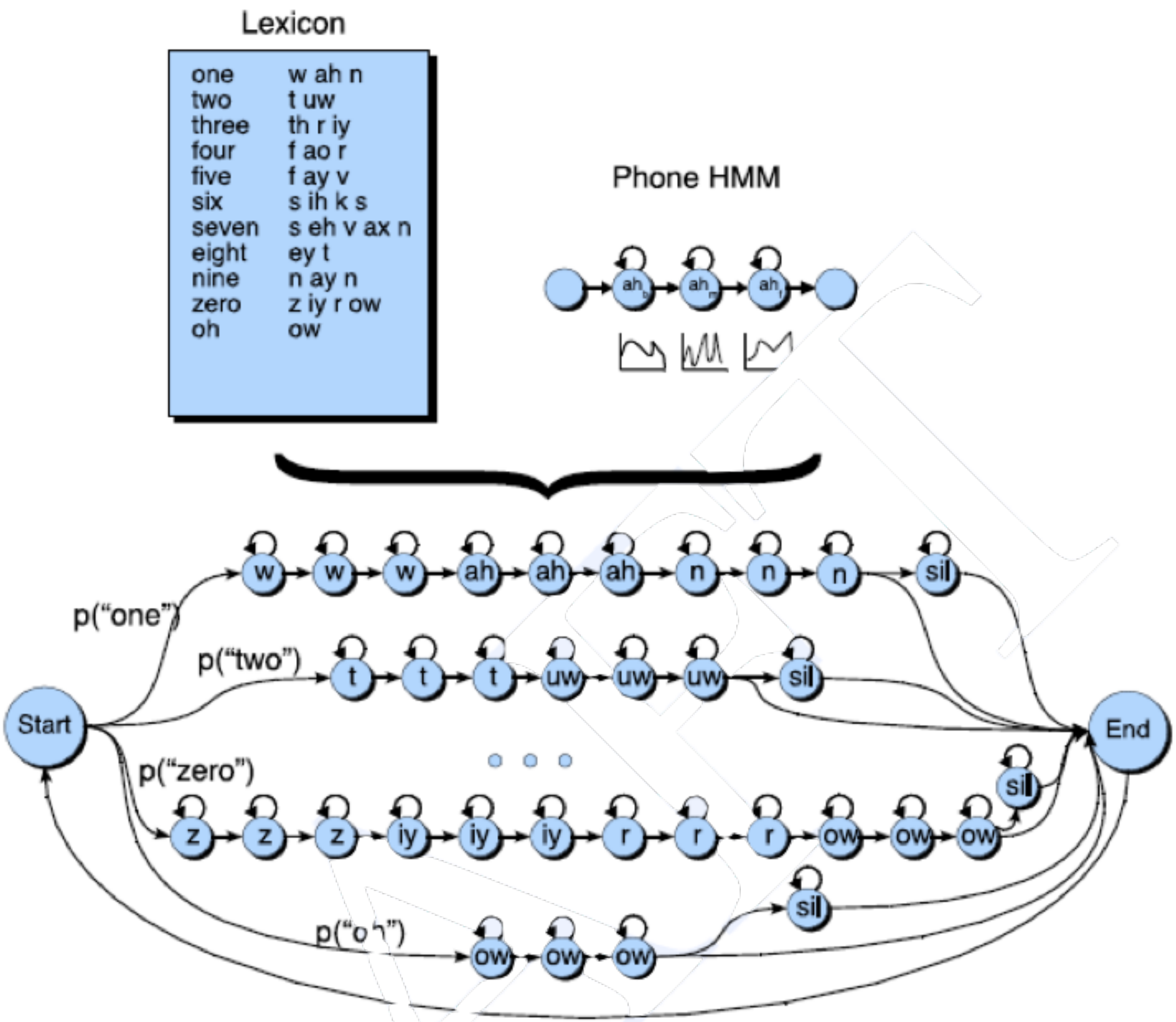
compute $p_{\Theta(s)}(\vec{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\vec{x_t})$, which is the standard

GMM output formula. Each state has its own parameters



3-state monophone (e.g., /s/)

Concatenating phones into words



Your task

- Use Bayes Net Tool Box: initHMM, loglikHMM, trainHMM, train a set of HMMs each representing a single phoneme.
- Find the HMM in your model set that produces the highest likelihood for each phone sequence of your data.
- Experiment with the number of states in your HMM, dimensionality of your data, number of mixtures in your GMMs, the amount of training data used, etc.
- Optionally, write your own version of loglikHMM (bonus) using Forward Algorithm.
- Write Levenshtein.m.

Tips

- During training you might get messages like

  `******likelihood decreased from X to Y!`

  This is an artifact of the way the toolbox estimates likelihoods.

  You can usually ignore these messages, but they may be indicative of non-ideal estimates – I don't get them.

- Each phone will usually take between **5** and **15** EM iterations.

- Training your HMMs can take up to **4** or **5** hours at 100% CPU.
  - Debug your code on a small subset of all training data.
  - Try using `screen`.

Agenda

- Background
    - Speech technology, in general
    - Acoustic phonetics
- Assignment 3
    - Speaker Recognition: Gaussian mixture models
    - Speech Recognition:
        - Continuous hidden Markov models
        - Transcription
        - Word-error rates with Levenshtein distance.

Word-error rates

- If somebody said
  REF: how to recognize speech
  but an ASR system heard
  HYP: how to wreck a nice beach
  how do we measure the error that occurred?
- One measure is #CorrectWords/#HypothesisWords
  e.g., 2/6 above
- Another measure is (S+I+D)/#ReferenceWords
  - S: # Substitution errors (one word for another)
  - I: # Insertion errors (extra words)
  - D: # Deletion errors (words that are missing).

Computing Levenshtein Distance

- In the example
  REF: how to recognize speech.
  HYP: how to wreck a nice beach
  How do we count each of S, I, and D?
- If "wreck" is a substitution error, what about "a" and "nice"?

Computing Levenshtein Distance

- In the example
  REF: how to recognize speech.
  HYP: how to wreck a nice beach
  How do we count each of S, I, and D?
  If “wreck” is a substitution error, what about “a” and “nice”?

- Levenshtein distance:

```
Initialize R[0,0] = 0, and R[i,j] = ∞ for all i=0 or j=0
for i=1..n (#ReferenceWords)
    for j=1..m (#Hypothesis words)
        R[i,j] = min(   R[i−1,j] + 1        (deletion)
                        R[i−1,j−1]          (only if words match)
                        R[i−1,j−1]+1        (only if words differ)
                        R[i,j−1] + 1    )   (insertion)
Return 100*R(n,m)/n
```


UNIVERSITY OF TORONTO

Levenshtein example

		how	to	wreck	a	nice	beach
	0	∞	∞	∞	∞	∞	∞
how	∞	0	1	2	3	4	5
to	∞						
recognize	∞						
speech	∞						

Levenshtein example

		how	to	wreck	a	nice	beach
	0	∞	∞	∞	∞	∞	∞
how	∞	0	1	2	3	4	5
to	∞	1	0	1	2	3	4
recognize	∞						
speech	∞						

Levenshtein example

		how	to	wreck	a	nice	beach
	0	∞	∞	∞	∞	∞	∞
how	∞	0	1	2	3	4	5
to	∞	1	0	1	2	3	4
recognize	∞	2	1	1	2	3	4
speech	∞						

Levenshtein example

		how	to	wreck	a	nice	beach
	0	∞	∞	∞	∞	∞	∞
how	∞	0	1	2	3	4	5
to	∞	1	0	1	2	3	4
recognize	∞	2	1	1	2	3	4
speech	∞	3	2	2	2	3	4

Word-error rate is $4/4 = 100\%$

2 substitutions, 2 insertions

Appendices

Multidimensional Gaussians, pt. 2

- If the i^{th} and j^{th} dimensions are statistically independent,

$$E(x[i]x[j]) = E(x[i])E(x[j])$$

  and

$$\Sigma[i,j] = 0$$

- If all dimensions are statistically independent, $\Sigma[i,j] = 0, \; \forall i \neq j$ and the covariance matrix becomes diagonal, which means

$$p(\vec{x}) = \prod_{i=1}^{d} p(x[i])$$

  where

$$p(x[i]) \sim N(\mu[i], \Sigma[i,i])$$
$$\Sigma[i,i] = \sigma^2[i]$$

MLE example – dD Gaussians

- The MLE estimates for parameters $\Theta = \langle \theta_1, \theta_2, \ldots, \theta_d \rangle$ given i.i.d. training data $X = \langle \vec{x_1}, \ldots, \vec{x_n} \rangle$ are obtained by maximizing the joint likelihood

$$L(X, \Theta) = p(X \mid \Theta) = p(\vec{x_1}, \ldots, \vec{x_n} \mid \Theta) = \prod_{i=1}^{n} p(\vec{x_i} \mid \Theta)$$

- To do so, we solve $\nabla_\Theta L(X, \Theta) = 0$, where

$$\nabla_\Theta = \left\langle \frac{\partial}{\partial \theta_1}, \ldots, \frac{\partial}{\partial \theta_d} \right\rangle$$

- Giving

$$\hat{\vec{\mu}} = \frac{\sum_{t=1}^{n} \vec{x_t}}{n} \qquad \hat{\Sigma} = \frac{\sum_{t=1}^{n} \left(\vec{x_t} - \hat{\vec{\mu}}\right)\left(\vec{x_t} - \hat{\vec{\mu}}\right)^T}{n}$$

MLE for Gaussian mixtures (pt1.5)

- Given $\log L(X,\Theta) = \sum_{t=1}^{N} \log p_\Theta(\vec{x}_t)$ and $p_\Theta(\vec{x}_t) = \sum_{m=1}^{M} \omega_m b_m(\vec{x}_t)$

  - Obtain an ML estimate, $\hat{\vec{\mu}}_m$, of the mean vector by maximizing $\log L(X, \vec{\mu}_m)$ w.r.t. $\mu_m[n]$

$$\frac{\partial \log L(X,\Theta)}{\partial \mu_m[n]} = \sum_{t=1}^{N} \frac{\partial}{\partial \mu_m[n]} \log p_\Theta(\vec{x}_t) = \sum_{t=1}^{N} \frac{1}{p_\Theta(\vec{x}_t)} \left[\frac{\partial}{\partial \mu_m[n]} \omega_m b_m(\vec{x}_t) \right]$$

- Why?

d of sum = sum of d

d rule for $\log_e$

d wrt μ_m is 0 for all other mixtures in the sum in $p_\Theta(\vec{x}_t)$