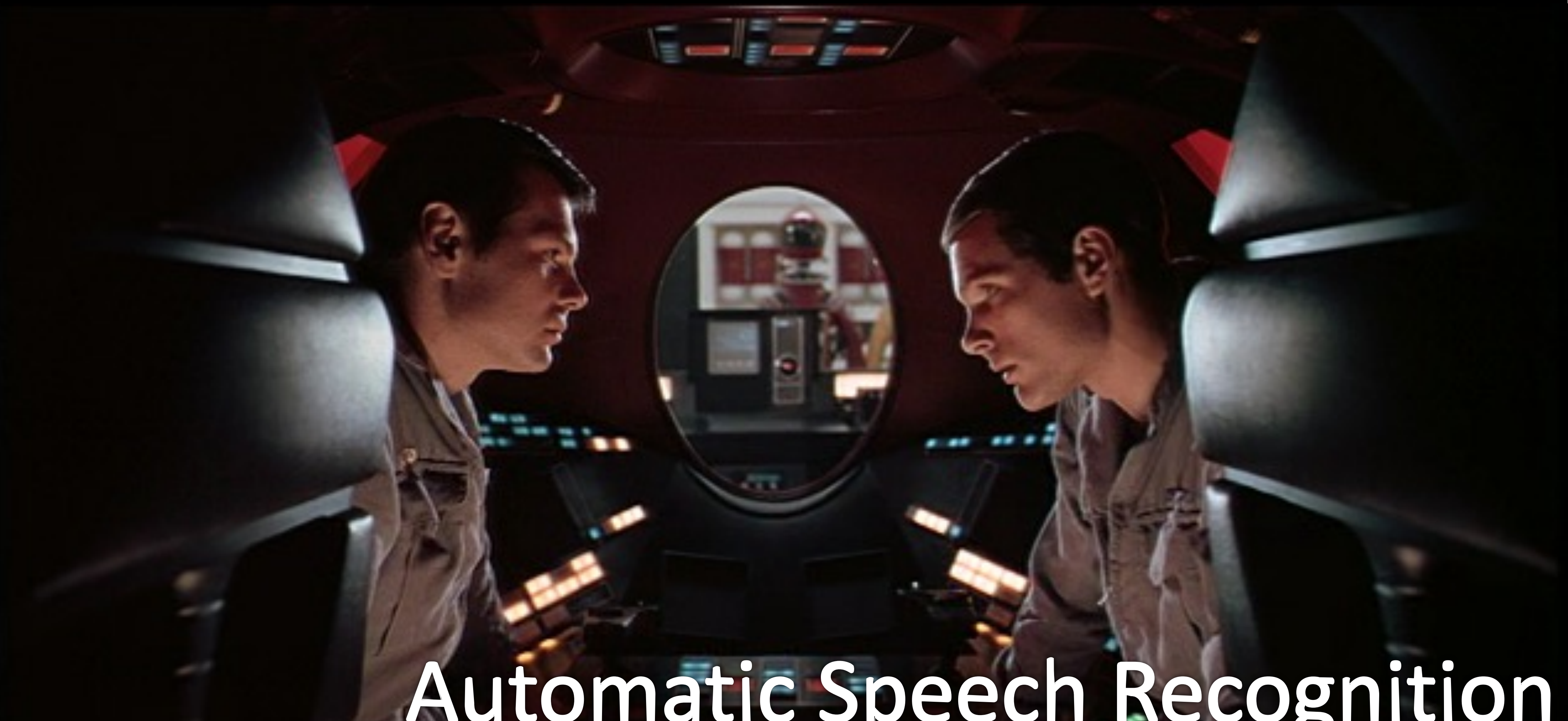


Automatic Speech Recognition

CSC401/2511 – Natural Language Computing – Spring 2022
Lecture 9
University of Toronto

Contents

Today we will discuss some building blocks:

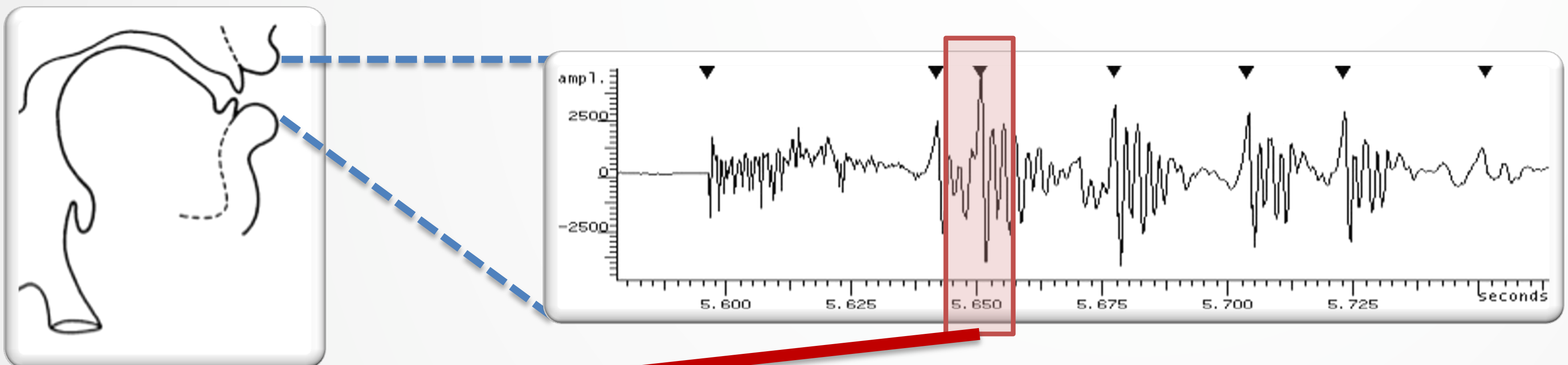
- Input features.
- Gaussian Mixtures Model.
- Clustering.

In next lecture (Wednesday) will discuss ASR systems:

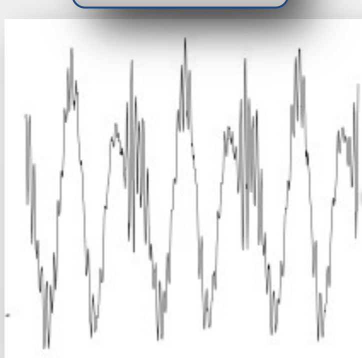
- Putting together an ASR system
- Evaluating with word error rate
- Neural speech recognition systems.

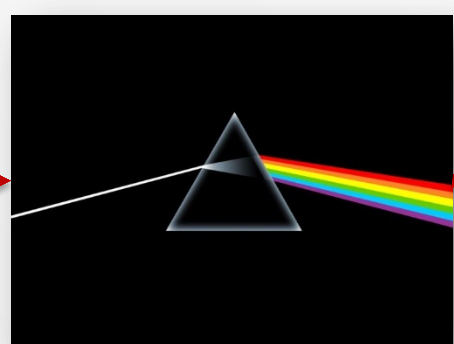
FEATURES

Recall our input to ASR

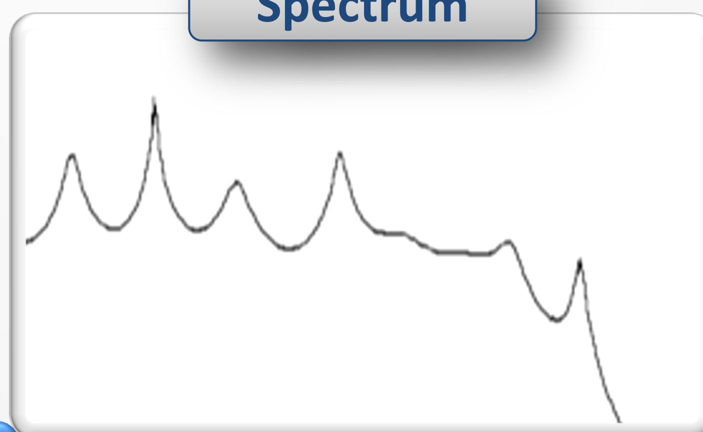


Frame





Spectrum



Is the spectrum the best input for our ASR systems?

Frequency (Hz)

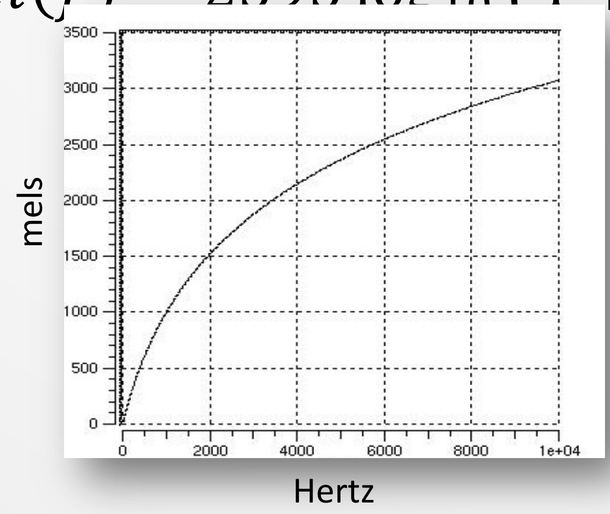
The mel-scale

- Human hearing is **not** equally sensitive to **all** frequencies.
  - We are **less** sensitive to frequencies > 1 kHz.

- A **mel** (after the word "melody") is a unit of pitch. Pairs of sounds which are **perceptually** equidistant in pitch are separated by an equal number of **mels**.

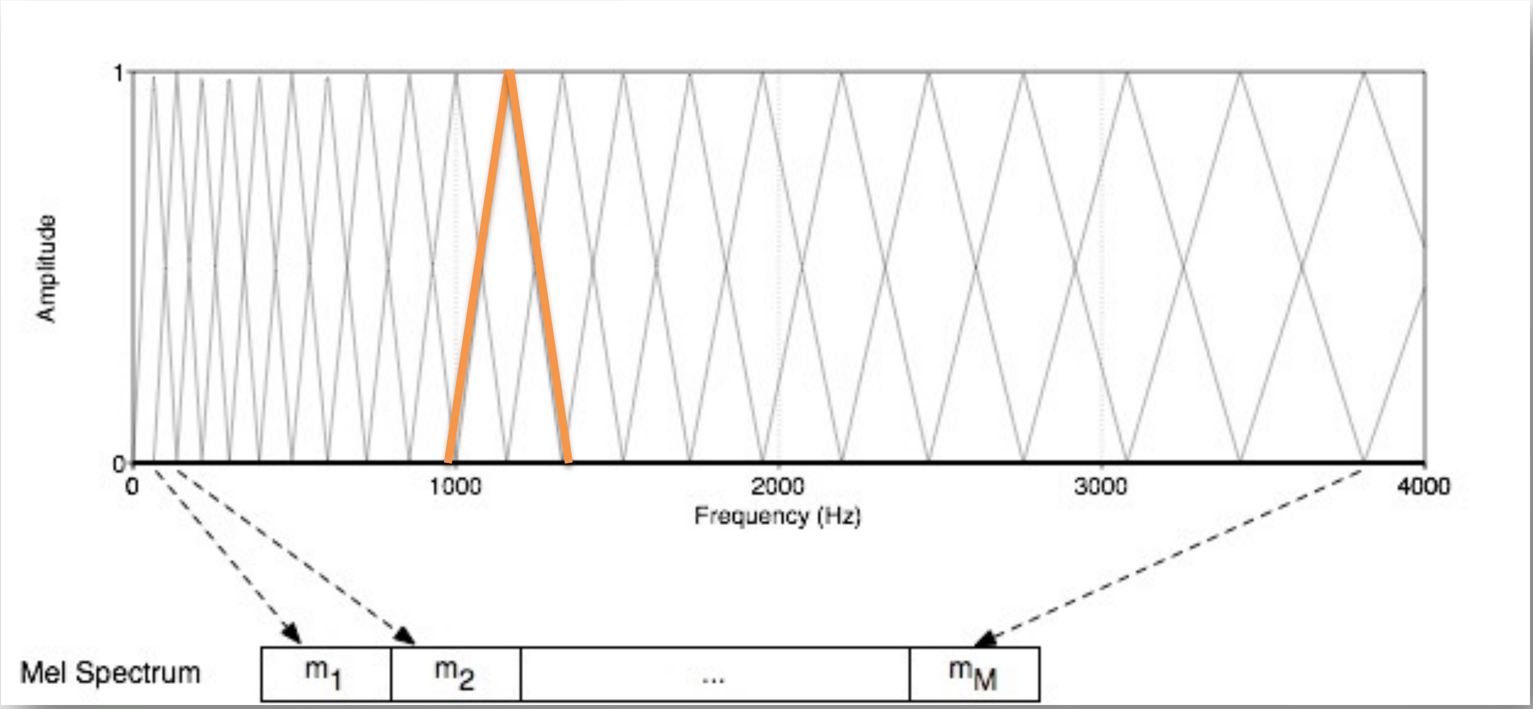
$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

(No need to memorize this either)



The mel-scale filter bank

- To **mimic** the response of the **human ear** (and because it *can* improve speech recognition), we often discretize the spectrum using M triangular **filters**.
  - **Uniform** in mel-scale, **logarithmic** in the frequency scale.



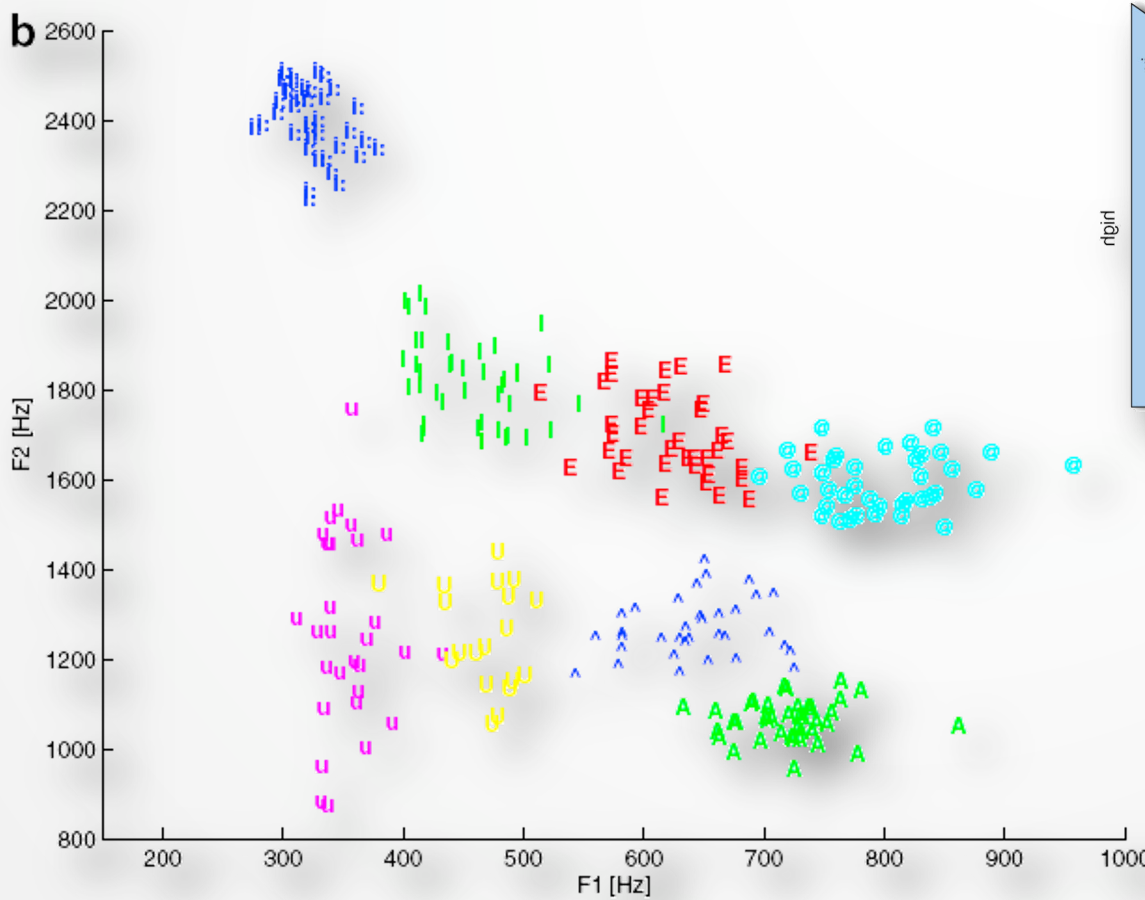
UNIVERSITY OF TORONTO

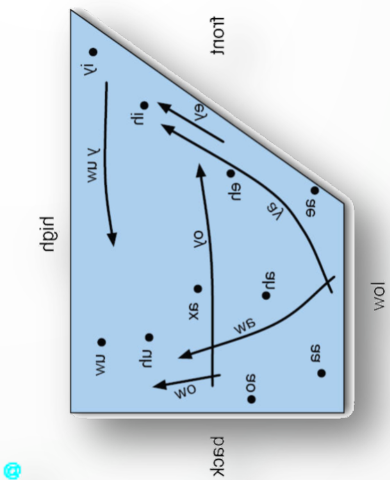
Aside - Mel-Frequency Cepstral Coefficients

- Earlier ASR required additional *Cepstral* processing on the Mel Spectrum
- Used to separate the **source** (glottal waveform) from **filter** (vocal tract resonances)
- MFCCs are used in Assignment 3
- Details on how to calculate them will be discussed in tutorial.
- Neural ASR usually uses the Mel-Spectrum as input
  - Good at **de-correlating** source and filter by itself

GAUSSIAN MIXTURES

Classifying speech sounds





Note: The vowel trapezoid's dimensions were physical

- Speech sounds can cluster. This graph shows vowels, each in their own colour, according to the 1st two formants.

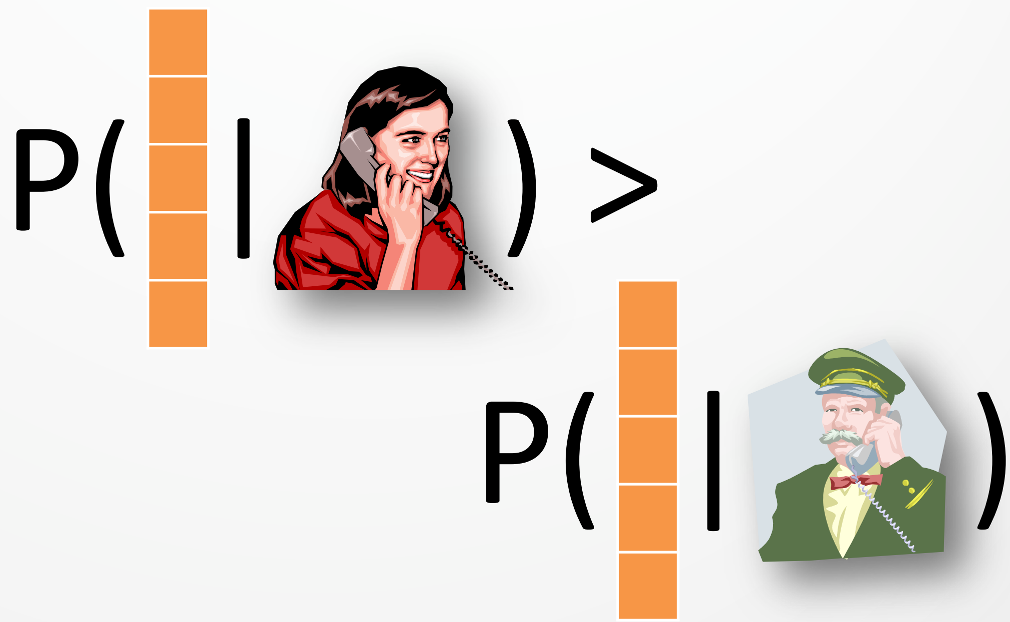
UNIVERSITY OF TORONTO

Classify speakers by cluster attributes

- Similarly, all of the speech produced by one **speaker** will cluster differently in the **Mel space** than speech from another speaker.
  - We can ∴ decide if a given observation comes from one speaker or another.

		Time, *t*			
		0	1	…	T
MFCC	1			…	
	2			…	
	3			…	
	…	…	…	…	…
	42			…	

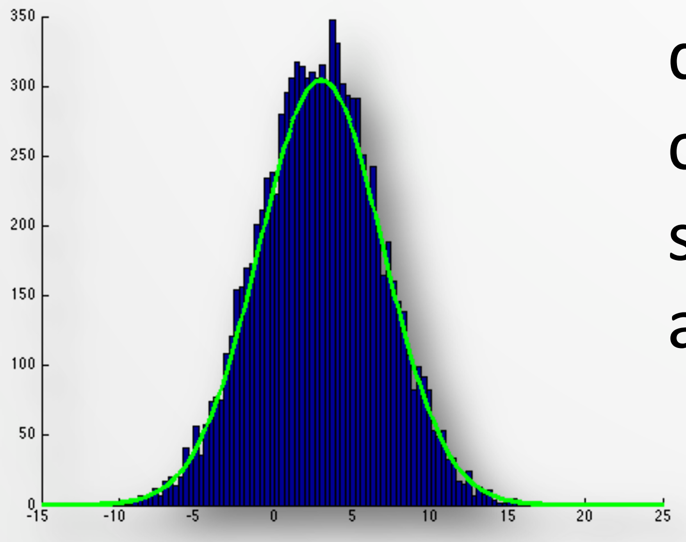
Observation matrix



UNIVERSITY OF TORONTO

Fitting continuous distributions

- Since we are operating with **continuous** variables, we need to **fit continuous probability** functions to a **discrete number** of observations.



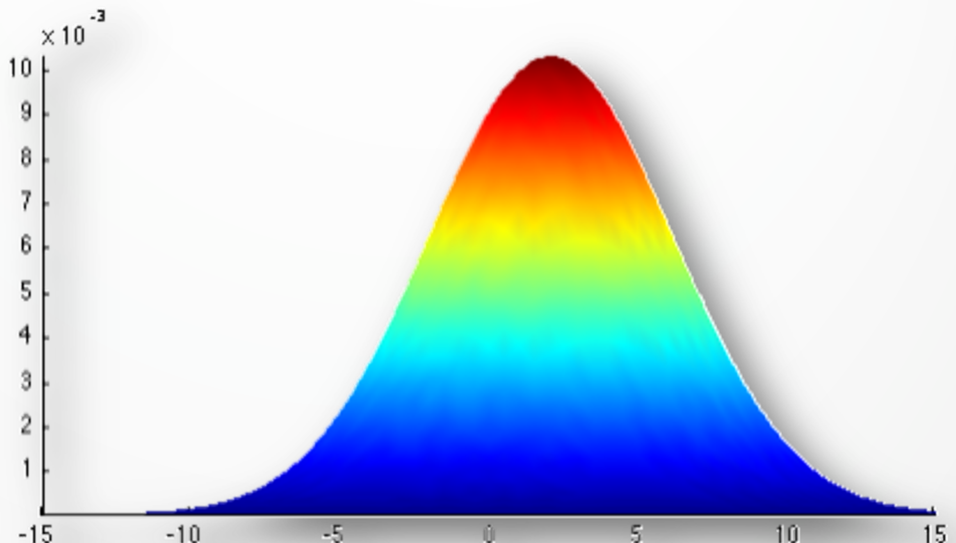
- If we *assume* the 1-dimensional data in **this histogram** is Normally distributed, we can fit a continuous Gaussian function simply in terms of the mean μ and variance σ^2.

UNIVERSITY OF TORONTO

(Aside) Univariate (1D) Gaussians

- Also known as **Normal** distributions, $N(\mu, \sigma)$

- $P(x;\mu,\sigma) = \dfrac{\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}{\sqrt{2\pi}\sigma}$



- The parameters we can modify are $\boldsymbol{\theta} = \langle \boldsymbol{\mu}, \boldsymbol{\sigma^2} \rangle$
  - $\mu = E(x) = \int x \cdot P(x)dx$ **(mean)**
  - $\sigma^2 = E\left((x-\mu)^2\right) = \int (x-\mu)^2 P(x)dx$ **(variance)**

But we don't have samples for all x…

UNIVERSITY OF TORONTO

Maximum likelihood estimation

- Given data $X = \{x_1, x_2, \ldots, x_n\}$, MLE produces an estimate of the parameters $\hat{\theta}$ by maximizing the **likelihood**, $L(X, \theta)$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \, L(X, \theta)$$

  where $\boldsymbol{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{P}(\boldsymbol{X}; \boldsymbol{\theta}) = \prod_{i=1}^{n} P(x_i; \theta)$.

- Since $L(X, \theta)$ provides a **surface** over all $\boldsymbol{\theta}$, in order to find the **highest likelihood**, we look at the derivative

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

  to see **at which point** the likelihood **stops growing**.

MLE with univariate Gaussians

- Estimate μ:

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

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

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

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

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

Multivariate Gaussians

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

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

the **mean** is

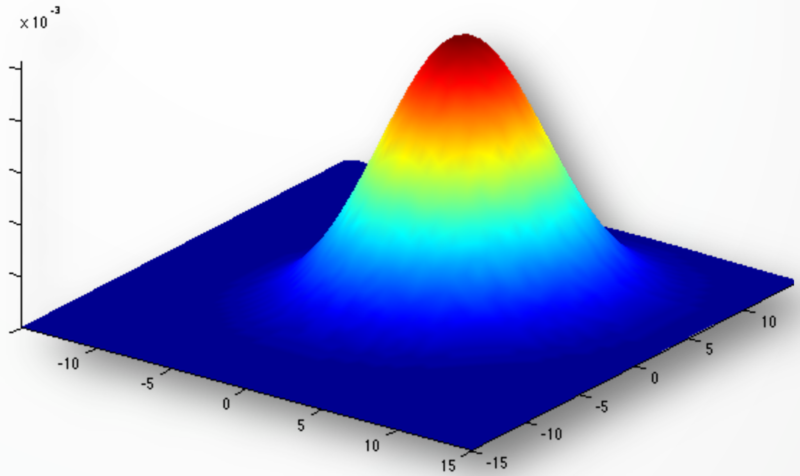
$$\vec{\mu} = E(\vec{x}) = \langle \mu[1], \mu[2], \ldots, \mu[d] \rangle$$

the **covariance matrix** is

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

and

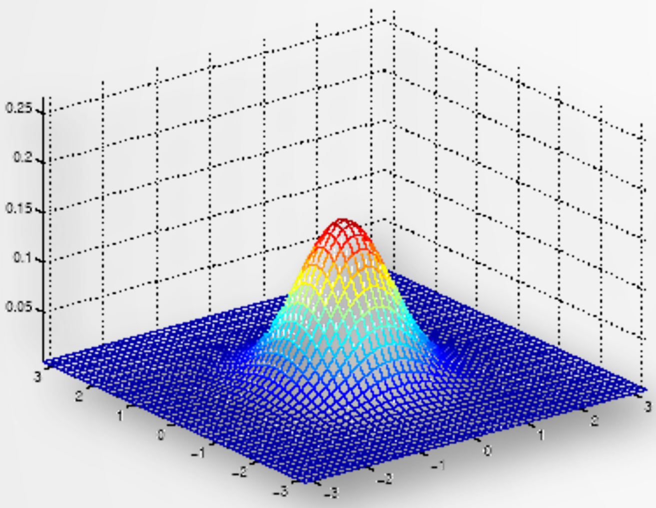
$$P(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^\intercal \Sigma^{-1} (\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{\frac{d}{2}} \, |\Sigma|^{\frac{1}{2}}}$$



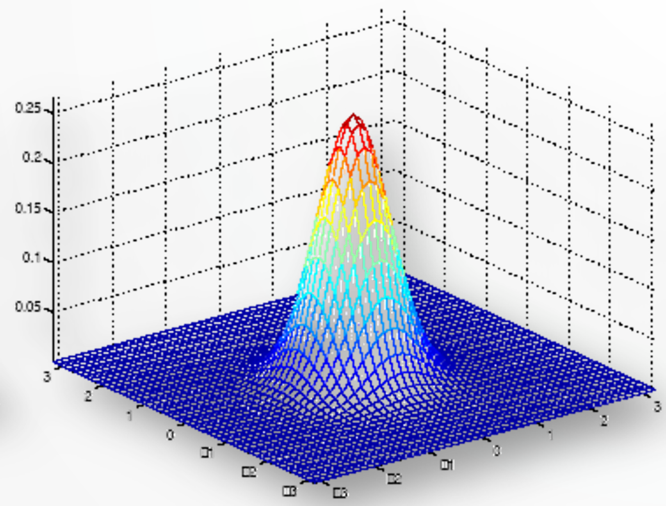
$A^\intercal$ is the **transpose** of A
A^{-1} is the **inverse** of A
$|A|$ is the **determinant** of A

UNIVERSITY OF TORONTO

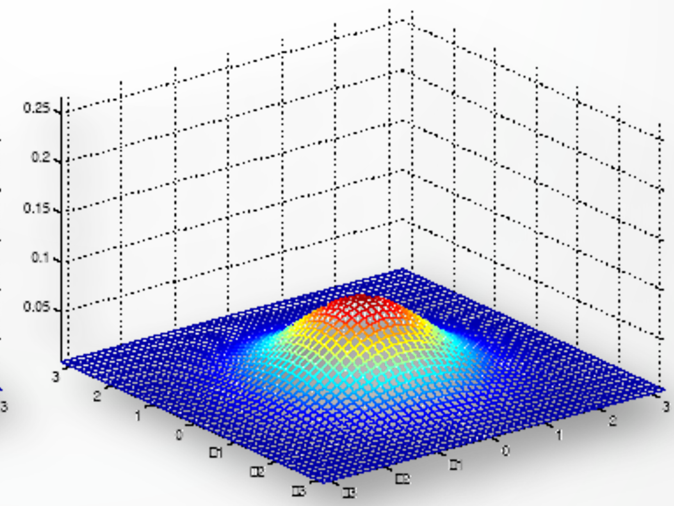
Intuitions of covariance



$$\mu = [0\ 0]$$
$$\Sigma = \mathbf{I}$$



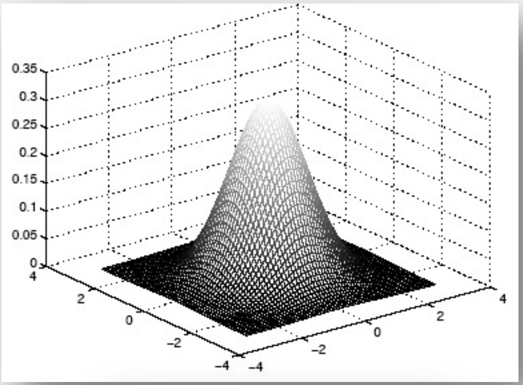
$$\mu = [0\ 0]$$
$$\Sigma = 0.6\mathbf{I}$$



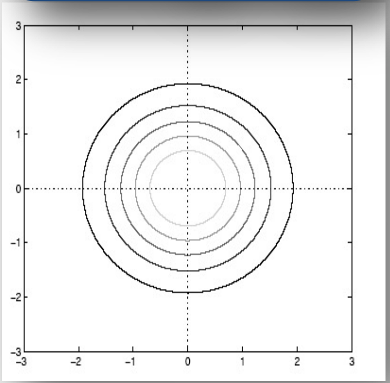
$$\mu = [0\ 0]$$
$$\Sigma = 2.0\mathbf{I}$$

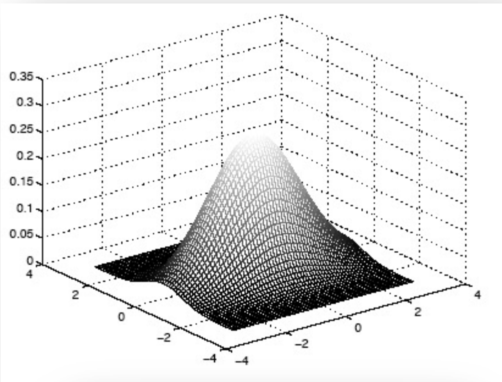
- As values in Σ become larger, the Gaussian spreads out.
- ($\mathbf{I}$ is the identity matrix)

Intuitions of covariance

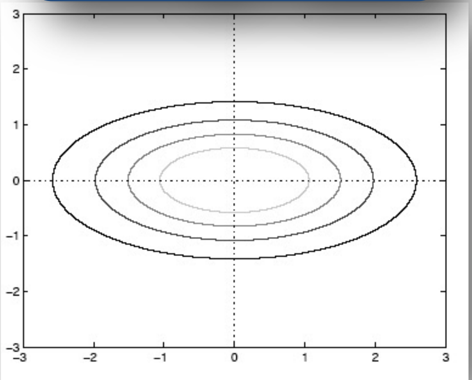


$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





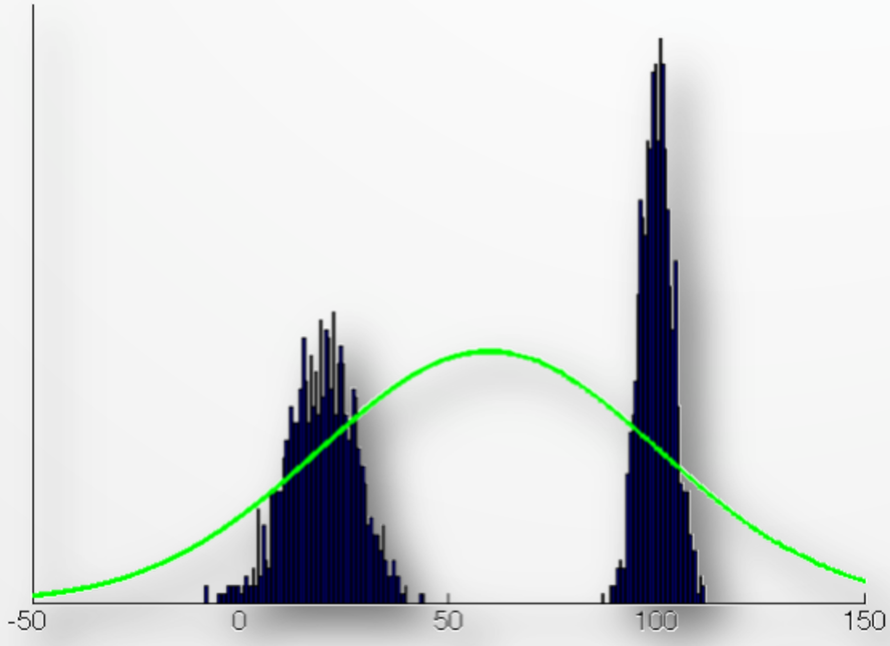
$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 0.6 \end{bmatrix}$$



- Different values on the diagonal result in different variances in their respective dimensions

Non-Gaussian observations

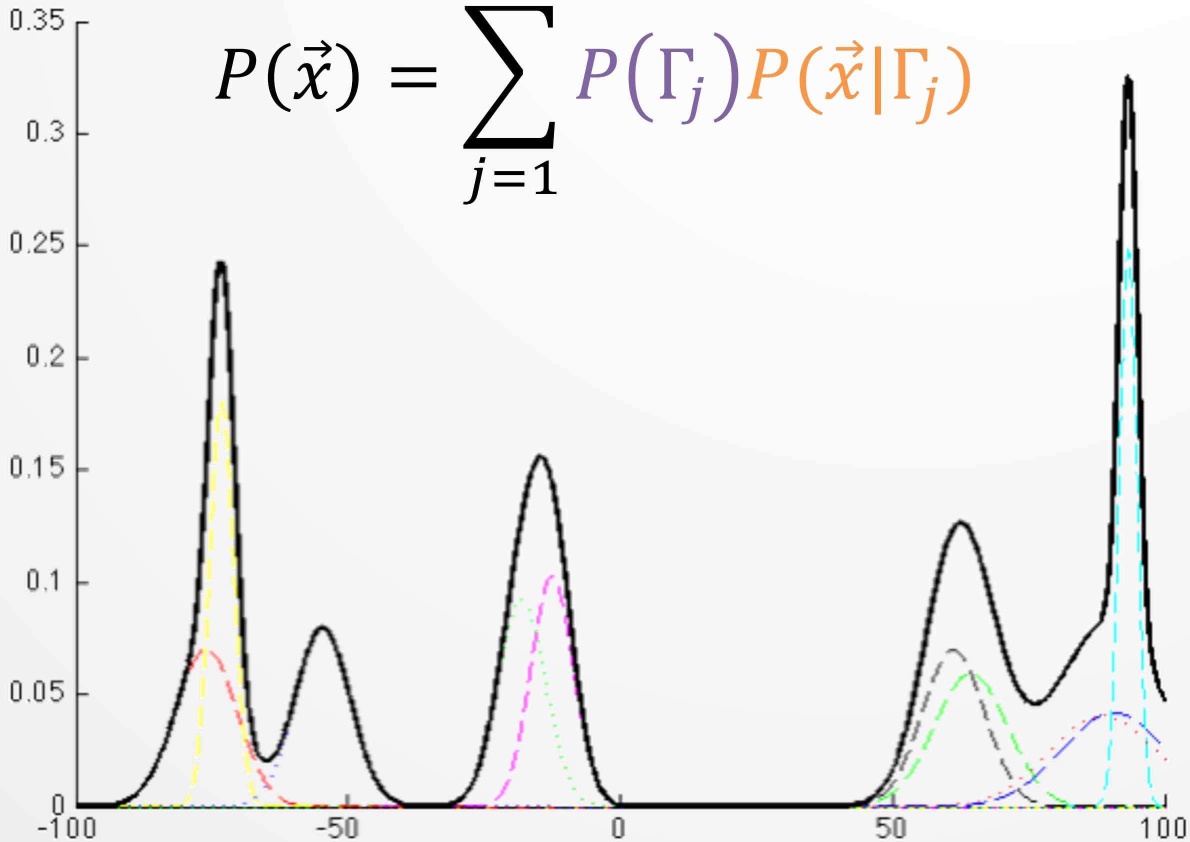
- Speech data are generally *not* unimodal.
- The observations below are **bimodal**, so fitting one Gaussian would not be representative.



Mixtures of Gaussians

- **Gaussian mixture models (GMMs)** are a weighted linear combination of M component Gaussians, $\langle \Gamma_1, \Gamma_2, \ldots, \Gamma_M \rangle$:

$$P(\vec{x}) = \sum_{j=1}^{M} P(\Gamma_j) P(\vec{x} | \Gamma_j)$$



UNIVERSITY OF TORONTO

Observation likelihoods

- Assuming MFCC dimensions are independent of one another, the **covariance matrix is diagonal** – i.e., 0 off the diagonal.
- Therefore, the probability of an observation vector given a Gaussian becomes

$$P(\vec{x}|\Gamma_m) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x[i]-\mu_m[i])^2}{\Sigma_m[i]}\right)}{(2\pi)^{\frac{d}{2}}\left(\prod_{i=1}^{d}\Sigma_m[i]\right)^{\frac{1}{2}}}$$

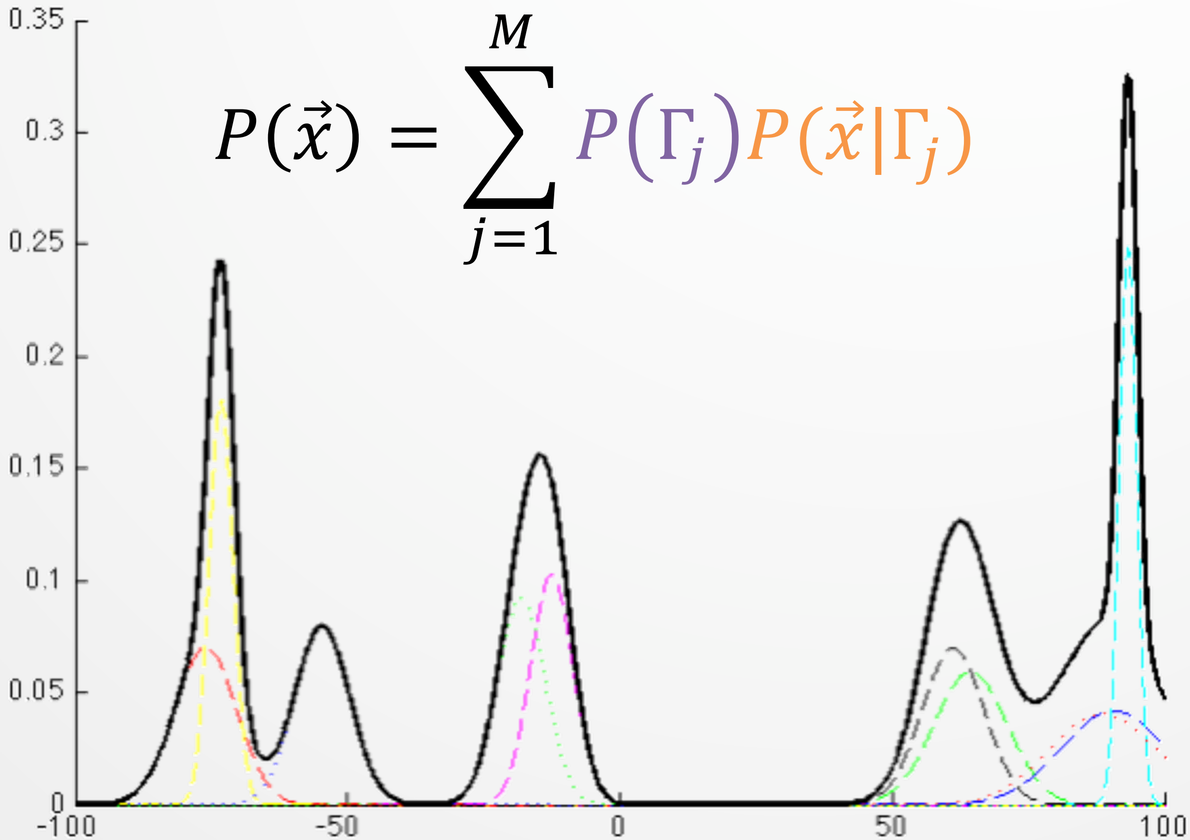
- *Imagine* that a GMM first *chooses a Gaussian*, then *emits an observation* from that Gaussian.

UNIVERSITY OF TORONTO

Mixtures of Gaussians

- If we knew *which* Gaussian generated each sample (which we don't), then $\langle \overrightarrow{\boldsymbol{\mu}_m}, \boldsymbol{\Sigma}_m \rangle$ can be learned by MLE.
- We must learn $P(\Gamma_j)$ as well.

$$P(\vec{x}) = \sum_{j=1}^{M} P(\Gamma_j) P(\vec{x}|\Gamma_j)$$



Expectation-Maximization for GMMs

- Overall idea:
  - First, initialize a set of model parameters.
  - “Expectation”: Compute the expected probabilities of observation, given these parameters.
  - “Maximization”: Update the parameters to maximize the aforementioned probabilities.
  - Repeat.
- Let’s look at the detailed steps in the next a few slides…

Expectation-Maximization for GMMs

- Let $\boldsymbol{\omega_m} = P(\Gamma_m)$ and $\boldsymbol{b_m}(\overrightarrow{\boldsymbol{x_t}}) = P(\overrightarrow{x_t}|\Gamma_m)$,

'weight'

'component observation likelihood'

$$P_\theta(\overrightarrow{x_t}) = \sum_{m=1}^{M} \omega_m b_m(\overrightarrow{x_t})$$

where $\boldsymbol{\theta} = \langle \boldsymbol{\omega_m}, \overrightarrow{\boldsymbol{\mu_m}}, \boldsymbol{\Sigma_m} \rangle$ for $m = 1..M$

- To estimate θ, we solve $\nabla_\theta \log L(X, \theta) = 0$ where

$$\log L(X, \theta) = \sum_{t=1}^{T} \log P_\theta(\overrightarrow{x_t}) = \sum_{t=1}^{T} \log \sum_{m=1}^{M} \omega_m b_m(\overrightarrow{x_t})$$

Expectation-Maximization for GMMs

- We **differentiate** the log likelihood function w.r.t . $\mu_m[n]$ and set this to 0 to find the value of $\mu_m[n]$ at which the likelihood stops growing.

$$\frac{\delta \log L(X,\theta)}{\delta \mu_m[n]} = \sum_{t=1}^{T} \frac{1}{P_\theta(\overrightarrow{x_t})} \left[\frac{\delta}{\delta \mu_m[n]} \omega_m b_m(\overrightarrow{x_t})\right] = 0$$

UNIVERSITY OF TORONTO

Expectation-Maximization for GMMs

- The **expectation step** gives us:

$$b_m(\overrightarrow{x_t}) = P(\overrightarrow{x_t}|\Gamma_m)$$

$$P(\Gamma_m|\overrightarrow{x_t};\theta) = \frac{\omega_m b_m(\overrightarrow{x_t})}{P_\theta(\overrightarrow{x_t})}$$

Proportion of overall probability contributed by m

- The **maximization step** gives us:

$$\widehat{\overrightarrow{\mu_m}} = \frac{\sum_t P(\Gamma_m|\overrightarrow{x_t};\theta)\overrightarrow{x_t}}{\sum_t P(\Gamma_m|\overrightarrow{x_t};\theta)}$$

$$\widehat{\Sigma_m} = \frac{\sum_t P(\Gamma_m|\overrightarrow{x_t};\theta)\overrightarrow{x_t}^2}{\sum_t P(\Gamma_m|\overrightarrow{x_t};\theta)} - \widehat{\overrightarrow{\mu_m}}^2$$

$$\widehat{\omega_m} = \frac{1}{T}\sum_{t=1}^{T} P(\Gamma_m|\overrightarrow{x_t};\theta)$$

Recall from slide 13, MLE wants:

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

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

UNIVERSITY OF TORONTO

Some notes…

- In the previous slide, the square of a vector, $\vec{a}^2$, is elementwise (i.e., `numpy.multiply`)
  - E.g., $[2, 3, 4]^2 = [4, 9, 16]$

- Since Σ is diagonal, it can be represented as a vector.

- Can $\widehat{\overrightarrow{\sigma_m^2}} = \frac{\sum_t P(\Gamma_m | \overrightarrow{x_t}; \theta) \overrightarrow{x_t}^2}{\sum_t P(\Gamma_m | \overrightarrow{x_t}; \theta)} - \widehat{\overrightarrow{\mu_m}}^2$ become negative?
  - No.
  - This is left as an exercise, but only if you're interested.

Speaker recognition

- **Speaker recognition**: *n*. the identification of a speaker among several speakers given only acoustics.

- Each **speaker** will produce speech according to **different** probability distributions.
  - We train **a Gaussian mixture model for each speaker**, given annotated data (mapping utterances to speakers).
  - We choose the speaker whose model gives the highest probability for an observation.

Recipe for GMM EM

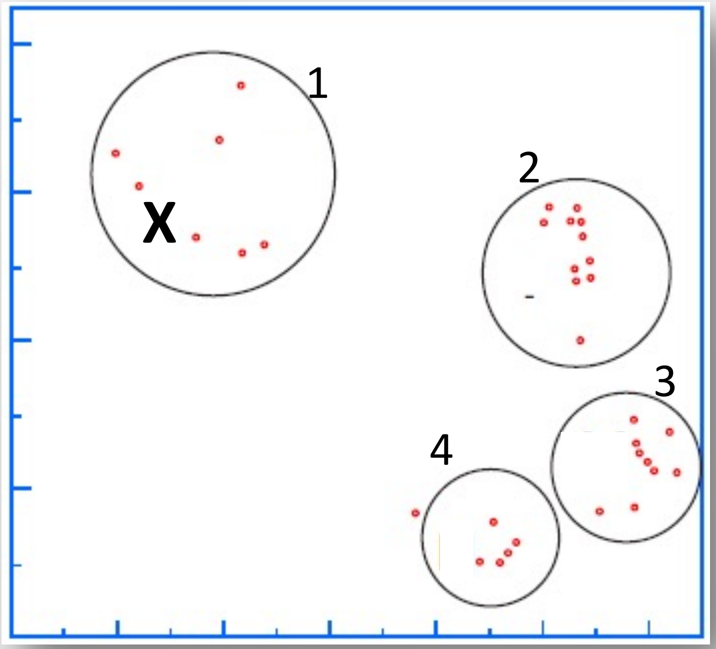
- For each speaker, we learn a GMM given all T frames of their training data.

1. Initialize:	Guess $\theta = \langle \omega_m, \overrightarrow{\mu_m}, \Sigma_m \rangle$ for $m = 1..M$ either uniformly, randomly, or by *k*-means clustering.
2. E-step:	Compute $b_m(\overrightarrow{x_t})$ and $P(\Gamma_m \mid \overrightarrow{x_t}; \theta)$.
3. M-step:	Update parameters for $\langle \omega_m, \overrightarrow{\mu_m}, \Sigma_m \rangle$ as described on slide 21.

CLUSTERING

Clustering

- Clustering is **unsupervised** learning.
  - Number and form of clusters often unknown.



- Observation X is in Cluster One, so we replace it with 1.

Aspects of clustering

- What **defines** a particular cluster?
  - Is there some **prototype** representing each cluster?

- What defines **membership** in a cluster?
  - Usually, some distance metric $d(x, y)$ (e.g., Euclidean distance).

- How well do clusters represent **unseen data**?
  - How is a new point assigned to a cluster?
  - How do we modify that cluster as a result?

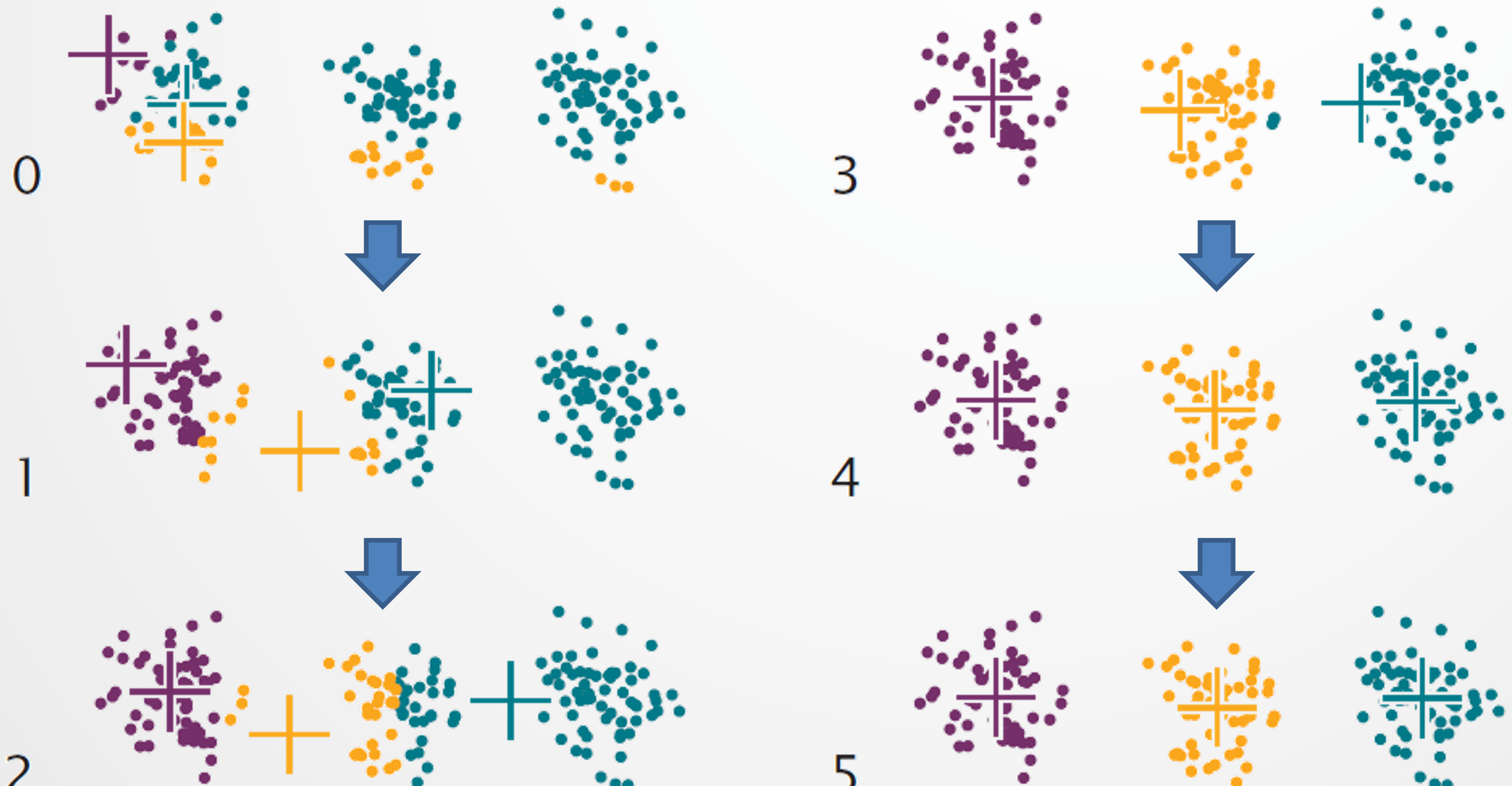
K-means clustering

- Used to group data into K clusters, $\{C_1, \dots, C_K\}$.
- Each cluster is represented by the **mean** of its assigned data.
  - (sometimes it's called the cluster's ***centroid***).
- Iterative algorithm converges to **local** optimum:
  1. **Select** K initial cluster means $\{\mu_1, \dots, \mu_K\}$ from among data points.
  2. **Until** (stopping criterion),
     a) **Assign** each data sample to closest cluster
     $$x \in C_i \quad if \quad d(x, \mu_i) \le d(x, \mu_j), \quad \forall i \neq j$$
     b) **Update** K means from assigned samples
     $$\mu_i = E(x) \, \forall \, x \in C_i, \qquad 1 \le i \le K$$

UNIVERSITY OF TORONTO

K-means example ($K = 3$)

- Initialize with a random selection of 3 data samples.
- Euclidean distance metric $d(x, \mu)$



UNIVERSITY OF TORONTO

K-means stopping condition

- The total **distortion**, $\mathcal{D}$, is the sum of squared error,

$$\mathcal{D} = \sum_{i=1}^{K} \sum_{x \in C_i} \|x - \mu_i\|^2$$

- $\mathcal{D}$ decreases between n^{th} and $(n+1)^{th}$ iteration.

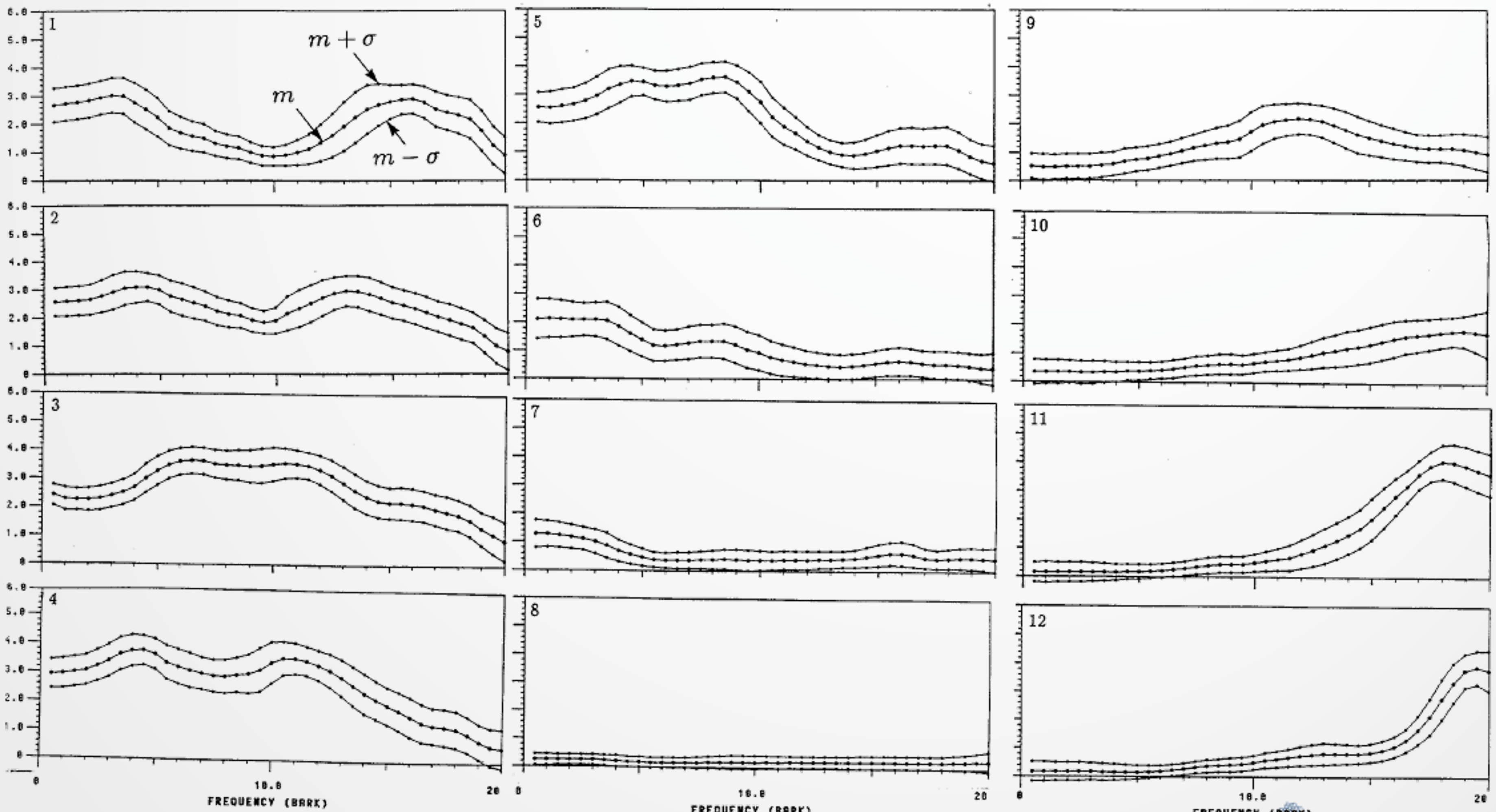
- We can stop training when $\mathcal{D}$ falls below some threshold $\mathcal{T}$.

$$1 - \frac{\mathcal{D}(n+1)}{\mathcal{D}(n)} < \mathcal{T}$$

UNIVERSITY OF TORONTO

Acoustic clustering example

- 12 clusters of spectra, after training.



UNIVERSITY OF TORONTO

Number of clusters

- The number of true clusters is unknown.
- We can iterate through various values of K.
  - As K approaches the size of the data, $\mathcal{D}$ approaches 0…

$K = 2$

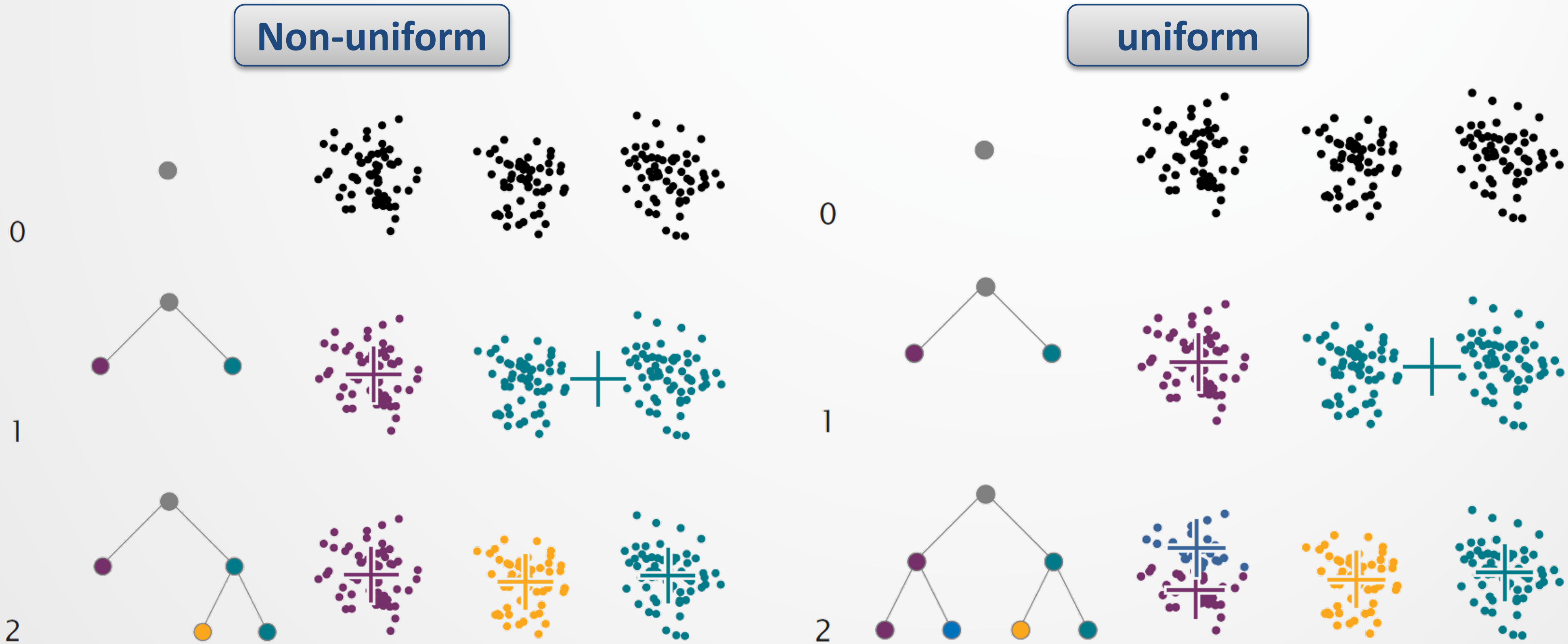
$K = 4$

Hierarchical clustering

- **Hierarchical clustering** clusters data into hierarchical 'class' structures.
- Two types: top-down (**divisive**) or bottom-up (**agglomerative**).
- Often based on greedy formulations.
- Hierarchical structure can be used for hypothesizing classes.

Divisive clustering

- Creates hierarchy by successively splitting clusters into smaller groups.



UNIVERSITY OF TORONTO

Agglomerative clustering

- **Agglomerative clustering** starts with N 'seed' clusters and iteratively combines these into a hierarchy.

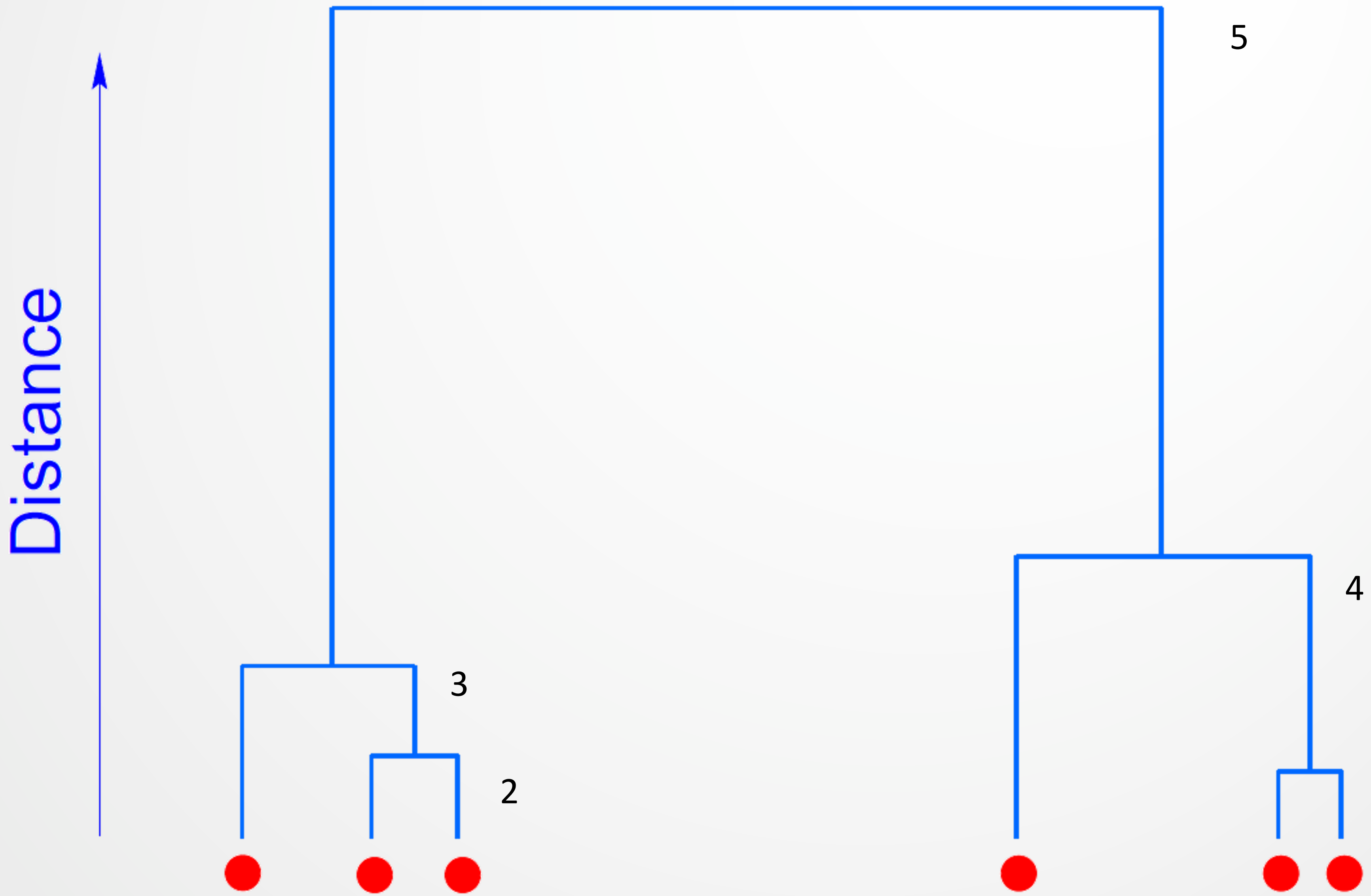
- On each iteration, the **two most similar** clusters are **merged** together to form a new **meta-cluster**.

- After $N - 1$ iterations, the hierarchy is complete.

- Often, when the similarity scores of new meta-clusters are tracked, the resulting graph (i.e., **dendrogram**) can yield insight into the natural grouping of data.

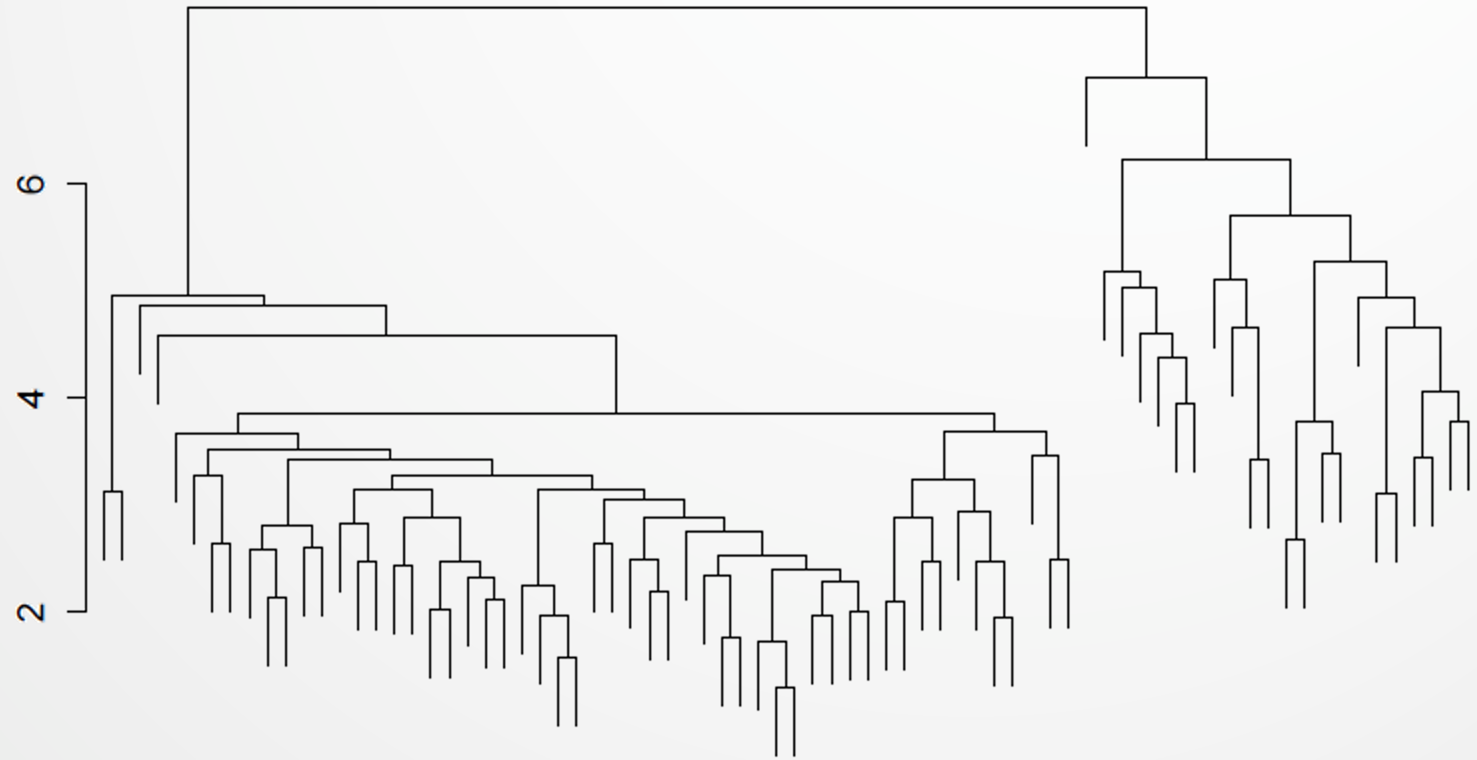
UNIVERSITY OF TORONTO

Dendrogram example



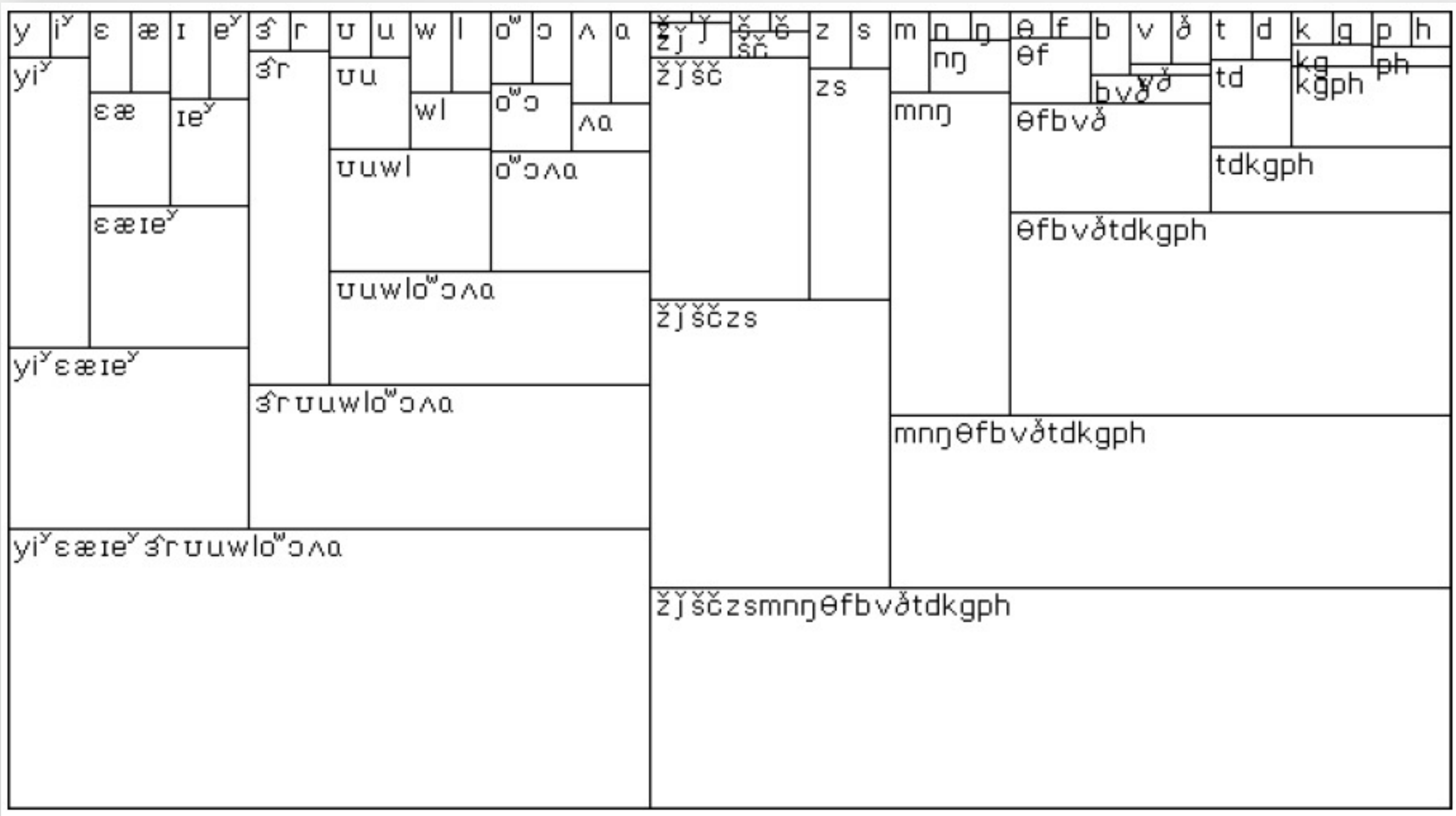
Speaker clustering

- 23 female and 53 male speakers from TIMIT.
- Data are vectors of average F1 and F2 for 9 vowels.
- Distance $d(C_i, C_j)$ is average of distances between members.



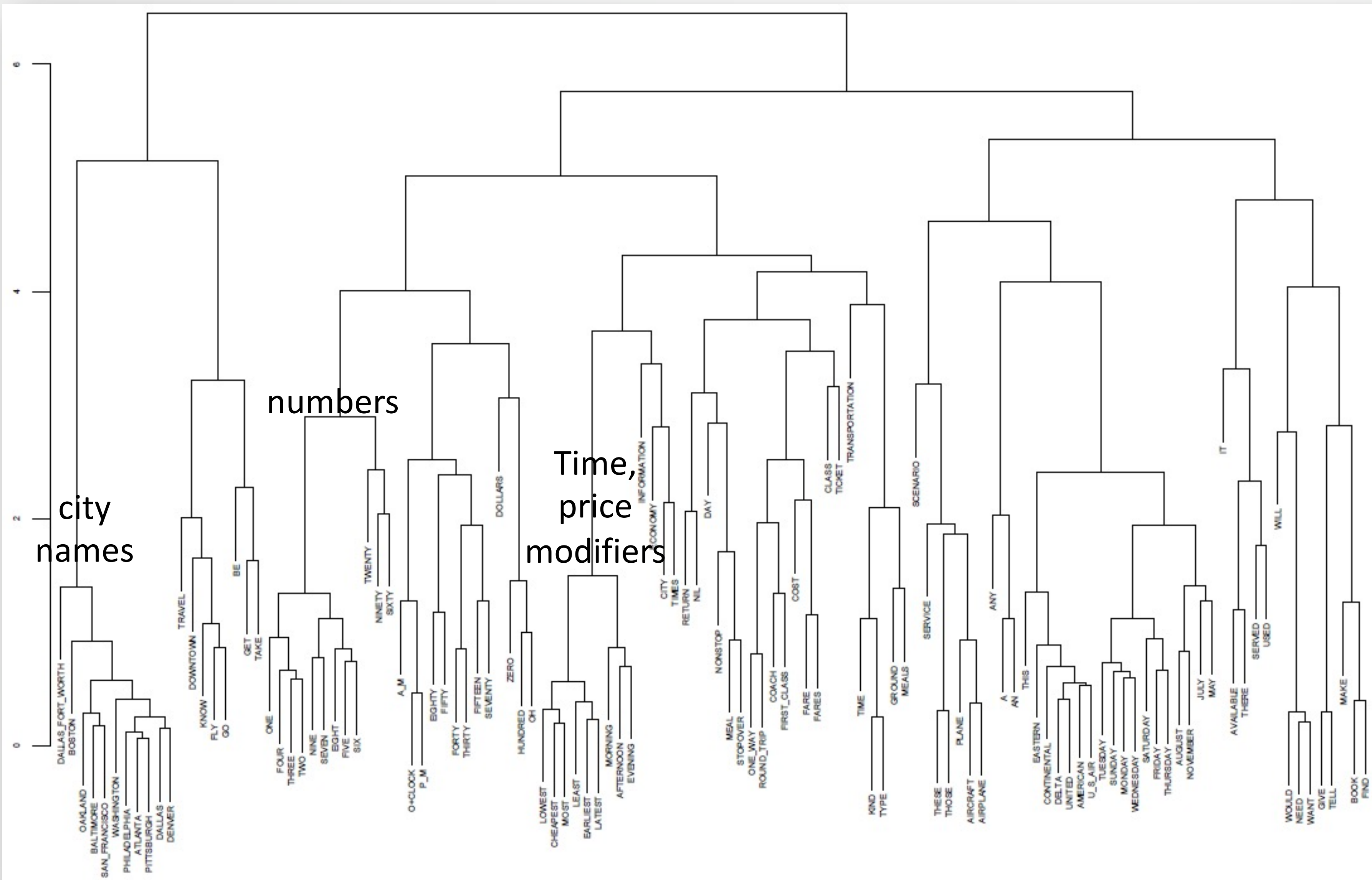
UNIVERSITY OF TORONTO

Acoustic-phonetic hierarchy



(this is basically an upside-down dendogram)

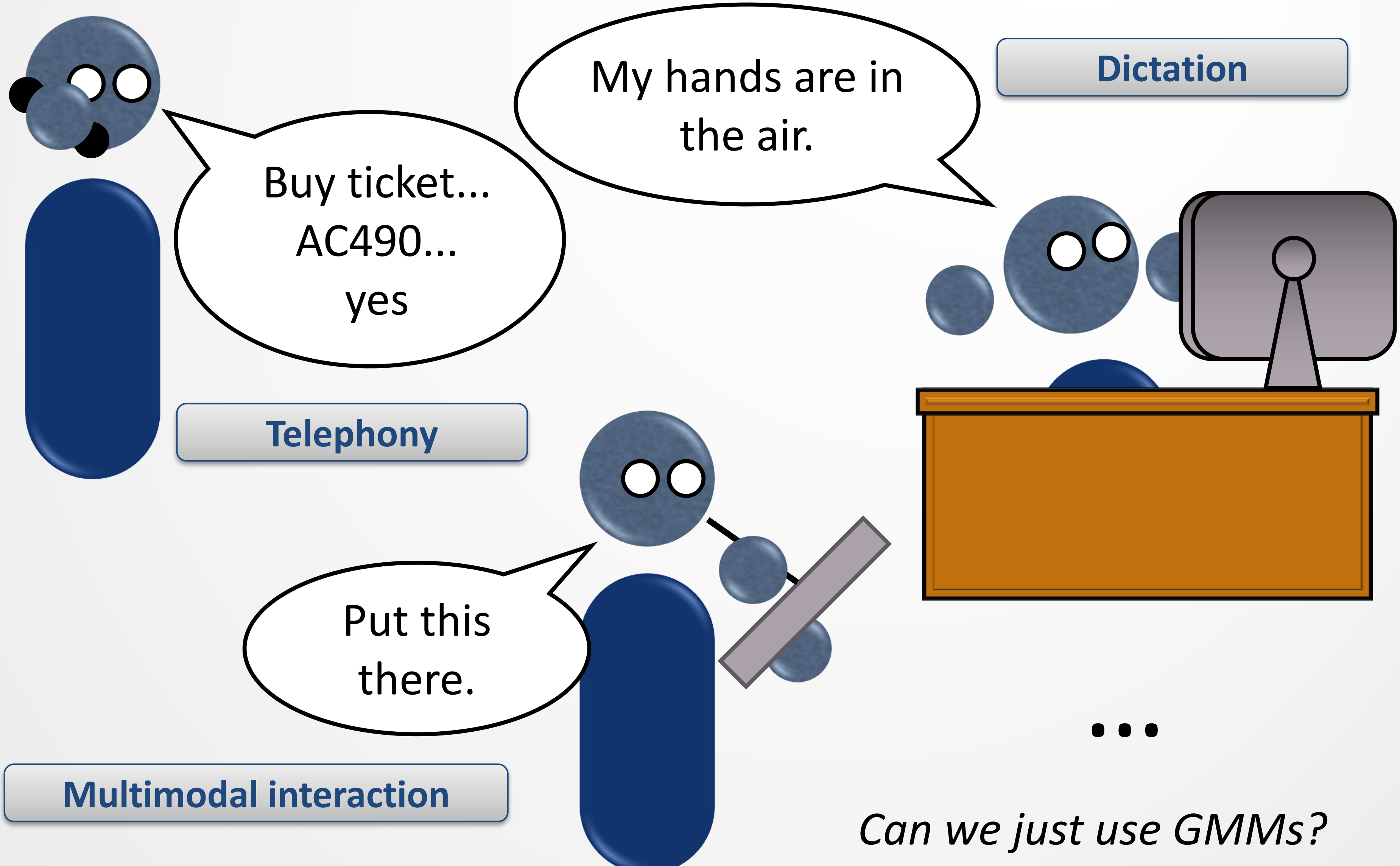
Word clustering



UNIVERSITY OF TORONTO

SPEECH RECOGNITION

Consider what we want speech to do



…

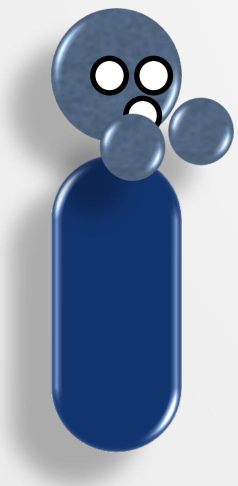
Can we just use GMMs?

Aspects of ASR systems in the world

- **Speaking mode**: **Isolated** word (e.g., "*yes*") vs. **continuous** (e.g., *"Hey Siri, ask Cortana for the weather")*
- **Speaking style**: **Read** speech vs. **spontaneous** speech; the latter contains many **dysfluencies** (e.g., stuttering, *uh*, *like*, …)
- **Enrolment**: **Speaker-dependent** (all training data from one speaker) vs. **speaker-independent** (training data from many speakers).
- **Vocabulary**: **Small** (<20 words) or **large** (>50,000 words).
- **Transducer**: Cell phone? Noise-cancelling microphone? Teleconference microphone?

Recognizing ~~speakers~~ phones

- A first idea: since GMM can be used to recognize speakers, it can be used to recognize phonemes.
  - For each frame (15~25ms), GMM classifies a phone.
  - Then we can look up a phone dictionary to assemble into words!



/ow ow ow ow ow p p p p ah ah ah ah ah n/

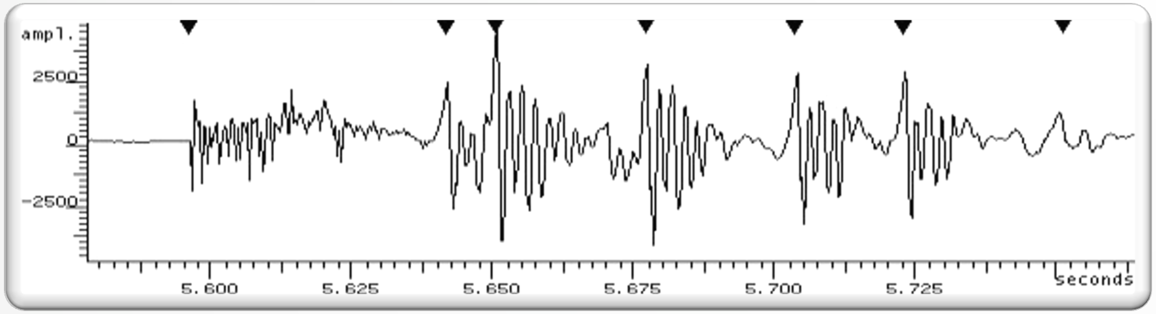
↓

/ow p ah n /

↓

Open

Speech is dynamic

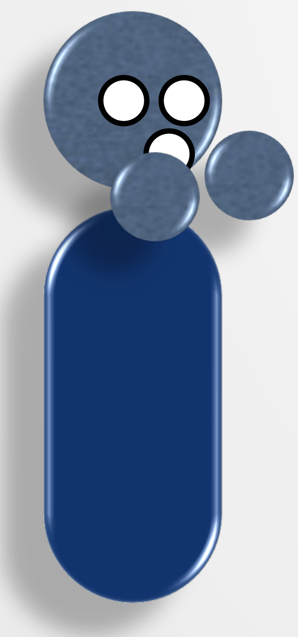


- Speech **changes** over time.
  - GMMs are good for high-level clustering, but they encode **no notion** of *order*, *sequence*, nor *time*.

- Speech is an expression of **language**.
  - We want to incorporate knowledge of how phonemes and words are ordered with **language models**.

UNIVERSITY OF TORONTO

Speech is sequences of phonemes (*)



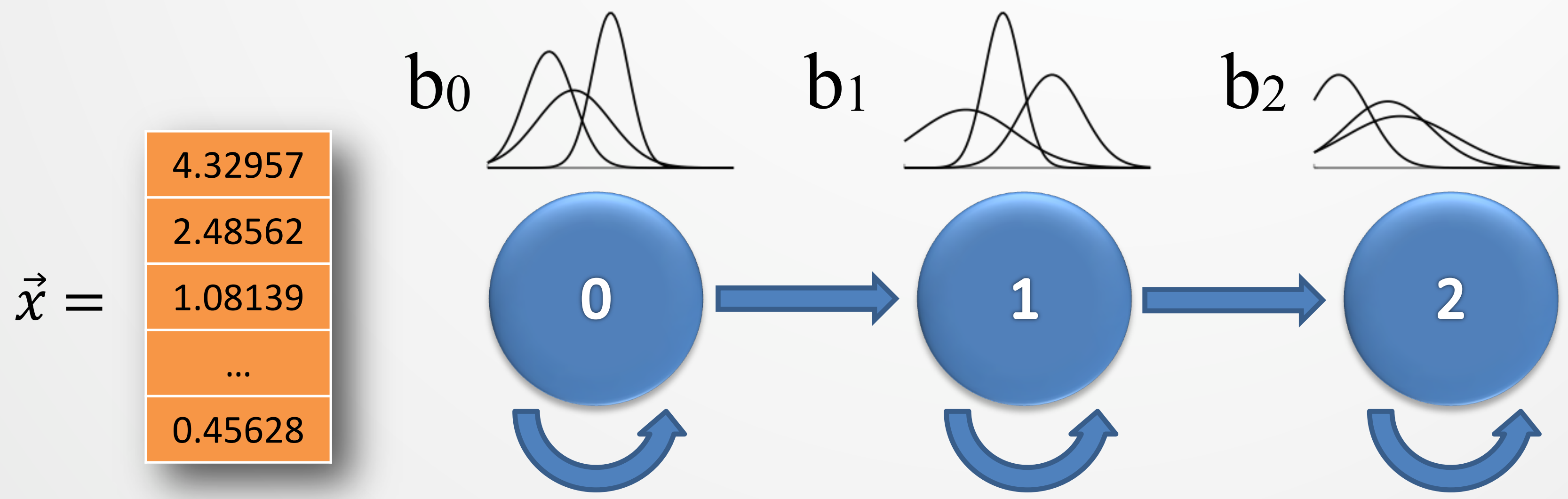
We want to convert a series of (e.g.) MFCC vectors into a sequence of phonemes.

(*) not really

UNIVERSITY OF TORONTO

Continuous HMMs (CHMM)

- A **continuous HMM** has observations that are distributed over continuous variables.
  - Observation probabilities, b_i, are also continuous.
  - E.g., here $b_0(\vec{x})$ tells us the probability of seeing the (multivariate) continuous observation $\vec{x}$ while in state 0.

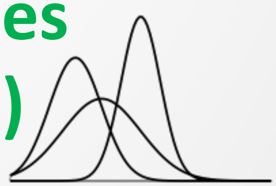


UNIVERSITY OF TORONTO

GMM-HMM as Continuous HMM

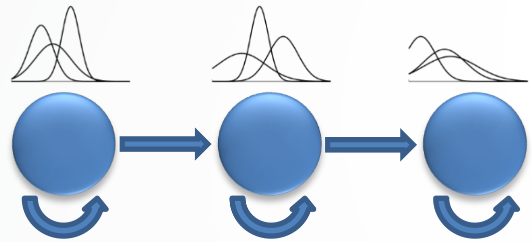
- Continuous HMMs are very similar to discrete HMMs.
  - $S = \{s_1, \ldots, s_N\}$: set of states (e.g., phonemes)
  - $X = \mathbb{R}^d$: **continuous observation space**

$$\theta \begin{cases} \Pi = \{\pi_1, \ldots, \pi_N\} & \text{: initial state probabilities} \\ A = \{a_{ij}\}, i, j \in S & \text{: state transition probabilities} \\ B = b_i(\vec{x}), i \in S, \vec{x} \in X & \text{: state output probabilities (i.e., Gaussian mixtures)} \end{cases}$$



yielding
  - $Q = \{q_0, \ldots, q_T\}, q_i \in S$: state sequence
  - $\mathcal{O} = \{o_0, \ldots, o_T\}, o_i \in X$: observation sequence

Using CHMMs



- As before, these HMMs are **generative** models that encode statistical knowledge of how output is **generated**.

- We **train** CHMMs with **Baum-Welch** (a type of Expectation-Maximization), as we did before with discrete HMMs.
  - Here, the observation parameters, $b_i(\vec{x})$, are adjusted using the GMM training 'recipe' from earlier.

- We find the best state sequences using **Viterbi**, as before.
  - Here, the best state sequence gives us a **sequence of phonemes**.

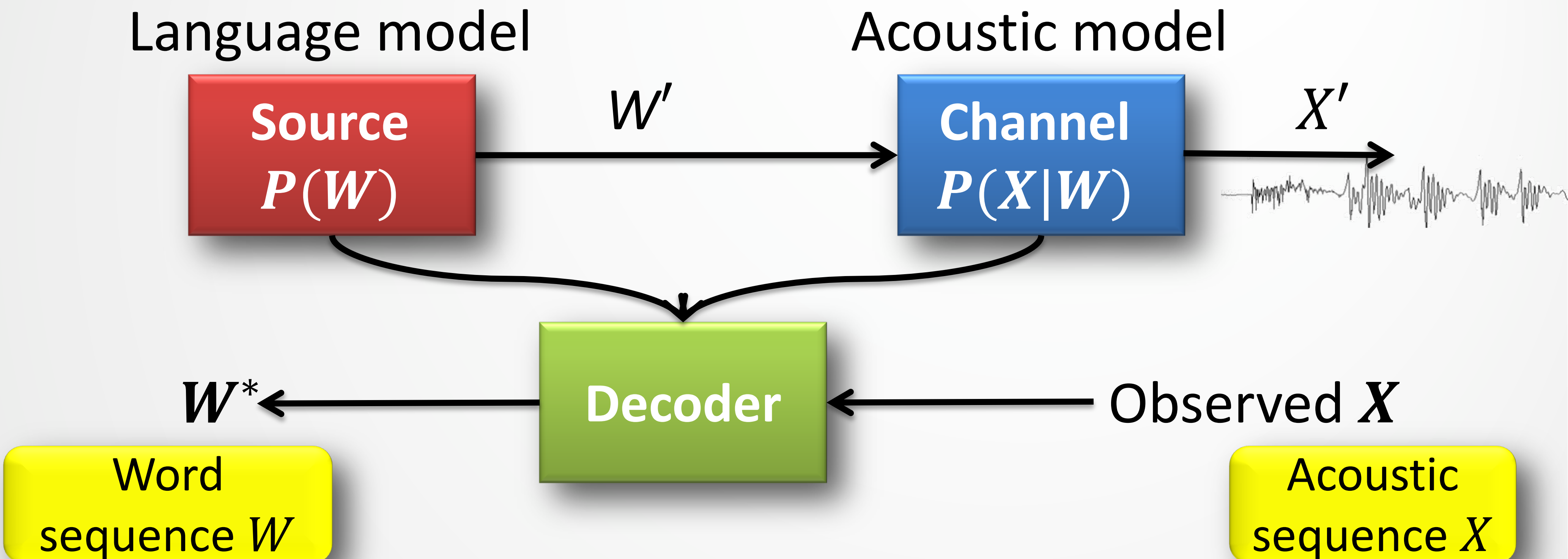
Phoneme dictionaries

- How do we convert our phoneme sequence into **words**?
- There are many **phonemic dictionaries** that map words to pronunciations (i.e., lists of phoneme sequences).
- The **CMU dictionary** (http://www.speech.cs.cmu.edu/cgi-bin/cmudict) is popular.
  - 127K words transcribed with the ARPAbet.
  - Includes some rudimentary **prosody markers**.

```
…
EVOLUTION          EH2 V AH0 L UW1 SH AH0 N
EVOLUTION(2)       IY2 V AH0 L UW1 SH AH0 N
EVOLUTION(3)       EH2 V OW0 L UW1 SH AH0 N
EVOLUTION(4)       IY2 V OW0 L UW1 SH AH0 N
EVOLUTIONARY       EH2 V AH0 L UW1 SH AH0 N EH2 R IY0
```


UNIVERSITY OF TORONTO

A noisy channel model for ASR



$$W^* = \underset{W}{\operatorname{argmax}} P(W|X) = \boxed{\underset{W}{\operatorname{argmax}} P(X|W)P(W)}$$

An HMM-based ASR system

$$W^* = \underset{W}{\operatorname{argmax}}\, P(W|X) = \underset{W}{\operatorname{argmax}}\, P(X|W)P(W)$$

- Use a CHMM to compute $P(X|W)$
  - This CHMM is trained on a phonemic corpus $\{\mathrm{W_d}, X_d\}$
- Use a language model to compute $P(W)$
  - The parameters of the language model is trained on a corpora $\{W_c\}$
- Use the CHMM and the LM jointly to find W^*.

Remember Viterbi

$\mathbf{0}$
$\mathbf{0}$
$\mathbf{0.06}$
$\mathbf{0}$
$\mathbf{0.08}$
$\mathbf{0}$

The best path to state s_j at time t, $\delta_j(t)$, depends on the best path to each possible previous state, $\delta_i(t-1)$, and their transitions to j, a_{ij}

$$\boldsymbol{\delta_j(t) = \max_i \left[\delta_i(t-1)a_{ij}\right. \boxed{b_j(\sigma_t)}}$$

$$\boldsymbol{\psi_j(t) = \operatorname*{argmax}_i \left[\delta_i(t-1)a_{ij}\right]}$$

Do these probabilities need to be GMMs?

$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

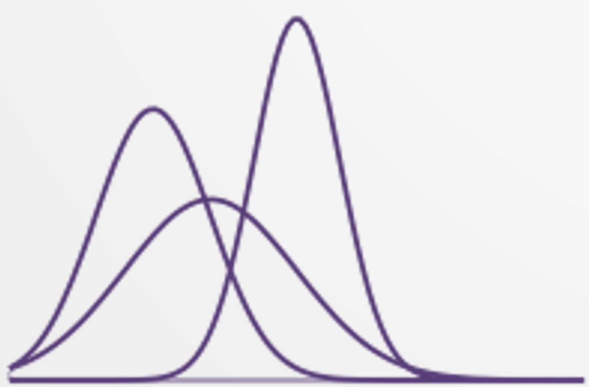
Observations, σ_t

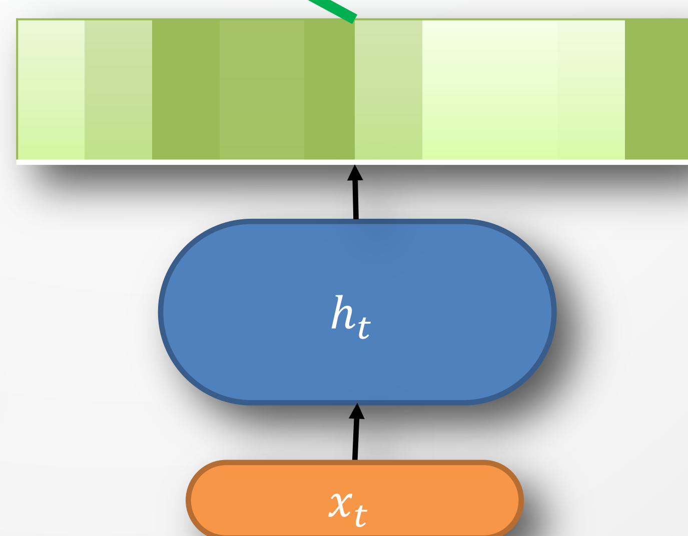
UNIVERSITY OF TORONTO

Replacing GMMs with DNNs

- Obtain $b_j(x) = \boldsymbol{p(x|s_j)}$ with a neural network.
- Instead of learning a continuous distribution directly, we can use Bayes' rule:

$$\boldsymbol{p(x|s_j)} = \frac{\boldsymbol{p(s_j|x)} \cdot p(x)}{p(s_j)}$$





UNIVERSITY OF TORONTO

Replacing GMMs with DNNs

- The probability of a word sequence W comes *loosely* from $P(X|W)$
- Considering the states $q_1, \ldots, q_T$ during the frames of a word:

$$\mathrm{P(X|W)} \approx \max_{q_1 \cdots q_T} \prod_{t=1}^{T} P(q_t|q_{t-1})\, P(x_t|q_t) \approx \max_{q_1 \cdots q_T} \prod_{t=1}^{T} P(q_t|q_{t-1}) \frac{P(q_t|x_t)}{P(q_t)}$$

HMM handles the temporal dependency

GMM

DNN

Training the DNN

- Maximize $P(q_t|x_t)$
- The order which we transition through states ($\approx$ phonemes) is known by the transcription (ignoring alternate pronunciations)
- At what frames these transitions happen are **unknown**
  - $\therefore q_t$ is unknown!
- One solution: bootstrapping
  - Use another model (e.g., GMM-HMM) to determine q_t
  - The acquired q_t is used to train the DNN.
- Other, advanced methods exist.

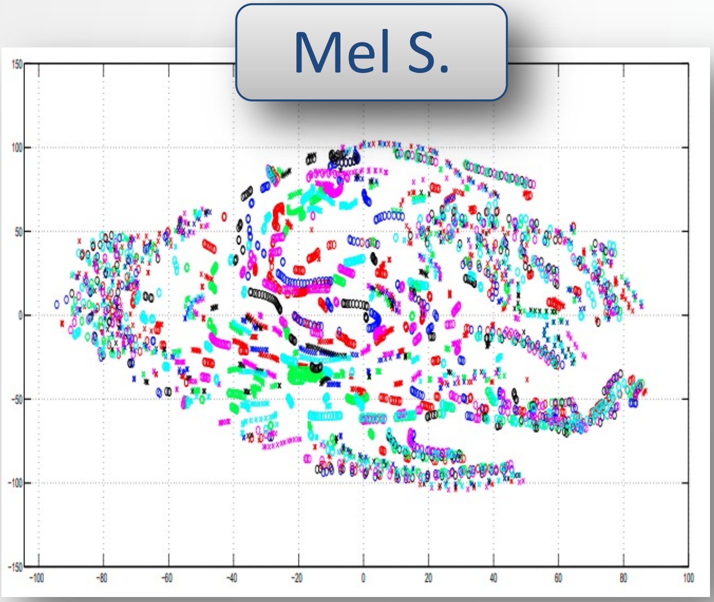
Hybrid DNN-HMM Systems

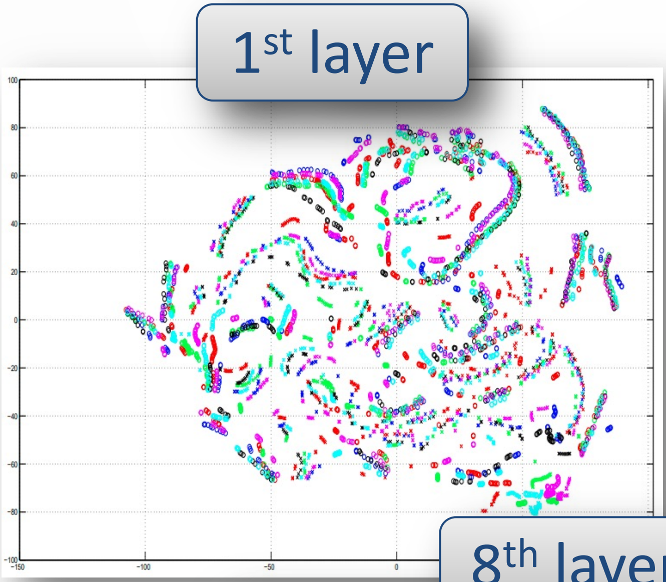
[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

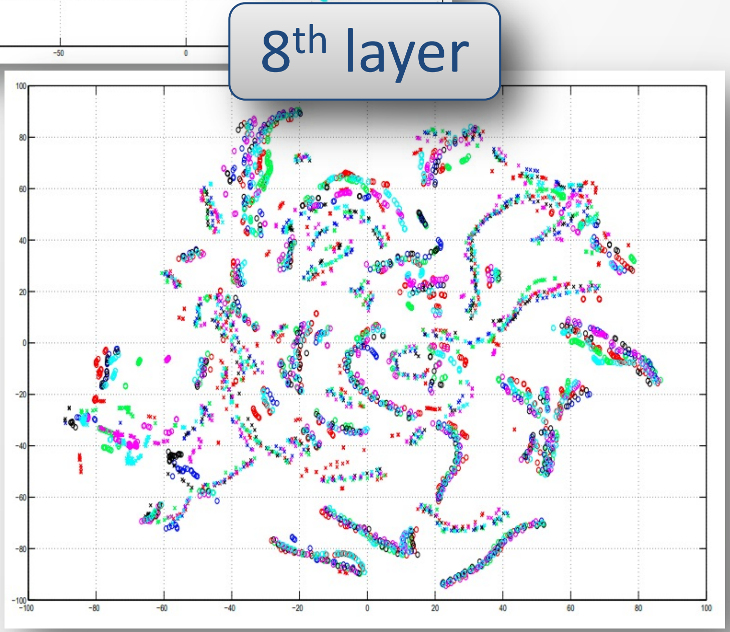
TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

G Hinton et al (Nov 2012). "Deep neural networks for acoustic modeling in speech recognition", IEEE Signal Processing Magazine, **29**(6):82–97. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6296526

Aside: What did these DNNs learn?





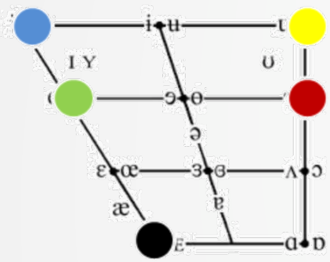


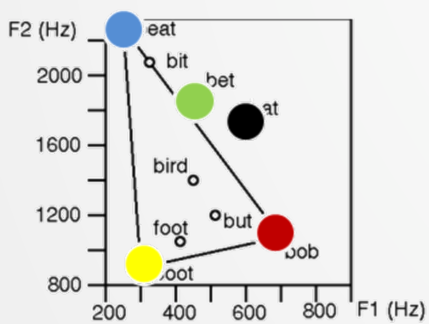
- t-SNE (stochastic neighbour embedding using t-distribution) visualizations in 2D (colours=speakers).
- Deeper layers encode information about the **segment**

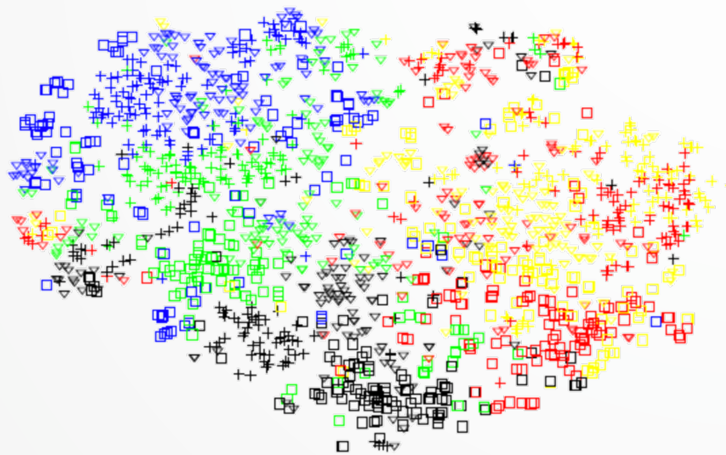
Mohamed, A., Hinton, G., & Penn, G. (2012). Understanding how deep belief networks perform acoustic modelling. In *ICASSP* (pp. 6–9).

UNIVERSITY OF TORONTO

Aside: What did these DNNs learn?







- DNN trained to classify phonemes
- t-SNE visualizations of hidden layer.
- Lower layers detect **manner** of articulation

Figure 1: Multilingual BN features of five vowels from French (+), German (□) and Spanish (▽): /a/ (black), /i/ (blue), /e/ (green), /o/ (red), and /u/ (yellow)

Vu, N. T., Weiner, J., & Schultz, T. (2014). Investigating the learning effect of multilingual bottle-neck features for ASR. *Interspeech*, 825–829.

End-to-end neural networks

- Neural networks are typically trained at the frame level.
  - This requires a separate training target for every frame, which in turn *requires the alignment between the audio and transcription sequences to be known*.
  - However, the alignment is only reliable once the classifier is trained.
- "End-to-end" ≈ an objective function that allows sequence transcription *without* requiring prior alignment between the **input** X (frames of audio) and **target** Y (output strings) sequences with arbitrary lengths, i.e.

$$P(Y|X)$$

- Target tokens can be words, sub-words, or just characters
- Two popular choices of $P(Y|X)$:
  1. Seq2seq (encoder/decoder, transformers)
  2. Connectionist Temporal Classification

UNIVERSITY OF TORONTO

End-to-end architectures for ASR

- The **same architectures** we saw in NMT work for ASR!
- Replace source embedding vector x_t with Mel spectrum vector
- Replace target sequence E with transcription sequence Y

…

And train with Connectionist Temporal Classification (in aside – please refer to https://distill.pub/2017/ctc/.

End-to-end architectures for ASR

Table 1. **Wall Street Journal Results.** All scores are word error rate/character error rate (where known) on the evaluation set. 'LM' is the Language model used for decoding. '14 Hr' and '81 Hr' refer to the amount of data used for training.

SYSTEM	LM	14 HR	81 HR
RNN-CTC	NONE	74.2/30.9	30.1/9.2
RNN-CTC	DICTIONARY	69.2/30.0	24.0/8.0
RNN-CTC	MONOGRAM	25.8	15.8
RNN-CTC	BIGRAM	15.5	10.4
RNN-CTC	TRIGRAM	13.5	8.7
RNN-WER	NONE	74.5/31.3	27.3/8.4
RNN-WER	DICTIONARY	69.7/31.0	21.9/7.3
RNN-WER	MONOGRAM	26.0	15.2
RNN-WER	BIGRAM	15.3	9.8
RNN-WER	TRIGRAM	13.5	8.2
BASELINE	NONE	—	—
BASELINE	DICTIONARY	56.1	51.1
BASELINE	MONOGRAM	23.4	19.9
BASELINE	BIGRAM	11.6	9.4
BASELINE	TRIGRAM	9.4	7.8
COMBINATION	TRIGRAM	—	6.7

DNN/HMM hybrid

Graves A, Jaitly N. (2014) Towards End-To-End Speech Recognition with Recurrent Neural Networks. JMLR Workshop Conf Proc, 32:1764–1772.

The open-source Kaldi ASR

- Kaldi is the *de-facto* open-source ASR toolkit: http://kaldi-asr.org
  - It has pretrained models, including the ASpIRE chain model trained on Fisher English, augmented with impulse responses and noises to create multi-condition training.
  - There are docker images for environments running Kaldi.
  - It often (anecdotally) performs better than Google's SpeechAPI.
  - It is originally in C++, but a wrapper (PyTorch-Kaldi) exists in the much easier Python.
  - Pro-sanity tip: don't read news about its progenitor.

UNIVERSITY OF TORONTO

EVALUATING SPEECH RECOGNITION

Evaluating ASR accuracy

- How can you tell how well an ASR system recognizes speech?
  - E.g., if somebody said

    Reference: *<u>how to</u> recognize speech*

    but an ASR system heard

    Hypothesis: *<u>how to</u> wreck a nice beach*

    how do we quantify the error?

- One measure is **word accuracy**: *#CorrectWords/#ReferenceWords*
  - E.g., $2/4$, above
  - This runs into problems similar to those we saw with SMT.
    - E.g., the hypothesis '*how to recognize speech boing boing boing boing boing*' has 100% accuracy by this measure.
    - Normalizing by *#HypothesisWords* also has problems…

UNIVERSITY OF TORONTO

Word-error rates (WER)

- ASR enthusiasts are often concerned with **word-error rate (WER)**, which counts different **kinds** of errors that can be made by ASR at the word-level.
  - **Substitution error**: One word being mistook for another e.g., '***shift***' given '*ship*'
  - **Deletion error**: An input word that is 'skipped' e.g. '*I Torgo*' given '*I* ***am*** *Torgo*'
  - **Insertion error**: A 'hallucinated' word that was not in the input. e.g., '*This* ***Norwegian*** *parrot is no more*' given '*This parrot is no more*'

UNIVERSITY OF TORONTO

Levenshtein distance

- The **Levenshtein** distance (and WER) is straightforward to calculate using dynamic programming

Allocate matrix $R[n+2, m+2]$ // where n is the number of reference words
// and m is the number of hypothesis words

Add <s> to beginning of each sequence, and </s> to their ends**.**

Fill [0:end] along the first row and column.

for $i := 1..n+1$ // #ReferenceWords

  for $j := 1..m+1$ // #Hypothesis words

    $R[i,j] := \min($ $R[i-1,j]+1,$ // **deletion**

      $R[i-1,j-1],$ // if the i^{th} reference word and
      // the j^{th} hypothesis word **match**

      $R[i-1,j-1]+1,$ // if they differ, i.e., **substitution**

      $R[i,j-1]+1\ )$ // **insertion**

Return $100\times R[n,m]/n$ // WER

Levenshtein distance – initialization

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1							
	to	2							
	recognize	3							
	speech	4							
	</s>	5							

The value at cell (i, j) is the **minimum** number of **errors** necessary to align i with j.

Levenshtein distance

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1	0						
	to	2							
	recognize	3							
	speech	4							
	</s>	5							

- $R[1,1] = \min(LEFT + 1, (0), ABOVE + 1) = 0$ (match)
- We put a little arrow in place to indicate the choice.
  - ‘Arrows’ are normally stored in a **backtrace matrix**.

Levenshtein distance

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1	0	1	2	3	4	5	6
	to	2							
	recognize	3							
	speech	4							
	</s>	5							

- We continue along for the first reference word…
  - These are all **insertion** errors

Levenshtein distance

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1	0	1	2	3	4	5	6
	to	2	1	0	1	2	3	4	5
	recognize	3	2	1	**1**	**2**	**3**	**4**	**5**
	speech	4							
	</s>	5							

- Since *recognize* ≠ *wreck*, we have a **substitution** error.
- At some points, you have >1 possible path as **indicated**.
  - We can prioritize types of errors arbitrarily.

UNIVERSITY OF TORONTO

Levenshtein distance

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1	0	1	2	3	4	5	6
	to	2	1	0	1	2	3	4	5
	recognize	3	2	1	1	2	3	4	5
	speech	4	3	2	2	2	3	4	5
	</s>	5	4	3	3	3	3	4	4

- And we finish the grid.
- There are $R[end, end] = 4$ word errors and a WER of $4/4 = 100\%$.
  - WER can be greater than 100% (relative to the reference).

Levenshtein distance

		hypothesis							
		<s>	how	to	wreck	a	nice	beach	</s>
Reference	<s>	0	1	2	3	4	5	6	7
	how	1	0	1	2	3	4	5	6
	to	2	1	0	1	2	3	4	5
	recognize	3	2	1	1	2	3	4	5
	speech	4	3	2	2	2	3	4	5
	</s>	5	4	3	3	3	3	4	4

- If we want, we can **backtrack** using our arrows (in a backtrace matrix).
- Here, we estimate 2 **substitution** errors and 2 **insertion** errors.

Summary

- We've seen how to:
  - **Extract useful speech features** with Mel-scale filter banks
  - **Model speech data** with Gaussian mixture models**.**
  - **Cluster** data with unsupervised algorithms.
  - **Recognize speech** with GMM-HMM, DNN-HMM and end-to-end DNN.
  - **Evaluate** ASR performance with Levenshtein distance.

- Next lecture: **synthesize** artificial speech.