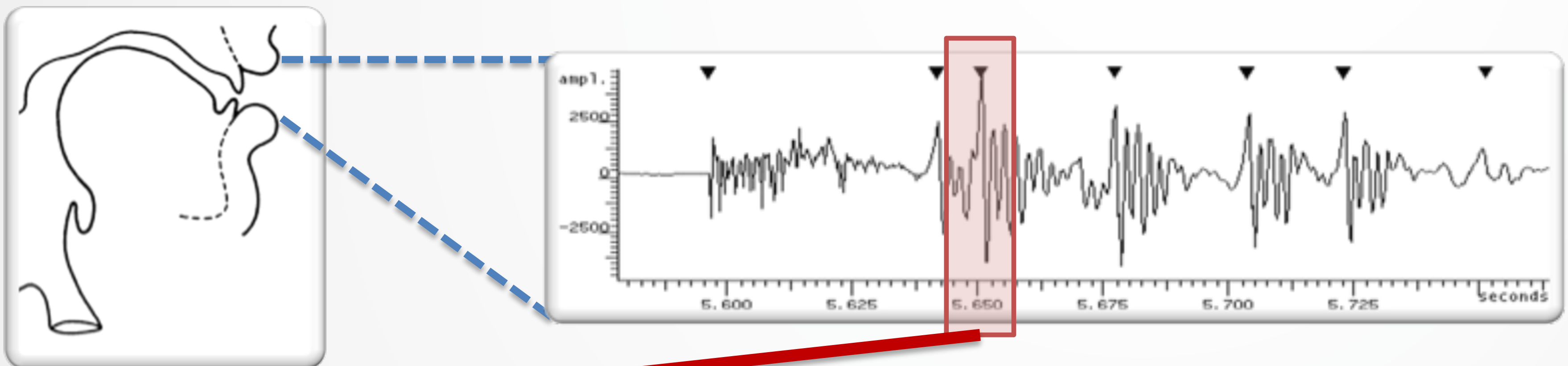
automatic speech recognition

CSC401/2511 – Natural Language Computing – Spring 2021

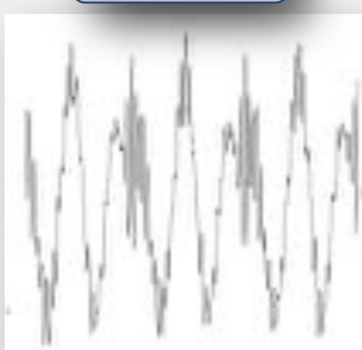
Lecture 8 Frank Rudzicz, Serena Jeblee, and Sean Robertson
University of Toronto

FEATURES

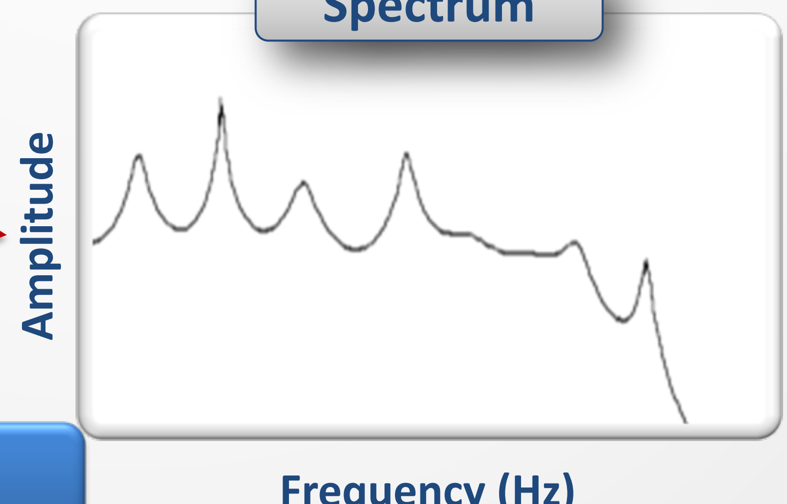
Recall our input to ASR



Frame



Spectrum



Is the spectrum the best input for our ASR systems?

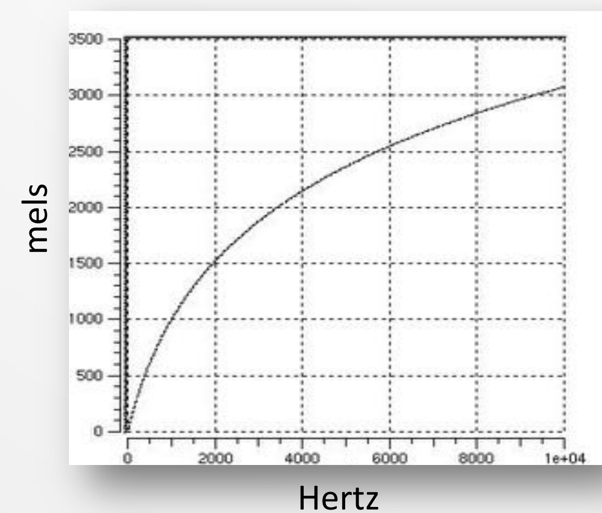
The Mel-scale

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

- A **mel** is a unit of pitch. Pairs of sounds which are **perceptually** equidistant in pitch are separated by an equal number of **mels**.

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

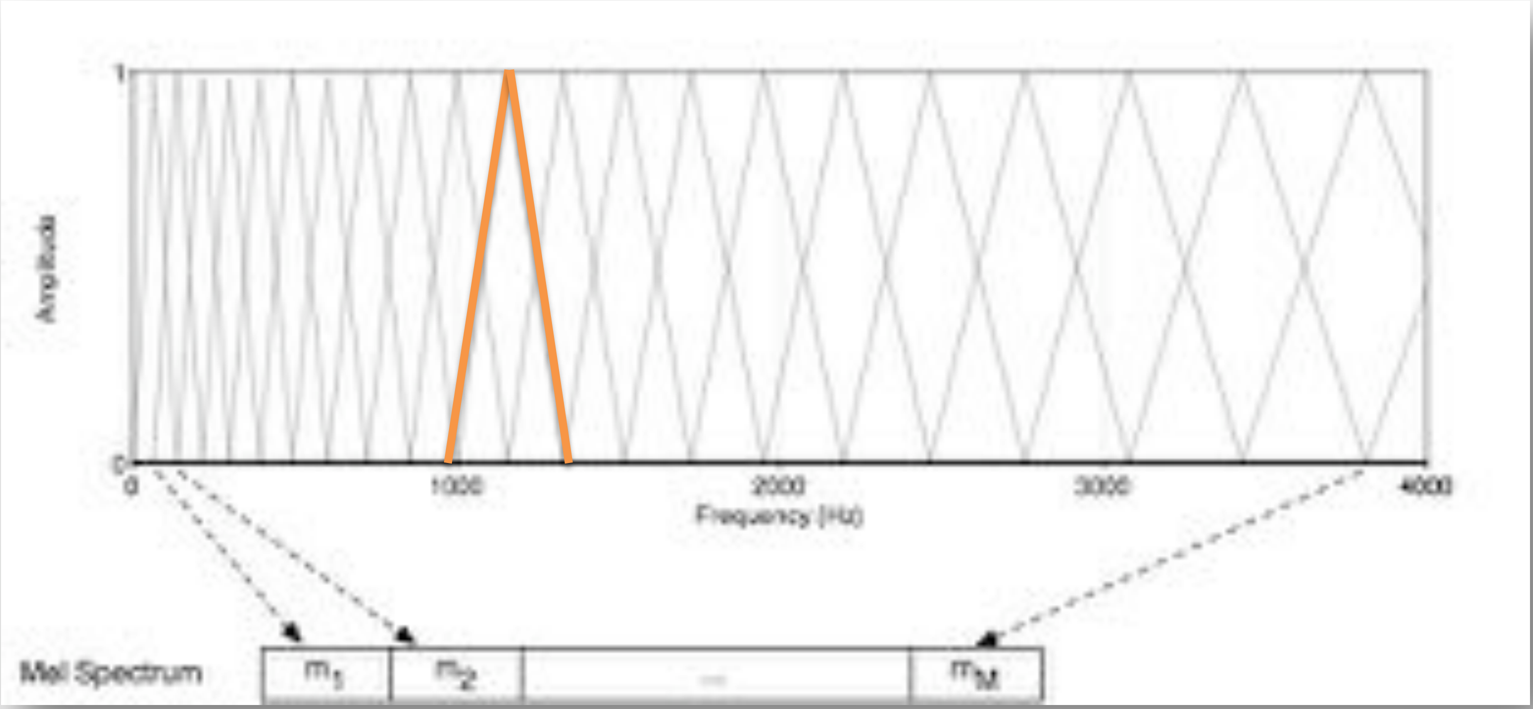
(No need to memorize this either)



UNIVERSITY OF TORONTO

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** spacing before 1 kHz, **logarithmic** after 1 kHz



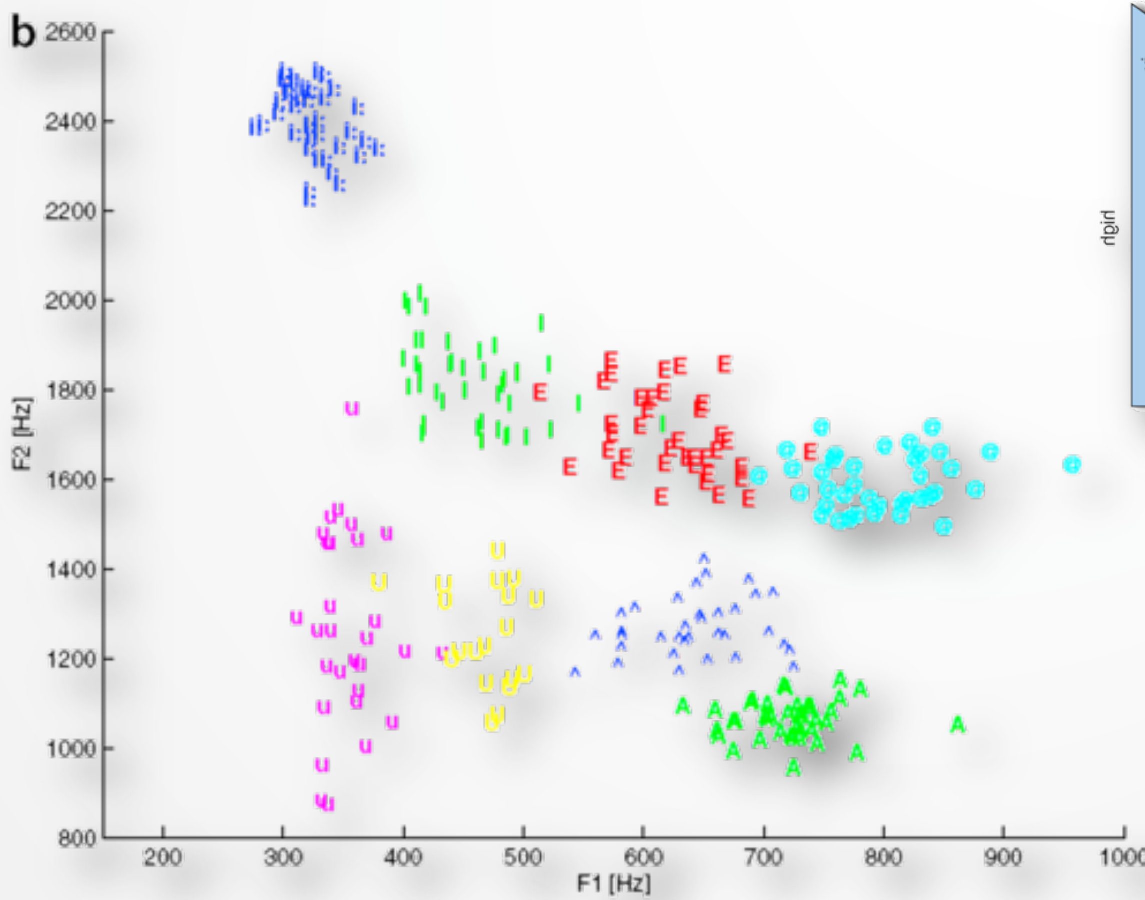
UNIVERSITY OF TORONTO

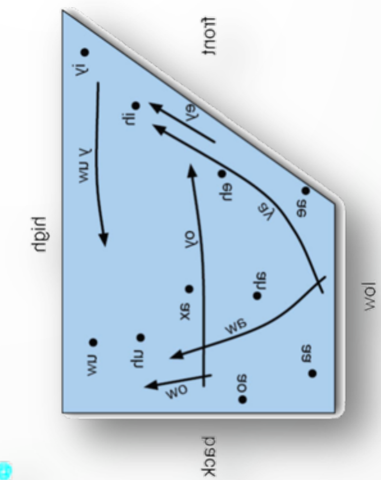
Aside - Mel-Frequency Cepstral Coefficients

- 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 can be found in the appendices (not tested)
- 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

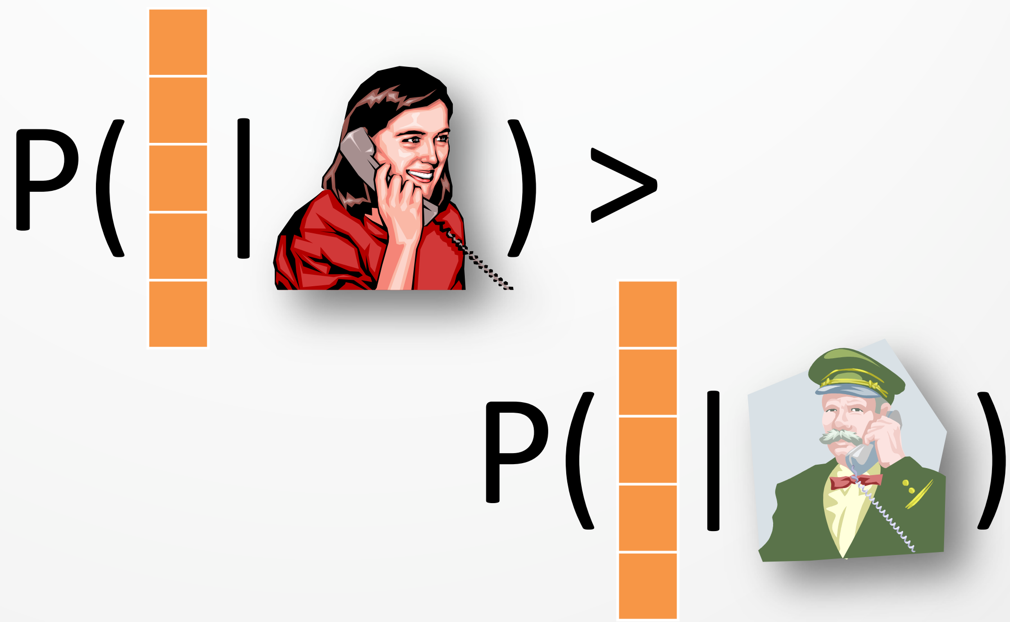
- Speech sounds can cluster. This graph shows vowels, each in their own colour, according to the 1st two formants.

Classifying speakers

- 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			…	

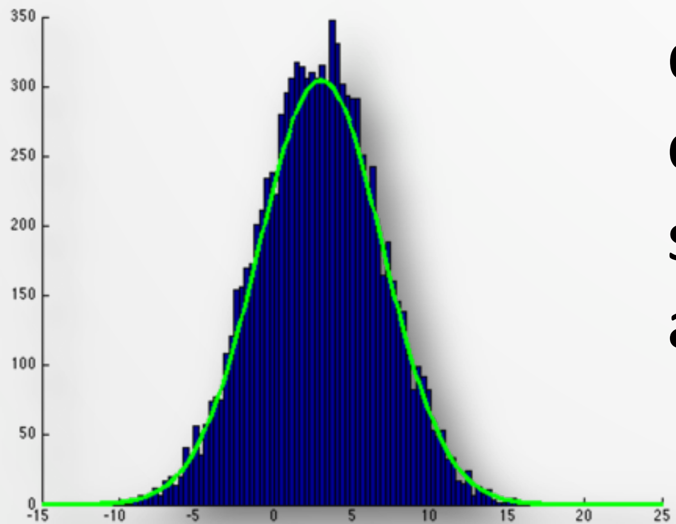
Observation matrix



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.



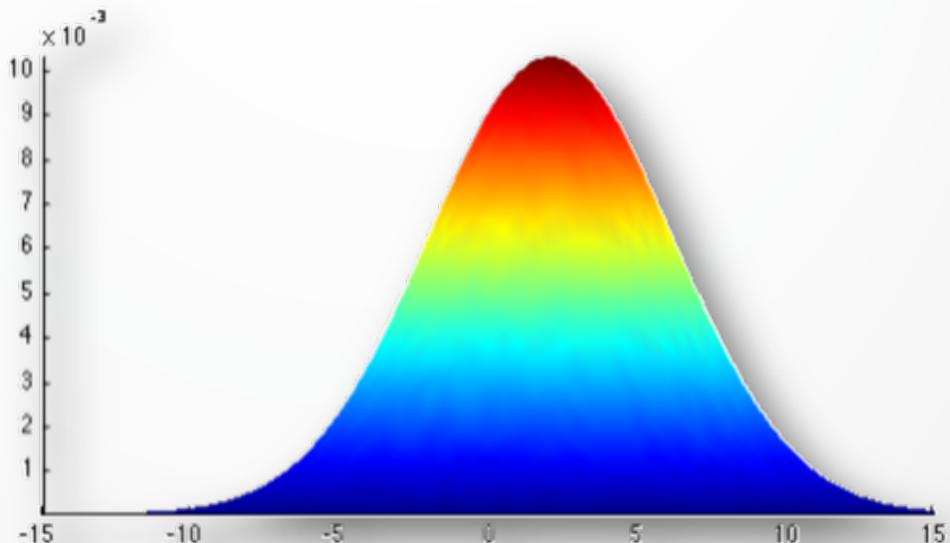
- If we *assume* the 1-dimensional data in **this histogram** is Normally distributed, we can fit a continuous Gaussian function simply in terms of the mean μ and variance σ^2.

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 $X = \{x_1, x_2, \dots, x_n\}$, 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 $\boldsymbol{L(X, \theta)} = \boldsymbol{P(X; \theta)} = \prod_{i=1}^{n} P(x_i; \theta)$.

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

UNIVERSITY OF TORONTO

Multivariate Gaussians

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

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

the **mean** is

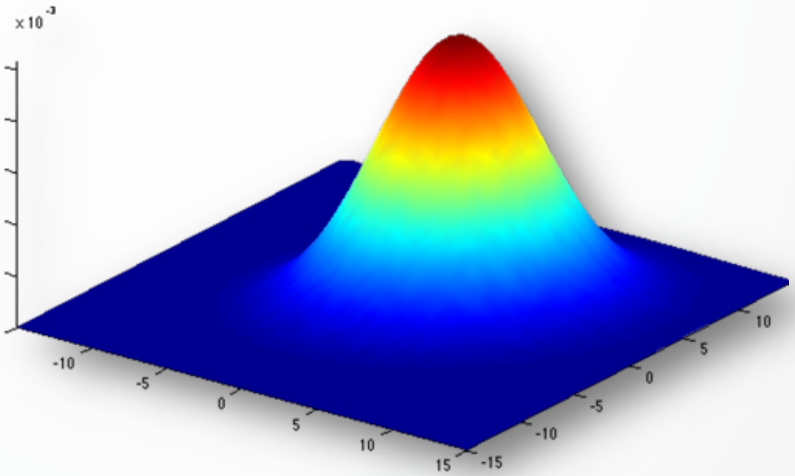
$$\vec{\mu} = E(\vec{x}) = \langle \mu[1], \mu[2], \dots, \mu[d] \rangle$$

the **covariance matrix** is

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

and

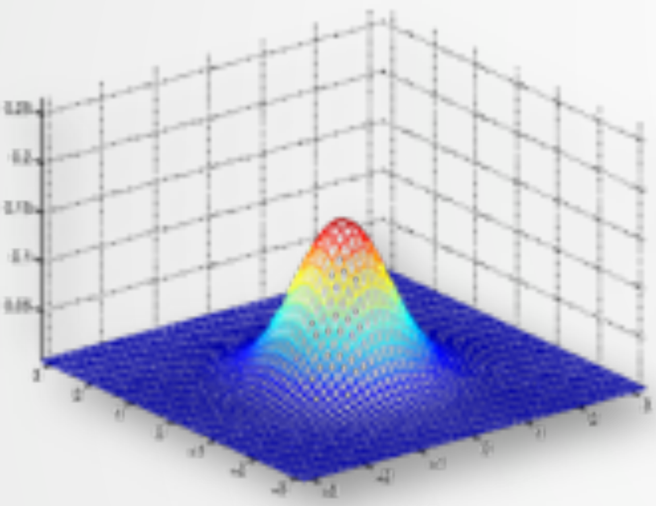
$$P(\vec{x}) = \frac{\exp\left(-\frac{(\vec{x}-\vec{\mu})^{\intercal}\Sigma^{-1}(\vec{x}-\vec{\mu})}{2}\right)}{(2\pi)^{\frac{d}{2}}\,|\Sigma|^{\frac{1}{2}}}$$



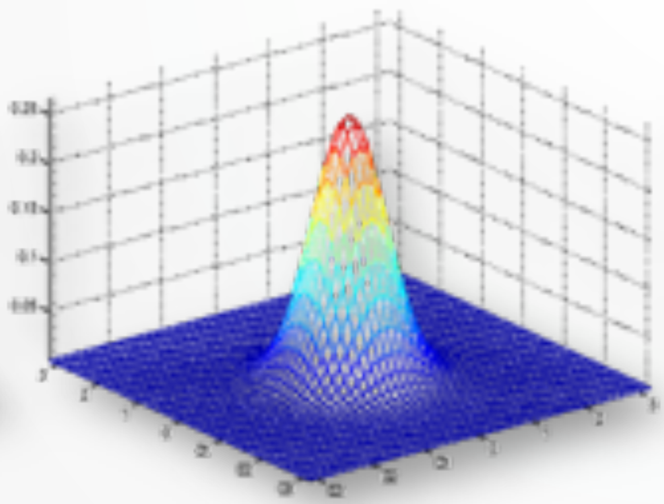
$A^{\intercal}$ is the **transpose** of A
A^{-1} is the **inverse** of A
$|A|$ is the **determinant** of A

UNIVERSITY OF TORONTO

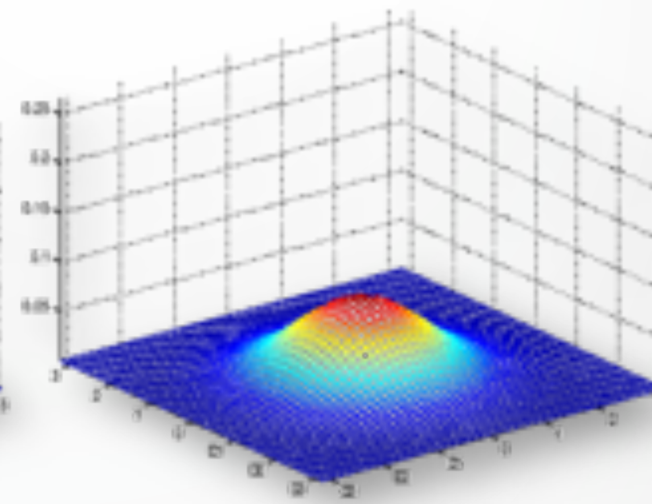
Intuitions of covariance



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



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

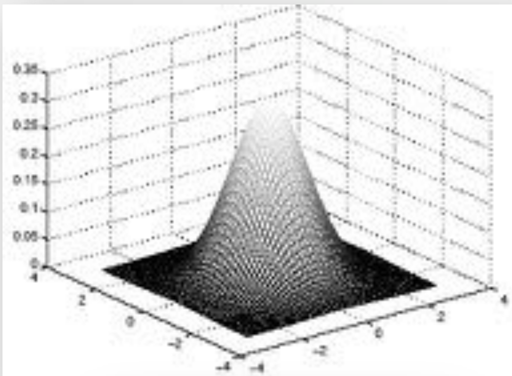


$\mu = [0\ 0]$
$\Sigma = 2.0\mathbf{I}$

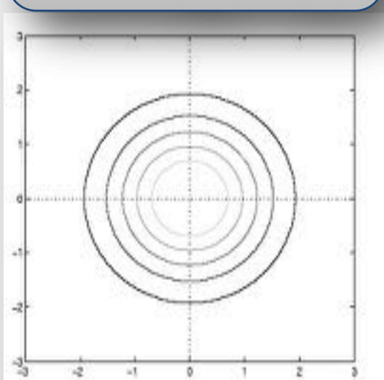
- As values in Σ become larger, the Gaussian spreads out.
- ($\mathbf{I}$ is the identity matrix)

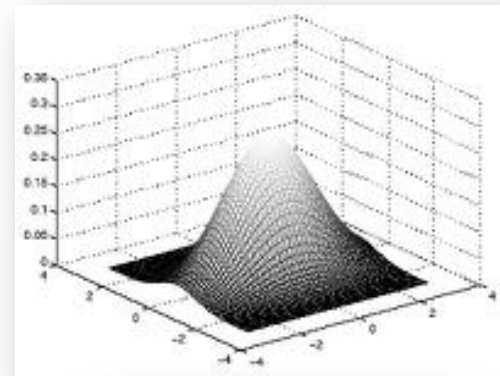
UNIVERSITY OF TORONTO

Intuitions of covariance

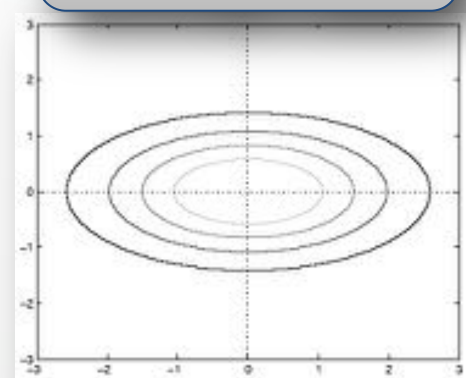


$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$





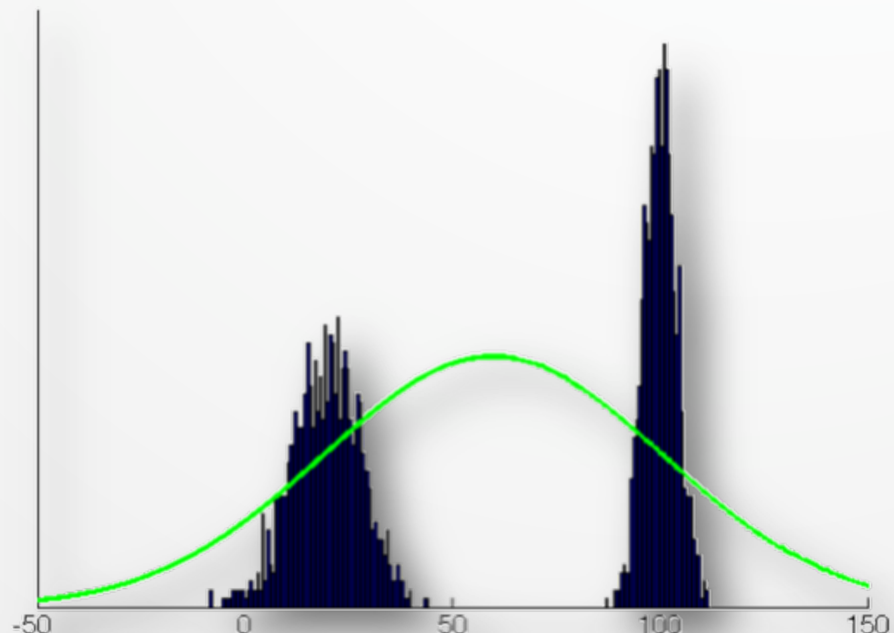
$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 0.6 \end{bmatrix}$$



- 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.

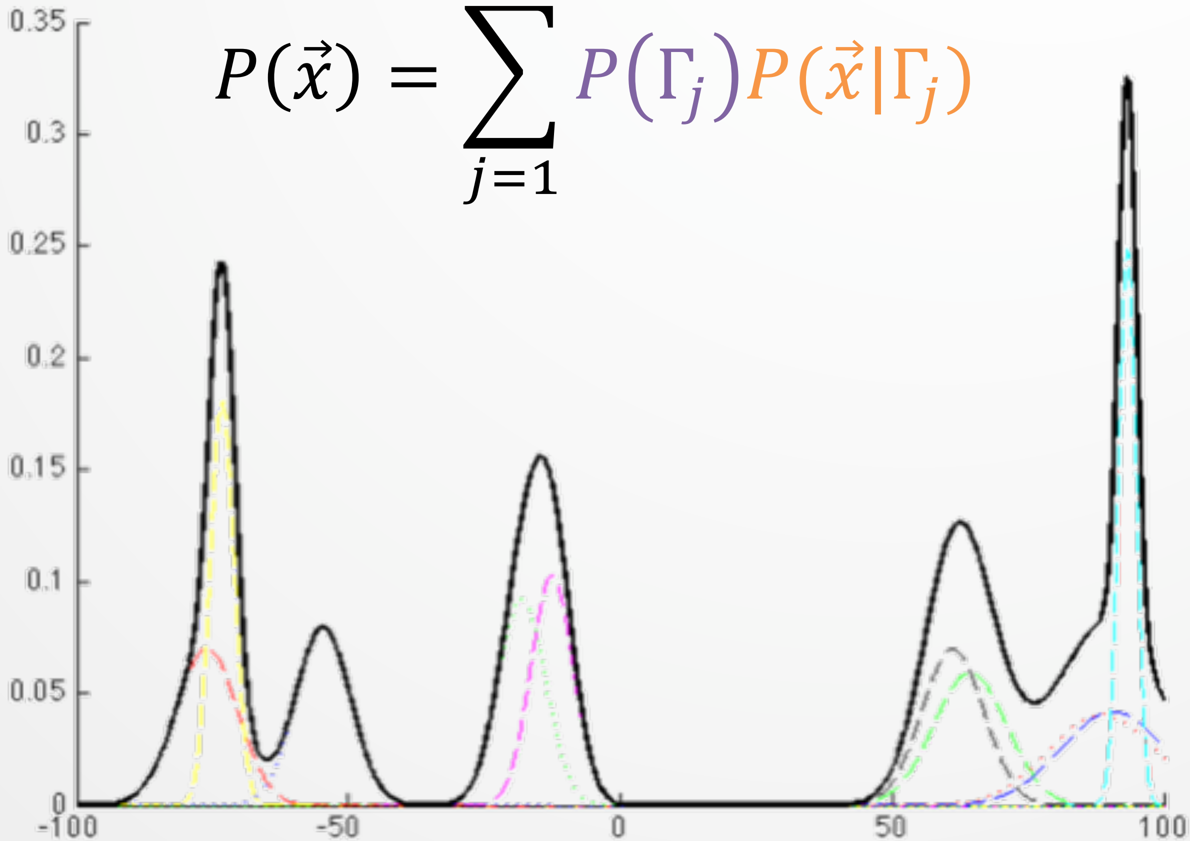


UNIVERSITY OF TORONTO

Mixtures of Gaussians

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

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



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 from slide 20 becomes

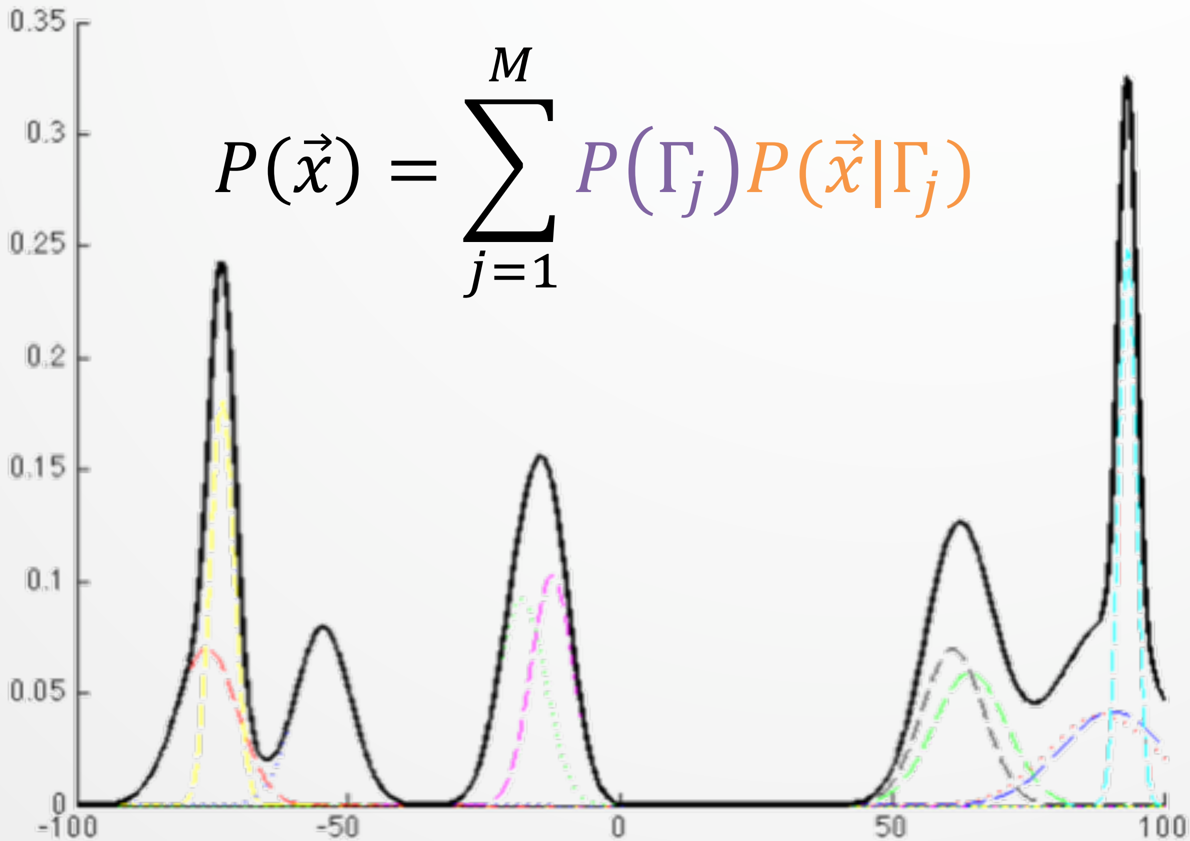
$$P(\vec{x}|\Gamma_m) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{d}\frac{(x[i]-\mu_m[i])^2}{\Sigma_m[i]}\right)}{(2\pi)^{\frac{d}{2}}\left(\prod_{i=1}^{d}\Sigma_m\,[i]\right)^{\frac{1}{2}}}$$

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

Mixtures of Gaussians

- If we knew *which* Gaussian generated each sample, we could learn $P(\Gamma_j)$ with MLE, but that data is **hidden**, so we must use…

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



Expectation-Maximization for GMMs

- If $\boldsymbol{\omega_m} = P(\Gamma_m)$ and $\boldsymbol{b}_m(\overrightarrow{\boldsymbol{x}_t}) = P(\overrightarrow{x_t}|\Gamma_m)$,

'component observation likelihood'

'weight'

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

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

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

UNIVERSITY OF TORONTO

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.

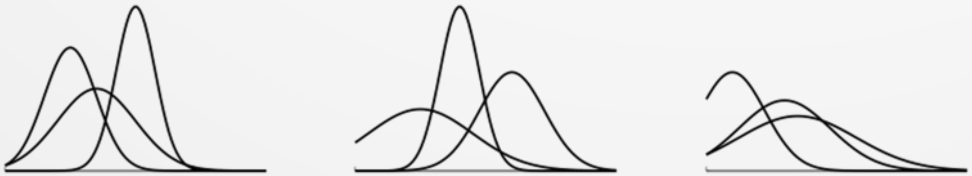
- Can $\widehat{\overrightarrow{\sigma_m^2}} = \frac{\sum_t P(\Gamma_m | \overrightarrow{x_t}; \theta) \overrightarrow{x_t}^2}{\sum_t P(\Gamma_m | \overrightarrow{x_t}; \theta)} - \widehat{\overrightarrow{\mu_m}}^2$ become negative?
  - No.
    - This is left as an exercise, but only if you're interested.

UNIVERSITY OF TORONTO

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.



UNIVERSITY OF TORONTO

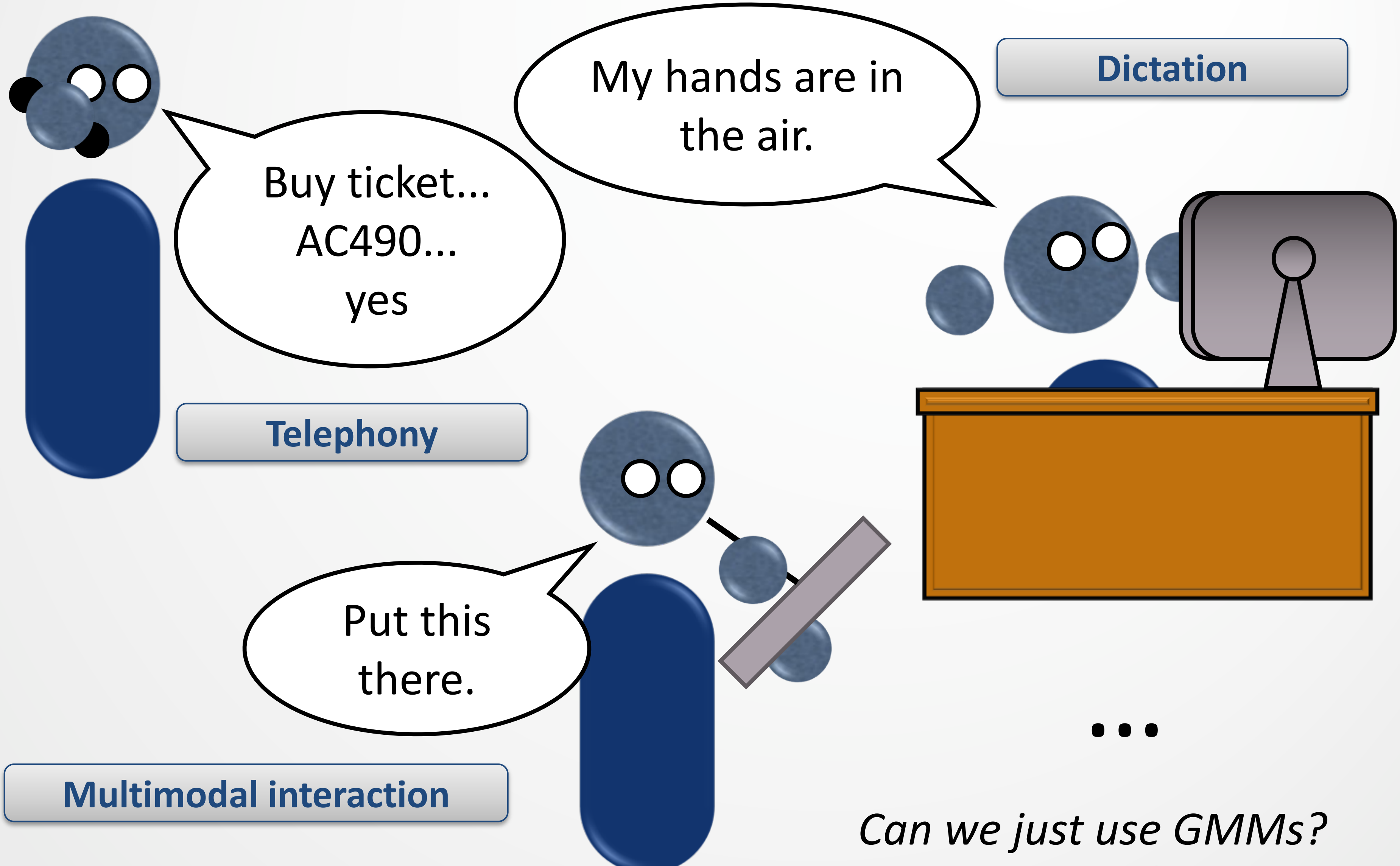
Recipe for GMM EM

- 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.

SPEECH RECOGNITION

Consider what we want speech to do

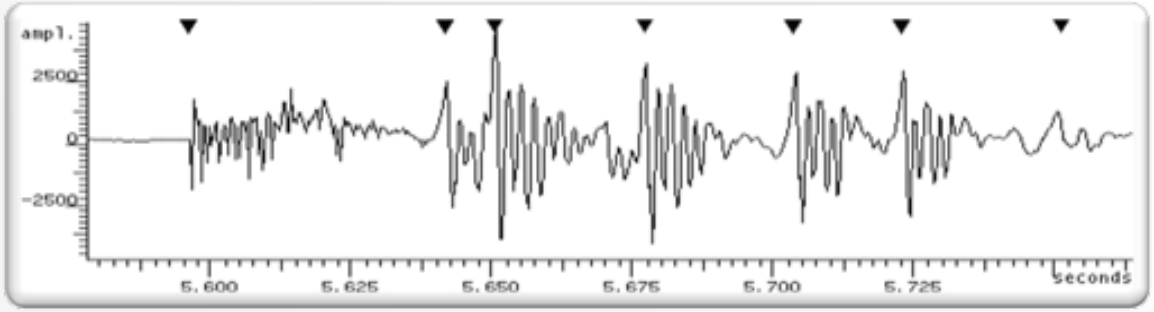


Can we just use GMMs?

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?

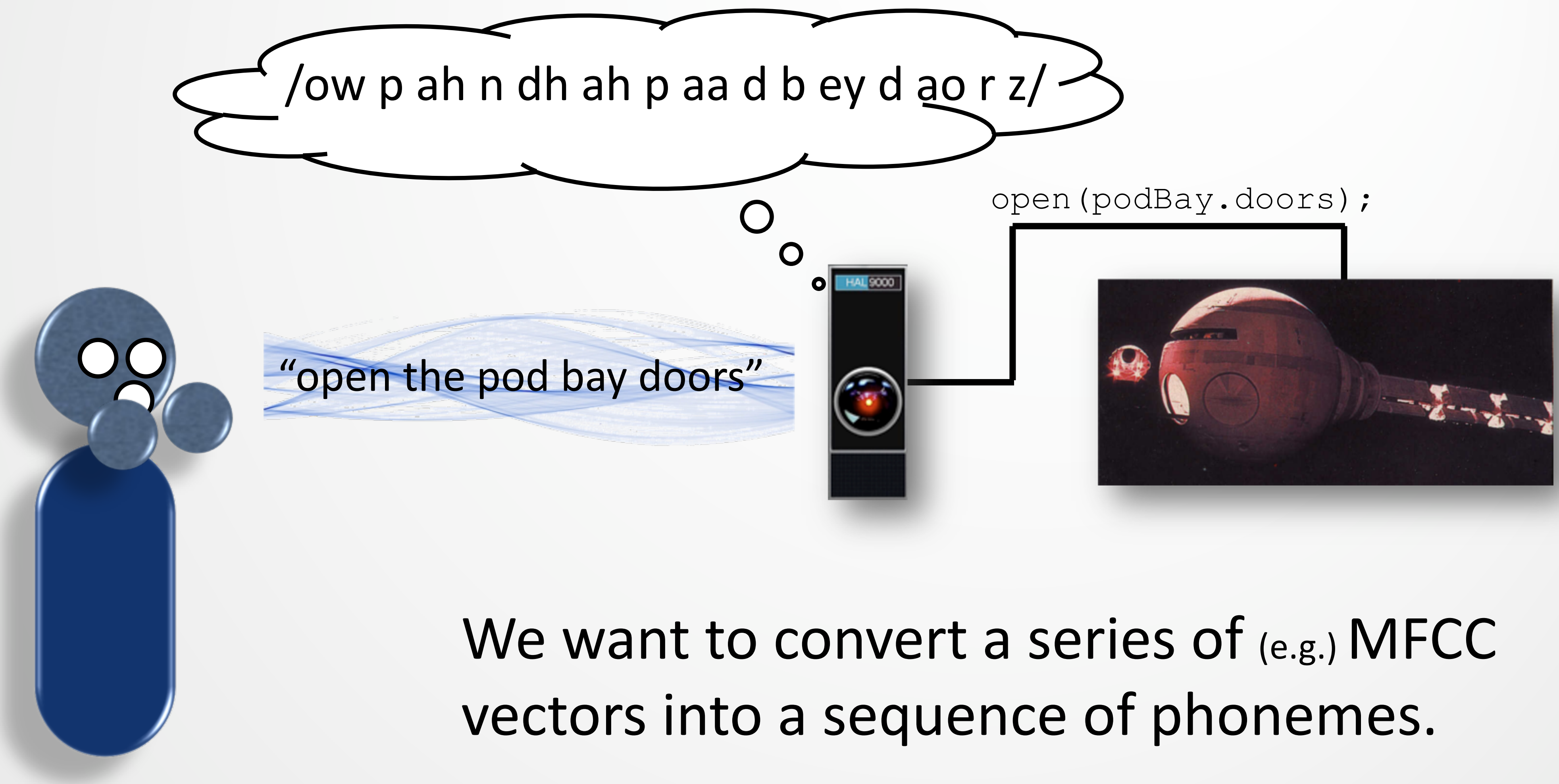
Speech is dynamic



- 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**.

Speech is sequences of phonemes (*)



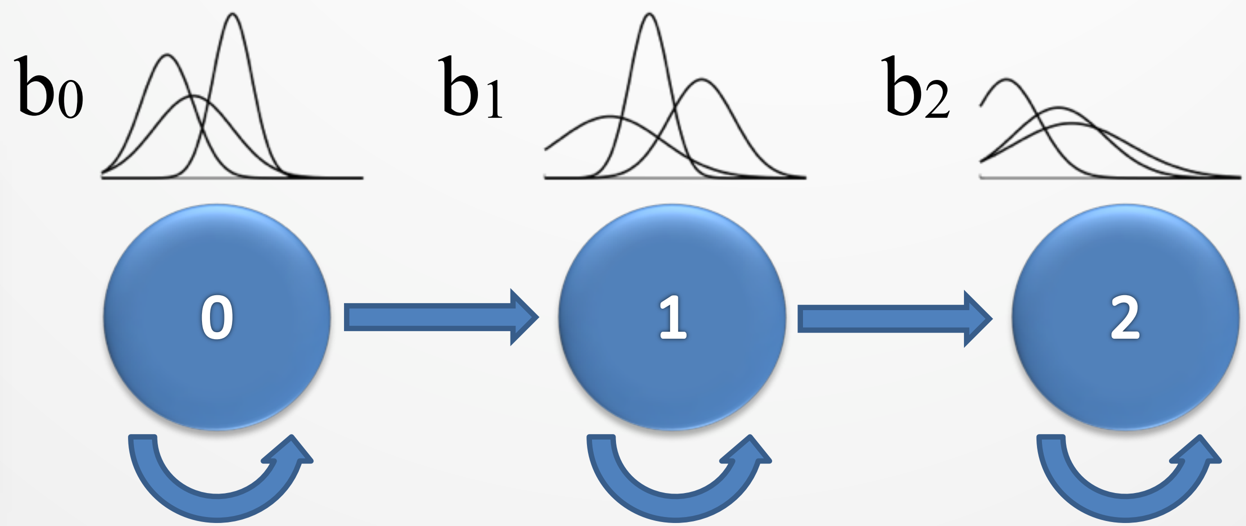
We want to convert a series of (e.g.) MFCC vectors into a sequence of phonemes.

(*) not really

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.

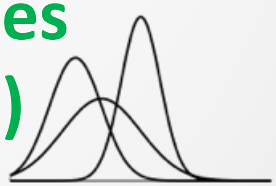
$$\vec{x} = \begin{bmatrix} 4.32957 \\ 2.48562 \\ 1.08139 \\ \ldots \\ 0.45628 \end{bmatrix}$$



Defining CHMMs

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

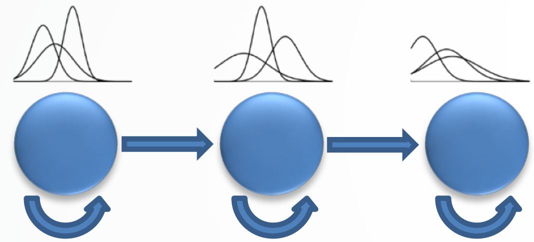
θ
  - $\Pi = \{\pi_1, \dots, \pi_N\}$: initial state probabilities
  - $A = \{a_{ij}\}, i, j \in S$: state transition probabilities
  - $B = b_i(\vec{x}), i \in S, \vec{x} \in X$: **state output probabilities (i.e., Gaussian mixtures)**



yielding
  - $Q = \{q_0, \dots, q_T\}, q_i \in S$: state sequence
  - $\mathcal{O} = \{\sigma_0, \dots, \sigma_T\}, \sigma_i \in X$: observation sequence

UNIVERSITY OF TORONTO

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.

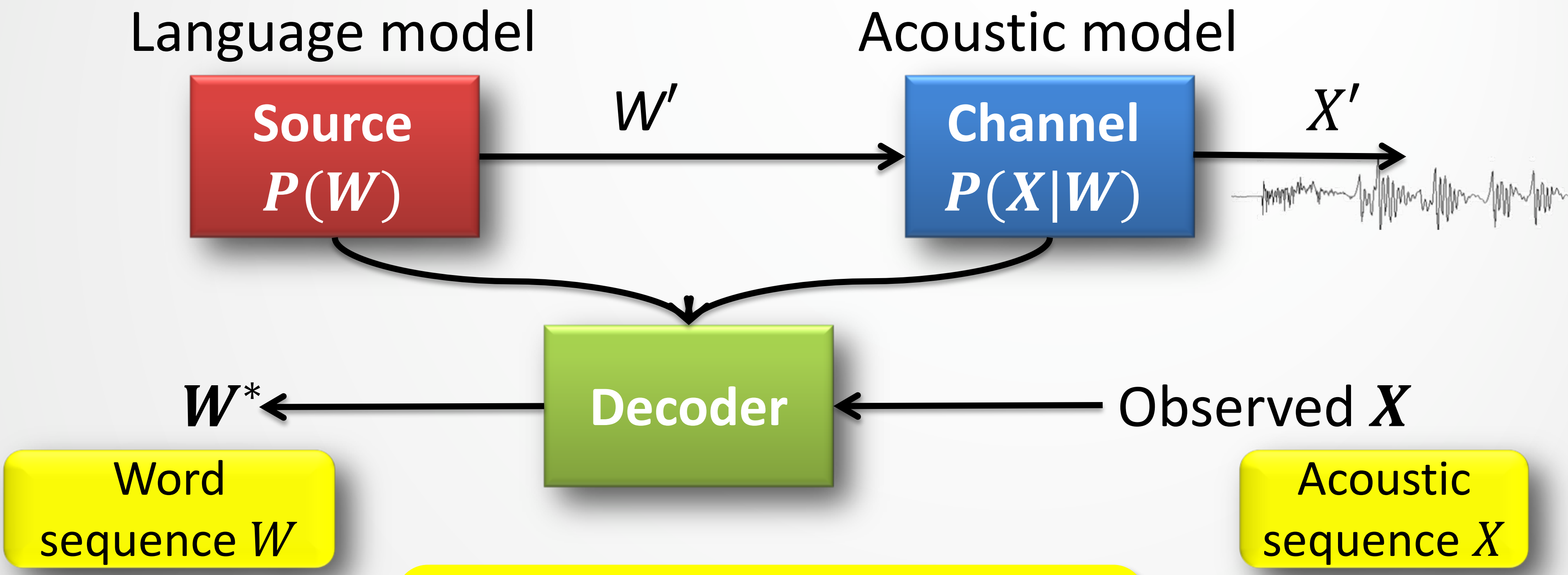
- We find the best state sequences using **Viterbi**, as before.
  - Here, the best state sequence gives us a **sequence of phonemes**

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
```

The noisy channel model for ASR

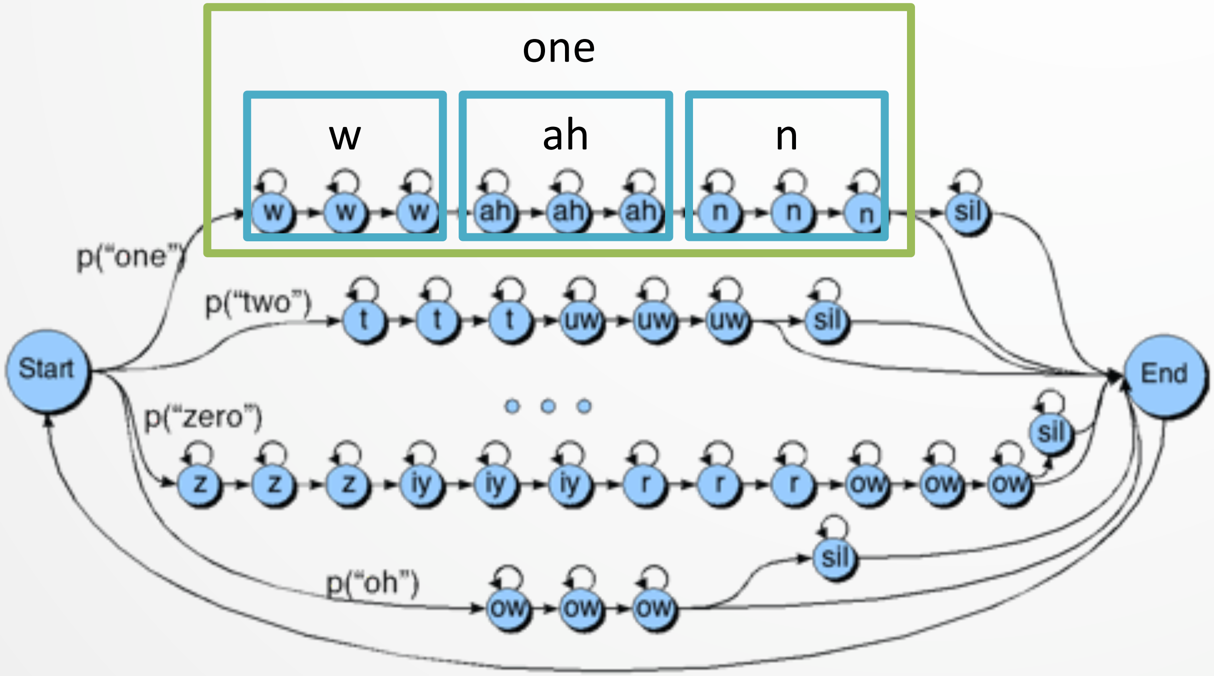


$$W^* = \underset{W}{\operatorname{argmax}} P(X|W)P(W)$$

How to encode $P(X|W)$?

Putting it together

- How do we combine the language model, phonemic dictionary, and CHMM together? → Nest them!



- Full details are an aside – see **Appendices** and J&M 2nd Ed.

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: *how to recognize speech*

    but an ASR system heard

    Hypothesis: *how to 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…

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*'

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

UNIVERSITY OF TORONTO

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

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							
	</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.

UNIVERSITY OF TORONTO

NEURAL SPEECH RECOGNITION

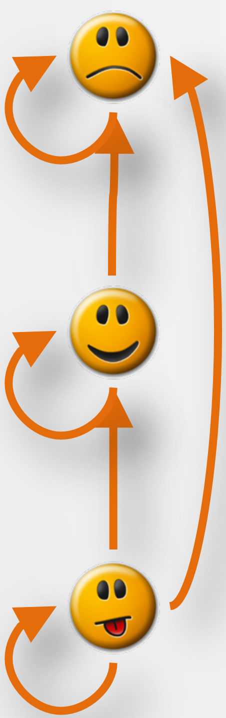
Remember Viterbi

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}

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

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

Do these probabilities need to be GMMs?



0
0

0.06
0

0.08
0

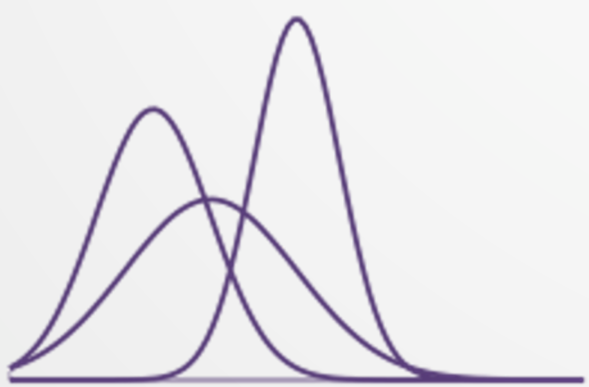
$\sigma_0 = ship$ $\sigma_1 = frock$ $\sigma_2 = tops$

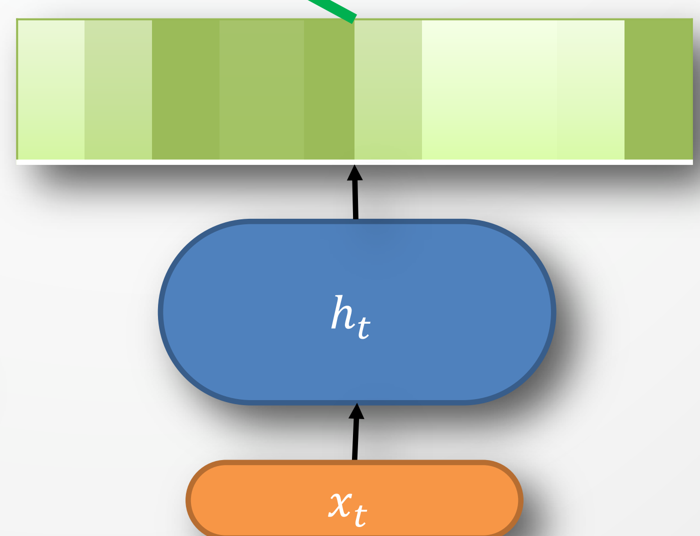
Observations, σ_t

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:

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

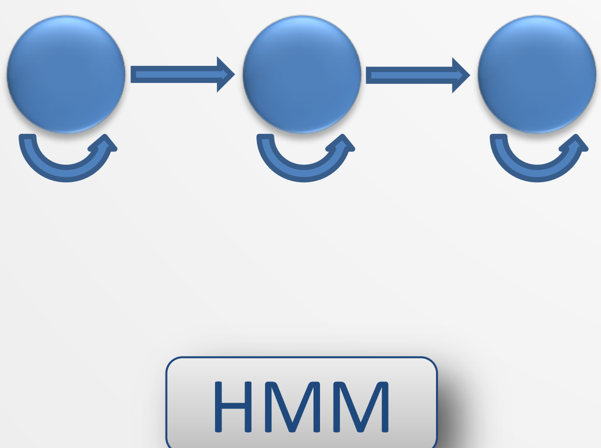


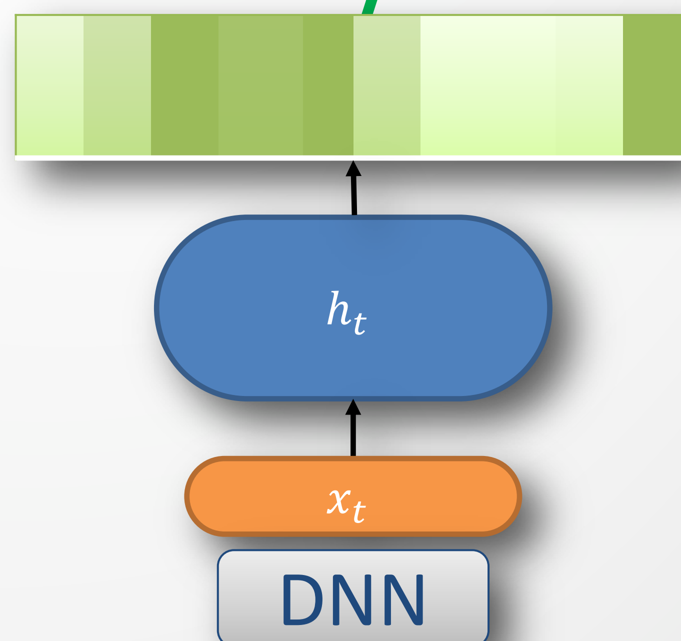


Replacing GMMs with DNNs

- The probability of a word sequence W comes *loosely* from $P(X|W)$

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





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!
- Solution: bootstrapping
  - Use another model to determine q_t
  - Often $\mathrm{argmax}_{q_{1\ldots T}} P_{CHMM}(q_{1\ldots T}, x_{1\ldots T})$ from GMM-HMM
    - $P_{CHMM}(q_t|q_{t-1})$ often stolen as well
    - … and $P(q_t)$
- Other, advanced methods exist

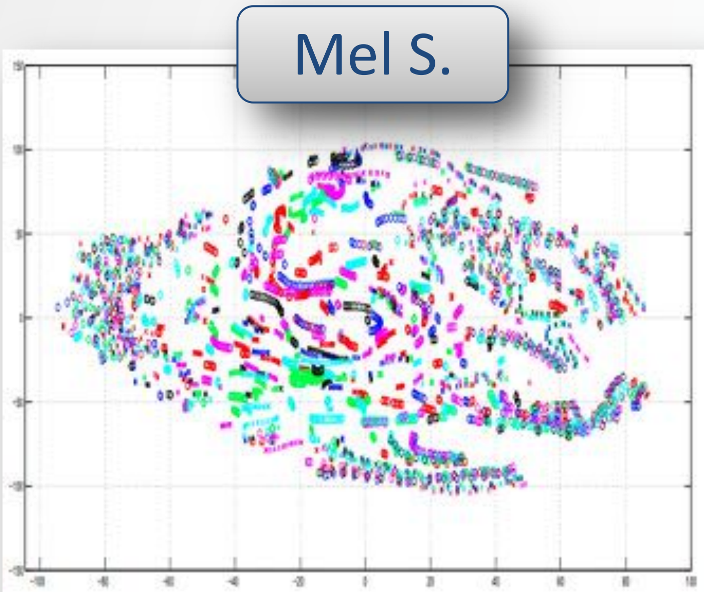
Hybrid HMM and DNN

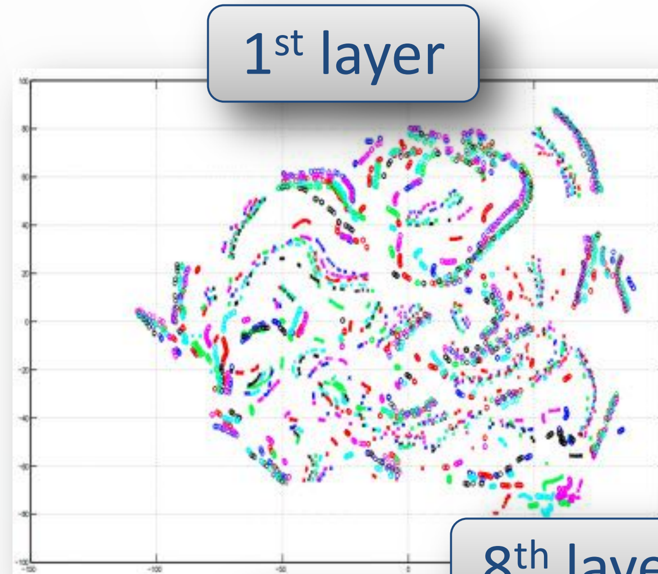
[TABLE 3] A COMPARISON OF THE PERCENTAGE WERs USING DNN-HMMs AND GMM-HMMs ON FIVE DIFFERENT LARGE VOCABULARY TASKS.

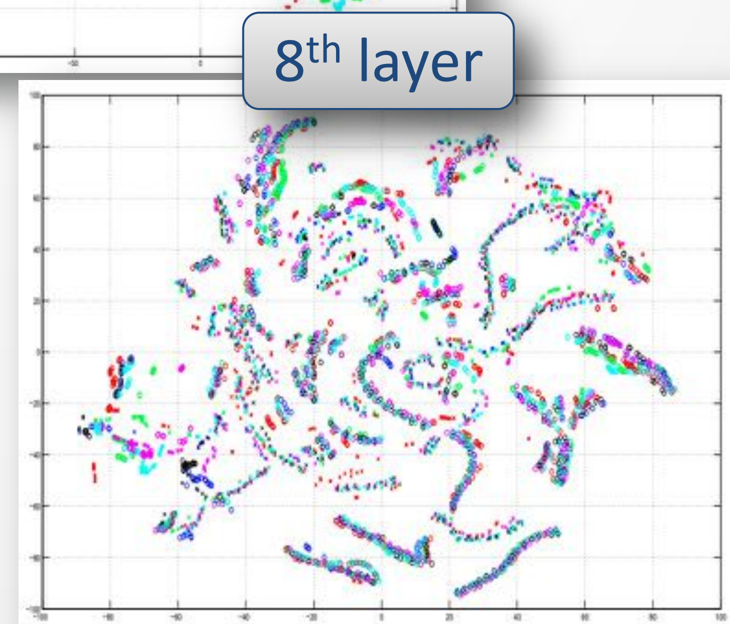
TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT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

What are these DNNs learning?







- 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).

What are these DNNs learning?

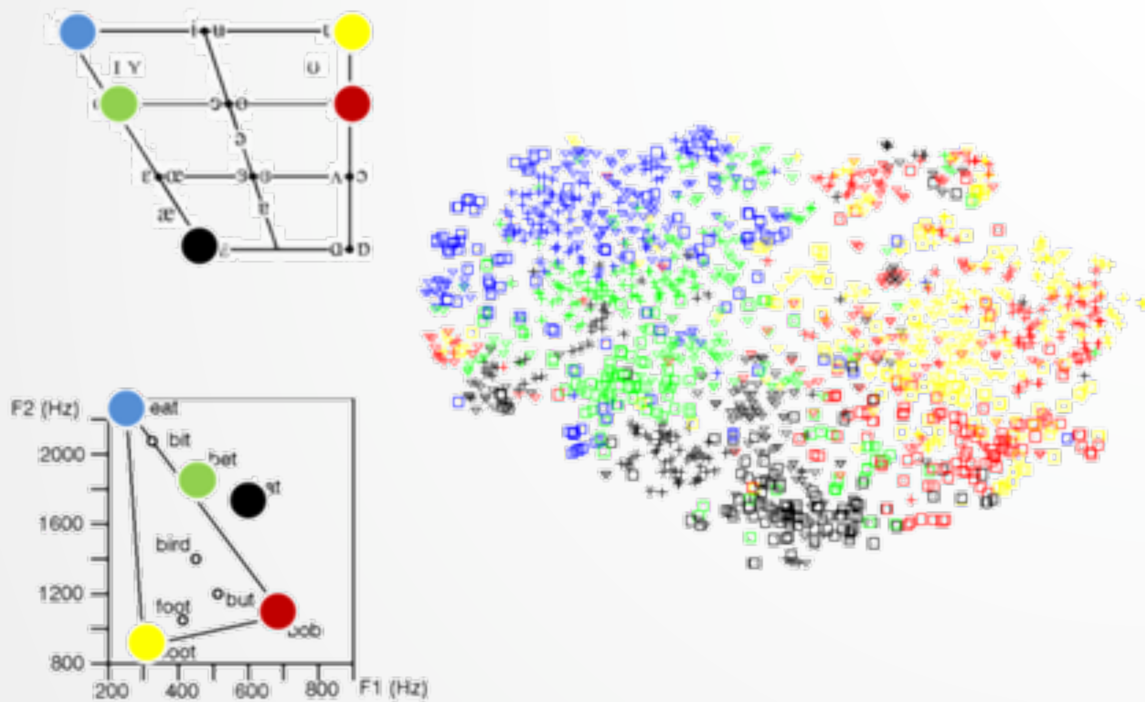


Figure 1: Multilingual BN features of five vowels from French (+), German (□) and Spanish (▽): /a/ (black), /i/ (blue), /e/ (green), /o/ (red), and /u/ (yellow)

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

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

Seq2seq architectures

- 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

…

That's it.

Aside – Listen, Attend, and Spell

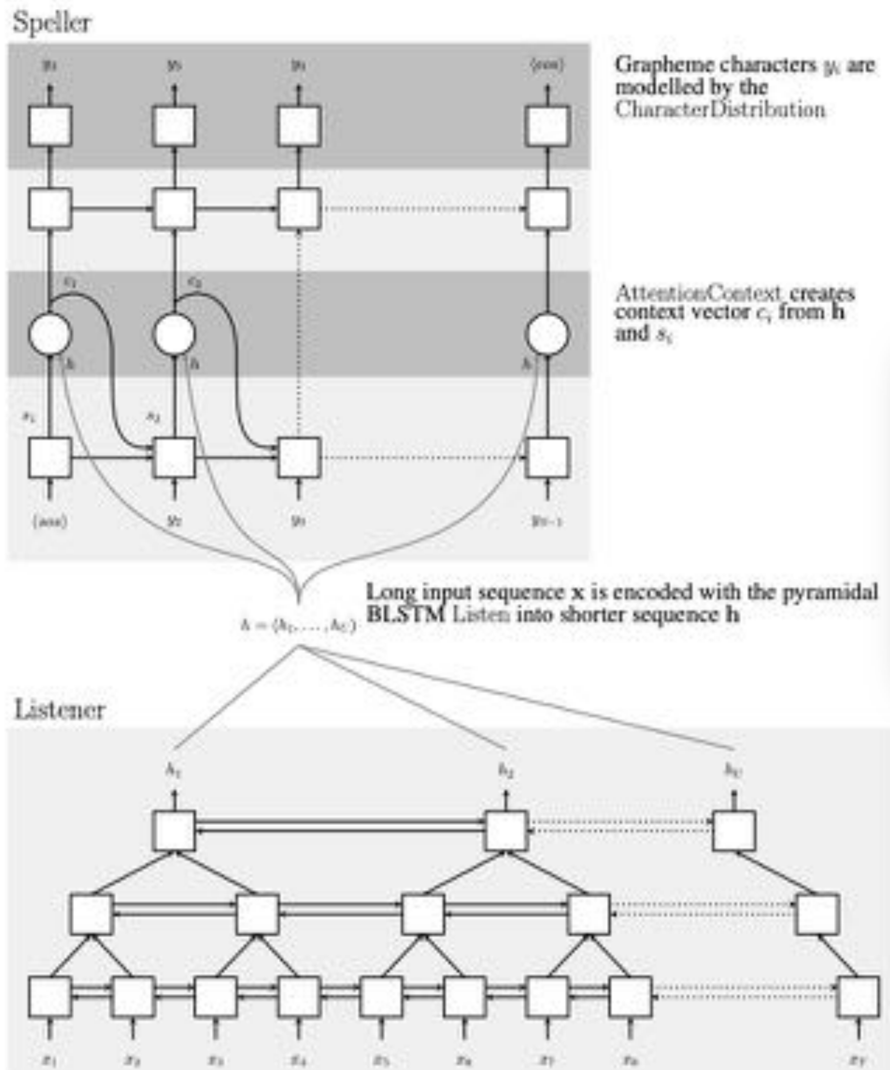


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence **x** into high level features **h**, the speller is an attention-based decoder generating the **y** characters from **h**.

Table 1: WER comparison on the clean and noisy Google voice search task. The CLDNN-HMM system is the state-of-the-art system, the Listen, Attend and Spell (LAS) models are decoded with a beam size of 32. Language Model (LM) rescoring was applied to our beams, and a sampling trick was applied to bridge the gap between training and inference.

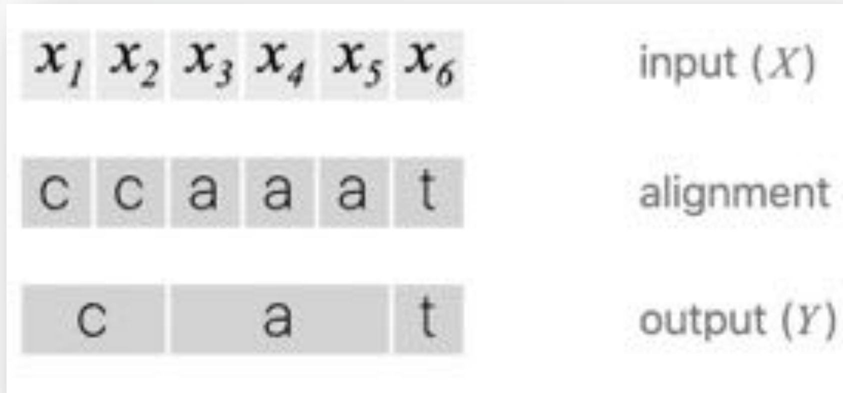
Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

https://arxiv.org/abs/1508.01211

UNIVERSITY OF TORONTO

Connectionist Temporal Classification

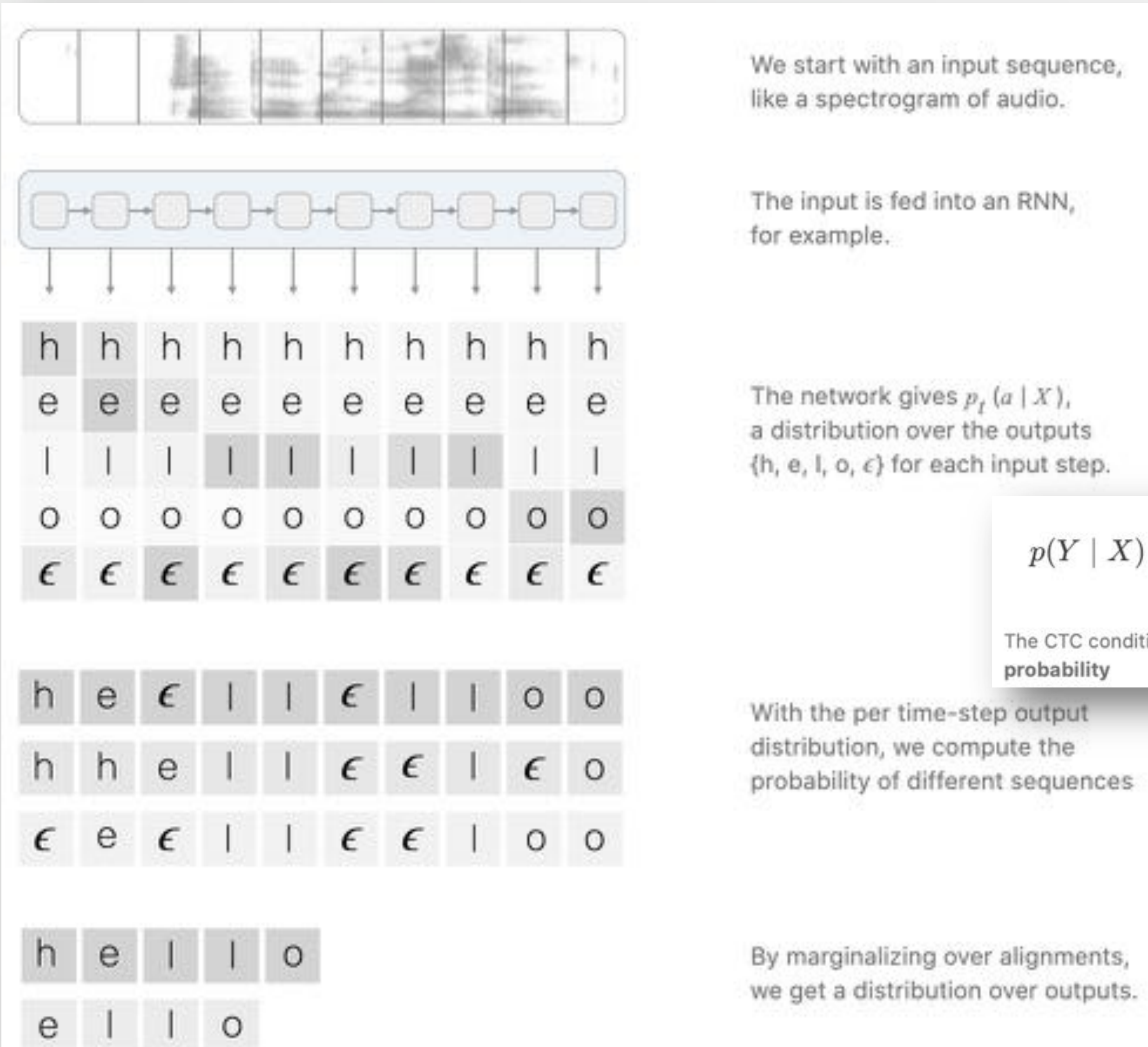
- Consider alignment:



- Not every input step needs an output. How can we collapse alignments for multi-character output (like, '*his*' vs '*hiss*')?
  - CTC introduces 'blank token' ϵ as a placeholder

See: https://distill.pub/2017/ctc/

Connectionist Temporal Classification



This is computed by an RNN

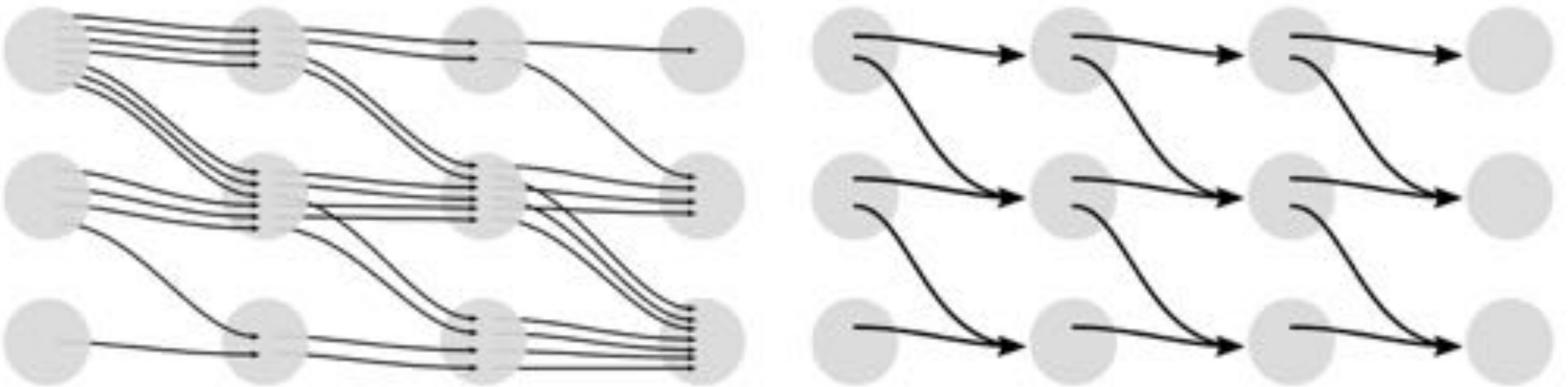
$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability** | **marginalizes** over the set of valid alignments | computing the **probability** for a single alignment step-by-step.

See: https://distill.pub/2017/ctc/

UNIVERSITY OF TORONTO

Connectionist Temporal Classification



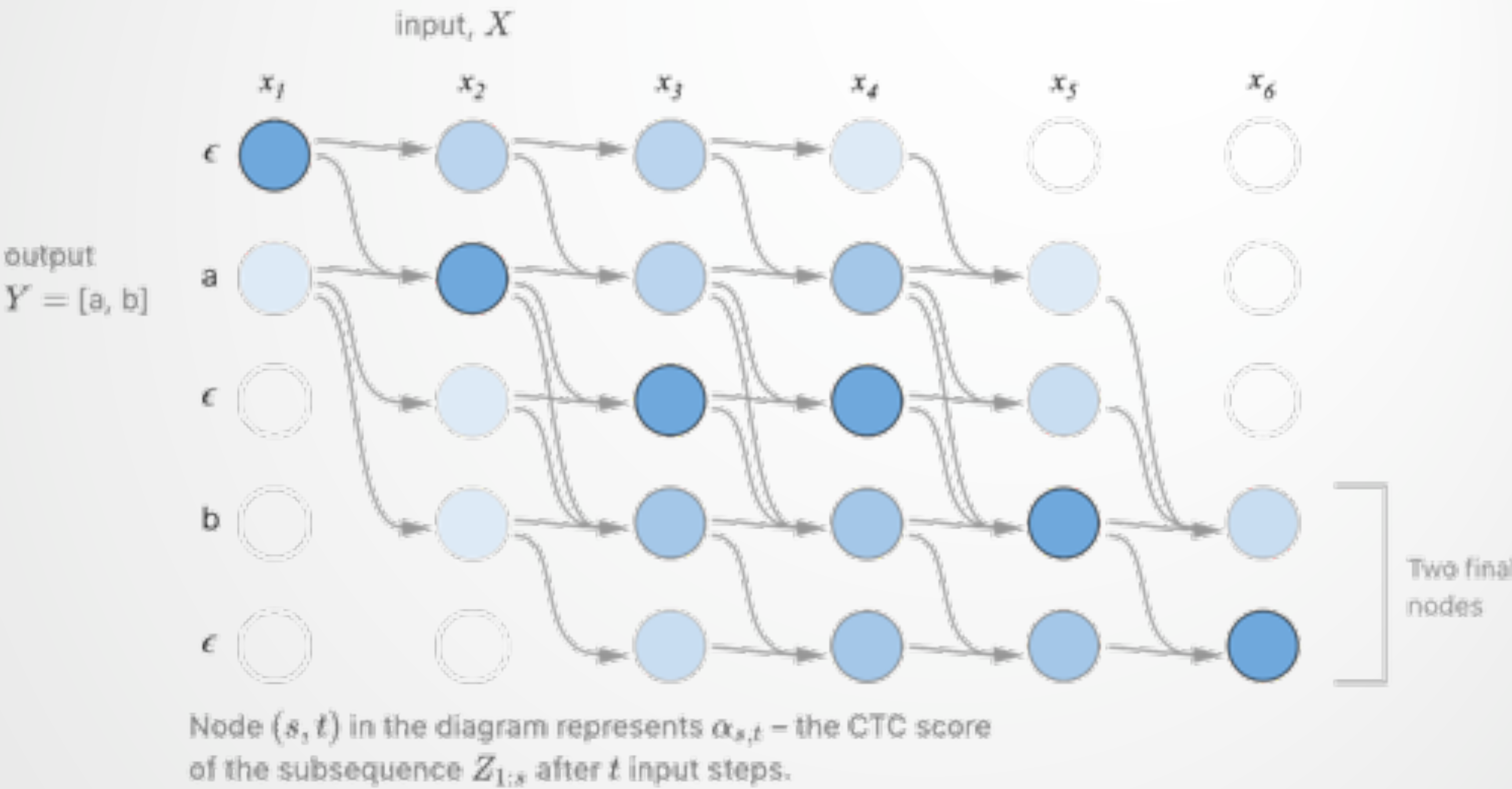
Summing over all alignments can be very expensive.

Dynamic programming merges alignments, so it's much faster.

See: https://distill.pub/2017/ctc/

UNIVERSITY OF TORONTO

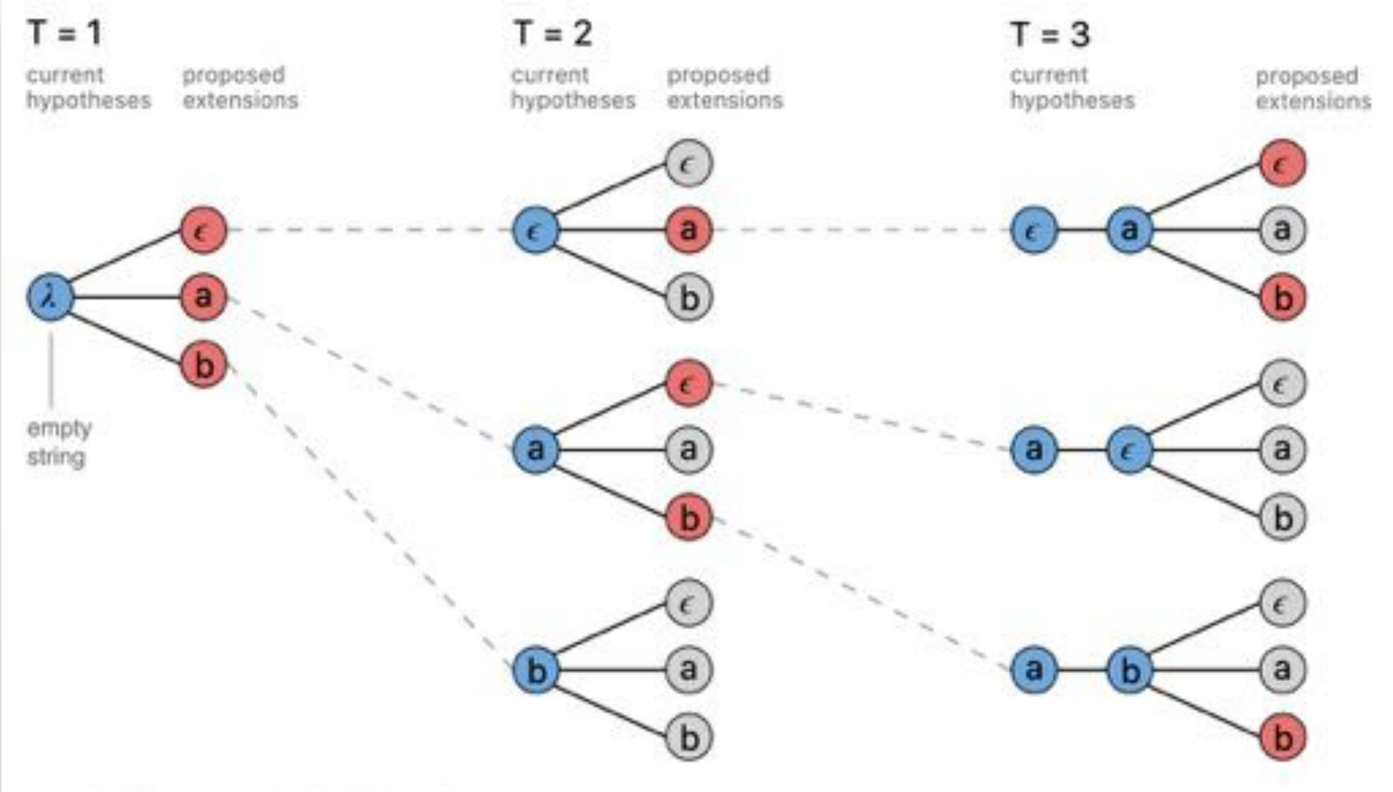
Connectionist Temporal Classification



See: https://distill.pub/2017/ctc/

Connectionist Temporal Classification

- It is still expensive to consider *all* possible alignments, and it is naïve to merely pick the max probability at each time step.
  - We therefore introduce a **beam search** (like in NMT)



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

See: https://distill.pub/2017/ctc/

End-to-end neural networks

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.

UNIVERSITY OF TORONTO

State-of-the-art?

Deep Speech: Scaling up end-to-end speech recognition

Awni Hannun*, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng

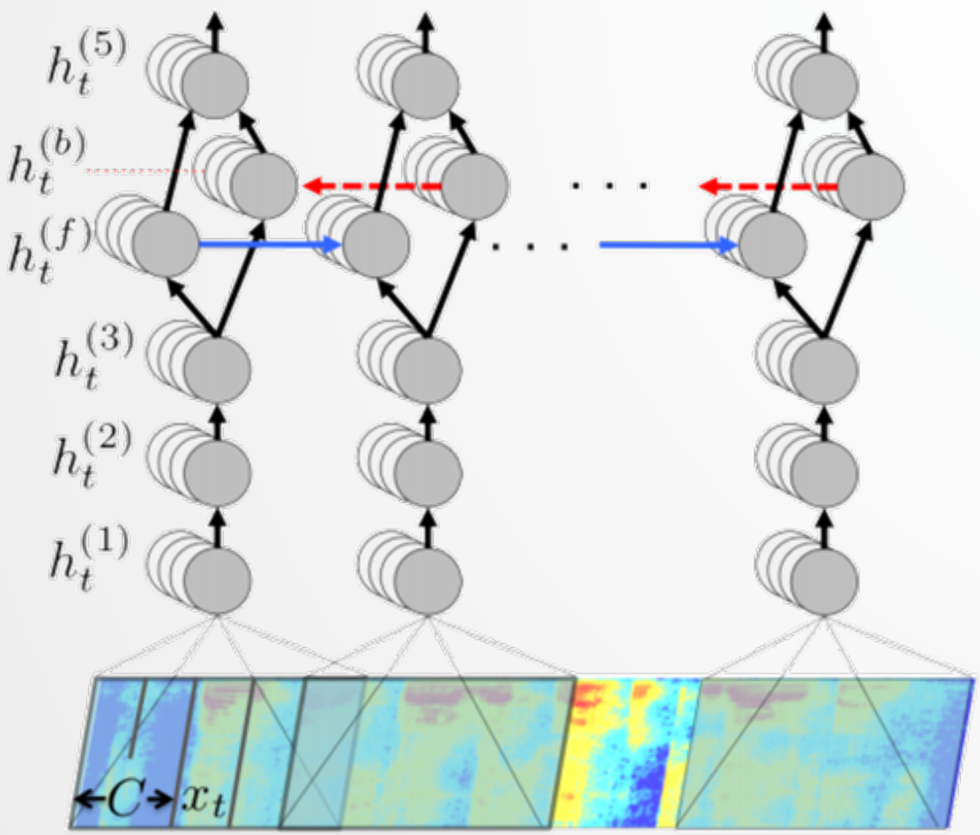
Baidu Research – Silicon Valley AI Lab

Abstract

We present a state-of-the-art speech recognition system developed using end-to-end deep learning. Our architecture is significantly simpler than traditional speech systems, which rely on laboriously engineered processing pipelines; these traditional systems also tend to perform poorly when used in noisy environments. In contrast, our system does not need hand-designed components to model background noise, reverberation, or speaker variation, but instead directly learns a function that is robust to such effects. We do not need a phoneme dictionary, nor even the concept of a "phoneme." Key to our approach is a well-optimized RNN training system that uses multiple GPUs, as well as a set of novel data synthesis techniques that allow us to efficiently obtain a large amount of varied data for training. Our system, called Deep Speech, outperforms previously published results on the widely studied Switchboard Hub5'00, achieving 16.0% error on the full test set. Deep Speech also handles challenging noisy environments better than widely used, state-of-the-art commercial speech systems.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Ng "Deep Speech: Scaling up end-to-end speech recognition", arXiv:1412.5567v2, 2014.

State-of-the-art?



- **Input**: spectrogram
- **Output**: characters (incl. space and null characters)

- No phonemes or vocabulary means no OOV words.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Ng "Deep Speech: Scaling up end-to-end speech recognition", arXiv:1412.5567v2, 2014.

State-of-the-art?

Model	SWB	CH	Full
Vesely et al. (GMM-HMM BMMI) [44]	18.6	33.0	25.8
Vesely et al. (DNN-HMM sMBR) [44]	12.6	24.1	18.4
Maas et al. (DNN-HMM SWB) [28]	14.6	26.3	20.5
Maas et al. (DNN-HMM FSH) [28]	16.0	23.7	19.9
Seide et al. (CD-DNN) [39]	16.1	n/a	n/a
Kingsbury et al. (DNN-HMM sMBR HF) [22]	13.3	n/a	n/a
Sainath et al. (CNN-HMM) [36]	11.5	n/a	n/a
Soltau et al. (MLP/CNN+I-Vector) [40]	**10.4**	n/a	n/a
Deep Speech SWB	20.0	31.8	25.9
Deep Speech SWB + FSH	12.6	**19.3**	**16.0**

Table 3: Published error rates (%WER) on Switchboard dataset splits. The columns labeled "SWB" and "CH" are respectively the easy and hard subsets of Hub5'00.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Ng "Deep Speech: Scaling up end-to-end speech recognition", arXiv:1412.5567v2, 2014.

Summary

- We've seen how to:
  - extract useful speech features with **Mel-scale filter banks**
  - cluster multi-modal speech data with **Gaussian mixture models.**
  - recognize speech with **hidden Markov models** and **neural networks**.
  - Recognize speech using only **end-to-end** neural networks.
  - evaluate ASR performance with **Levenshtein** distance.

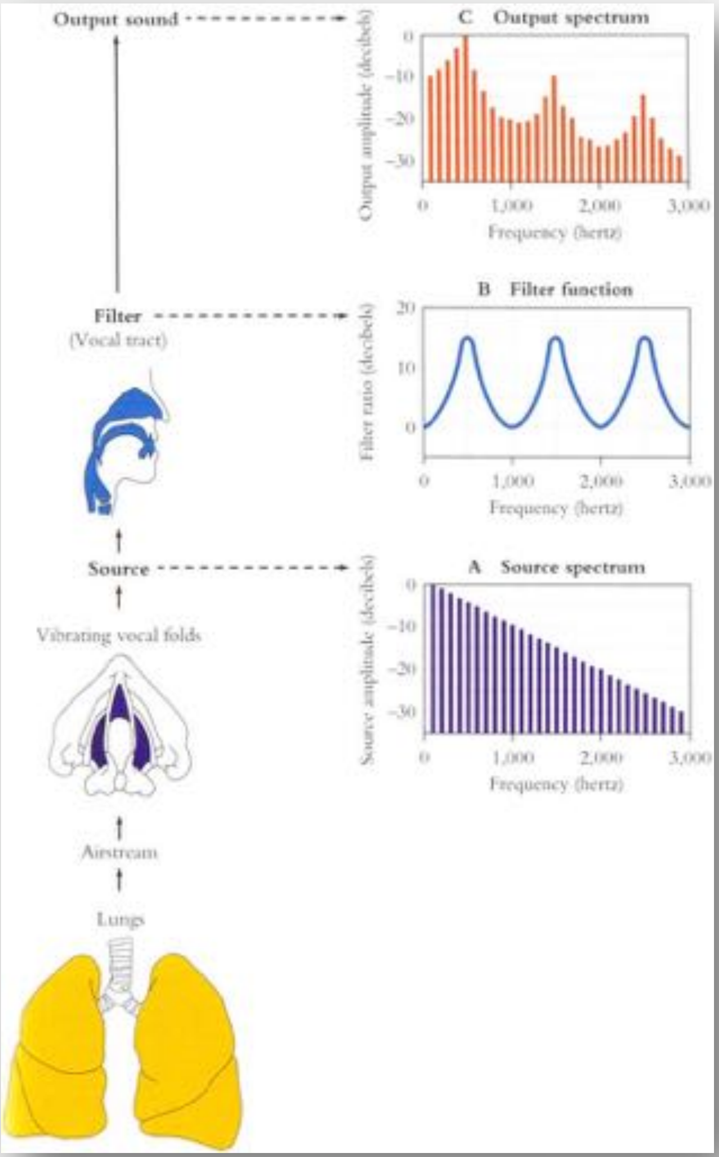
- Next, we'll see how to **synthesize** artificial speech.

APPENDICES

(EVERYTHING THAT FOLLOWS IS AN *ASIDE*. NOT ON THE EXAM.)

APPENDIX: CEPSTRUM AND MFCCS

Source and filter

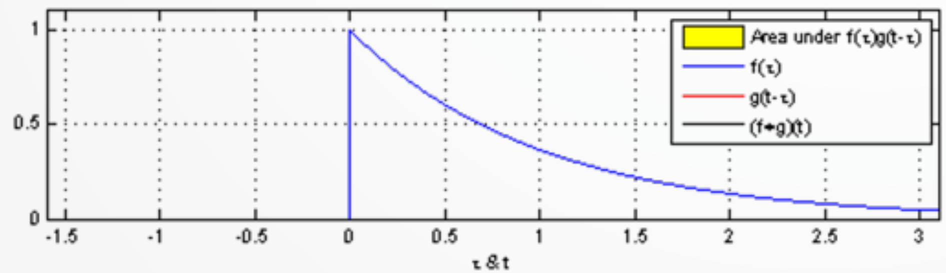


- The **acoustics** of speech are produced by a glottal pulse waveform (the **source**) passing through a vocal tract whose shape modifies that wave (the **filter**).
- The **shape** of the vocal tract is more important to phoneme recognition.
  - ***We want to separate the source from the filter in the acoustics.***

UNIVERSITY OF TORONTO

Source and filter

- Since speech is assumed to be the output of a linear time invariant system, it can be described as a **convolution**.
  - **Convolution**, $x * y$, is beyond the scope of this course, but can be conceived as the modification of one signal by another.



  - For **speech signal** $x[n]$, **glottal signal** $g[n]$, and **vocal tract transfer** $v[n]$ with **spectra** $X[z]$, $G[z]$, and $V[z]$, respectively :

$$x[n] = g[n] * v[n]$$
$$X[z] = G[z]V[z]$$
$$\log X[z] = \log G[z] + \log V[z]$$

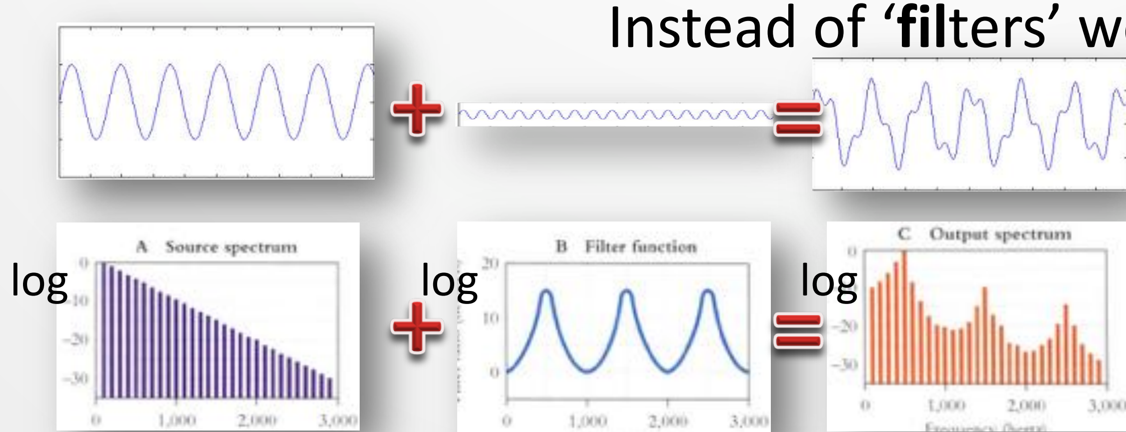
We've separated the source and filter into two terms!

UNIVERSITY OF TORONTO

The cepstrum

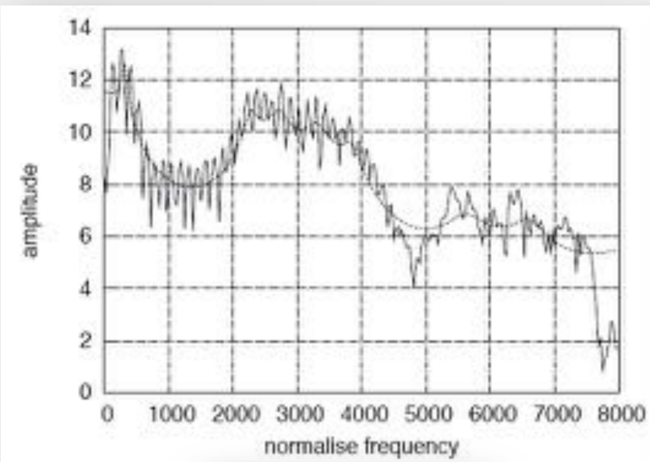
- We **separate** the **source** and the **filter** by *pretending* the log of the **spectrum** is actually a *time domain* signal.
  - the log spectrum $\log X[z]$ is a **sum** of the log spectra of the **source** and **filter**, i.e., a **superposition**;
    finding *its* spectrum will allow us to **isolate** these components.

- **Cepstrum**: *n.* the spectrum of the log of the spectrum.
  - Fun fact: 'ceps' is the reverse of 'spec'.
    Instead of '**fil**ters' we have '**lif**ters'…

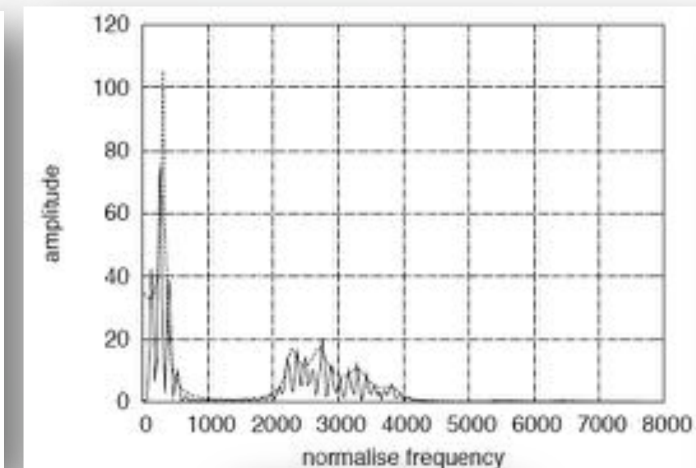


UNIVERSITY OF TORONTO

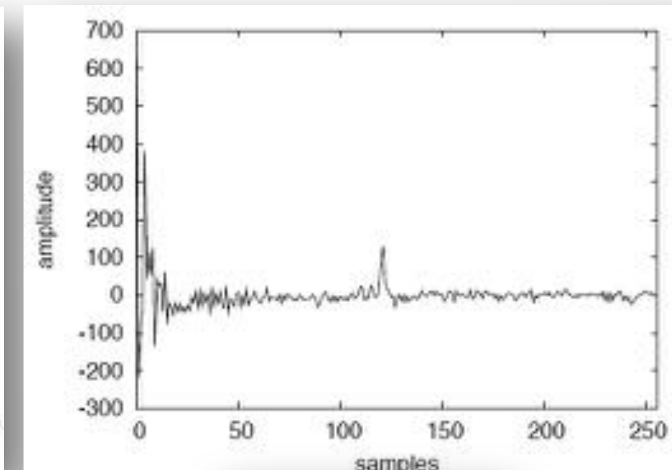
The cepstrum



Spectrum



Log spectrum



Cepstrum

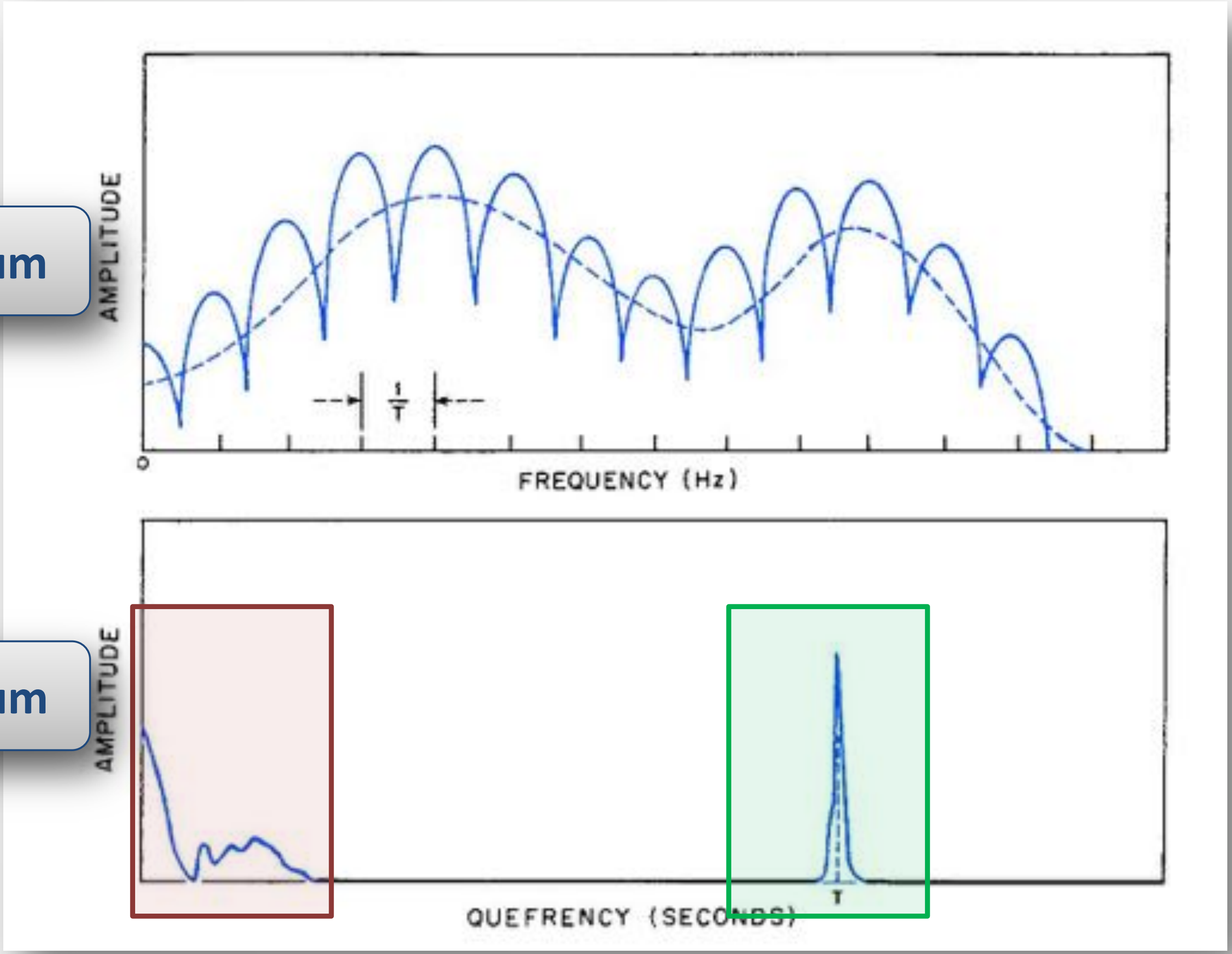
- The domain of the cepstrum is **quefrency** (a play on the word ‘*fre<u>que</u>ncy*’).

UNIVERSITY OF TORONTO

The cepstrum

Spectrum

Cepstrum



This is due to the vocal tract shape

This is due to the glottis

Pictures from John Coleman (2005)

Mel-frequency cepstral coefficients

- **Mel-frequency cepstral coefficients (MFCCs)** are a popular representation of speech used in ASR.
  - They are the **spectra** of the logarithms of the **Mel-scaled filtered spectra** of the **windows** of the **waveform**.

UNIVERSITY OF TORONTO

MFCCs in practice

- An observation vector of MFCCs often consists of
  - The **first 13 cepstral coefficients** (i.e., the first 13 dimensions produced by this method),
  - An additional **overall energy** measure,
  - The **velocities** (δ) of each of those 14 dimensions,
    - i.e., the rate of change of each coefficient at a given time
  - The **accelerations** ($\delta\delta$) of each of original 14 dimensions.

- The result is that at a timeframe t we have an observation MFCC vector of (13+1)·3 = 42 dimensions.
  - This vector is what is used by our ASR systems…

Advantages of MFCCs

- The cepstrum produces **highly uncorrelated features** (every dimension is useful).
  - This includes a **separation** of the **source** and **filter**.

- Historically, the cepstrum *has been* **easier to learn** than the spectrum for phoneme recognition.

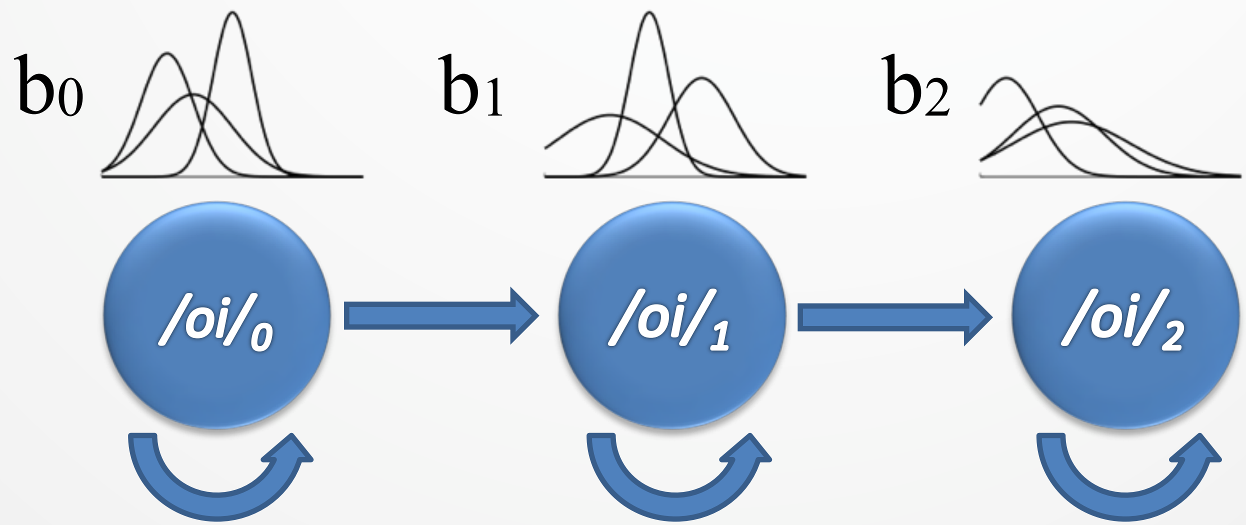
- “tl;dr: Use Mel-scaled filter banks if the [ML] algorithm is not susceptible to highly correlated input. Use MFCCs if the [ML] algorithm is susceptible to correlated input.” - Haytham Fayek

UNIVERSITY OF TORONTO

APPENDIX: PHONEME HMMS AND COMPOSITION

Phoneme HMMs

- Phonemes *change* over time – we model these dynamics by building one HMM for *each* phoneme.
  - Tristate phoneme models are popular.
    - The centre state is often the 'steady' part.



tristate phoneme model (e.g., */oi/*)

Phoneme HMMs

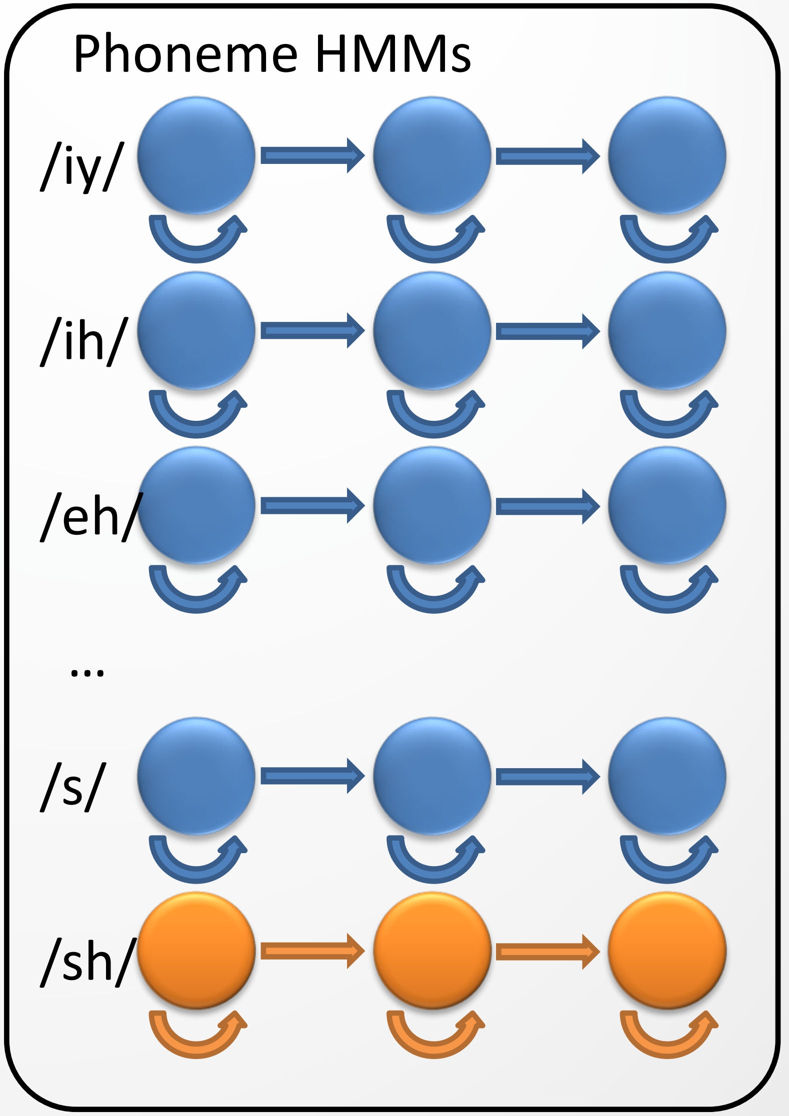
- We train each phoneme HMM using ***all*** sequences of that phoneme.

t_1	t_2	phn
...		
64	85	ae
85	**96**	**sh**
96	102	epi
102	106	m
...		

annotation

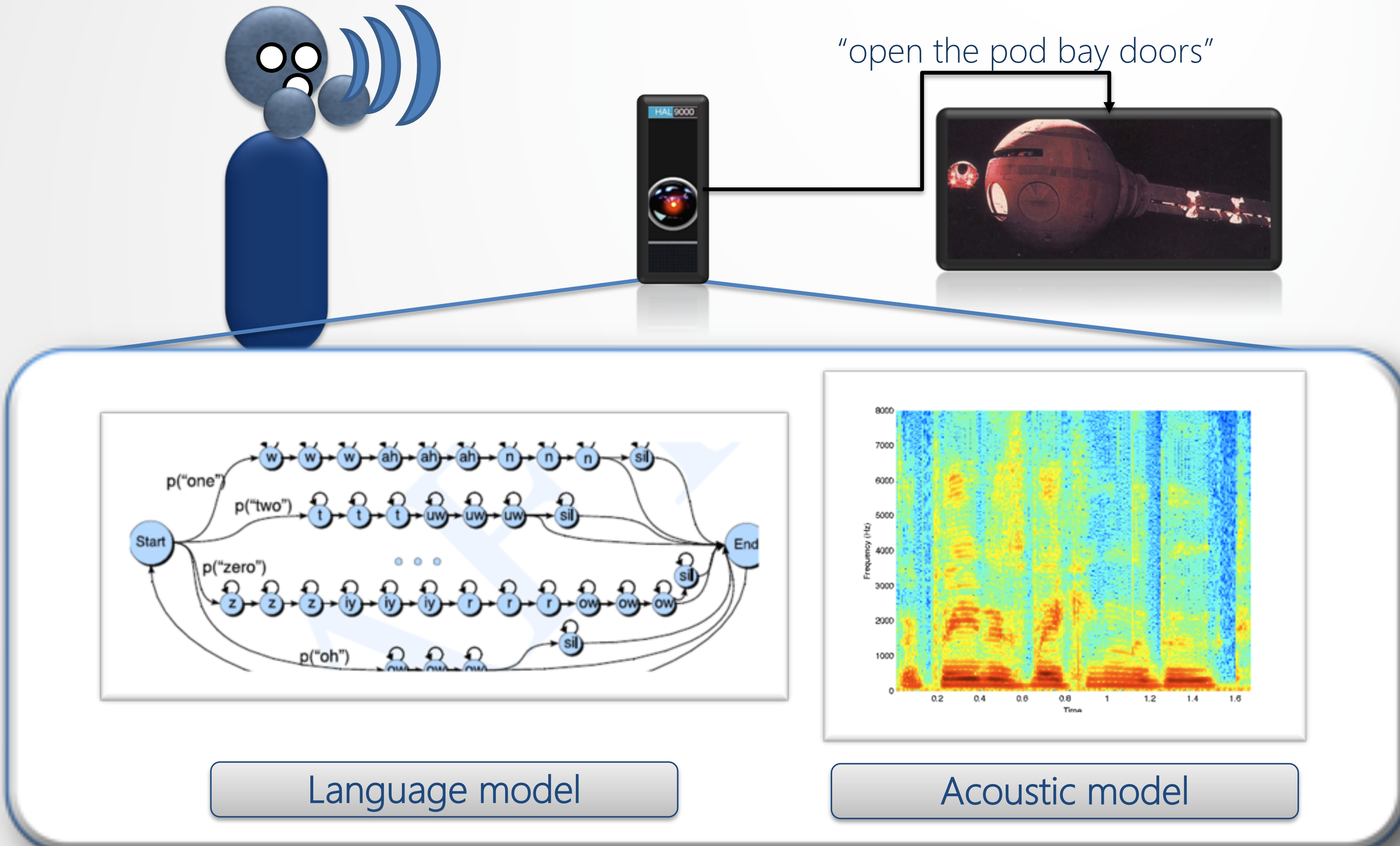
		Time, t				
		...	85	...	96	...
MFCC	1	...		...		...
	2	...		...		...
	3	...		...		...
	...	...	...	...	...	...
	42	...		...		...

observations



UNIVERSITY OF TORONTO

Putting it together



Combining models

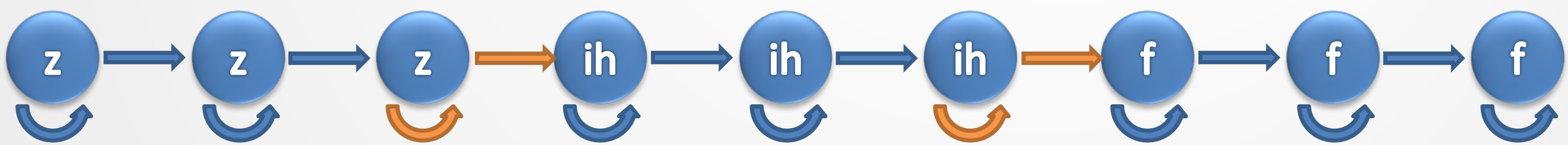
- We can learn an *N*-gram **language model** from word-level transcriptions of speech data.
  - These models are discrete and are trained using MLE.

- Our phoneme HMMs together constitute our **acoustic model**.
  - Each phoneme HMM tells us how a phoneme 'sounds'.

- We can **combine** these models by **concatenating** phoneme HMMs together according to a known lexicon.
  - We use a word-to-phoneme dictionary.

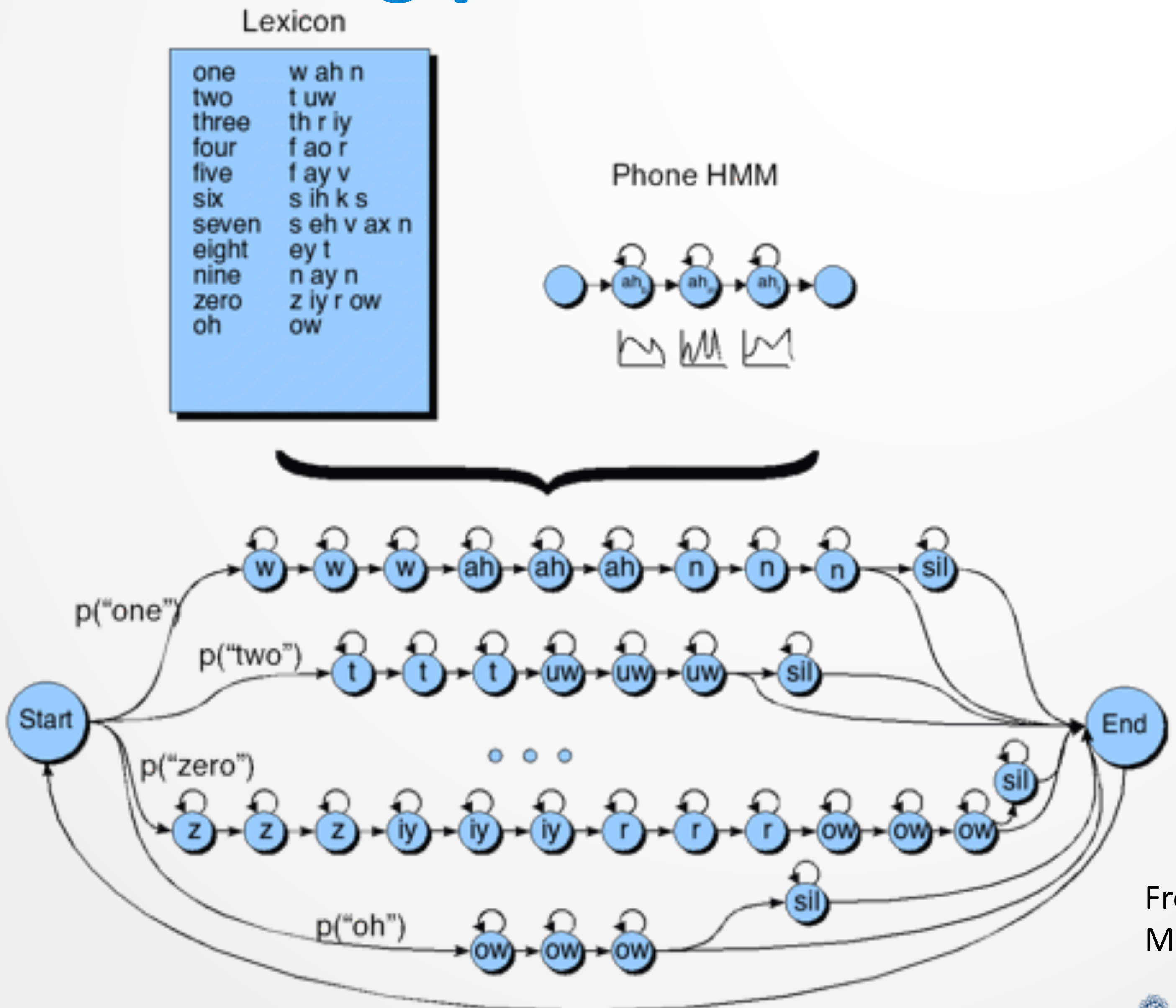
Combining models

- If we know how phonemes combine to make words, we can simply **concatenate** together our phoneme models by inserting and **adjusting** transition weights.
  - e.g., *Zipf* is pronounced */z ih f/*, so…



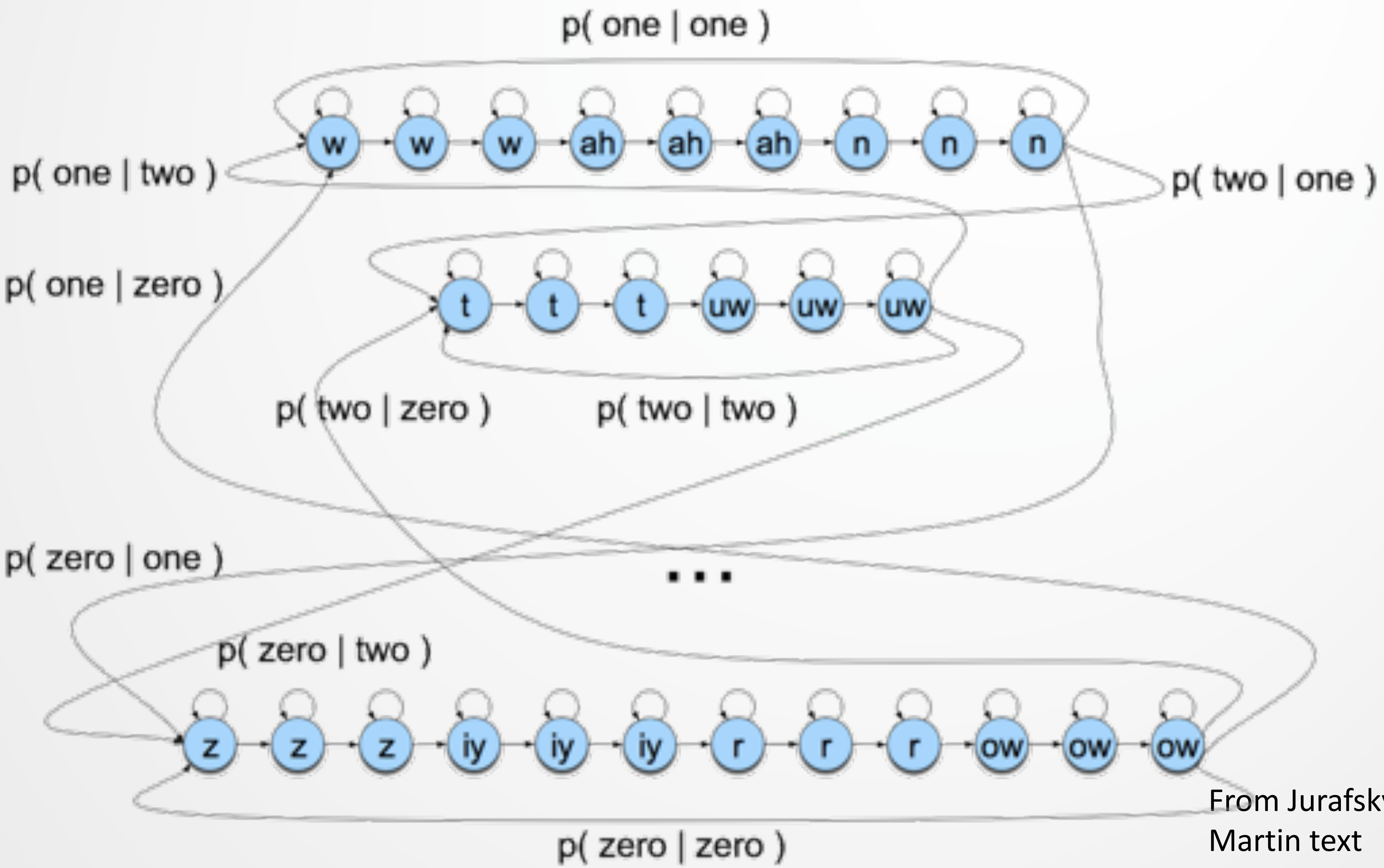
(It's more complicated: 1) the HMMs are often more complex,
2) they often represent phonemes *in context* of other phonemes
3) …)

Concatenating phoneme models



From Jurafsky & Martin text

Bigram models

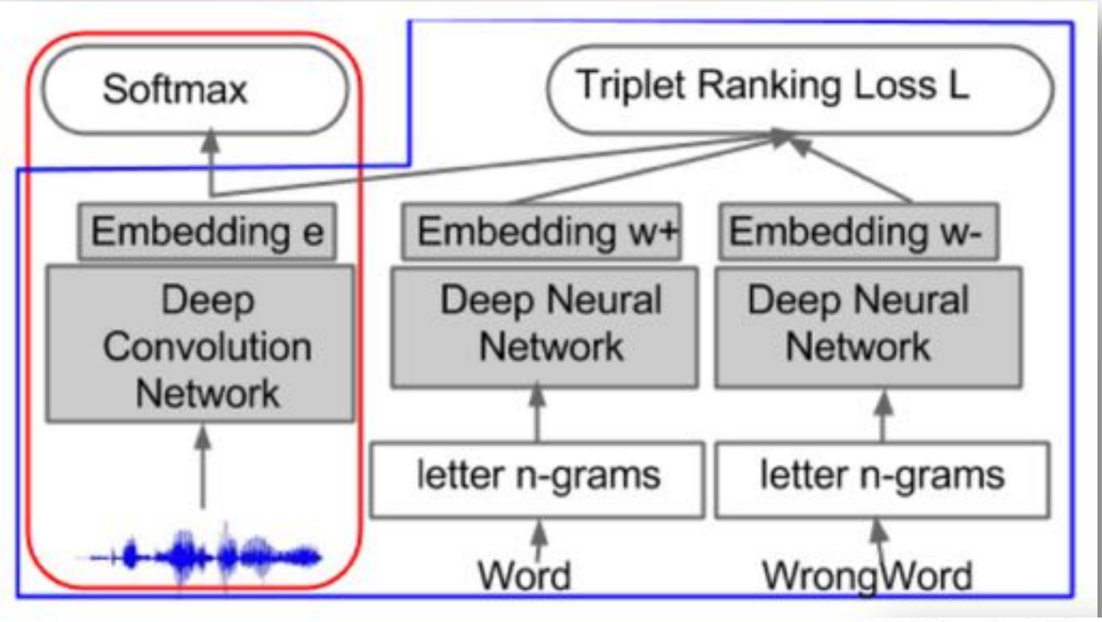


From Jurafsky & Martin text

UNIVERSITY OF TORONTO

APPENDIX: OTHER NEURAL ARCHITECTURES AND IMPLEMENTATIONS

End-to-end hybrids



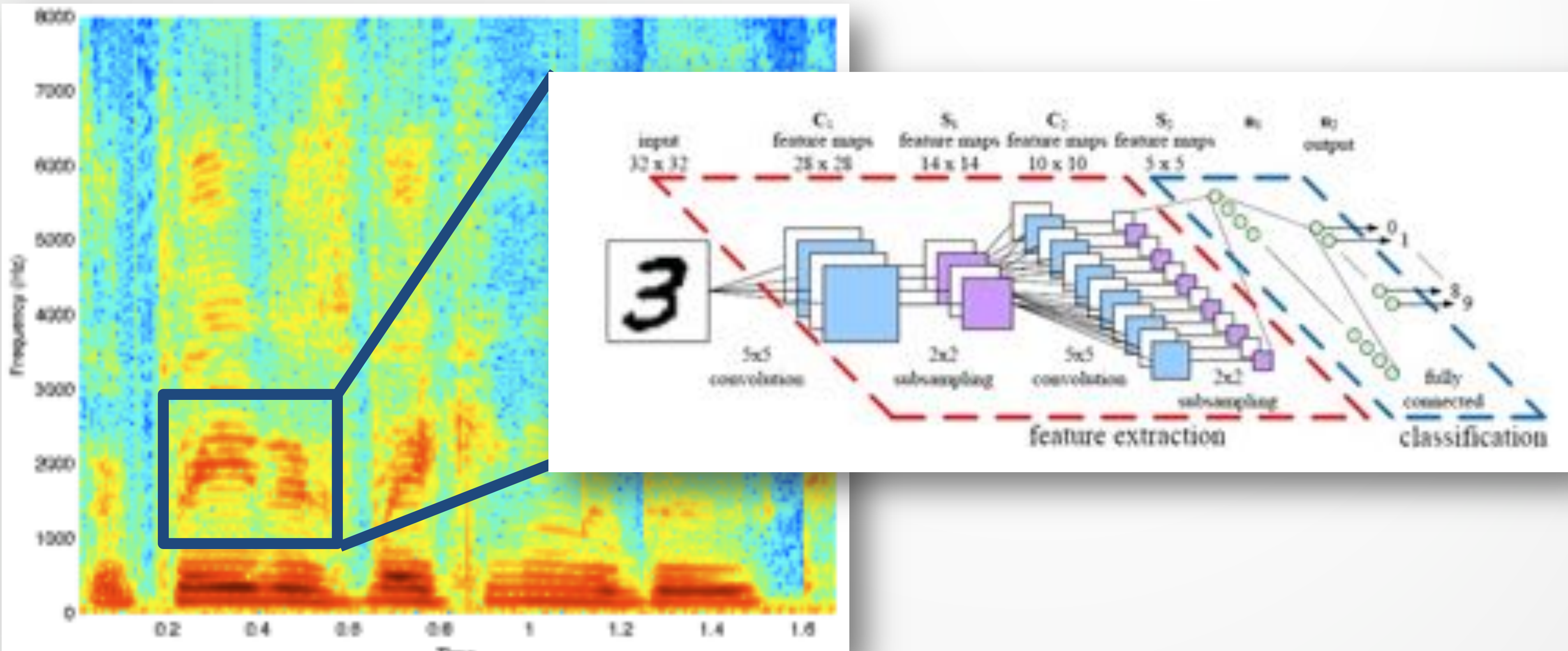
- Get word boundaries from some external tool.
- Train word/characters and acoustics simultaneously.
- Obtain up to 0.11% improvement in error rates

Table 2: *Word Error Rates for the three compared models, with two different values of the beam search parameter.*

Model	WER	
	beam=11	beam=15
Baseline	10.16	9.70
Word embedding model	11.2	11.1
Combination	10.07	9.59

Bengio, S., & Heigold, G. (2014). Word Embeddings for Speech Recognition, *Interspeech*

Convolutional Neural Networks



- Spectrograms are kinds of images, so let's use the kinds of neural networks used in computer vision.

UNIVERSITY OF TORONTO

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.
  - My favourite incarnation uses I-Vectors to account for the speaker.
  - 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.

Keeping up

LAS, Transformers, and the RNN-T (extends CTC) are reaching state of the art, e.g.

1712.01769v6 [cs.CL] 23 Feb 2018

STATE-OF-THE-ART SPEECH RECOGNITION WITH SEQUENCE-TO-SEQUENCE MODELS

Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, Michiel Bacchiani

Google, USA

{chungchengc,tsainath,yonghui,prabhavalkar,drpng,zhifengc,anjuli ronw,kanishkarao,kgonina,ndjaitly,boboli,chorowski,michiel}@google.com

ABSTRACT

Attention-based encoder-decoder architectures such as Listen, Attend, and Spell (LAS), subsume the acoustic, pronunciation and language model components of a traditional automatic speech recognition (ASR) system into a single neural network. In previous work, we have shown that such architectures are comparable to state-of-the-art ASR systems on dictation tasks, but it was not clear if such architectures would be practical for more challenging tasks such as voice search. In this work, we explore a variety of structural and optimization improvements to our LAS model which significantly improve performance. On the structural side, we show that word piece models can be used instead of graphemes. We also introduce a multi-head attention architecture, which offers improvements over the commonly-used single-head attention. On the optimization side, we explore synchronous training, scheduled sampling, label smoothing, and minimum word error rate optimization, which are all shown to improve accuracy. We present results with a unidirectional LSTM encoder for streaming recognition. On a 12, 500 hour voice search task, we find that the proposed changes improve the WER from 9.2% to 5.6%, while the best conventional system achieves 6.7%; on a dictation task our model achieves a WER of 4.1% compared to 5% for the conventional system.

on a large vocabulary continuous speech recognition (LVCSR) task. The goal of this paper is to explore various structure and optimization improvements to allow sequence-to-sequence models to significantly outperform a conventional ASR system on a voice search task.

Since previous work showed that LAS offered improvements over other sequence-to-sequence models [6], we focus on improvements to the LAS model in this work. The LAS model is a single neural network that includes an *encoder* which is analogous to a conventional acoustic model, an *attender* that acts as an alignment model, and a *decoder* that is analogous to the language model in a conventional system. We consider both modifications to the model structure, as well as in the optimization process. On the structure side, first, we explore word piece models (WPM) which have been applied to machine translation [7] and more recently to speech in RNN-T [8] and LAS [9]. We compare graphemes and WPM for LAS, and find modest improvement with WPM. Next, we explore incorporating multi-head attention [10], which allows the model to learn to attend to multiple locations of the encoded features. Overall, we get 13% relative improvement in WER with these structure improvements.

On the optimization side, we explore a variety of strategies as well. Conventional ASR systems benefit from discriminative sequence training, which optimizes criteria more closely related to

https://arxiv.org/pdf/1712.01769.pdf

For **HOT** news and architectures, see https://github.com/syhw/wer_are_we

APPENDIX: SPEAKER ADAPTATION

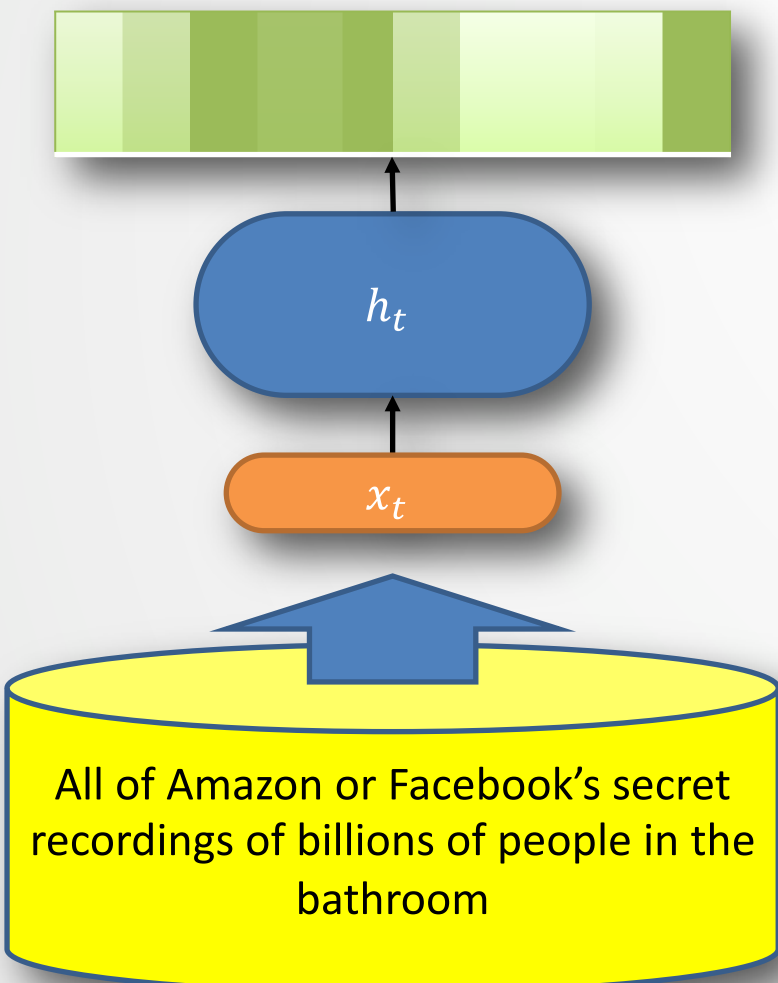
Speaker adaptation

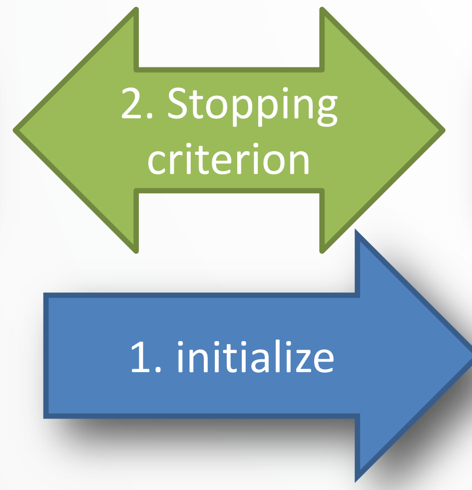
- Given a neural ASR system trained with many speakers, we want to adapt to the voice of a new individual.
- We know how to do this with HMMs
  - e.g., with interpolation, or (aside) with MAP or MLLR training.

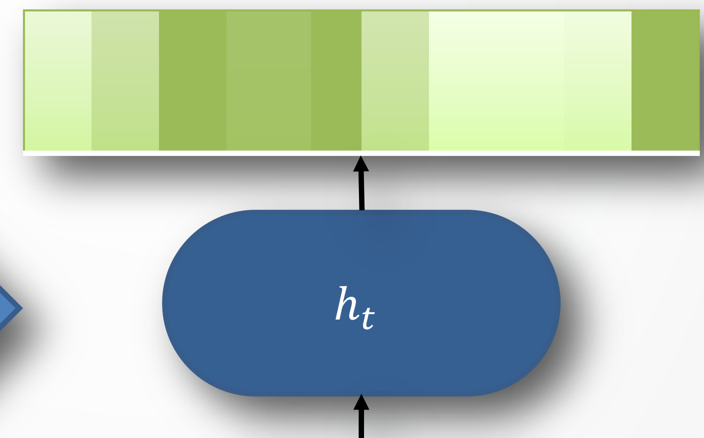
- DNNs need *lots* of data to be useful, but we can adapt:
  - **Conservative**: re-train whole DNN, with some constraints
  - **Transformative**: only retrain one layer (or a few)
  - **Speaker-aware**: do not really train the parameters

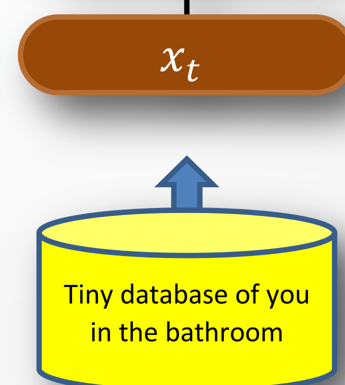
With notes from Hung-yi Lee

Conservative speaker adaptation





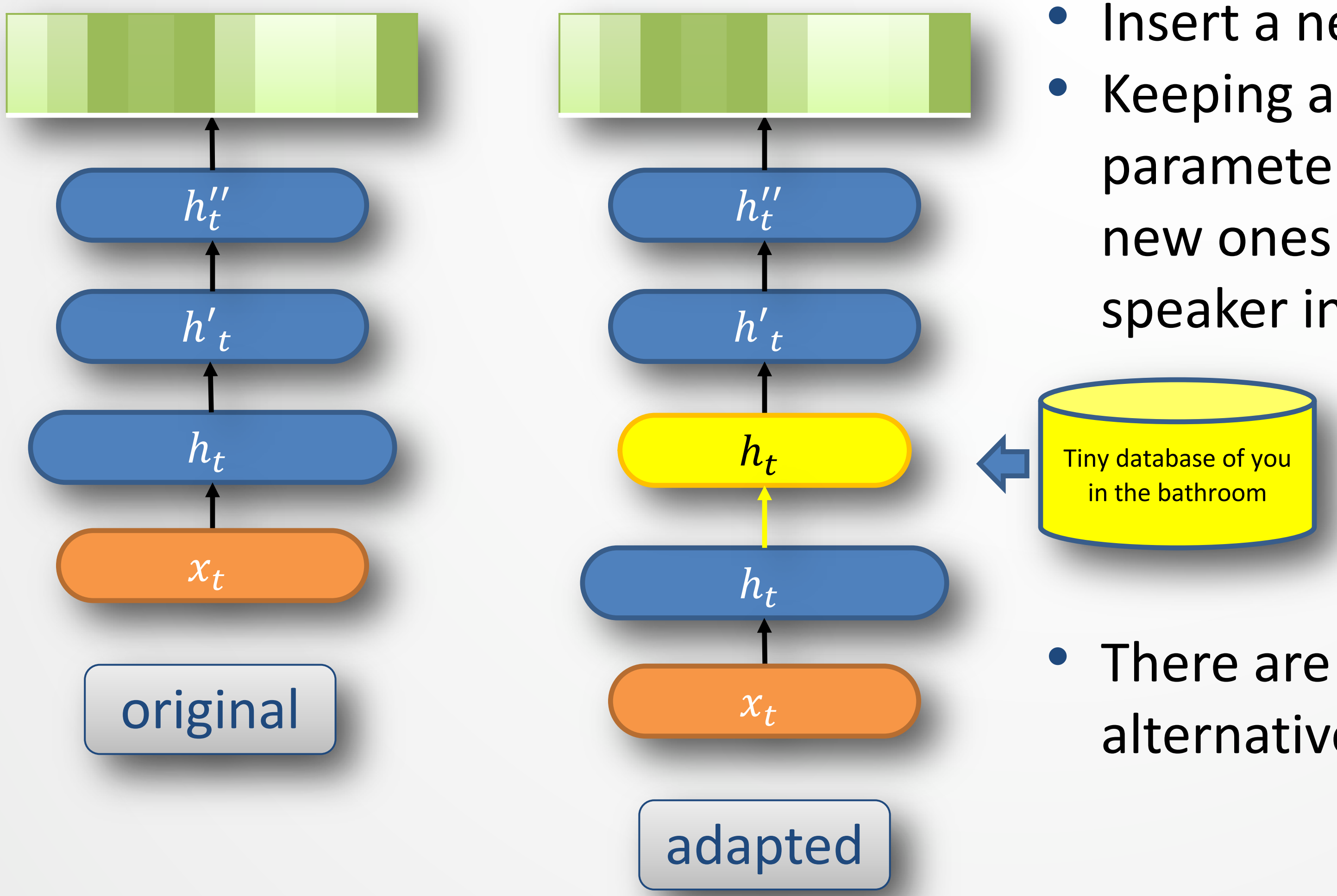




- Stopping criteria can exist on output, parameters, or meta-aspects of training

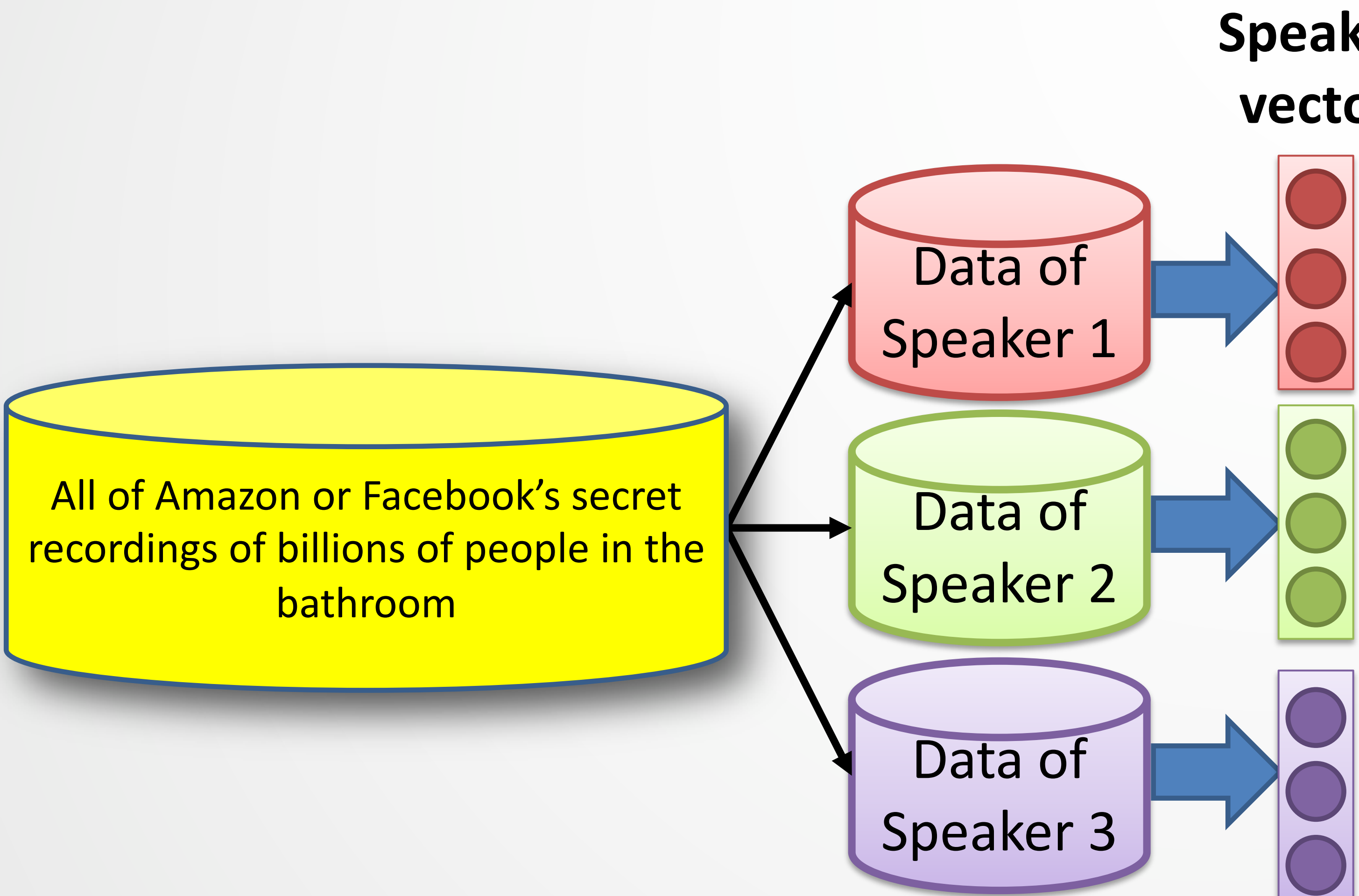
UNIVERSITY OF TORONTO

Transformative speaker adaptation



- Insert a new layer.
- Keeping all other parameters fixed, train the new ones to normalize speaker information.
- There are many alternatives…

Speaker-aware training



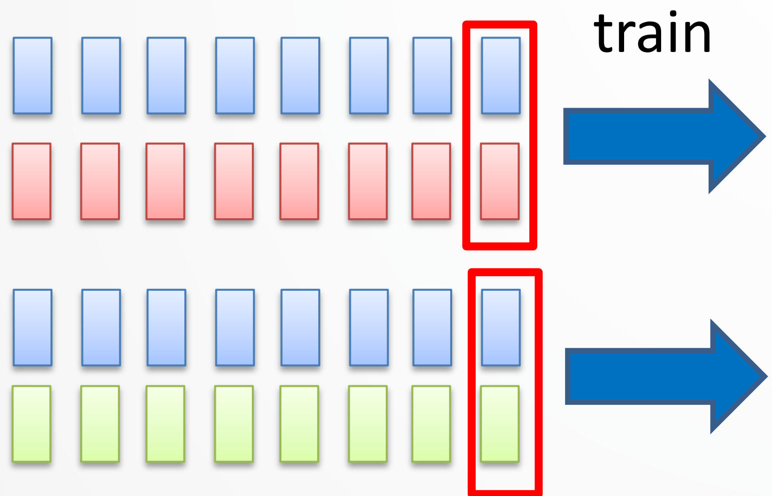
- Fixed length low dimension vectors, obtained in a variety of ways.
- Note we can segment things by recording device, noise, etc.
- This can be used to remove the **channel effect.**

Senior, A., & Lopez-Moreno, I. (2014). Improving DNN speaker independence with I-vector inputs. ICASSP, 225–229. https://doi.org/10.1109/ICASSP.2014.6853591

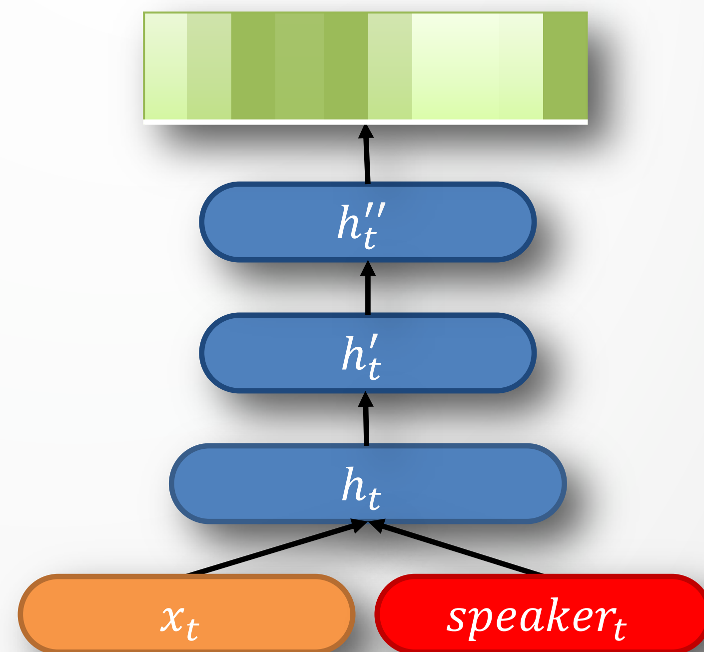
Speaker-aware training

Training data:

Speaker 1



Speaker 2



Acoustic features augmented with speaker vectors

Testing data:

All speakers use the same DNN model

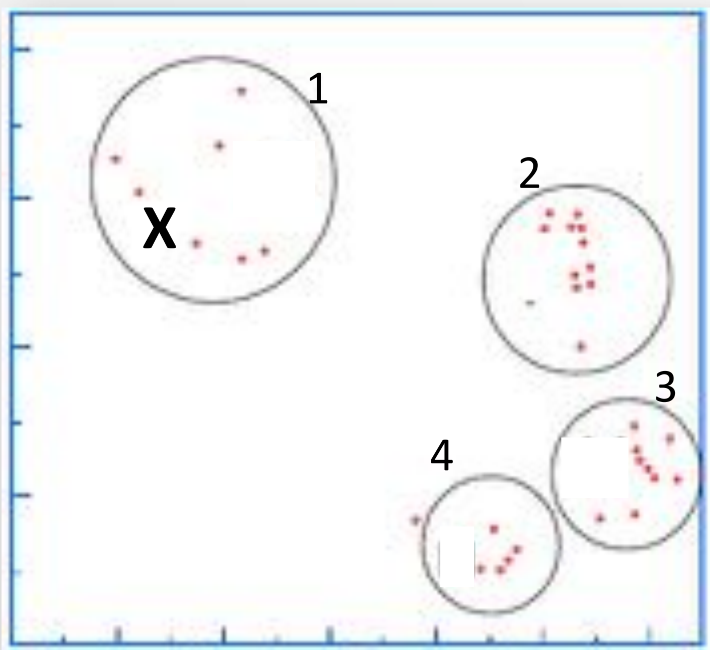
Different speakers augmented by different features

UNIVERSITY OF TORONTO

APPENDIX: CLUSTERING

Clustering

- **Quantization** involves turning possibly **multi-variate** and **continuous** representations into **univariate discrete** symbols.
  - Reduced storage and computation costs.
  - Potentially tremendous loss of information.



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

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

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

     $$x \in C_i \quad if \quad d(x, \mu_i) \leq d(x, \mu_j), \quad \forall i \neq j$$

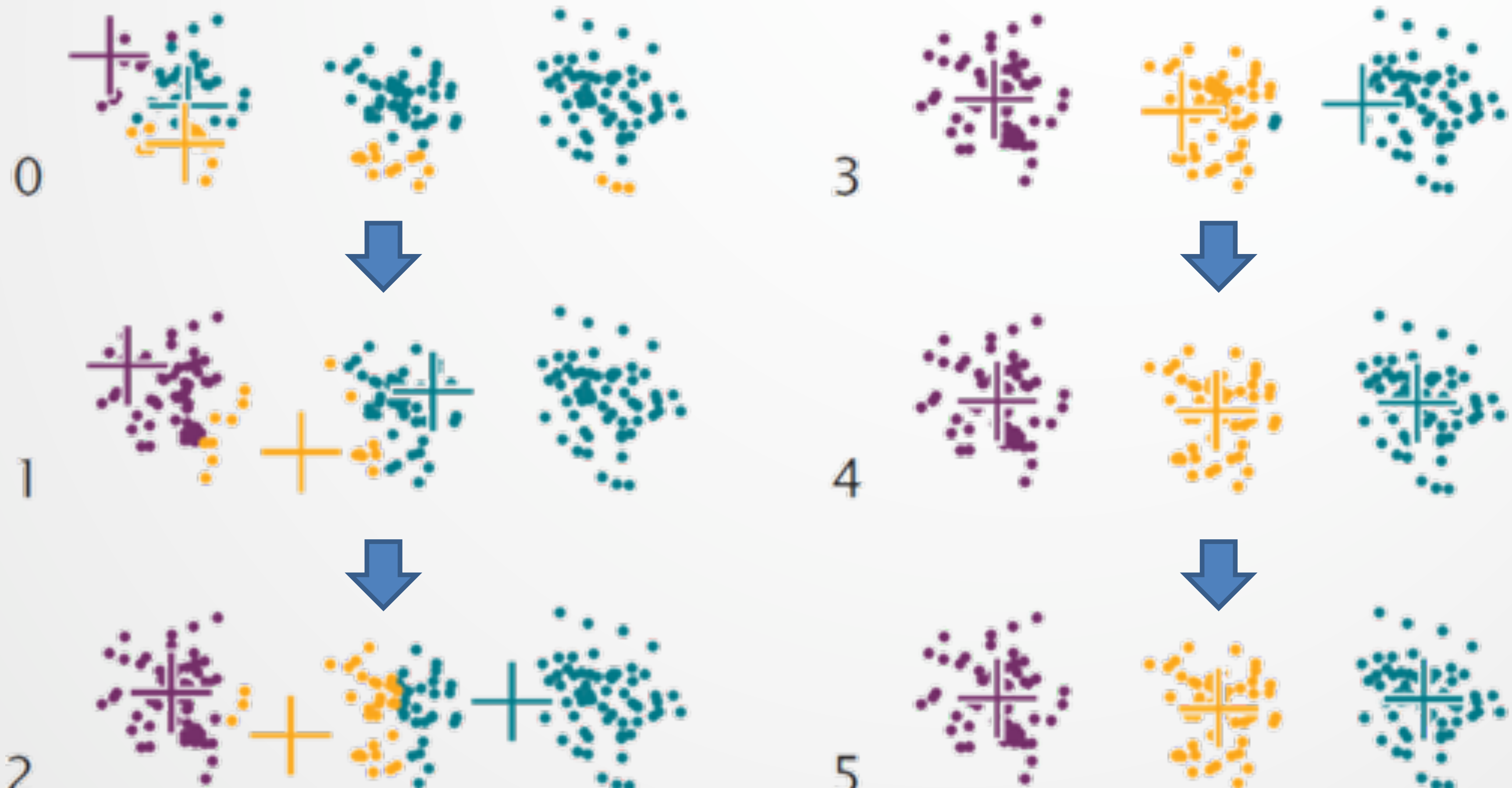
     b) **Update** K means from assigned samples

     $$\mu_i = E(x) \; \forall \, x \in C_i, \qquad 1 \leq i \leq K$$

UNIVERSITY OF TORONTO

K-means example ($K = 3$)

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



K-means stopping condition

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

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

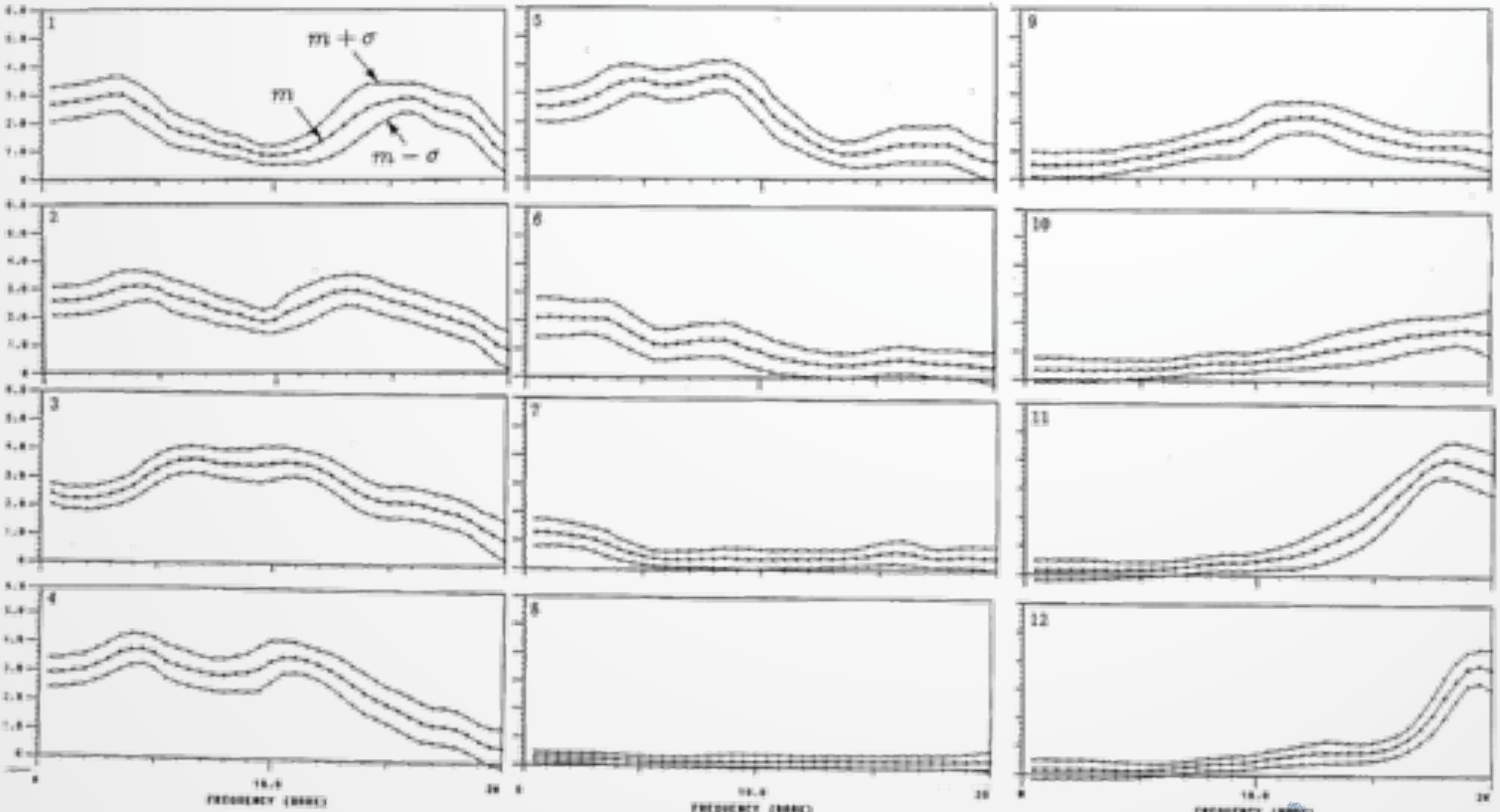
- $\mathcal{D}$ decreases between n^{th} and $(n+1)^{th}$ iteration.

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

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

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…

$K = 2$

$K = 4$

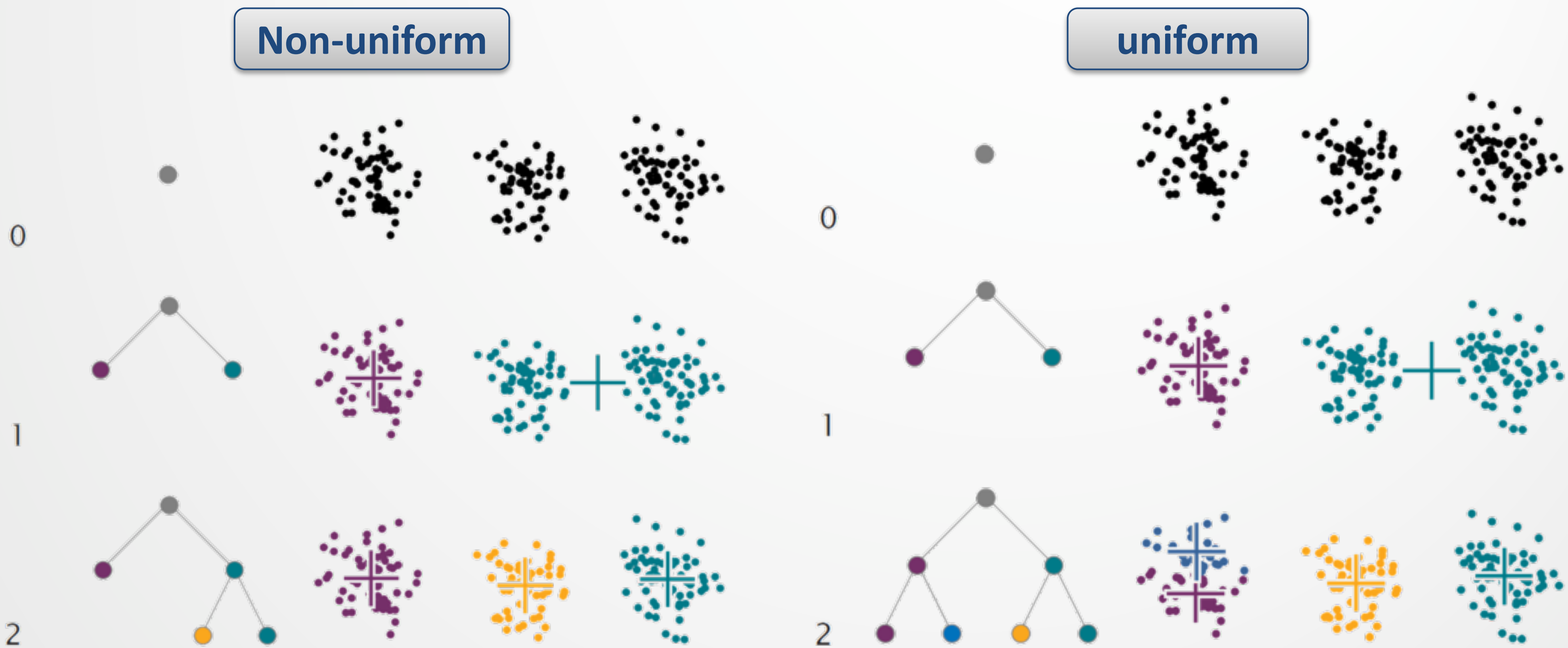
UNIVERSITY OF TORONTO

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.

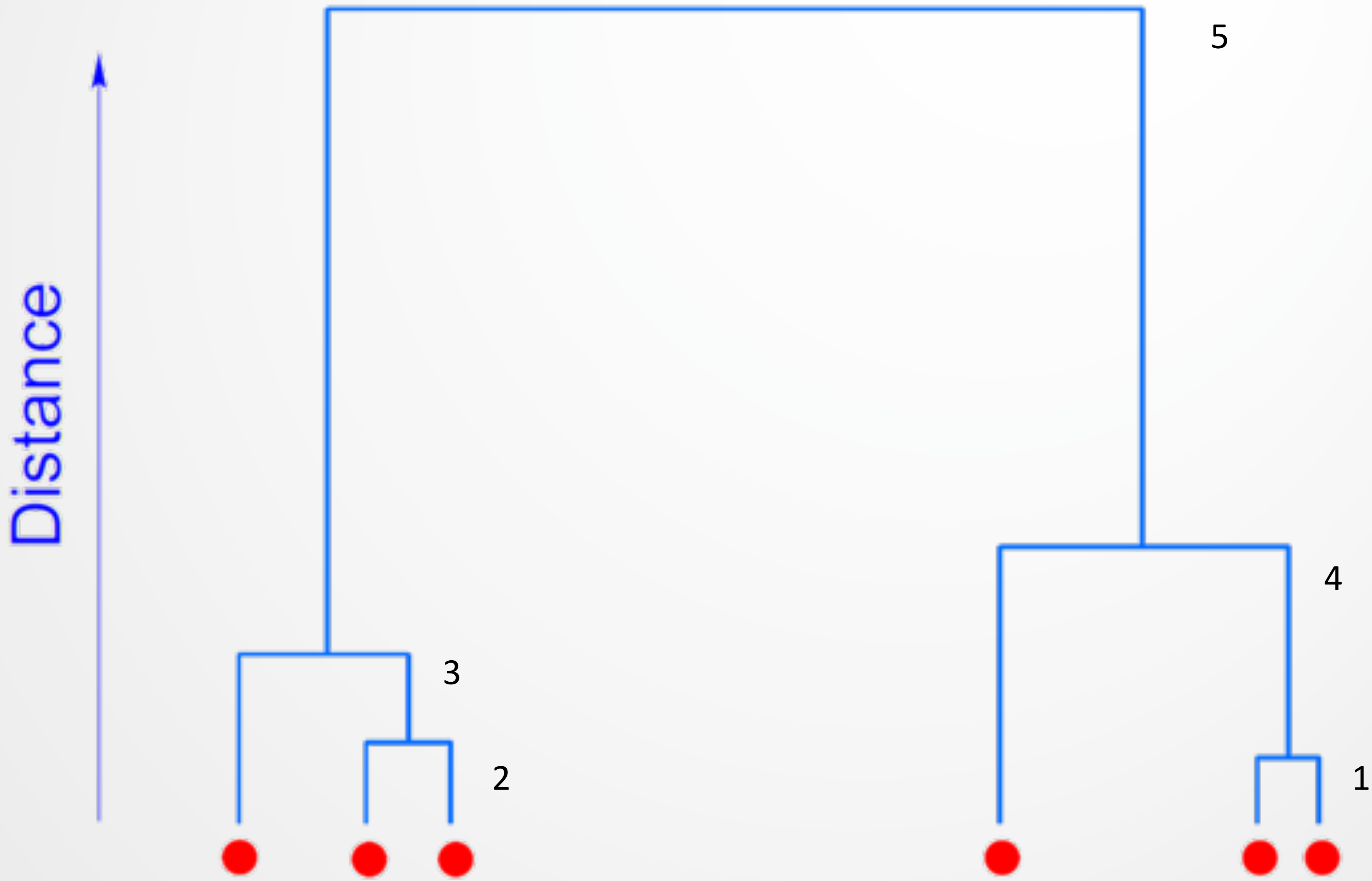


UNIVERSITY OF TORONTO

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., **dendogram**) can yield insight into the natural grouping of data.

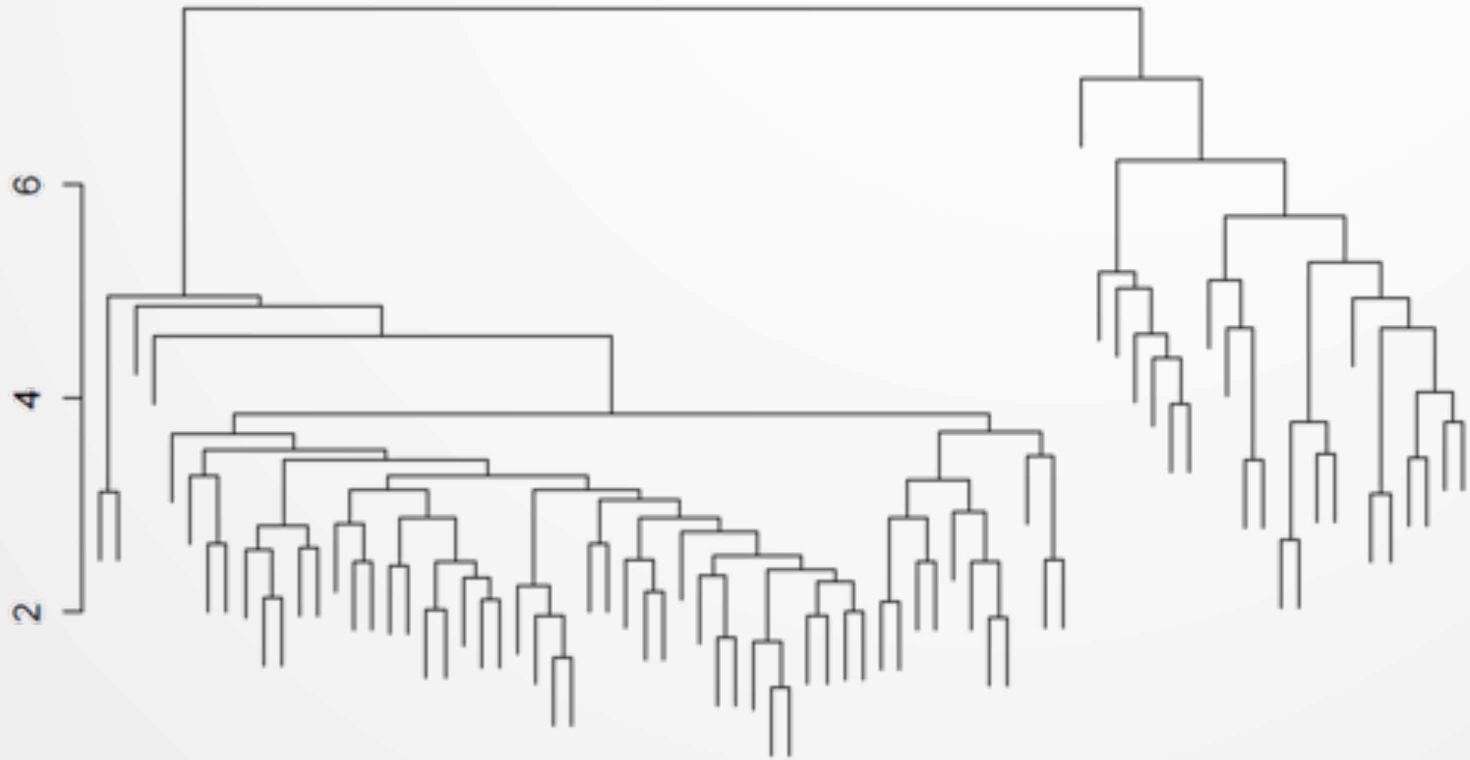
Dendogram example



UNIVERSITY OF TORONTO

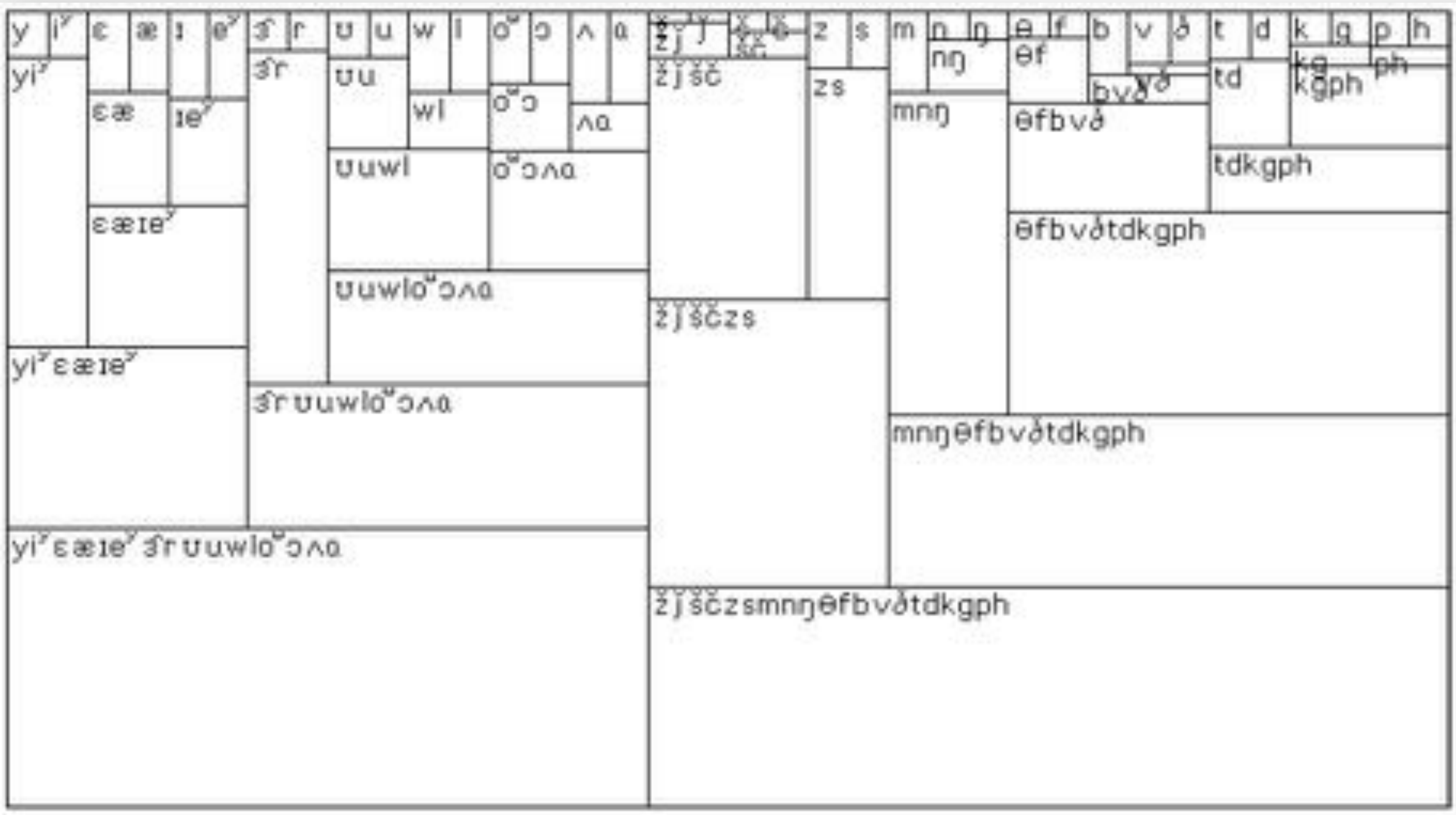
Speaker clustering

- 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)

UNIVERSITY OF TORONTO

Word clustering