ENGR-E 511; ENGR-E 399

Machine Learning for Signal Processing

Module 07:

Hidden Markov Models

Minje Kim

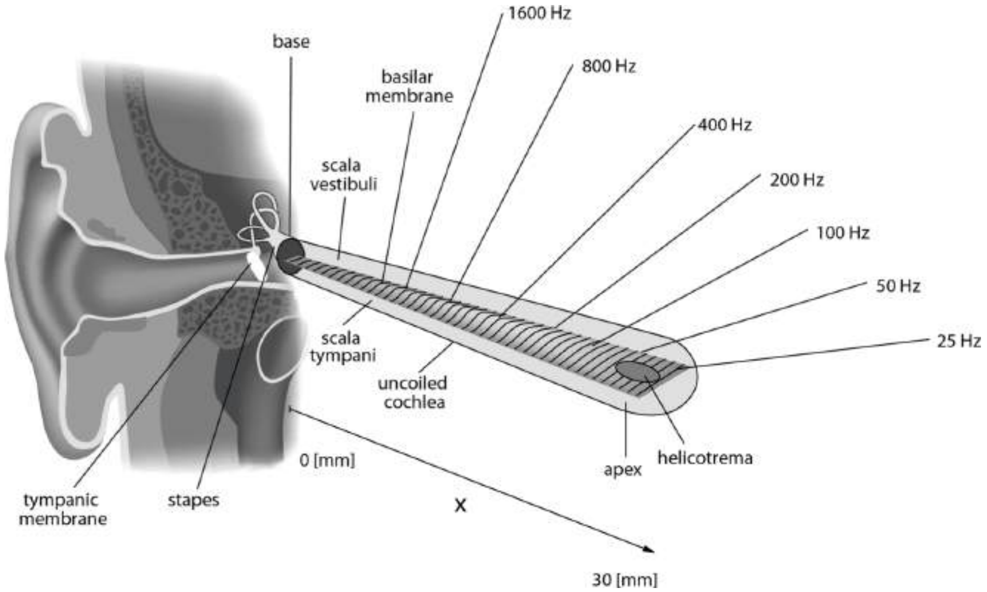
Department of Intelligent Systems Engineering

Email: minje@indiana.edu

Website: http://minjekim.com

Research Group: http://saige.sice.indiana.edu

Meeting Request: http://doodle.com/minje



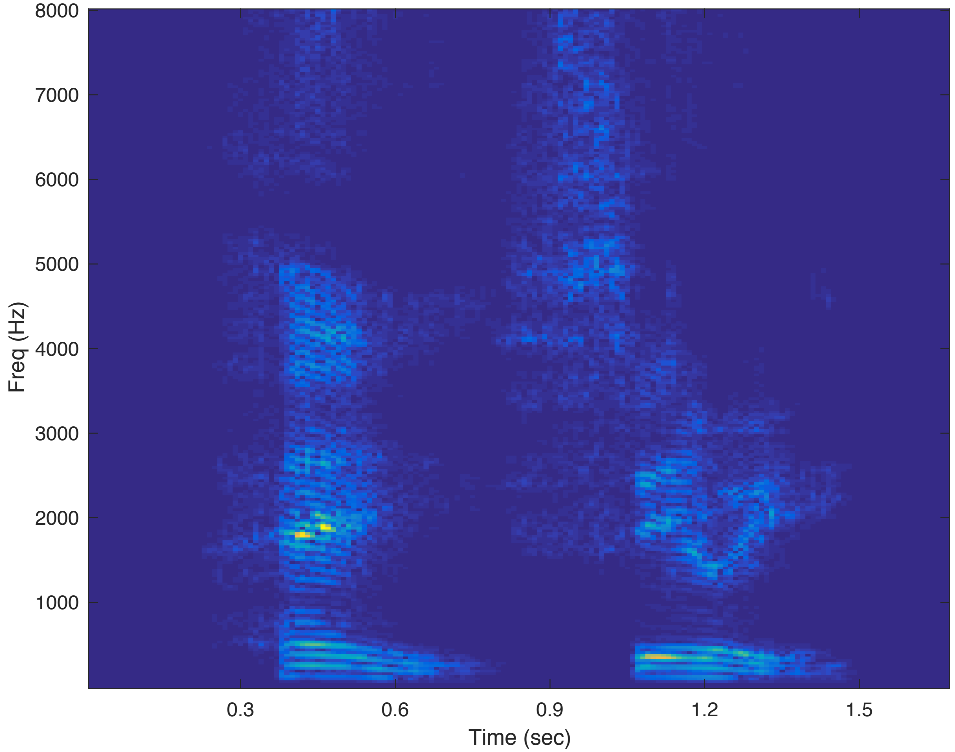
INDIANA UNIVERSITY

SCHOOL OF INFORMATICS, COMPUTING, AND ENGINEERING

Keyword Spotting

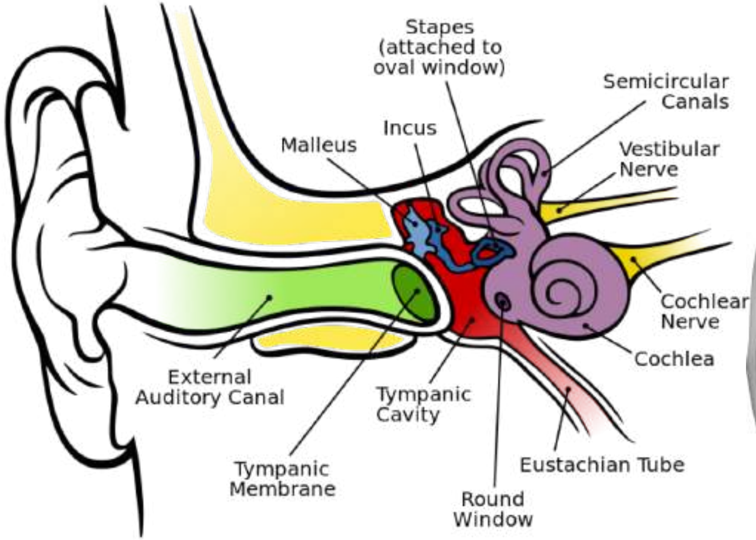
- Preprocessing

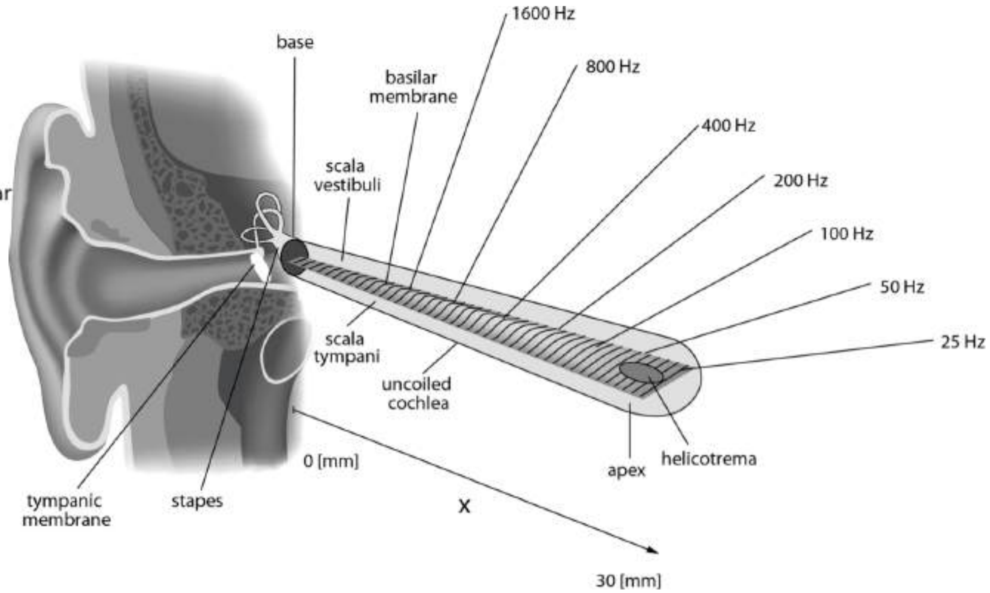
- I like Siri
  - But, she doesn't like me back
- What should I do to build my own keyword spotter?



Keyword Spotting
- Preprocessing

- Problem with STFT
  - Too many dimensions
  - Even before that, our auditory system is not linear





https://en.wikipedia.org/wiki/Electrocochleography
https://ocw.mit.edu/courses/health-sciences-and-technology/hst-730-molecular-biology-for-the-auditory-system-fall-2002/hst-730f02.jpg

Keyword Spotting

- Preprocessing

- Nonlinear frequency?

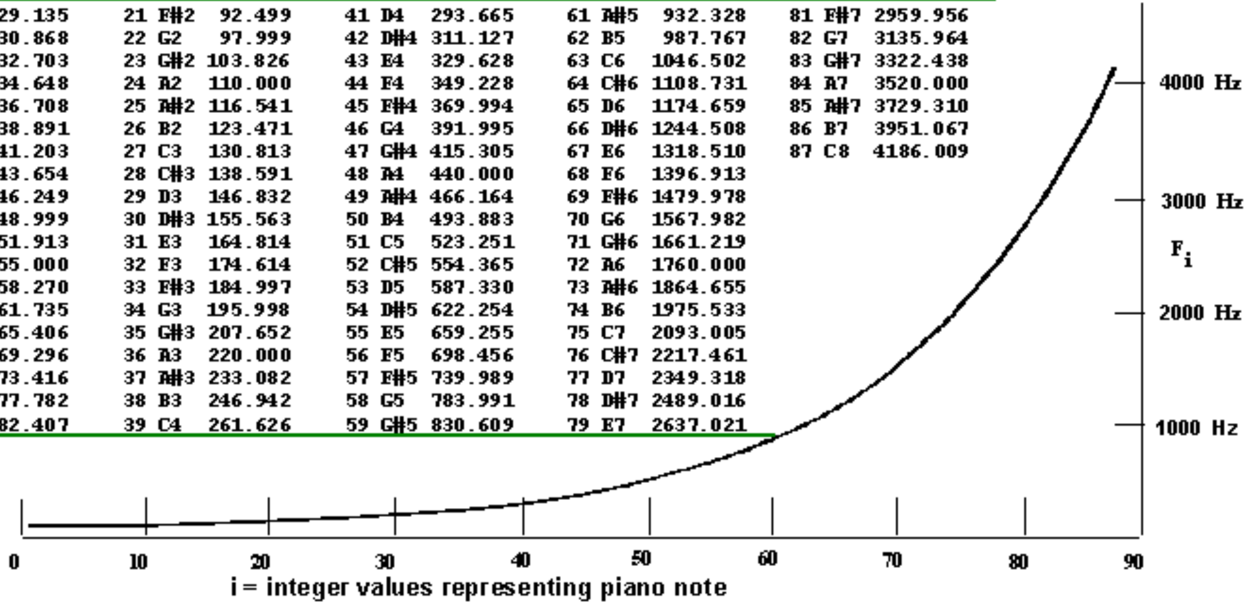
$f_1 = 622.25$ $f_2 = 698.46$ $f_3 = 783.99$

$\underbrace{f_1 \quad f_2}_{\text{Pitch difference}}$ $\underbrace{f_2 \quad f_3}_{\text{Pitch difference}}$

Pitch difference
$f_2 - f_1 = d_{21}$

Pitch difference
$f_3 - f_2 = d_{32}$

0	A0	27.500	20	F2	87.307	40	C#4	277.183	60	A5	880.000	80	F7	2793.826
1	A#0	29.135	21	F#2	92.499	41	D4	293.665	61	A#5	932.328	81	F#7	2959.956
2	B0	30.868	22	G2	97.999	42	D#4	311.127	62	B5	987.767	82	G7	3135.964
3	C1	32.703	23	G#2	103.826	43	E4	329.628	63	C6	1046.502	83	G#7	3322.438
4	C#1	34.648	24	A2	110.000	44	F4	349.228	64	C#6	1108.731	84	A7	3520.000
5	D1	36.708	25	A#2	116.541	45	F#4	369.994	65	D6	1174.659	85	A#7	3729.310
6	D#1	38.891	26	B2	123.471	46	G4	391.995	66	D#6	1244.508	86	B7	3951.067
7	E1	41.203	27	C3	130.813	47	G#4	415.305	67	E6	1318.510	87	C8	4186.009
8	F1	43.654	28	C#3	138.591	48	A4	440.000	68	F6	1396.913			
9	F#1	46.249	29	D3	146.832	49	A#4	466.164	69	F#6	1479.978			
10	G1	48.999	30	D#3	155.563	50	B4	493.883	70	G6	1567.982			
11	G#1	51.913	31	E3	164.814	51	C5	523.251	71	G#6	1661.219			
12	A1	55.000	32	F3	174.614	52	C#5	554.365	72	A6	1760.000			
13	A#1	58.270	33	F#3	184.997	53	D5	587.330	73	A#6	1864.655			
14	B1	61.735	34	G3	195.998	54	D#5	622.254	74	B6	1975.533			
15	C2	65.406	35	G#3	207.652	55	E5	659.255	75	C7	2093.005			
16	C#2	69.296	36	A3	220.000	56	F5	698.456	76	C#7	2217.461			
17	D2	73.416	37	A#3	233.082	57	F#5	739.989	77	D7	2349.318			
18	D#2	77.782	38	B3	246.942	58	G5	783.991	78	D#7	2489.016			
19	E2	82.407	39	C4	261.626	59	G#5	830.609	79	E7	2637.021			



Ideal frequencies for equally tempered 12-tone chromatic scale.

$$F_i = 2^{(i/12)} \cdot C$$

Where C = ideal frequency of A_0 = 27.5 Hz

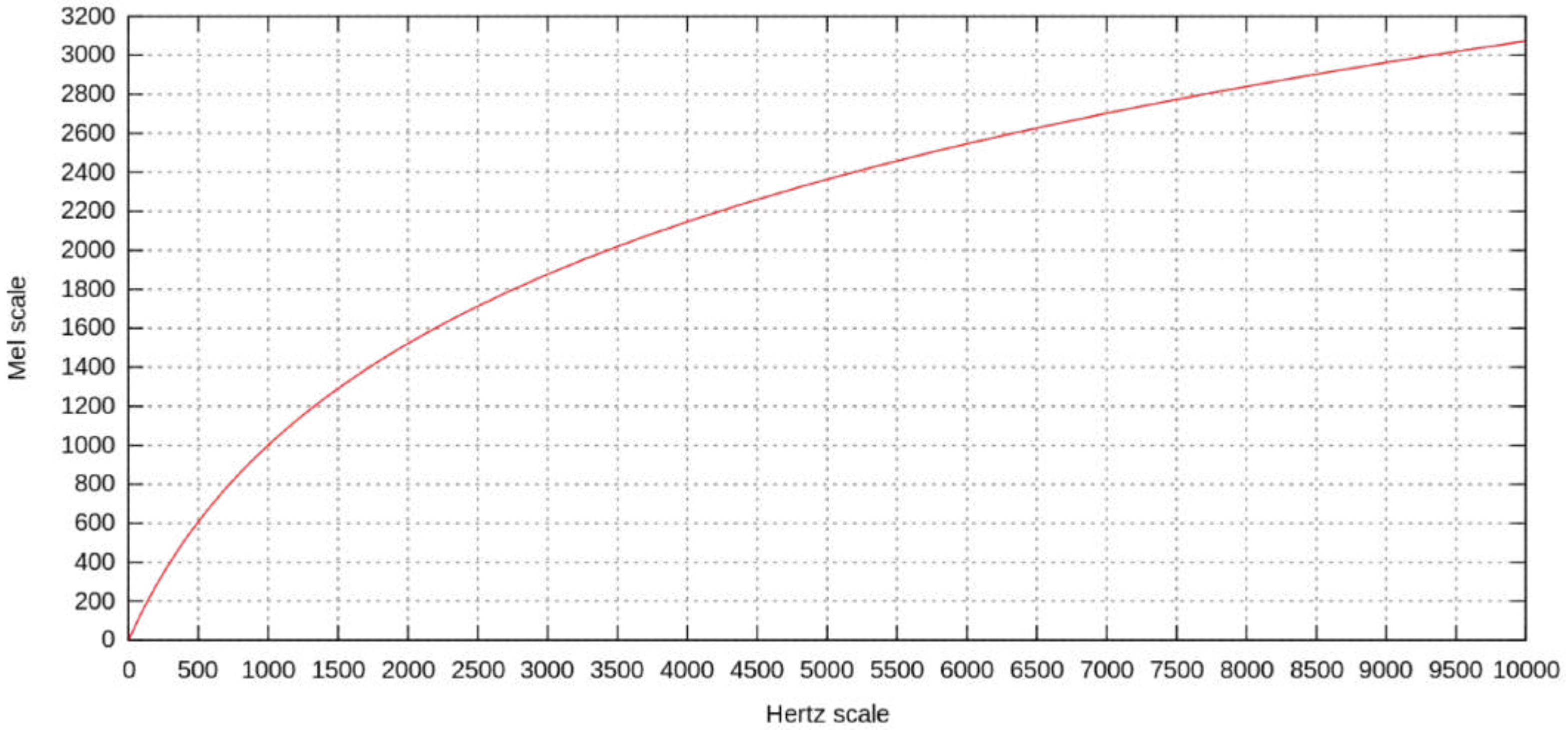
- $d_{21} = d_{32}$?
  - $d_{21} = 76.21 \neq d_{32} = 85.53$

- Human auditory system
  - More sensitive to the changes in lower freq.

Keyword Spotting
- Preprocessing

- Mel scale

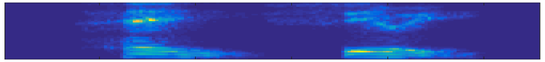
$$mel(f) = a \ln\left(1 + \frac{f}{b}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$



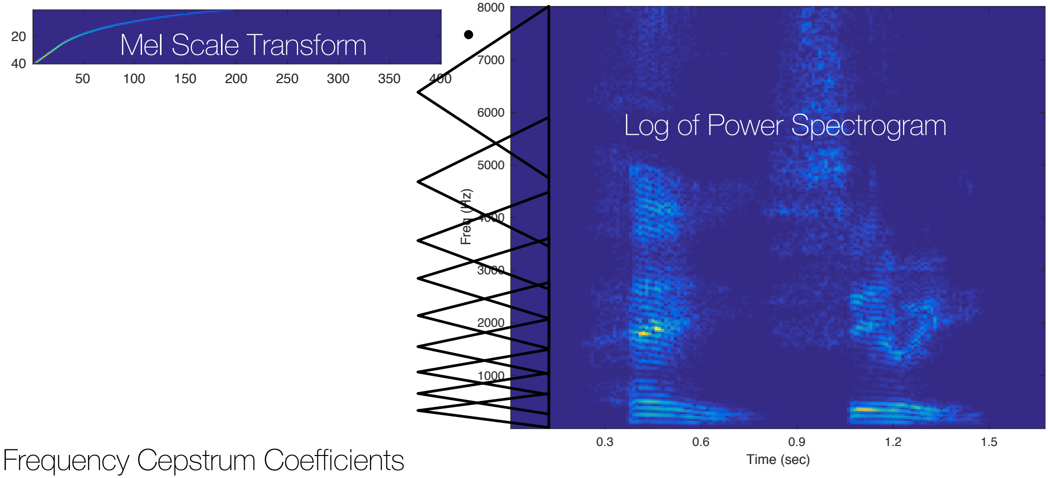
Keyword Spotting

- Preprocessing

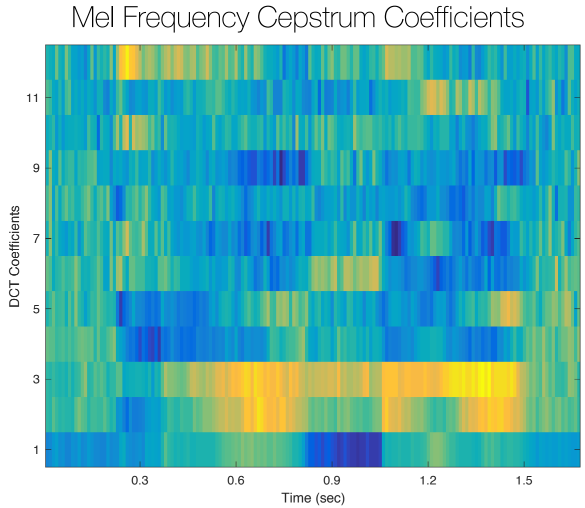
- Mel scale spectra



=



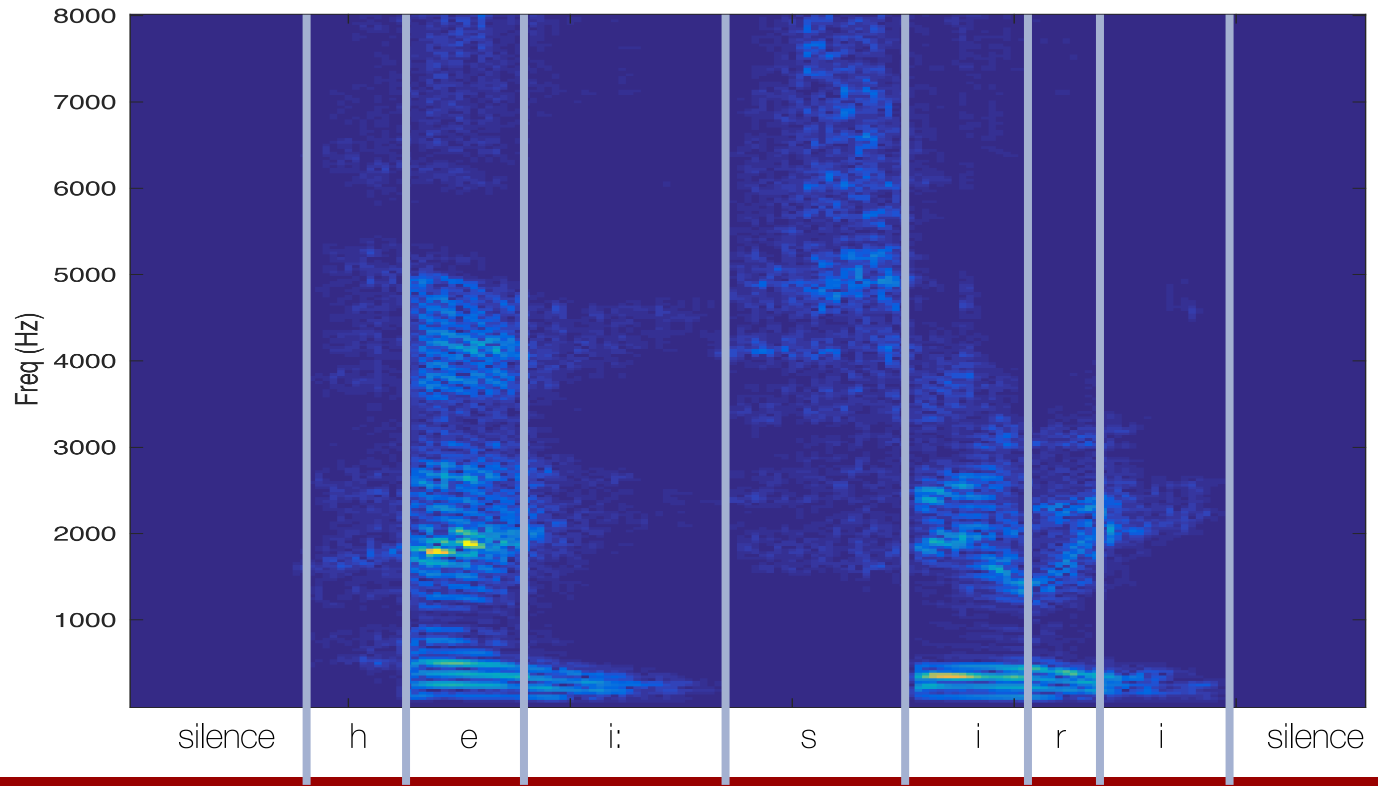
- Still too many dimensions
  - PCA?
    - Instead, DCT
- Mel Frequency Cepstrum Coefficients (MFCC)
- From now on I will use spectrogram in the following slides to visualize the speech signal
- But, actual processing was done on the MFCC matrix
  - I just don't know how to intuitively interpret MFCC



Training from a Sequence: a Preview

- Smoothing clustering with transition probabilities

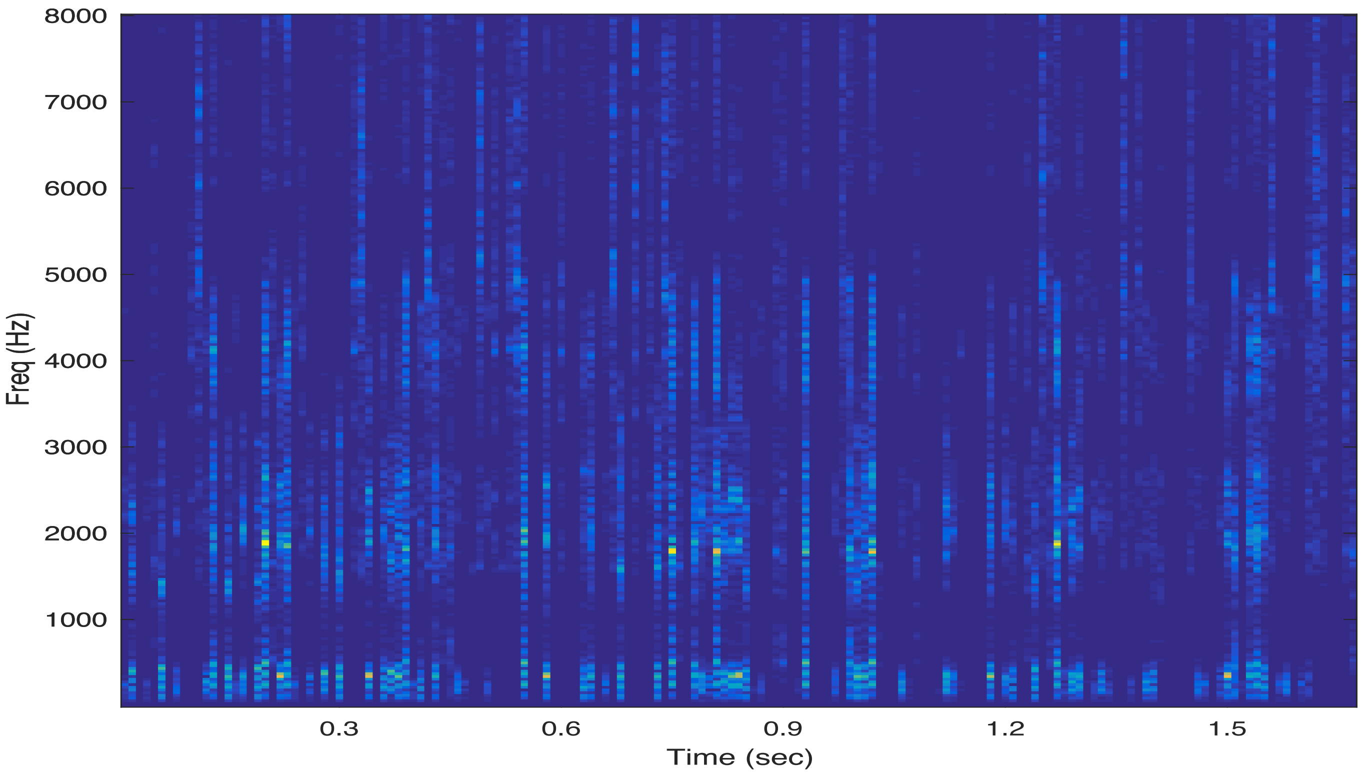
- I want to build a system that recognize my keyword. What should I do?
  - Prepare training signals
- Then what?
  - I feel like learning a model out of this 12X167 data matrix.
  - What should I do?
- Clustering!
  - But to build a model, not for grouping
  - Something similar to learning GMM for classification
- Let me do clustering manually to build ground-truth labels



Training from a Sequence: a Preview

- Smoothing clustering with transition probabilities

- A big question
  - Can I do the same clustering if I scramble the column vectors?
  - Seems difficult
- Ideal clustering considers the temporal structure
  - A phoneme should sustain for a while, etc
- For GMM or Kmeans, the algorithms doesn't know the order of the data
  - Random shuffling will result in same results
  - We may need an algorithm that benefits from the temporal structure
- Will revisit this problem later in this lecture



Case 1: Decoding a Sequence

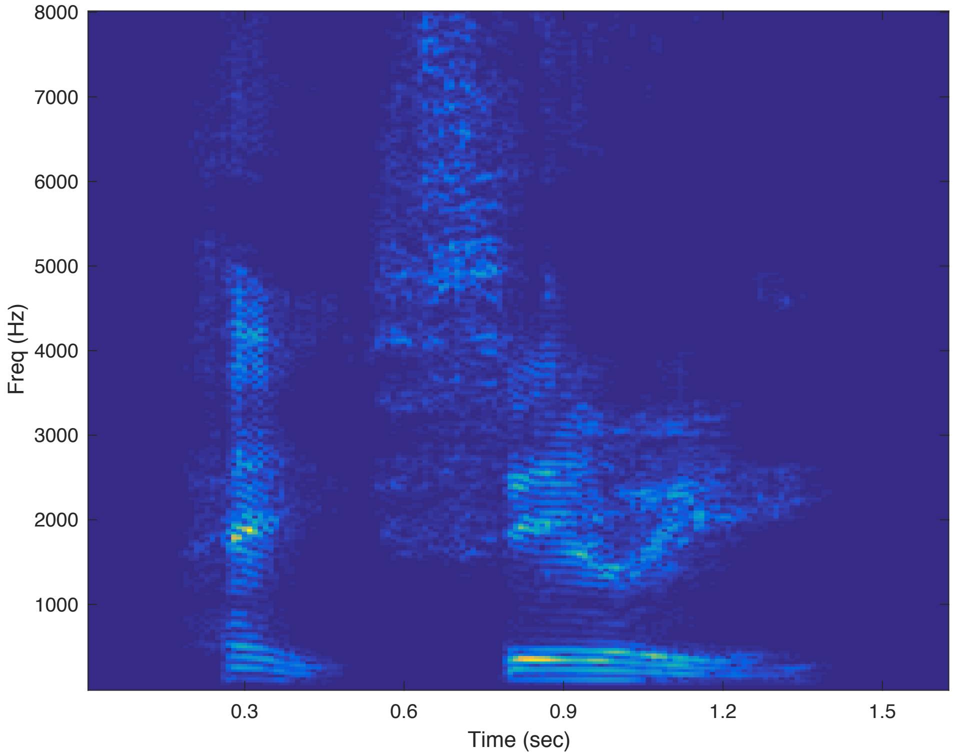
- Smoothing classification with transition probabilities

- For convenience, let's assume that I know the ground-truth cluster means
  - So, we're going to assume that training is done
  - We'll revisit the training part later

- Then the problem boils down to a **series** of phoneme classification job
  - I observe a new sequence, a test signal
  - I want to assign the these new MFCC spectra to the clusters I (manually) defined before


train

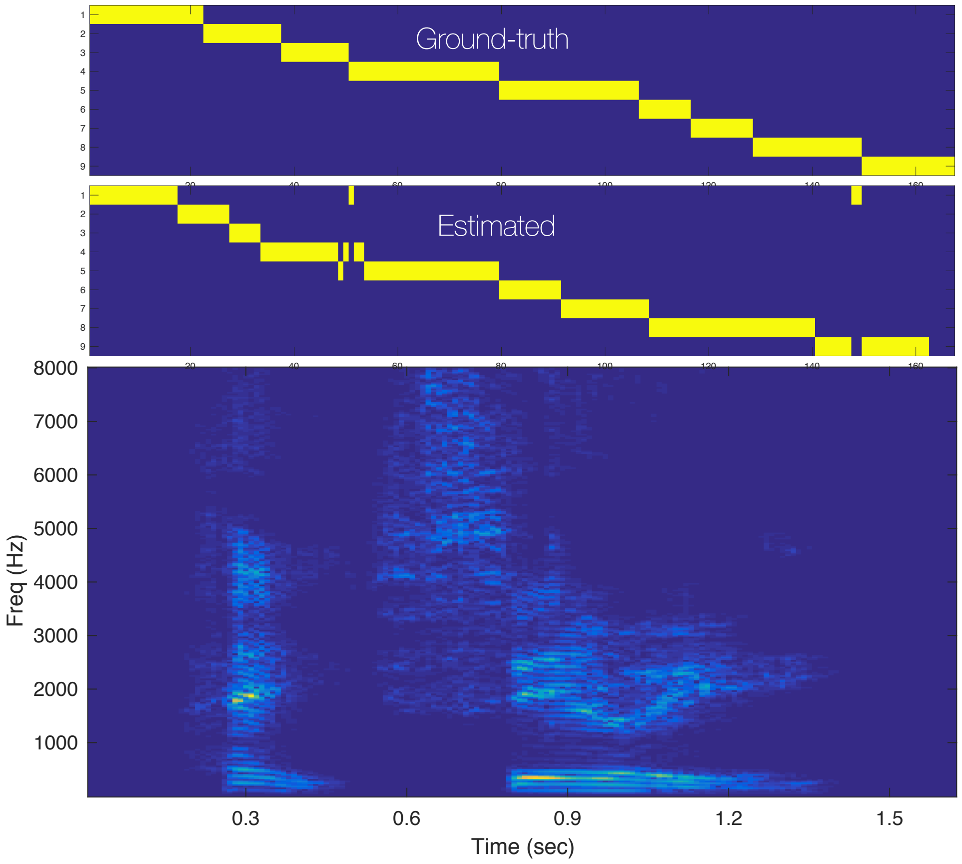

test



Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

- This seems like an easy naïve Bayes classification
  - As we assume known parameters
- But it's not
- This is a **decoding** problem
  - Find out the best **sequence of states** that might have generated the observed sequence
  - In our case…
    - Observation: MFCCs
    - States: Phonemes



Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

- Why is decoding difficult?
- In a perfect world the **emission** probabilities are clean and representative

$$P(\omega_{j_t}|\boldsymbol{x}_t) = \frac{\boxed{P(\boldsymbol{x}_t|\omega_{j_t})}P(\omega_{j_t})}{\sum_{\omega_{j_t}=1}^{J} P(\boldsymbol{x}_t|\omega_{j_t})P(\omega_{j_t})}$$

- In the real world we need to deal with a test signal that can be different from the model
  - Same for all classification problem $P(\tilde{\boldsymbol{x}}_t|\omega_{j_t}) = P(\boldsymbol{x}_t|\omega_{j_t})?$
- We tried our best as for the data generation procedure
  - i.e. we used ground-truth labels $P(\boldsymbol{x}_t|\omega_{j_t})$
  - But it wasn't good enough
- What can we do to improve the performance?
  - $P(\omega_{j_t}) \neq P(\omega_{j_t}|\omega_{j_{t-1}}, \omega_{j_{t-2}}, \cdots, \omega_{j_1}) = P(\omega_{j_t}|\omega_{j_{t-1}})$
  - A first-order **Markov** model

Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

- The first-order Markov model for our example (in English)
  - "If the previous frame was C1 (silence), the current one would be C1 with a high probability"

$$P(\omega_{j_t} = 1 | \omega_{j_{t-1}} = 1) = 0.8$$

  - "But with a small probability, there's a chance that the current frame can be C2 ('h')"

$$P(\omega_{j_t} = 2 | \omega_{j_{t-1}} = 1) = 0.2$$

  - "I'm pretty sure that the current frame won't be any of the phonemes from C3 to C9"

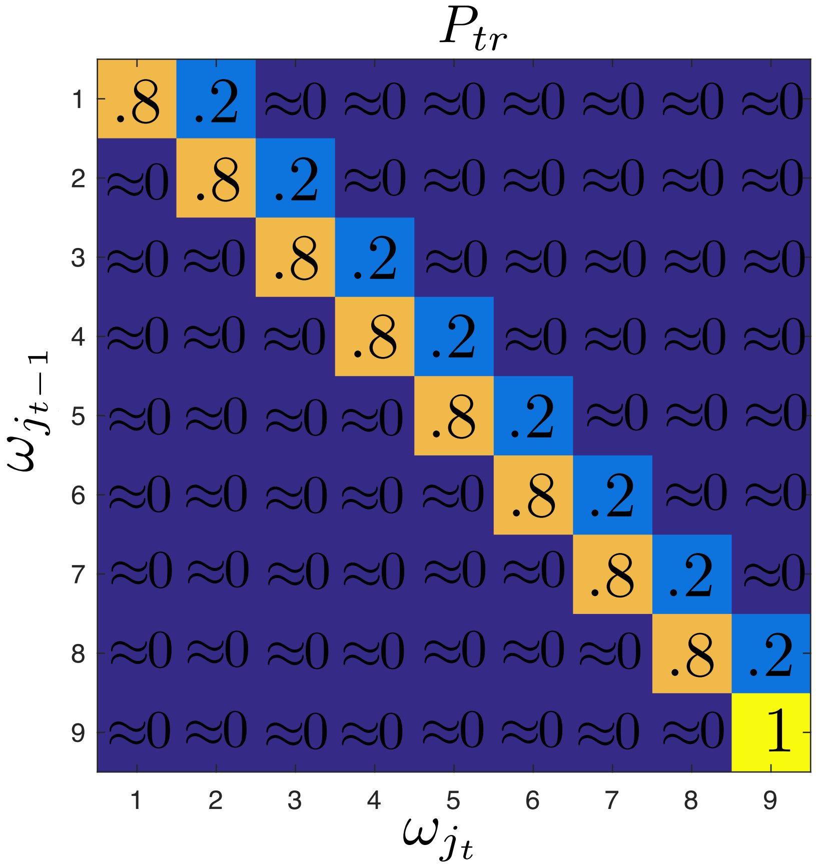
$$P(\omega_{j_t} = 3 | \omega_{j_{t-1}} = 1) = P(\omega_{j_t} = 4 | \omega_{j_{t-1}} = 1) =$$
$$\cdots = P(\omega_{j_t} = 9 | \omega_{j_{t-1}} = 1) = 0$$

- This kind of intuition can build a set of **transition probabilities**
  - It works like *a priori* distribution about the states (clusters) before even seeing the observations

Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

- The transition probability matrix



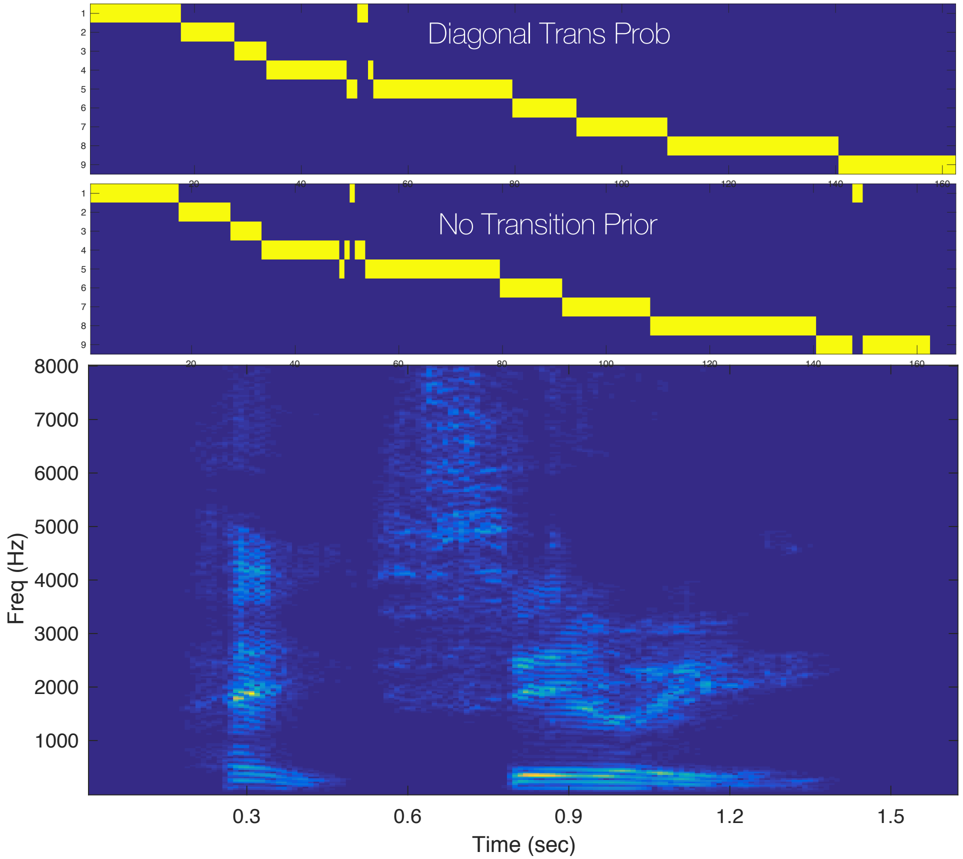
Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

- At a given time frame I have an observation $\boldsymbol{x}_t$
  - The goal is to estimate the posterior probability (actually the joint prob) at every frame

$$P(\omega_{j_t}|\tilde{\boldsymbol{x}}_t) \propto P(\tilde{\boldsymbol{x}}_t|\omega_{j_t})P(\omega_{j_t}|\omega_{j_{t-1}})$$
$$= \mathcal{N}(\tilde{\boldsymbol{x}}_t; \boldsymbol{\mu}_{j_t}, \boldsymbol{\Sigma}_{j_t})P(\omega_{j_t}|\omega_{j_{t-1}})$$

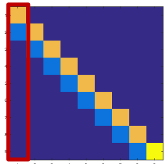
- In English
  - Even if my Gaussian distribution says that the frame should be C1,
  - I ignore it and assign it to C9 as its previous frame was C9

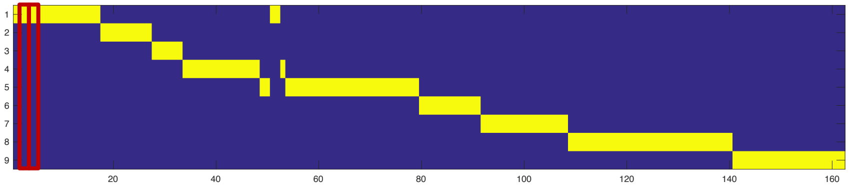


Case 1: Decoding a Sequence

- Smoothing classification with transition probabilities

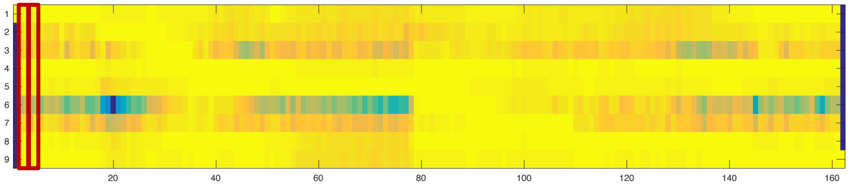
$P_{tr}^{\top}$





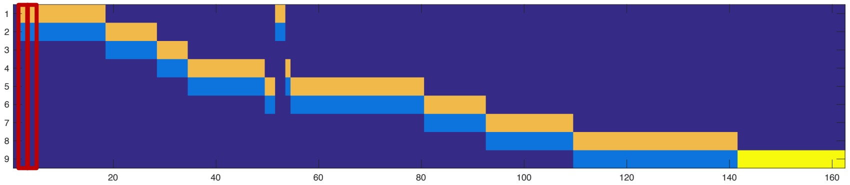
$$\tilde{\boldsymbol{U}}_{(j,t)} = \begin{cases} 1 & \text{if } P_{\boldsymbol{U}(j,t)} \odot \boldsymbol{U}_{(j,t)} \\ & \quad = \max_j P_{\boldsymbol{U}(j,t)} \odot \boldsymbol{U}_{(j,t)} \\ 0 & \text{otherwise} \end{cases}$$

Final labels from posterior dist



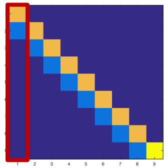
$$P_{\boldsymbol{U}} \odot \boldsymbol{U}$$

$$\log P_{\boldsymbol{U}} \odot \boldsymbol{U} = \log P_{\boldsymbol{U}} + \log \boldsymbol{U}$$

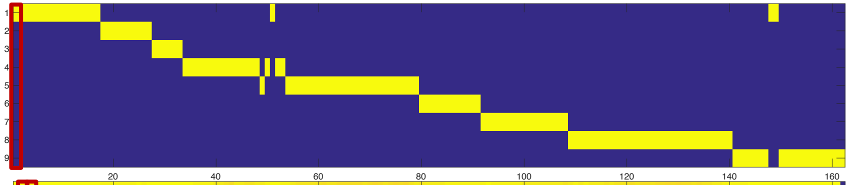


$$P_{\boldsymbol{U}} = P(\omega_{j_t} | \omega_{j_{t-1}}) = P_{tr}^{\top} \tilde{\boldsymbol{U}}_{(:,t-1)}$$

$P_{tr}^{\top}$

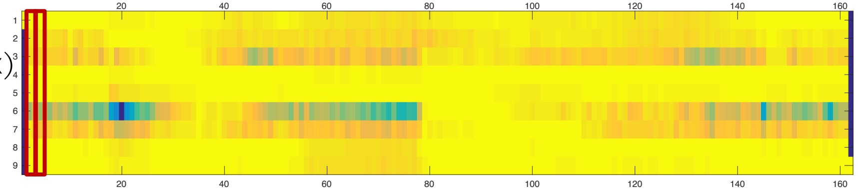


Transition matrix
(Note: no time index)



$$\bar{\boldsymbol{U}}_{(j,t)} = \begin{cases} 1 & \text{if } \boldsymbol{U}_{(j,t)} = \max_j \boldsymbol{U}_{(j,t)} \\ 0 & \text{otherwise} \end{cases}$$

State labels from likelihood



$$\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t | \omega_{jt})$$

Likelihood from uniform transition prob. (Naïve Bayes)

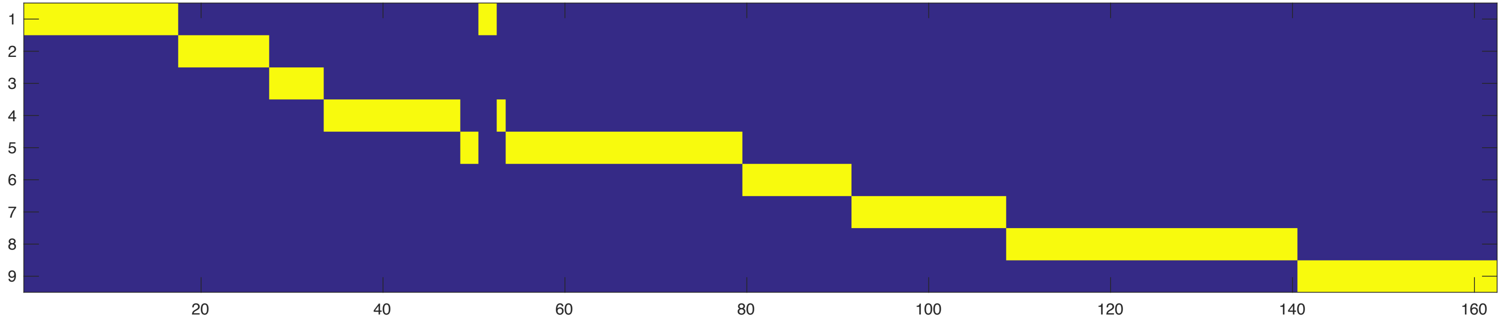
Case 1: Decoding a Sequence
- Smoothing classification with transition probabilities

- For t-th frame
  - Calculate the log likelihood $\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t | \omega_{j_t} = j)$
  - Calculate the prior
    - For this we need to know $\tilde{\boldsymbol{U}}_{(j,t-1)}$

$$P_{\boldsymbol{U}(j,t)} = P(\omega_{j_t} | \omega_{j_{t-1}}) \quad = P_{tr}^\top \tilde{\boldsymbol{U}}_{(:,t-1)}$$

Keeps j'-th column of $P_{tr}^\top$
where j' is the label of the (t-1)-th state

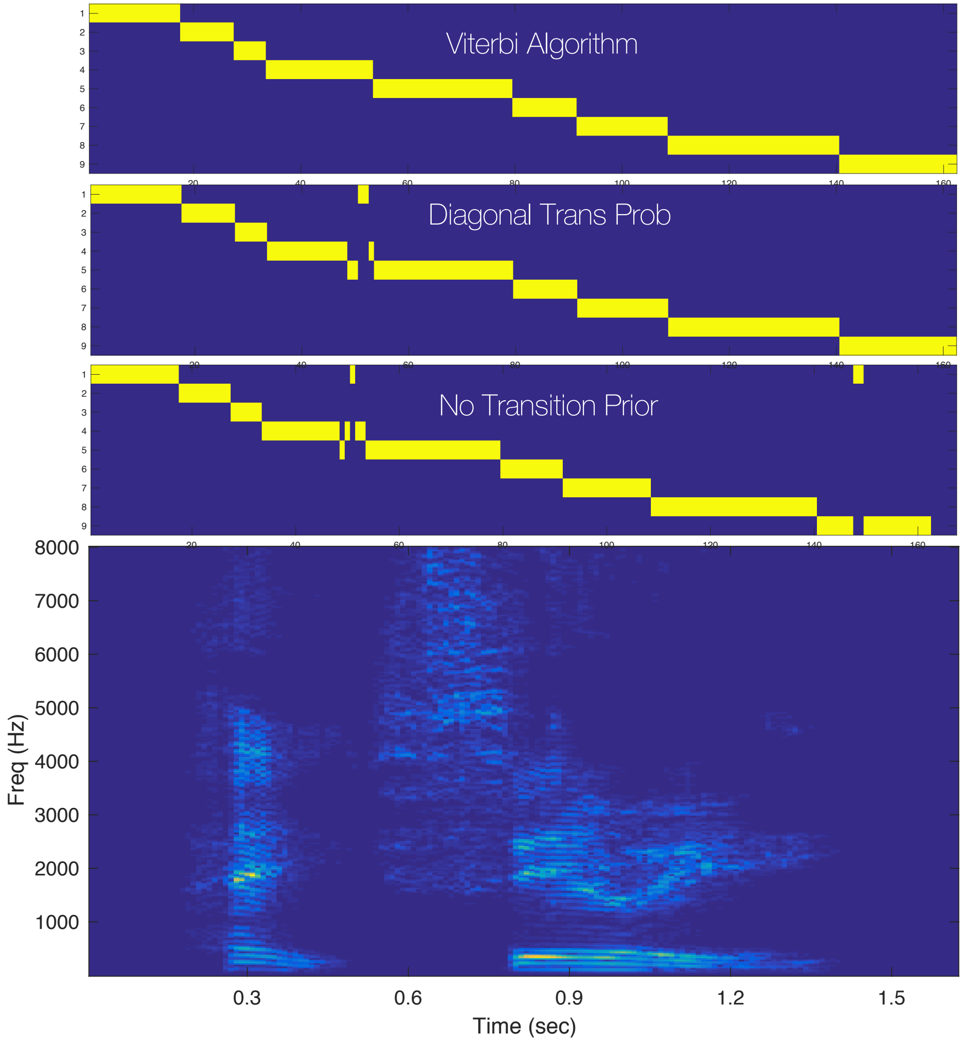
  - Calculate the posterior $P_{\boldsymbol{U}(:,t)} \odot \boldsymbol{U}_{(:,t)}$
  - Find the final state $\tilde{\boldsymbol{U}}_{(j,t)} = \begin{cases} 1 & \text{if } P_{\boldsymbol{U}(j,t)} \odot \boldsymbol{U}_{(j,t)} = \max_j P_{\boldsymbol{U}(j,t)} \odot \boldsymbol{U}_{(j,t)} \\ 0 & \text{otherwise} \end{cases}$
- Do you like this approach?
  - If you're wrong in the earlier frames, there's no way to fix it



Case 1: Decoding a Sequence

- The Viterbi algorithm

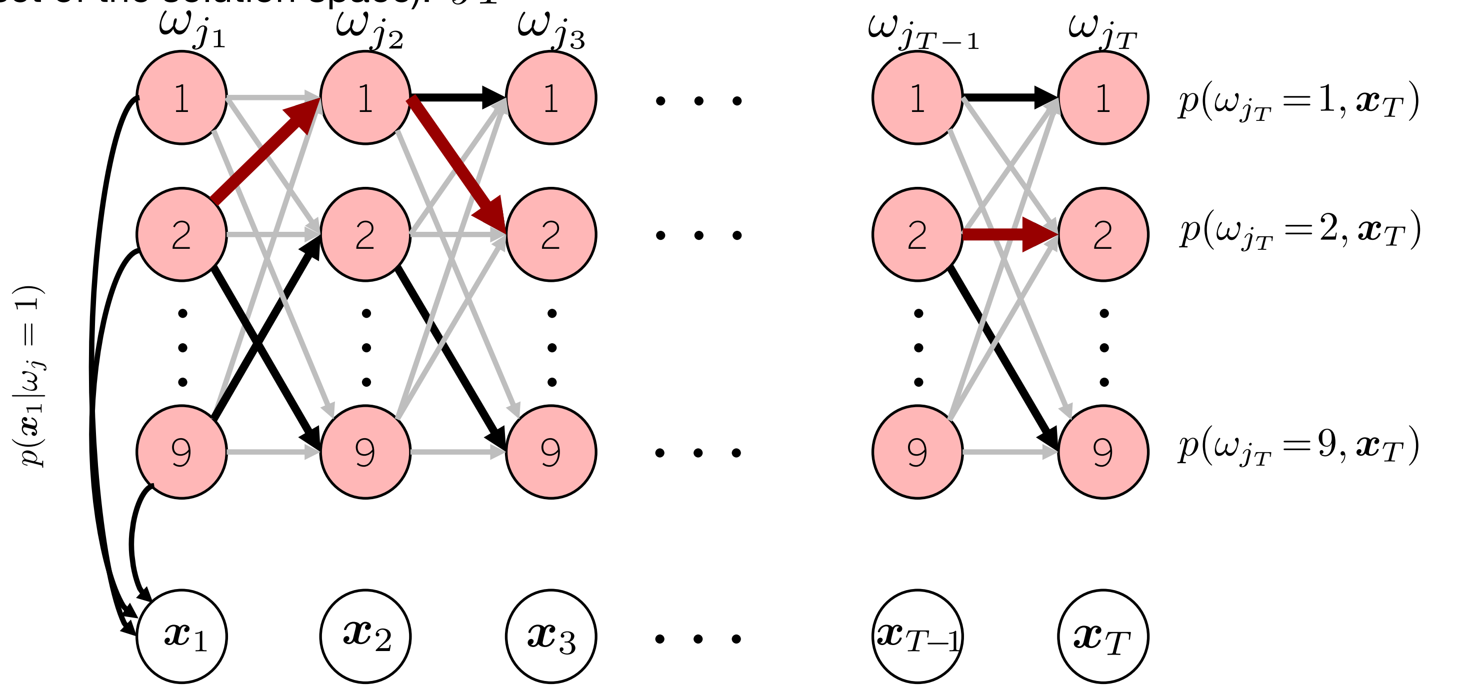
- With the Viterbi algorithm we find the **best path**
  - Calculate the posterior probabilities and back track



Case 1: Decoding a Sequence

- The Viterbi algorithm

- A dynamic programming algorithm
  - The entire solution space: J^T
  - Our candidates (subset of the solution space): JT



Each joint prob chooses the best transition

$p(\omega_{j_1} = 1|\boldsymbol{x}_1) \propto p(\boldsymbol{x}_1|\omega_{j_1} = 1)$

$p(\omega_{j_2} = 9, \boldsymbol{x}_2) = p(\boldsymbol{x}_2|\omega_{j_2} = 9) \cdot \boxed{P_{tr(9,2)}} \cdot p(\omega_{j_1} = 2, \boldsymbol{x}_1)$

Best path coming to 9 is from 2 (at t=2)

$p(\omega_{j_T} = 2, \boldsymbol{x}_T) = p(\boldsymbol{x}_T|\omega_{j_T} = 2) \cdot P_{tr(2,2)} \cdot p(\omega_{j_{T-1}} = 2, \boldsymbol{x}_{T-1})$

Case 1: Decoding a Sequence

- The Viterbi algorithm

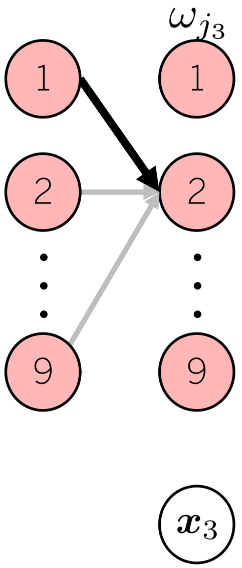
- The Viterbi algorithm:
- Prepare the log-likelihoods $\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t | \omega_{jt})$
- Calculate the posterior (joint) probability from the left (earlier frames) to the right
  - For j-th state at t-th frame there are J different possible paths from the previous frame
  - The Viterbi algorithm chooses and keeps the best path from t-1 to t

$$p(\omega_{j_t}, \boldsymbol{x}_t) \propto \max_{j_{t-1}} p(\boldsymbol{x}_t | \omega_{j_t}) \cdot p(\omega_{j_t} | \omega_{j_{t-1}}) \cdot p(\omega_{j_{t-1}}, \boldsymbol{x}_{t-1})$$

$$\log p(\omega_{j_t}, \boldsymbol{x}_t) = \max_{j_{t-1}} \log p(\boldsymbol{x}_t | \omega_{j_t}) + \log p(\omega_{j_t} | \omega_{j_{t-1}}) + \log p(\omega_{j_{t-1}}, \boldsymbol{x}_{t-1}) + const.$$

- Backtracking
  - The final J states at t=T have their final posterior probabilities
  - Pick the best $p(\omega_{j_T}, \boldsymbol{x}_T)$, e.g. $\arg\max_{j_T} p(\omega_{j_T}, \boldsymbol{x}_T) = 9$
  - What was $\omega_{j_{T-1}}$ that led to it?
  - What was $\omega_{j_{T-2}}$ that led to the best?
  - …
- This procedure is based on a recursive assumption

$$\log p(\omega_{j_t}, \boldsymbol{x}_t) = \log p(\boldsymbol{x}_t | \omega_{j_t}) + \max_{j_{t-1}} \left\{ \log p(\omega_{j_{t-1}}, \boldsymbol{x}_t) + \log p(\omega_{j_t} | \omega_{j_{t-1}}) \right\}$$



Case 1: Decoding a Sequence

- The Viterbi algorithm

- You have a model, maybe a naïve Bayes
  - Or, any model that assumes a set of latent variables (classes)
    - The model also assumes each class has a unique generative model and observations are from them
  - Without any temporal structure this is a classification problem
- With the n-th order Markov model on the states

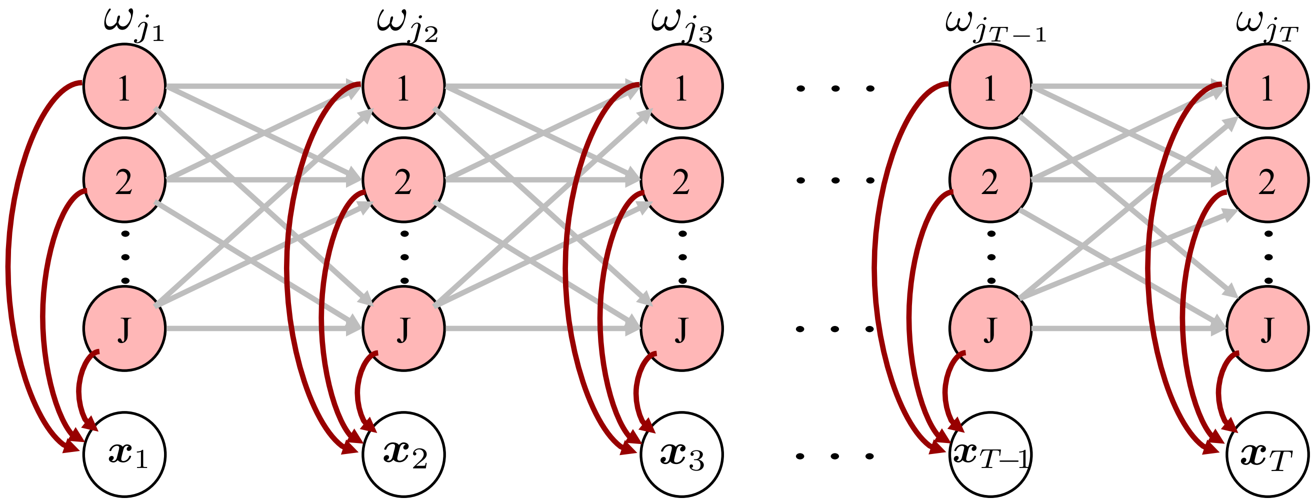
$$P(\omega_{j_t}|\omega_{j_{t-1}}, \omega_{j_{t-2}}, \cdots, \omega_{j_1}) = P(\omega_{j_t}|\omega_{j_{t-1}})$$

  - This turns into a decoding problem
  - It's still a classification problem
    - But you need to smooth your labels based on the transition probabilities
- The Viterbi algorithm does this labeling job in two paths
  - Better than naïve forward-path-only smoothing
- The decoding problem assumes
  - You know the LV-specific generative models
  - You know the transition probabilities
    - **What if you don't?**

Case 2: Learning from a Sequence

- Hidden Markov Models (HMM)

- Let me properly define an HMM first



- The observations: $\boldsymbol{x}_t \in \mathbb{R}^{D\times 1}$
- The hidden states: ω_{j_t} ← Decoding (Viterbi) procedure estimates this (given the following)
- Transition probabilities: $p(\omega_{j_t}|\omega_{j_{t-1}}) \in \mathbb{R}^{J\times J}$ ← HMM learning estimates this
- Emission probabilities: $p(\boldsymbol{x}_t|\omega_{j_t}) \in \mathbb{R}^{D\times J}$ ← HMM learning estimates this
- Initial priors: $p(\omega_{j_1}) \in \mathbb{R}^{J\times 1}$ ← HMM learning estimates this

Case 2: Learning from a Sequence

- The Viterbi reestimation algorithm

- Initialize $p(\omega_{j_t}|\omega_{j_{t-1}})$ with random numbers
- Prepare the log-likelihoods $\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t|\omega_{jt})$
- Calculate the posterior probability using the local best paths

$$\log p(\omega_{j_t}, \boldsymbol{x}_t) = \max_{j_{t-1}} \log p(\boldsymbol{x}_t|\omega_{j_t}) + \log p(\omega_{j_t}|\omega_{j_{t-1}}) + \log p(\omega_{j_{t-1}}, \boldsymbol{x}_{t-1}) + const.$$

- Backtracking
  - Estimate the best path given the data and the parameter estimation:

$$\Omega_i = \{\omega_{j_1} = 1, \omega_{j_2} = 1, \cdots, \omega_{j_{t-1}} = 3, \omega_{j_t} = 4, \omega_{j_{t+1}} = 4, \cdots \omega_{j_T} = 9\}$$

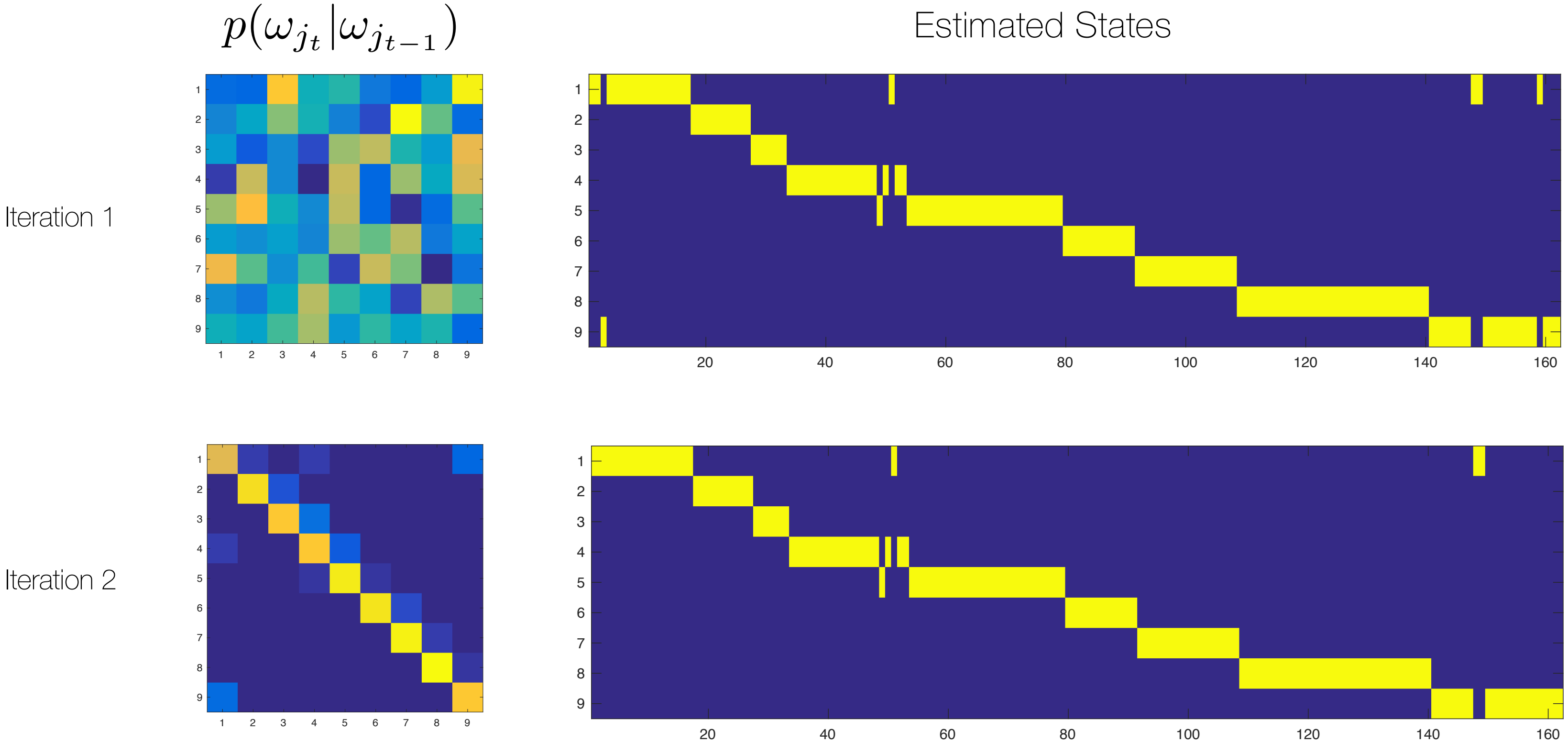
- Then, it's easy to count these values
  - $n_{i|j}$: number of transitions from state *j* to *i*
  - $n_{|j}$: number of transitions originated from state *j*

- Finally $p(\omega_{j_t}|\omega_{j_{t-1}}) \leftarrow \dfrac{n_{i|j}}{n_{|j}}$

Case 2: Learning from a Sequence

- The Viterbi reestimation algorithm

- Demo



Case 2: Learning from a Sequence
- The Viterbi reestimation algorithm

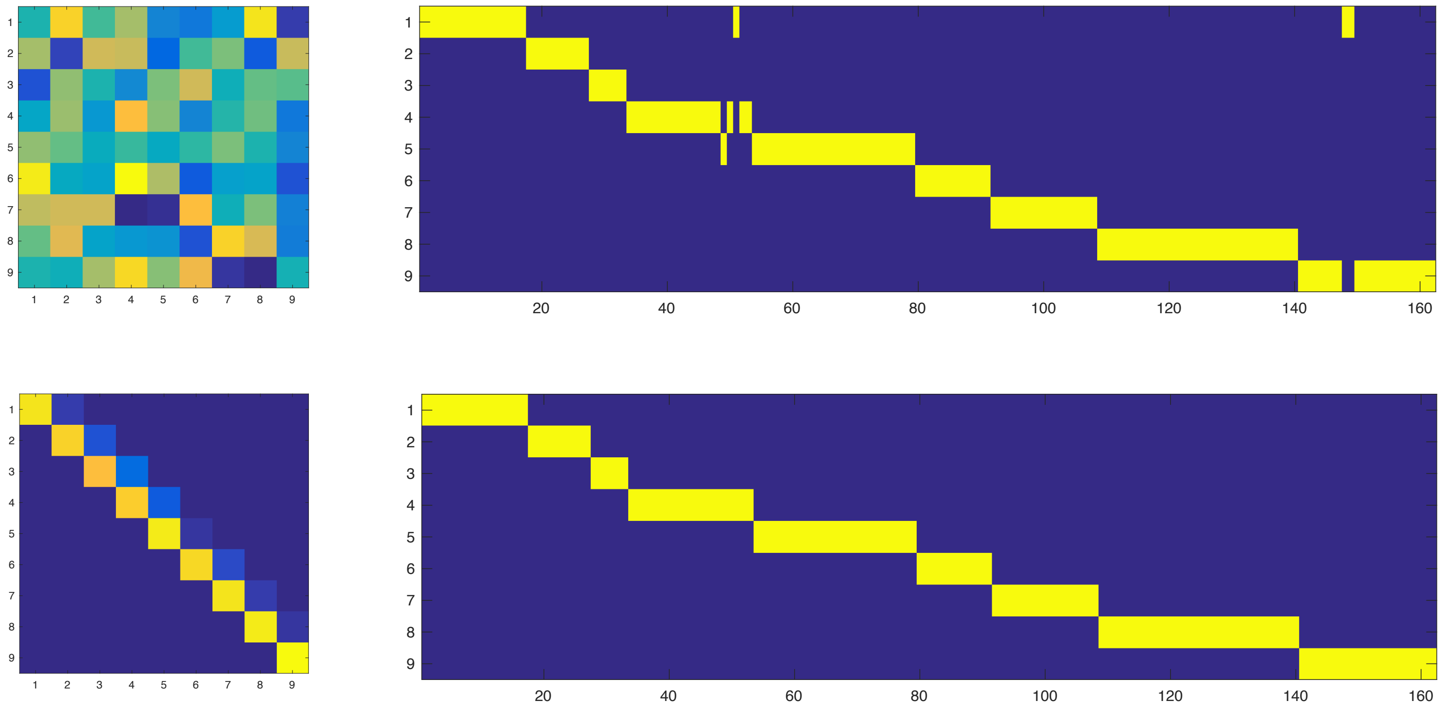
- Prior for the diagonals (erase off-diagonals at every iteration)

$p(\omega_{j_t}|\omega_{j_{t-1}})$

Estimated States

Iteration 1

Iteration 2

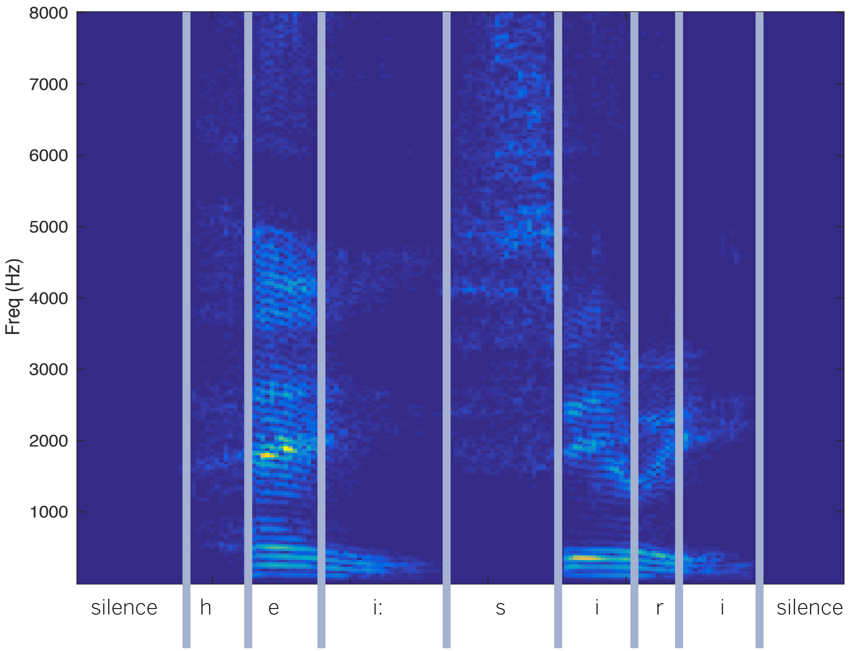


Case 2: Learning from a Sequence

- The Viterbi reestimation algorithm

- Why was this so easy?
  - We saw a convergence after only one iteration
  - B/C we started from a good set of emission probabilities using training data
  - If you remember, I manually cluster the MFCC frames to get the cluster-specific distribution
- But HMM learning can be seen as a clustering problem (or a mixture of distributions)
  - So, we must be able to learn the other parameters, too
  - I mean the emission probabilities

$$p(\boldsymbol{x}_t|\omega_{j_t}) \in \mathbb{R}^{D\times J}$$



Case 2: Learning from a Sequence

- The Viterbi reestimation algorithm (full)

- Initialize $p(\omega_{j_t}|\omega_{j_{t-1}})$ with random numbers
- Prepare the log-likelihoods $\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t|\omega_{jt})$
- Calculate the posterior probability using the best paths

$$\log p(\omega_{j_t}, \boldsymbol{x}_t) = \max_{j_{t-1}} \log p(\boldsymbol{x}_t|\omega_{j_t}) + \log p(\omega_{j_t}|\omega_{j_{t-1}}) + \log p(\omega_{j_{t-1}}, \boldsymbol{x}_{t-1}) + const.$$

- Backtracking
  - Estimate the best path given the data and the parameter estimation:

$$\Omega_i = \{\omega_{j_1}=1, \omega_{j_2}=1, \cdots, \omega_{j_{t-1}}=3, \omega_{j_t}=4, \omega_{j_{t+1}}=4, \cdots \omega_{j_T}=9\}$$

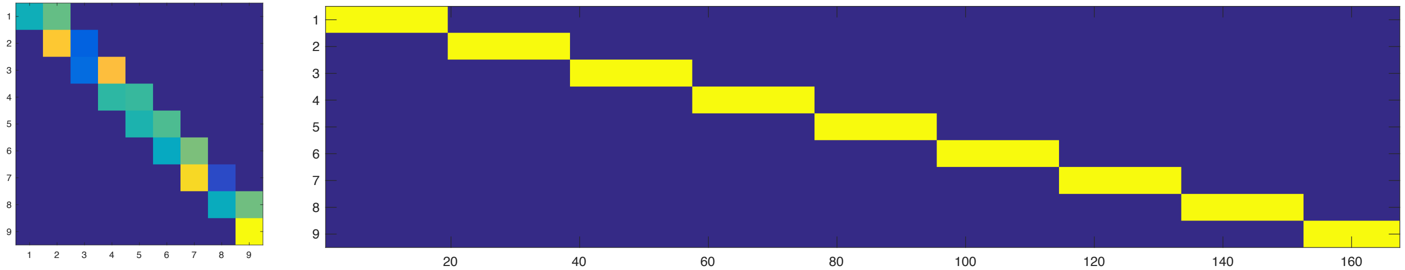
- Then, it's easy to count these values
  - $n_{i|j}$: number of transitions from state *j* to *i*
  - $n_{|j}$: number of transitions originated from state *j*
  - $n_{i|}$: number of transitions terminated at state i
- Finally $p(\omega_{j_t}|\omega_{j_{t-1}}) \leftarrow \frac{n_{i|j}}{n_{|j}}$
- For the observations associated with state j, do your parameter estimation
  - e.g. Maximum likelihood estimation
  - In my case the emission probability is $p(\boldsymbol{x}_t|\omega_{j_t}) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$
  - Therefore, $\boldsymbol{\mu}_i = \frac{1}{n_{i|}} \sum_{t=1}^{T} \mathcal{I}(\omega_{j_t} = i)\boldsymbol{x}_t$ $\quad \mathcal{I}(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$

Case 2: Learning from a Sequence

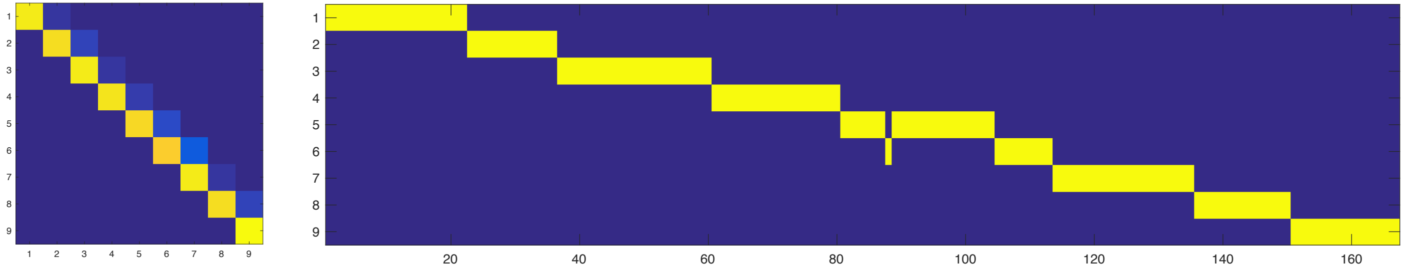
- The Viterbi reestimation algorithm (full)

- This time the Gaussian parameters are updated as well

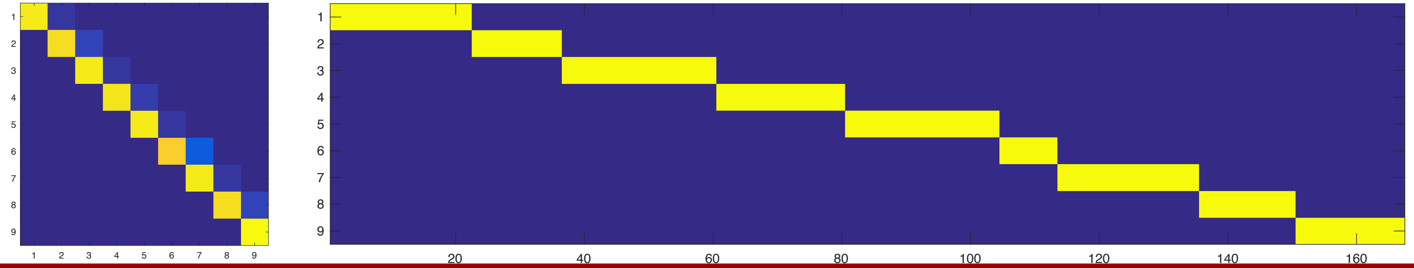
Iteration 1



Iteration 2



Iteration 5



Case 3: Evaluating a Sequence

- Recognition using the best path

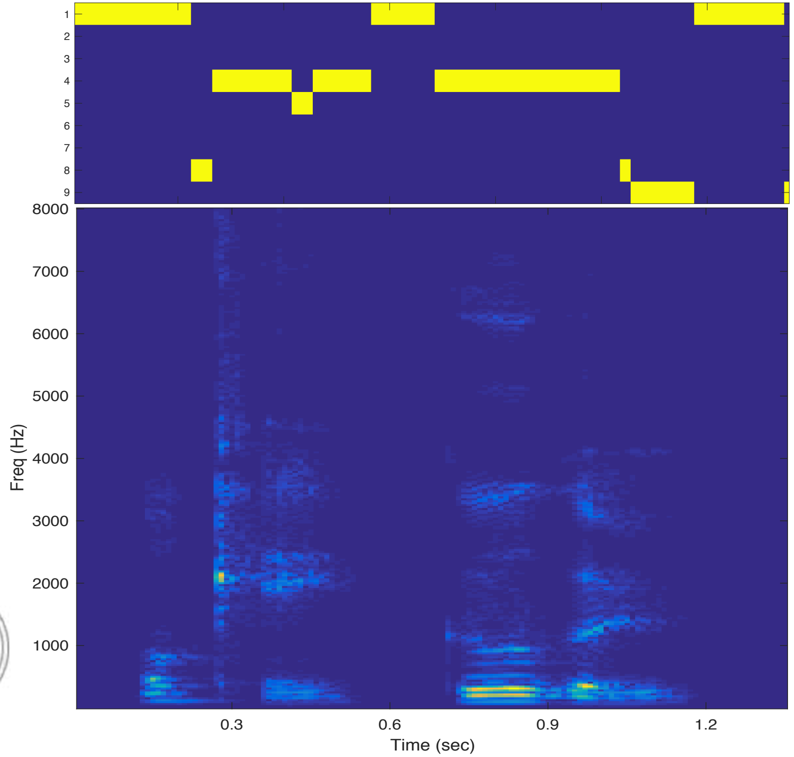
- Now you know how to find the best path for a test signal
  - And it comes with a posterior probability of the state transitions

$$p(\Omega_i)p(\boldsymbol{X}|\Omega_i) \qquad \Omega_i : \text{i-th (best) path out of } J^T \text{ possible paths}$$

- Another test signal
  - Log of the posterior prob of the best path:

    -13469
  - Compare this with that of the test signal that says “Hey, Siri”:

    -3230
- The best path found is not good enough to model the new signal
  - The new test signal might be far from the training signal
- Can we be more careful?



Case 3: Evaluating a Sequence

- Recognition using all paths

- The forward pass:
- Prepare the log-likelihoods $\log \boldsymbol{U}_{(j,t)} = \log P(\tilde{\boldsymbol{x}}_t|\omega_{jt})$
- ~~Calculate the joint probability from the left (earlier frames) to the right~~
  - ~~For j-th state at t-th frame there are J different possible paths from the previous frame~~
  - ~~The Viterbi algorithm chooses and keeps the best path from t-1 to t~~

$$\cancel{\log p(\omega_{j_t}, \boldsymbol{x}_t) = \max_{j_{t-1}} \log p(\boldsymbol{x}_t|\omega_{j_t}) + \log p(\omega_{j_t}|\omega_{j_{t-1}}) + \log p(\omega_{j_{t-1}}, \boldsymbol{x}_{t-1}) + const.}$$

- The forward algorithm calculates from all paths and sum them up

$$\alpha_{j,1} = P(\omega_{j_1})p(\boldsymbol{x}_1|\omega_{j_1}) = p(\boldsymbol{x}_1, \omega_{j_1})$$

$$\alpha_{j,2} = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \omega_{j_2}) = \sum_{\omega_{j_1}} p(\boldsymbol{x}_1, \boldsymbol{x}_2, \omega_{j_2}, \omega_{j_1}) = p(\boldsymbol{x}_2|\omega_{j_2}) \sum_{\omega_{j_1}} p(\boldsymbol{x}_1, \omega_{j_1})p(\omega_{j_2}|\omega_{j_1})$$

$$= p(\boldsymbol{x}_2|\omega_{j_2}) \sum_{\omega_{j_1}} \alpha_{j,1} p(\omega_{j_2}|\omega_{j_1})$$

$$\alpha_{j,3} = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \omega_{j_3}) = \sum_{\omega_{j_2}} p(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \omega_{j_3}, \omega_{j_2}) = p(\boldsymbol{x}_3|\omega_{j_3}) \sum_{\omega_{j_2}} p(\boldsymbol{x}_1, \boldsymbol{x}_2, \omega_{j_2})p(\omega_{j_3}|\omega_{j_2})$$

$$= p(\boldsymbol{x}_3|\omega_{j_3}) \sum_{\omega_{j_2}} \alpha_{j,2} p(\omega_{j_3}|\omega_{j_2})$$

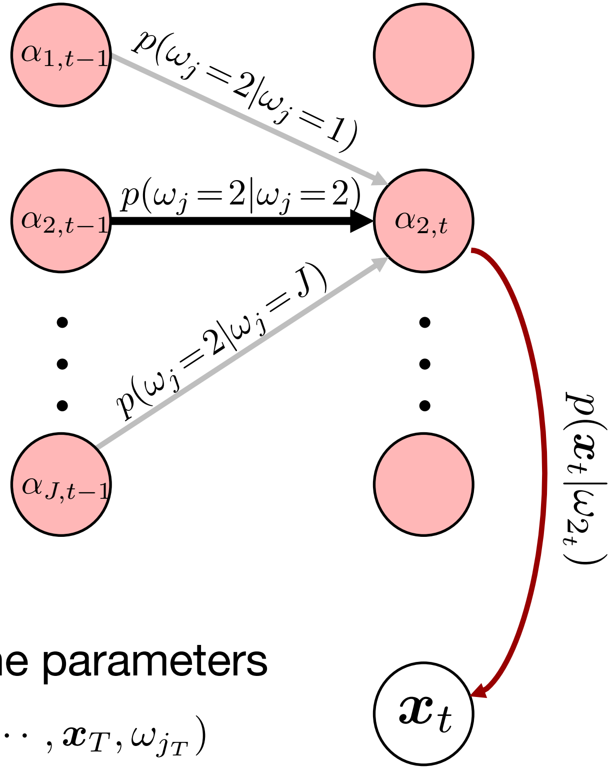
  - Where am I going? $\alpha_{j,t} = p(\boldsymbol{x}_t|\omega_{j_t}) \sum_{\omega_{j_{t-1}}} \alpha_{j,t-1} p(\omega_{j_t}|\omega_{j_{t-1}})$

Case 3: Evaluating a Sequence

- Recognition using all paths

- The forward pass

$$\alpha_{j,t} = p(\boldsymbol{x}_t|\omega_{j_t}) \sum_{\omega_{j_{t-1}}} \alpha_{j,t-1} p(\omega_{j_t}|\omega_{j_{t-1}})$$



- For your information..
  - If you replace the summation with maximum

$$\alpha_{j,t} = p(\boldsymbol{x}_t|\omega_{j_t}) \max_{j_{t-1}} \alpha_{j,t-1} p(\omega_{j_t}|\omega_{j_{t-1}})$$

  - It corresponds to the forward pass of the Viterbi algorithm

- In the end, we get the probability of observing the data given the parameters

$$\alpha_{j,T} = p(\boldsymbol{x}_T|\omega_{j_T}) \sum_{\omega_{j_{T-1}}} \alpha_{j,T-1} p(\omega_{j_T}|\omega_{j_{T-1}}) = p(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T, \omega_{j_T})$$

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T) = \sum_j \alpha_{j,T} \qquad p(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T; \boldsymbol{\Theta}) = \sum_j \alpha_{j,T}$$

  - If $\sum_j \alpha_{j,T}$ is high, the model fits the data better

Case 2: Learning from a Sequence (Revisited)

- The Baum-Welch algorithm

- Now we have a soft version of the forward pass
  - Let's use this for the learning task
- The backward pass

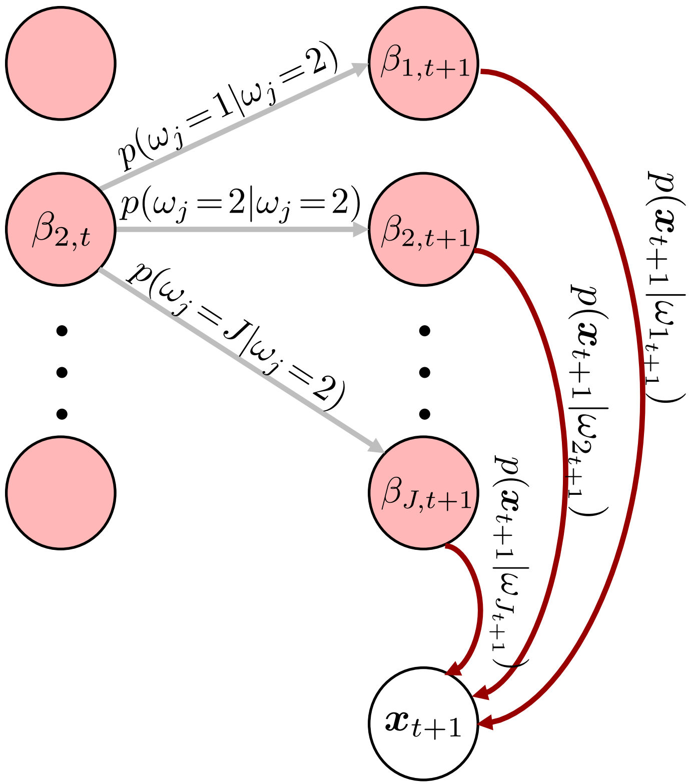
$$\beta_{j,T} = 1$$

$$\beta_{j,t} = p(\boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_T | \omega_{j_t})$$

$$= \sum_{\omega_{j_{t+1}}} p(\boldsymbol{x}_{t+1}, \cdots, \boldsymbol{x}_T, \omega_{j_{t+1}} | \omega_{j_t})$$

$$= \sum_{\omega_{j_{t+1}}} p(\boldsymbol{x}_{t+1} | \omega_{j_{t+1}}) p(\boldsymbol{x}_{t+2}, \cdots, \boldsymbol{x}_T | \omega_{j_{t+1}}) p(\omega_{j_{t+1}} | \omega_{j_t})$$

$$= \sum_{\omega_{j_{t+1}}} p(\boldsymbol{x}_{t+1} | \omega_{j_{t+1}}) \beta_{j_{t+1}} p(\omega_{j_{t+1}} | \omega_{j_t})$$

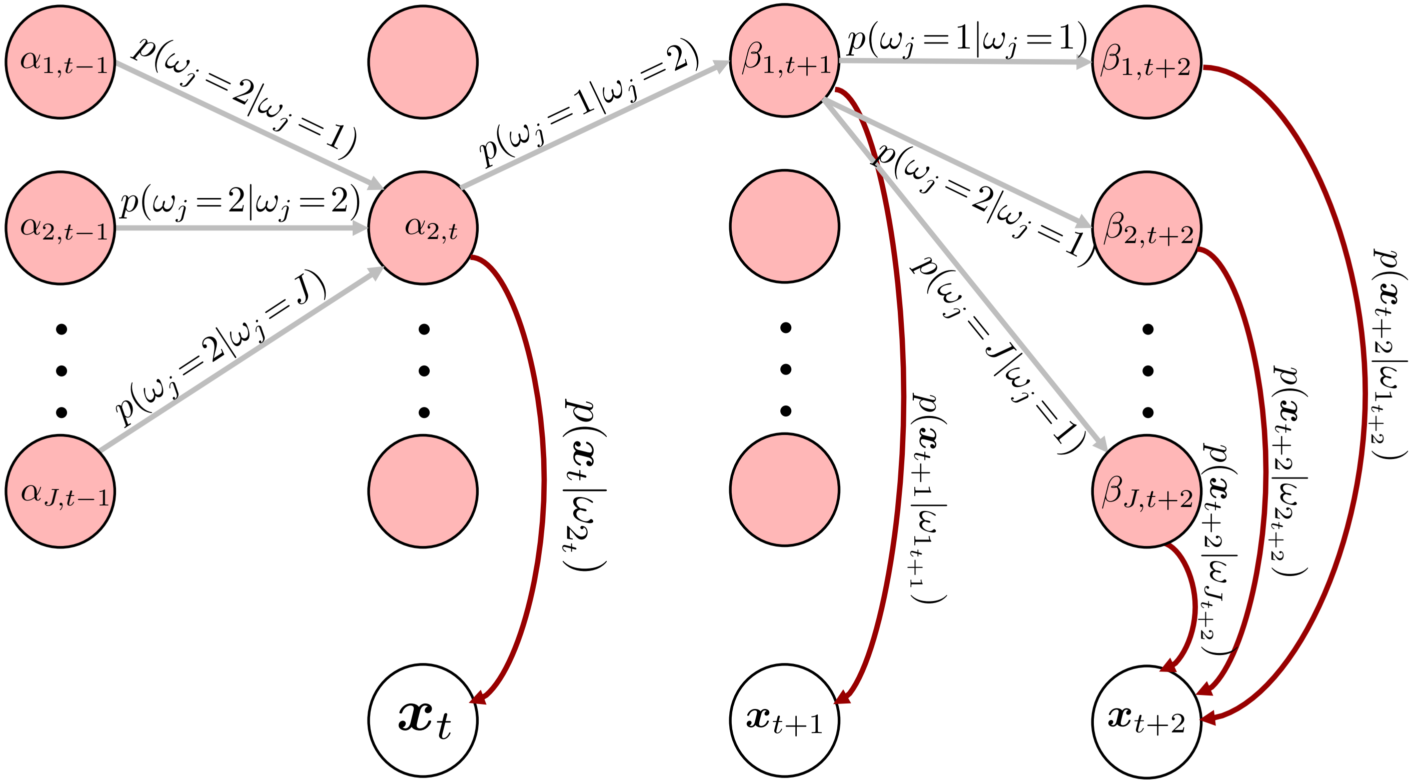


- Let's combine the passes

Case 2: Learning from a Sequence (Revisited)

- The Baum-Welch algorithm

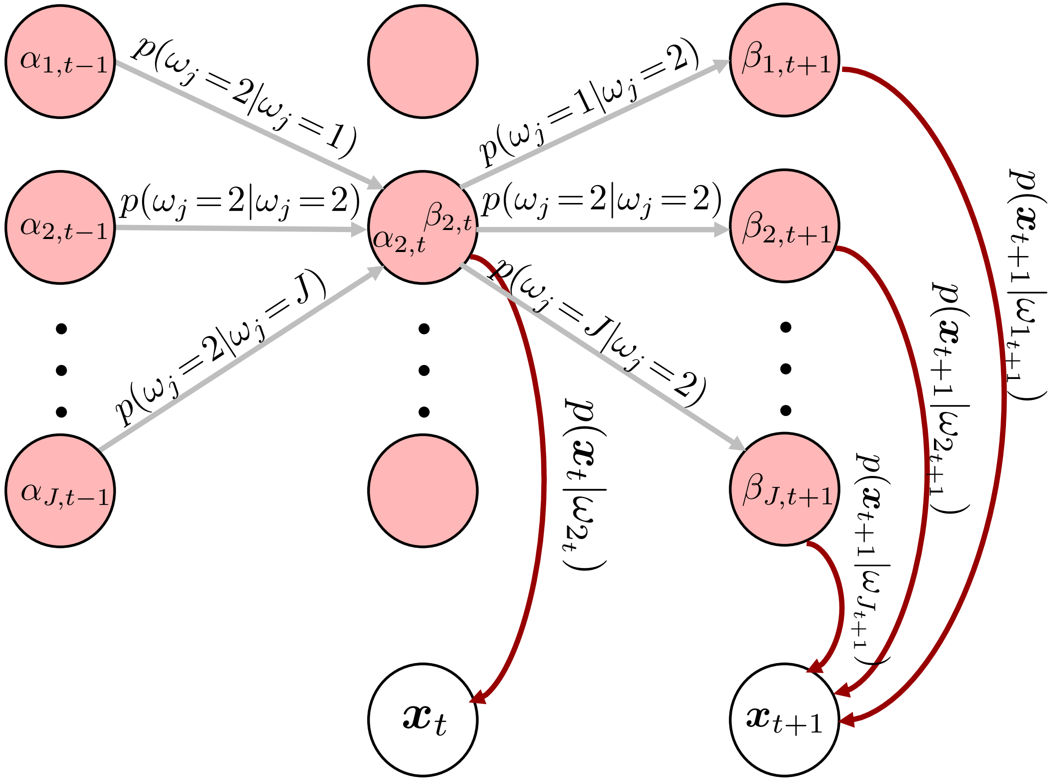
○ $$p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, \omega_{2_t}, \boldsymbol{x}_{t+1}, \omega_{1_{t+1}}, \boldsymbol{x}_{t+2}, \cdots, \boldsymbol{x}_T)$$
$$= \alpha_{2,t}\, p(\omega_j = 1 | \omega_j = 2)\, p(\boldsymbol{x}_{t+1} | \omega_{1_{t+1}}) \beta_{1,t+1}$$



Case 2: Learning from a Sequence (Revisited)

- The Baum-Welch algorithm

- $p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T, \omega_{2_t}) = \alpha_{2,t}\beta_{2,t}$



Case 2: Learning from a Sequence (Revisited)

- The Baum-Welch algorithm

$$p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T, \omega_{2_t}) = \alpha_{2,t}\beta_{2,t}$$

$$p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t, \omega_{2_t}, \boldsymbol{x}_{t+1}, \omega_{1_{t+1}}, \boldsymbol{x}_{t+2}, \cdots, \boldsymbol{x}_T) = \alpha_{2,t} p(\omega_j = 1 | \omega_j = 2) p(\boldsymbol{x}_{t+1} | \omega_{1_{t+1}}) \beta_{1,t+1}$$

- Transition matrix estimation
  - Get the pseudocount

$$p(\omega_1|\omega_2) \leftarrow \frac{\sum_t p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T, \omega_{2_t}, \omega_{1_{t+1}})}{\sum_t p(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T, \omega_{2_t})} = \frac{\sum_t \alpha_{2,t} p(\omega_j = 1 | \omega_j = 2) p(\boldsymbol{x}_{t+1} | \omega_{1_{t+1}}) \beta_{1,t+1}}{\sum_t \alpha_{2,t}\beta_{2,t}}$$

- Emission probability
  - Use the posterior prob $p(\omega_{j_t} | \boldsymbol{x}_1 \cdots \boldsymbol{x}_T) \propto \gamma_{j,t} = \alpha_{j,t}\beta_{j,t}$
  - Remember $p(\boldsymbol{x}_t | \omega_{j_t}) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$

$$\boldsymbol{\mu}_j = \frac{\sum_{t=1}^T \gamma_{j,t} \boldsymbol{x}_t}{\sum_{t=1}^T \gamma_{j,t}}$$

$$\boldsymbol{\mu}_{j'} = \frac{1}{n_{i|}} \sum_{t=1}^T \mathcal{I}(\omega_{j_t} = j') \boldsymbol{x}_t$$

$$\mathcal{I}(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

- EM with a more complex E-step!

Case 2: Learning from a Sequence (Revisited)

- The Baum-Welch algorithm

- What happens if my emission probabilities are not following Gaussian?
  - There's no guarantee
  - Is there a flexible probabilistic distribution that can model ANY data?
  - Gaussian mixture models
- Eventually, a full HMM will be
  - A mixture of GMM
  - Each state has a corresponding GMM as its emission probability
  - You do another EM for the parameter estimation
    - Instead of the MLE step

Recap

- Three problems in Hidden Markov Models (HMM)
  - Decoding: given the observation sequence and model parameters, find out the best sequence of hidden states
    - I said it's classification by seeing each hidden state as a class, but in practice it's not
  - Evaluation: calculate the probability of observing the data given the model parameters
    - An HMM model per class is a more common classification setting
  - Learning: learning the model parameters from data
    - Corresponds to training
- Decoding
  - Viterbi algorithm
    - More efficient than an exhaustive search $O(J^T)$; uses the best path out of $O(JT)$; backtracking
- Evaluation
  - Using the best path
    - Do Viterbi and pick up the probability of the best path
  - Forward algorithm
    - Soften up the decision. Gather up all the other probabilities coming from the suboptimal paths.
- Learning
  - The Viterbi reestimation
    - Do Viterbi and count the number of frames per state. Use this to do the clustering to reestimate the parameters.
  - The Baum-Welch algorithm
    - Using the forward pass and backward pass, calculate the probability of having j-th state given the entire observation sequence. This is your E-step.

Reading

- Chapter 9
- L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, Feb 1989.
  doi: 10.1109/5.18626

Thank You!