Proc. Digital del Continguts Musicals

Session 2:
Sequence Recognition

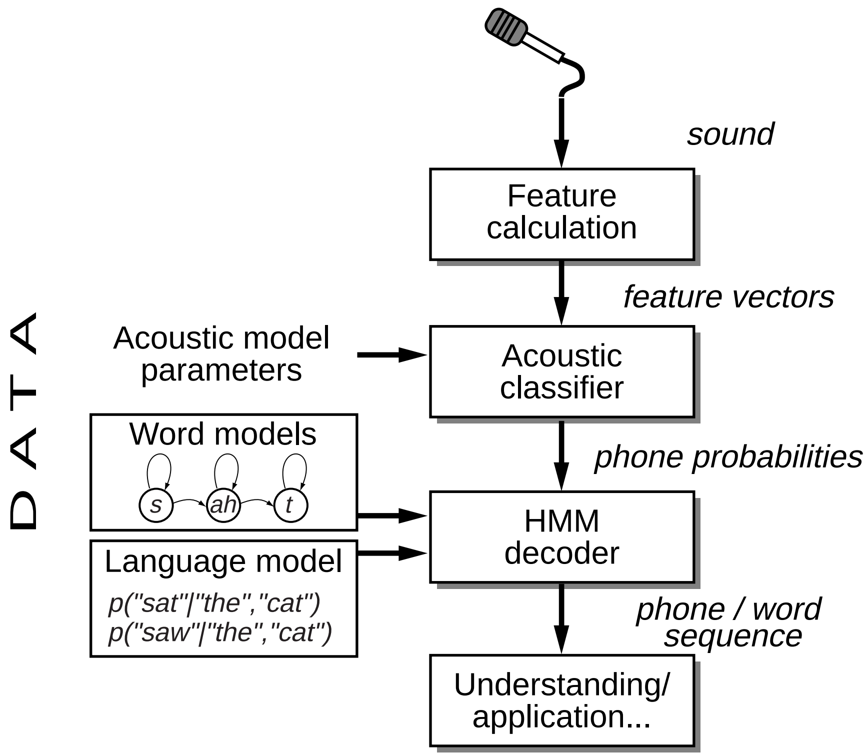
1. **Speech Recognition**
2. **Sequence Modeling**
3. **Hidden Markov Models**
4. **Chord Sequence Recognition**

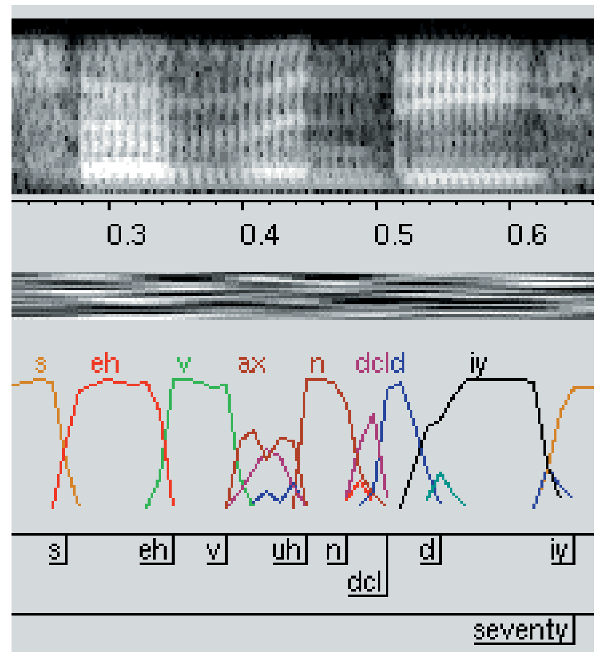
Dan Ellis <dpwe@ee.columbia.edu>
http://www.ee.columbia.edu/~dpwe/muscontent/

Laboratory for Recognition and Organization of Speech and Audio
Columbia University, New York
Spring 2003

1 Speech Recognition

- **Standard speech recognition**

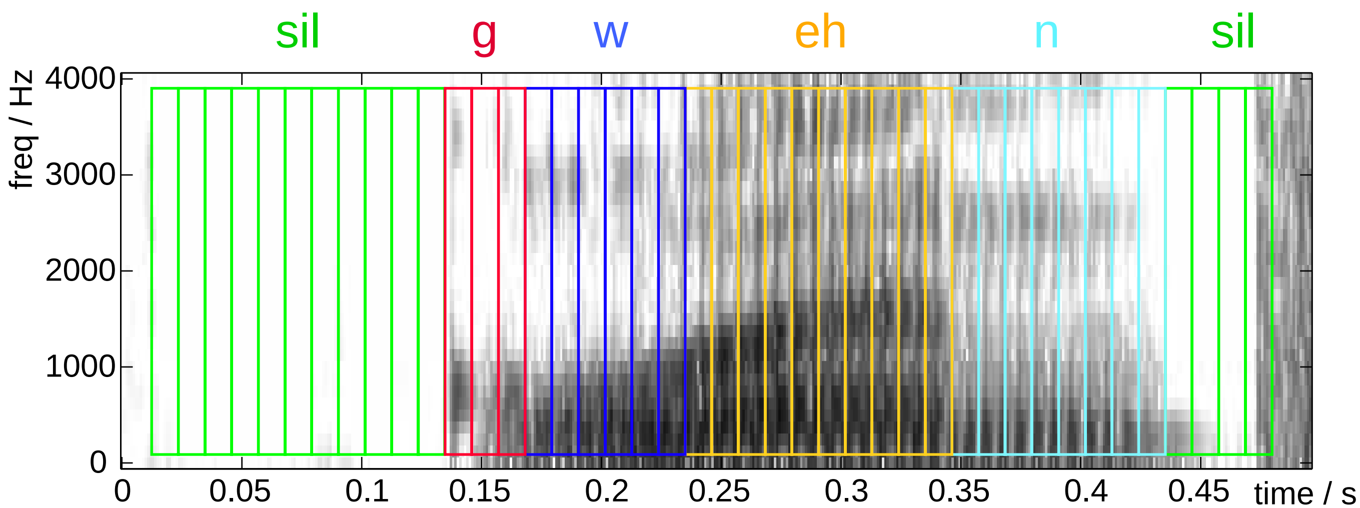




- **‘State of the art’ word-error rates (WERs):**
  - 2% (dictation) - 30% (telephone conversations)

Speech units

- **Speech is highly variable**
  - simple templates won't do
  - spectral variation (voice quality)
  - *time-warp* problems
- **Match short segments (states), allow repeats**
  - model with pseudo-stationary slices of ~ 10 ms



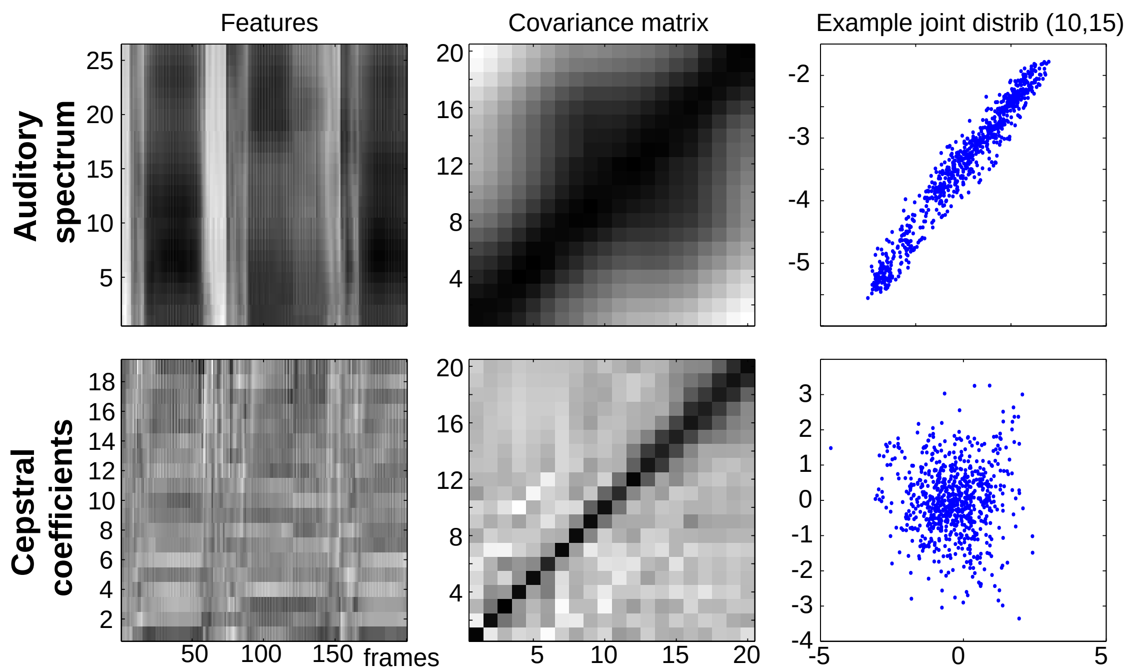
- **Speech models are distributions** $p(X|q)$

Speech features: Cepstra

- **Idea: Decorrelate & summarize spectral slices:**

$$X_m[l] = IDFT\{\log|S[mH, k]|\}$$

  - easier to model:

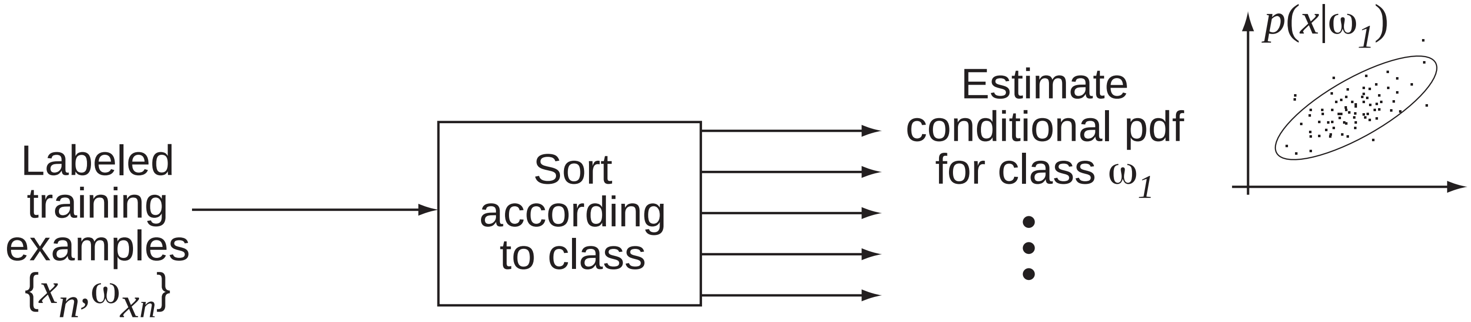


  - C_0 'normalizes out' average log energy

- **Decorrelated pdfs fit diagonal Gaussians**
  - DCT is close to PCA for log spectra

Acoustic model training

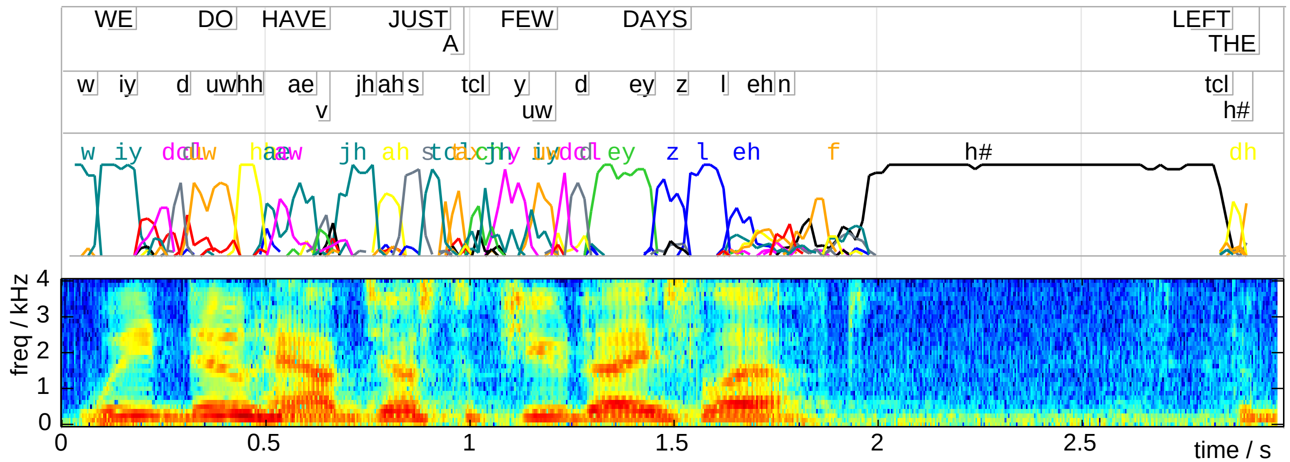
- **Goal: describe $p(X|q)$ with e.g. GMMs**



- **Training data labels from:**
  - manual phonetic annotation
  - ‘best path’ from earlier classifier (Viterbi)
  - EM: joint estimation of labels & pdfs

HMM decoding

- **Feature vectors cannot be reliably classified into phonemes**



- **Use top-down constraints to get good results**
  - allowable phonemes
  - dictionary of known words
  - grammar of possible sentences
- **Decoder searches all possible state sequences**
  - at least notionally; pruning makes it possible

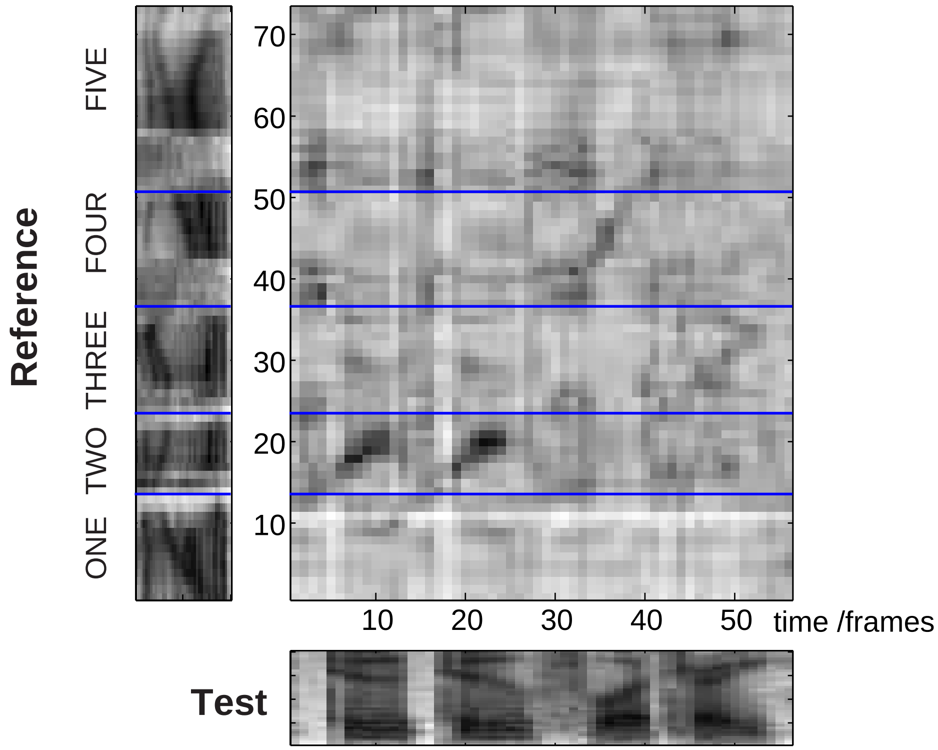
Outline

1. Speech Recognition
2. **Sequence Modeling**
   - Dynamic Time Warp
   - probabilistic framework
3. Hidden Markov Models
4. Chord Sequence Recognition

Sequence Modeling

2

- **Template matching:**
  Framewise comparison of input & templates:

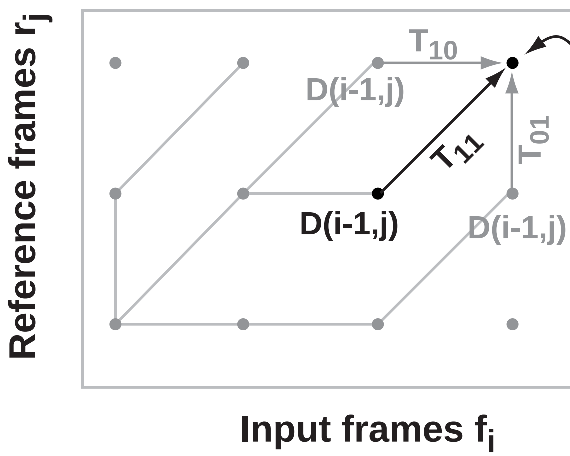


  - distance metric?
  - comparison between frames?
  - constraints?

Dynamic Time Warp (DTW)

- **Find lowest-cost constrained path:**
  - matrix d(i,j) of distances between input frame f_i and reference frame r_j
  - allowable predecessors & transition costs T_{xy}

Lowest cost to (i,j)



$$D(i,j) = d(i,j) + \min\left\{\begin{array}{l} D(i-1,j) + T_{10} \\ D(i,j-1) + T_{01} \\ D(i-1,j-1) + T_{11} \end{array}\right\}$$

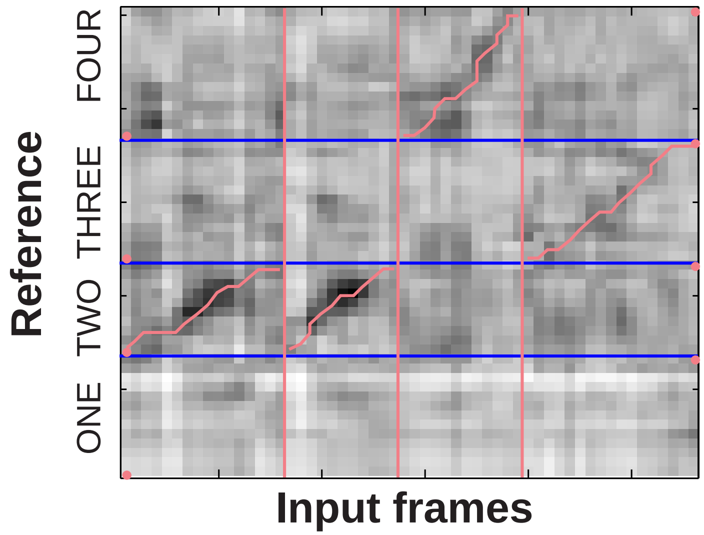
Local match cost

Best predecessor (including transition cost)

- **Best path via traceback from final state**
  - have to store predecessors for (almost) every (i,j)

DTW-based recognition

- **Reference templates for each possible word**
- **Isolated word:**
  - mark endpoints of input word
  - calculate scores through each template (+prune)
  - choose best
- **Continuous speech**
  - one matrix of template slices;
    special-case constraints at word ends



Lab ROSA
Laboratory for the Recognition and Organization of Speech and Audio

DTW-based recognition (2)

- **+ Successfully handles timing variation**
- **+ Able to recognize speech at reasonable cost**
- **- Distance metric?**
  - pseudo-Euclidean space?
- **- Warp penalties?**
- **- How to choose templates?**
  - several templates per word?
  - choose ‘most representative’?
  - align and average?

$\rightarrow$ **need a *rigorous* foundation...**

Statistical sequence recognition

- **DTW limited because it's hard to optimize**
  - interpretation of distance, transition costs?
- **Need a theoretical foundation: Probability**
- **Formulate recognition as MAP choice among models:**

$$M^* = \underset{M_j}{\operatorname{argmax}}\ p(M_j | X, \Theta)$$

  - X = observed features
  - M_j = word-sequence models
  - Θ = all current parameters

Statistical formulation (2)

- **Can rearrange via Bayes' rule (& drop $p(X)$):**

$$M^* = \underset{M_j}{\operatorname{argmax}}\ p(M_j | X, \Theta)$$

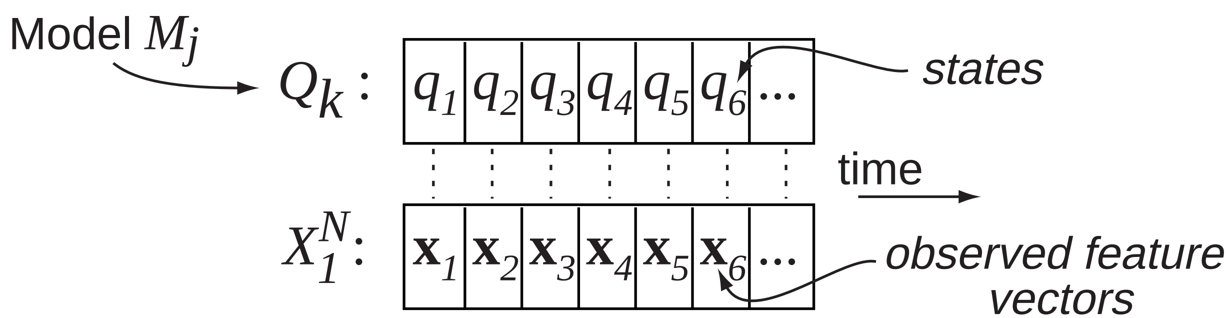
$$= \underset{M_j}{\operatorname{argmax}}\ p(X | M_j, \Theta_A) p(M_j | \Theta_L)$$

  - $p(X \mid M_j)$ = likelihood of obs'v'ns under model
  - $p(M_j)$ = prior probability of model
  - Θ_A = acoustics-related model parameters
  - Θ_L = language-related model parameters

- **Questions:**
  - what form of model to use for $p(X | M_j, \Theta_A)$?
  - how to find Θ_A (training)?
  - how to solve for M_j (decoding)?

State-based modeling

- **Assume** **discrete-state** **model for the speech:**
  - observations are divided up into time frames
  - model → states → observations:



- **Probability of observations given model is:**

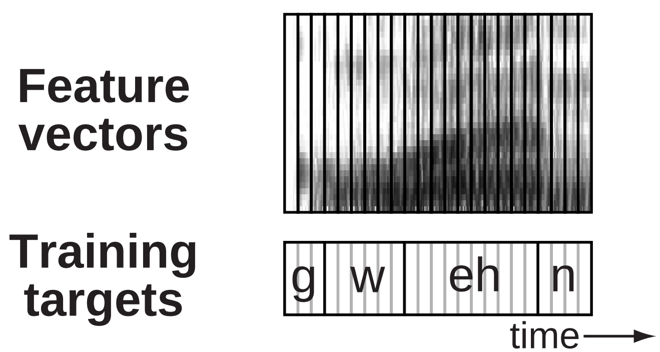
$$p(X|M_j) = \sum_{\text{all } Q_k} p(X_1^N \,\big|\, Q_k, M_j) \cdot p(Q_k \,\big|\, M_j)$$

  - sum over all possible state sequences Q_k

- **How do observations depend on states?**
  How do state sequences depend on model?
  What is 'in' the model?

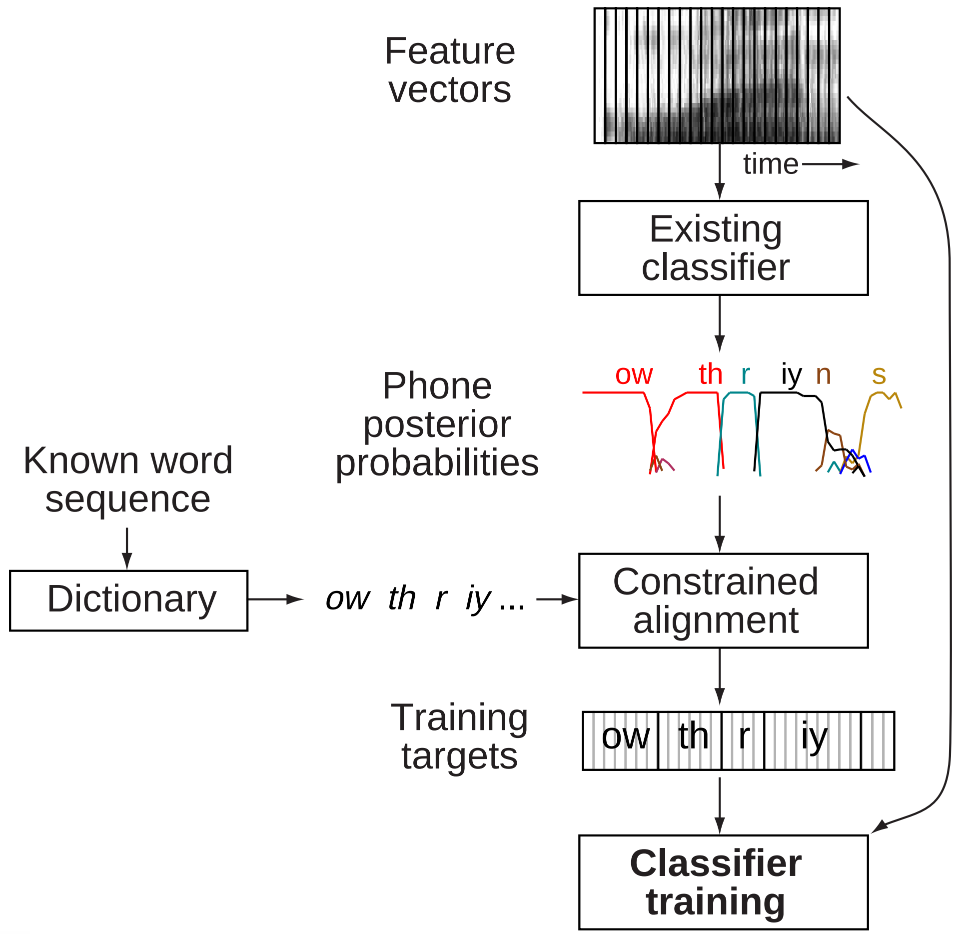
Defining classifier targets

- **Choice of {q^i} can make a big difference**
  - must support recognition task
  - must be a practical classification task
- **Hand-labeling is one source...**
  - 'experts' mark spectrogram boundaries
- **...Forced alignment is another**
  - 'best guess' with existing classifiers, given words
- **Result is *targets* for each training frame:**



Forced alignment

- **Best labeling given existing classifier constrained by known word sequence**



Gaussian Mixture Models vs. Neural Nets

- **GMMs fit distribution of features under states:**
  - *separate* 'likelihood' model for each state q^i

$$p(\mathbf{x}|q^k) = \frac{1}{(\sqrt{2\pi})^d |\Sigma_k|^{1/2}} \cdot \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x}-\mu_k)\right]$$

  - match any distribution given enough data

- **Neural nets estimate posteriors directly**

$$p(q^k|\mathbf{x}) = F[\sum_j w_{jk} \cdot F[\sum_j w_{ij} x_i]]$$

  - parameters set to *discriminate* classes

- **Posteriors & likelihoods related by Bayes' rule:**

$$p(q^k|\mathbf{x}) = \frac{p(\mathbf{x}|q^k) \cdot Pr(q^k)}{\sum_j p(\mathbf{x}|q^j) \cdot Pr(q^j)}$$

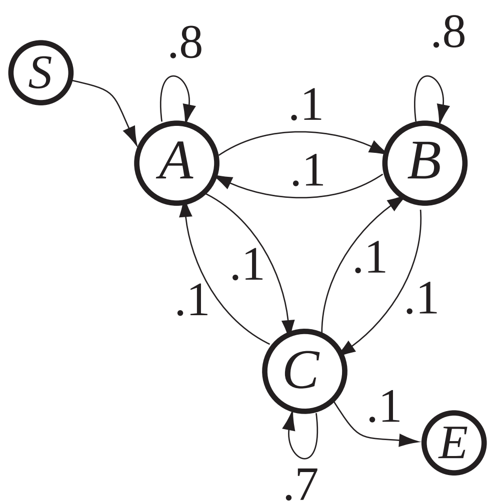
Outline

1. Speech Recognition
2. Sequence Modeling
3. **Hidden Markov Models**
   - generative HMMs
   - HMM model fitting
   - .. applied to Singing Detection
4. Chord Sequence Recognition

3

Hidden Markov Models

- **A (first order) Markov model is a finite-state system whose behavior depends *only on the current state***
- **E.g. *generative* Markov model:**

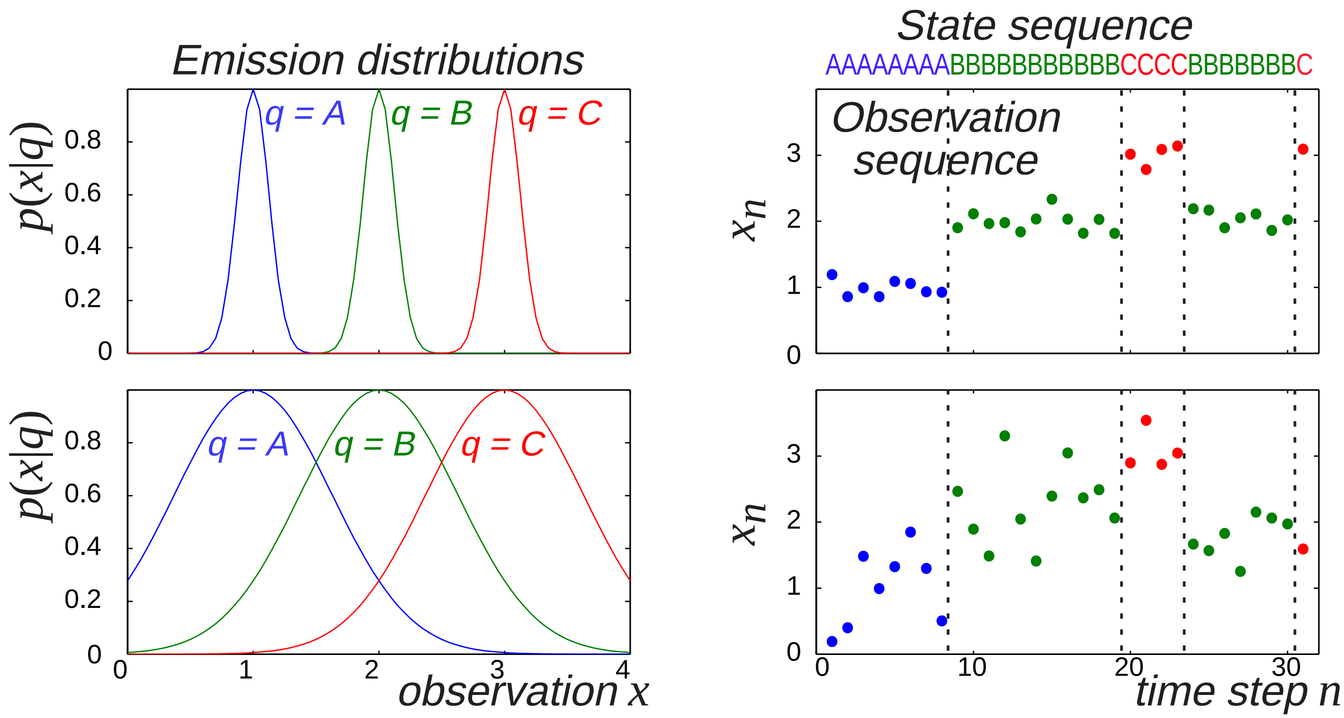


$p(q_{n+1}\mid q_n)$	q_{n+1}: S	A	B	C	E
q_n: S	0	1	0	0	0
A	0	.8	.1	.1	0
B	0	.1	.8	.1	0
C	0	.1	.1	.7	.1
E	0	0	0	0	1

S A A A A A A A A B B B B B B B B B C C C C B B B B B B C E

Hidden Markov models

- **= Markov model where state sequence $Q = \{q_n\}$ is not directly observable (= 'hidden')**
- **But, observations X *do* depend on Q:**
    - x_n is rv that depends on current state: $p(x|q)$



- can still tell *something* about state seq...

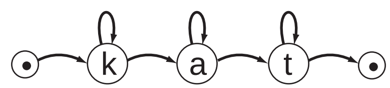
(Generative) Markov models (2)

- **HMM is specified by:**

- states q^i

- transition probabilities a_{ij}

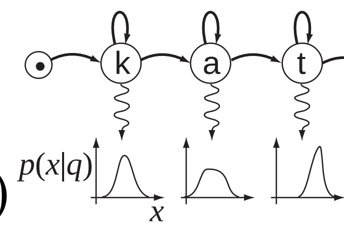
$$p(q_n^j | q_{n-1}^i) \equiv a_{ij}$$



	k	a	t	•
•	1.0	0.0	0.0	0.0
k	0.9	0.1	0.0	0.0
a	0.0	0.9	0.1	0.0
t	0.0	0.0	0.9	0.1

- emission distributions $b_i(x)$

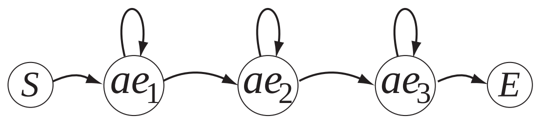
$$p(x | q^i) \equiv b_i(x)$$

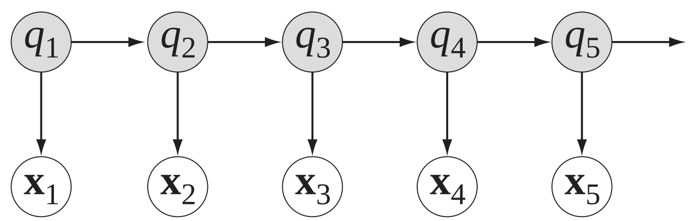


+ (initial state probabilities $p(q_1^i) \equiv \pi_i$)

Markov models for speech

- **Speech models** M_j
  - typ. left-to-right HMMs (sequence constraint)
  - observation & evolution are conditionally independent of rest given (hidden) state q_n





  - *self-loops* for time dilation

Markov models for sequence recognition

- **Independence of observations:**
  - observation x_n depends only current state q_n

$$p(X|Q) = p(x_1, x_2, \ldots x_N | q_1, q_2, \ldots q_N)$$

$$= p(x_1|q_1) \cdot p(x_2|q_2) \cdot \ldots p(x_N|q_N)$$

$$= \prod_{n=1}^{N} p(x_n|q_n) = \prod_{n=1}^{N} b_{q_n}(x_n)$$

- **Markov transitions:**
  - transition to next state q_{i+1} depends only on q_i

$$p(Q|M) = p(q_1, q_2, \ldots q_N | M)$$

$$= p(q_N|q_1 \ldots q_{N-1}) p(q_{N-1}|q_1 \ldots q_{N-2}) \ldots p(q_2|q_1) p(q_1)$$

$$= p(q_N|q_{N-1}) p(q_{N-1}|q_{N-2}) \ldots p(q_2|q_1) p(q_1)$$

$$= p(q_1) \prod_{n=2}^{N} p(q_n|q_{n-1}) = \pi_{q_1} \prod_{n=2}^{N} a_{q_{n-1} q_n}$$

Model fit calculation

- **From 'state-based modeling':**

$$p(X|M_j) = \sum_{\text{all } Q_k} p(X_1^N \big| Q_k, M_j) \cdot p(Q_k \big| M_j)$$

- **For HMMs:**

$$p(X|Q) = \prod_{n=1}^{N} b_{q_n}(x_n)$$

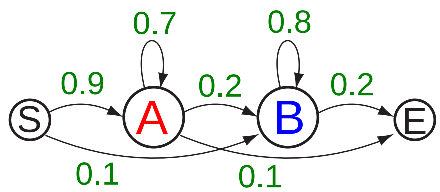
$$p(Q|M) = \pi_{q_1} \cdot \prod_{n=2}^{N} a_{q_{n-1} q_n}$$

- **Hence, solve for M^* :**
  - calculate $p(X|M_j)$ for each available model, scale by prior $p(M_j) \rightarrow p(M_j|X)$

- **Sum over *all* Q_k ???**

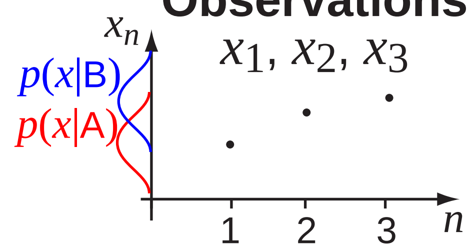
Summing over all paths

Model M_1



	S	A	B	E
S	•	0.9	0.1	•
A	•	0.7	0.2	0.1
B	•	•	0.8	0.2
E	•	•	•	1

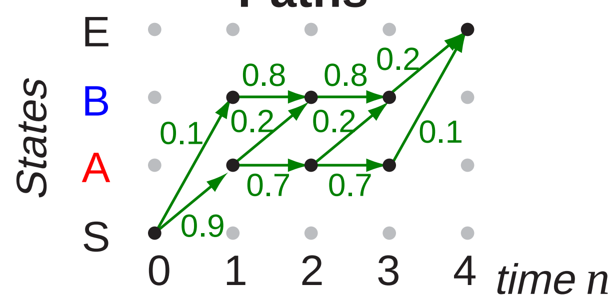
Observations



Observation likelihoods

$p(x\|q)$	x_1	x_2	x_3
q A	2.5	0.2	0.1
q B	0.1	2.2	2.3

Paths



All possible 3-emission paths Q_k from S to E

q_0	q_1	q_2	q_3	q_4	$p(Q \mid M) = \prod_n p(q_n \mid q_{n-1})$	$p(X \mid Q,M) = \prod_n p(x_n \mid q_n)$	$p(X,Q \mid M)$
S	A	A	A	E	.9 x .7 x .7 x .1 = **0.0441**	2.5 x 0.2 x 0.1 = 0.05	0.0022
S	A	A	B	E	.9 x .7 x .2 x .2 = 0.0252	2.5 x 0.2 x 2.3 = 1.15	0.0290
S	A	B	B	E	.9 x .2 x .8 x .2 = 0.0288	2.5 x 2.2 x 2.3 = 12.65	**0.3643**
S	B	B	B	E	.1 x .8 x .8 x .2 = 0.0128	0.1 x 2.2 x 2.3 = 0.506	0.0065
					Σ = 0.1109		$\boxed{\Sigma = p(X \mid M) = \mathbf{0.4020}}$

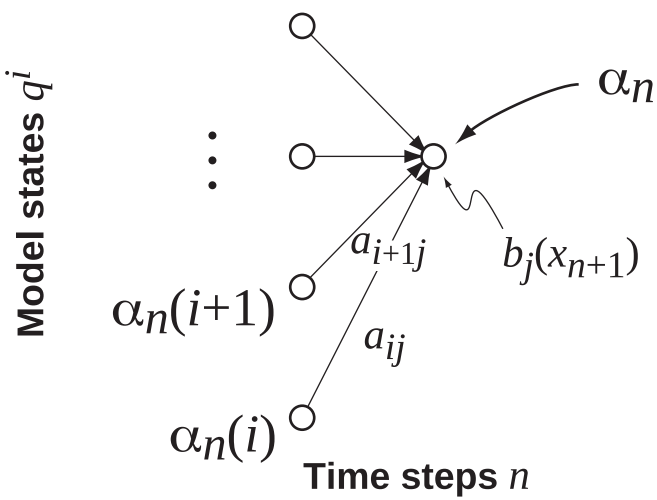
The ‘forward recursion’

- **Dynamic-programming-like technique to calculate sum over all** Q_k

- **Define** $\alpha_n(i)$ **as the probability of *getting to* state** q^i **at time step** n **(by any path):**

$$\alpha_n(i) = p(x_1, x_2, \ldots x_n, q_n = q^i) \equiv p(X_1^n, q_n^i)$$

- **Then** $\alpha_{n+1}(j)$ **can be calculated recursively:**



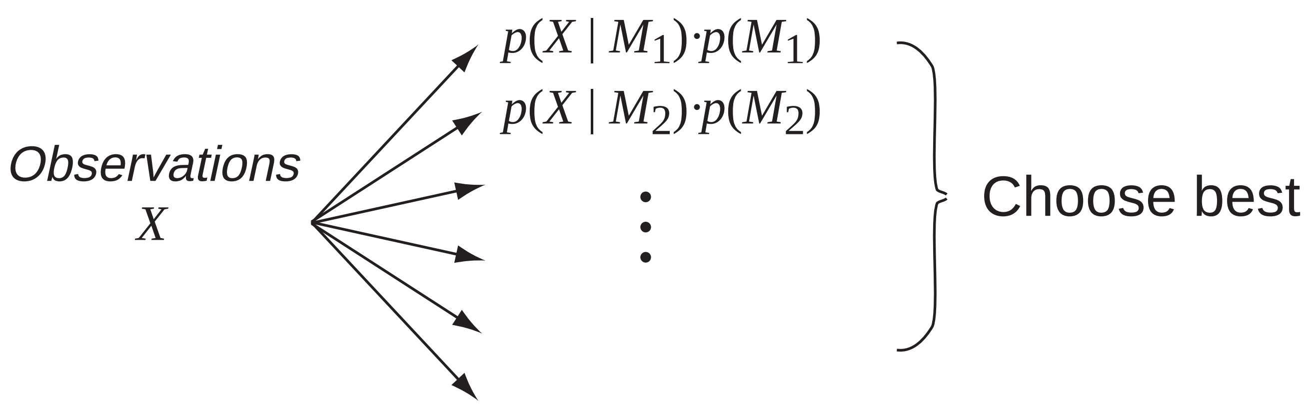
$$\alpha_{n+1}(j) = \left[\sum_{i=1}^{S} \alpha_n(i) \cdot a_{ij}\right] \cdot b_j(x_{n+1})$$

Forward recursion (2)

- **Initialize** $\alpha_1(i) = \pi_i \cdot b_i(x_1)$

- **Then total probability** $p(X_1^N | M) = \sum_{i=1}^{S} \alpha_N(i)$

→ **Practical way to solve for** $p(X \mid M_j)$ **and hence perform recognition**



Optimal path

- **May be interested in actual q_n assignments**
  - which state was 'active' at each time frame
  - e.g. phone labelling (for training?)
- **Total probability is over *all* paths...**
- **... but can also solve for single *best* path = "Viterbi" state sequence**
- **Probability along best path to state q_{n+1}^j:**

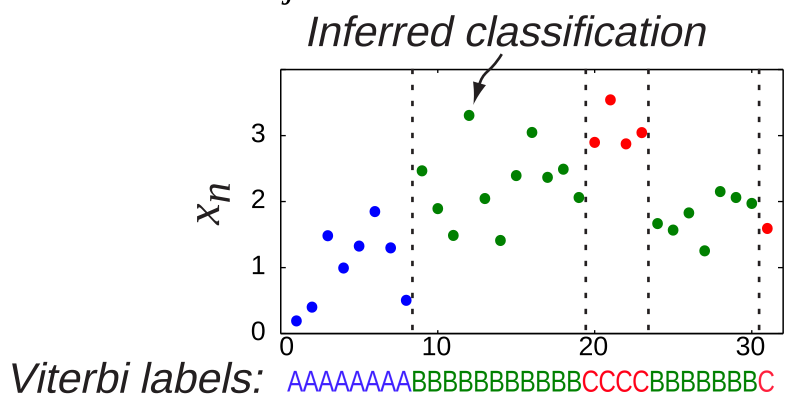
$$\alpha_{n+1}^{*}(j) = \left[\max_i \left\{\alpha_n^{*}(i) a_{ij}\right\}\right] \cdot b_j(x_{n+1})$$

  - backtrack from final state to get best path
  - final probability is product only (no sum) $\rightarrow$ log-domain calculation just summation
- **Total probability often dominated by best path:**

$$p(X, Q^{*} | M) \approx p(X|M)$$

Interpreting the Viterbi path

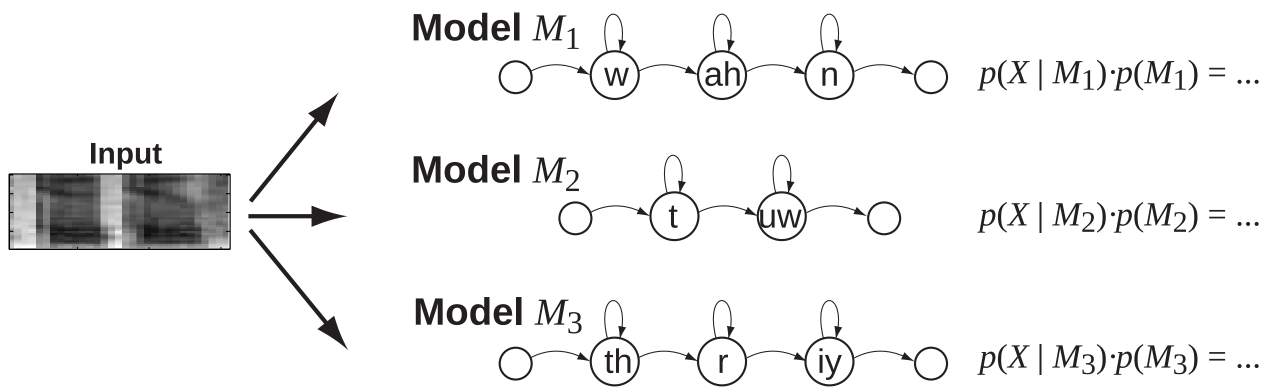
- **Viterbi path assigns each x_n to a state q^i**
  - performing classification based on $b_i(x)$
  - ... at the same time as applying transition constraints a_{ij}



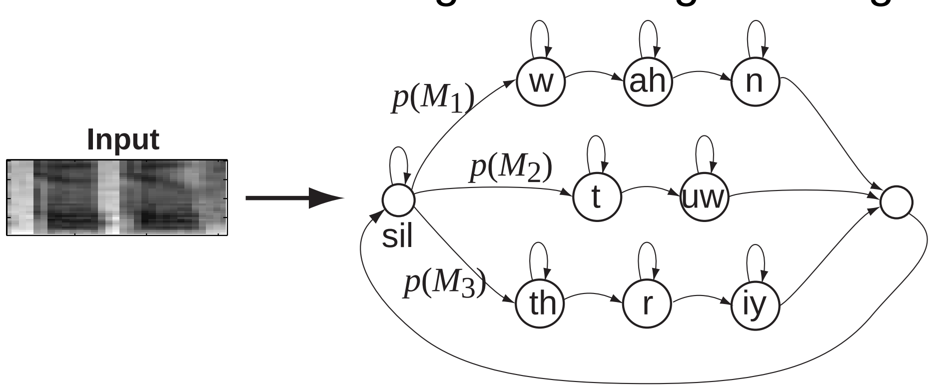
- **Can be used for segmentation**
  - train an HMM with 'garbage' and 'target' states
  - decode on new data to find 'targets', boundaries
- **Can use for (heuristic) training**
  - e.g. train classifiers based on labels...

Recognition with HMMs

- **Isolated word**
  - choose best $p(M|X) \propto p(X|M)p(M)$

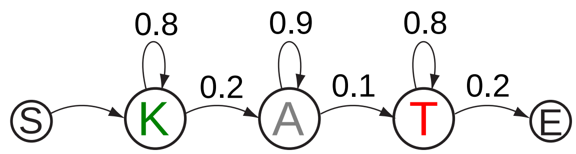


- **Continuous speech**
  - Viterbi decoding of one large HMM gives words

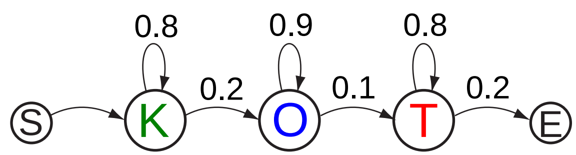


HMM example: Different state sequences

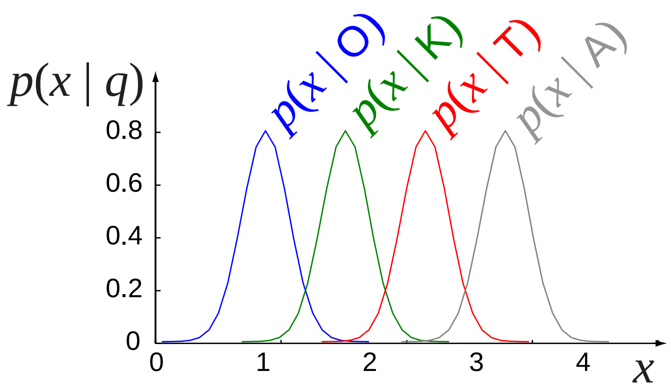
Model M_1



Model M_2

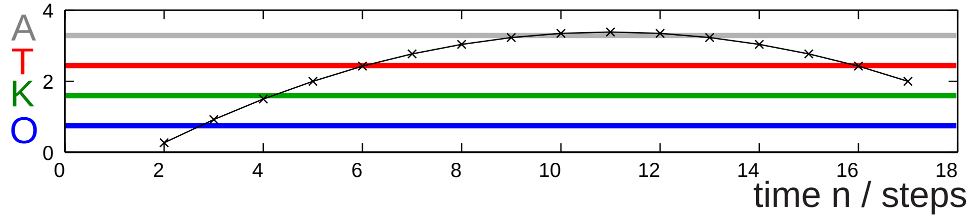


Emission distributions



Model inference: Emission probabilities

Observation sequence x_n



Model M_1

$\log p(X \mid M) = -32.1$

state alignment

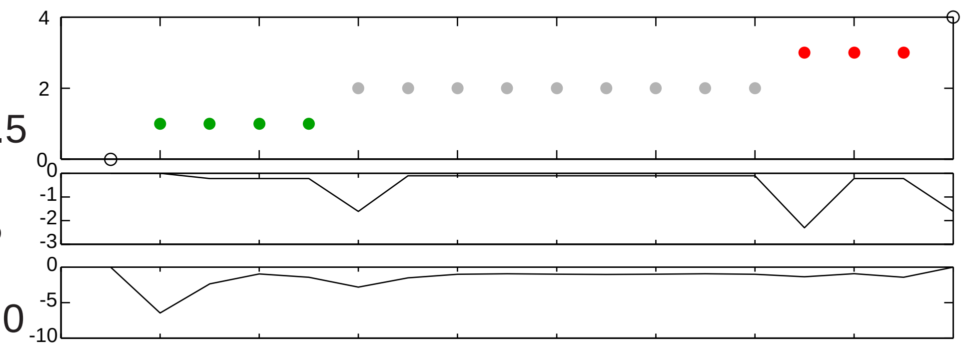
$\log p(X,Q^* \mid M) = -33.5$

log trans.prob

$\log p(Q^* \mid M) = -7.5$

log obs.l'hood

$\log p(X \mid Q^*,M) = -26.0$



Model M_2

$\log p(X \mid M) = -47.0$

state alignment

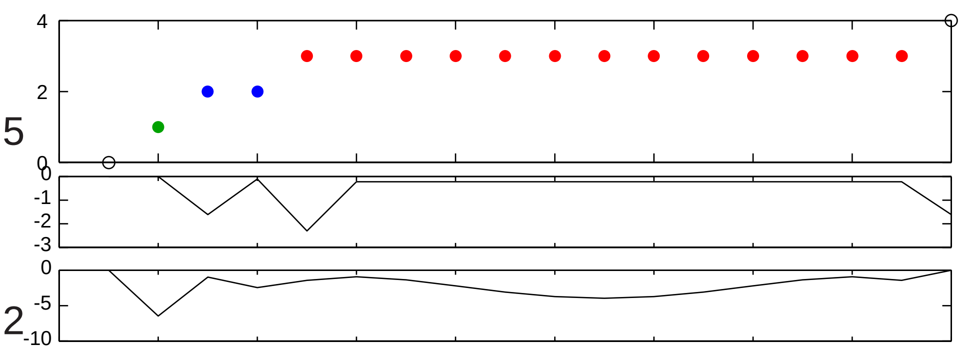
$\log p(X,Q^* \mid M) = -47.5$

log trans.prob

$\log p(Q^* \mid M) = -8.3$

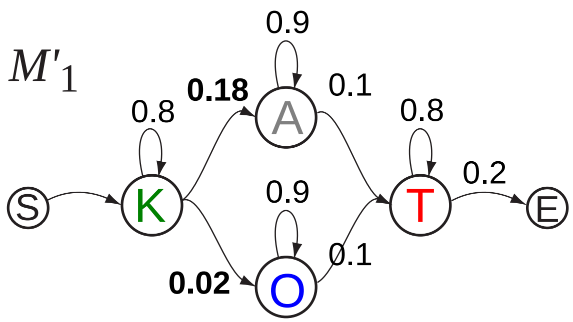
log obs.l'hood

$\log p(X \mid Q^*,M) = -39.2$

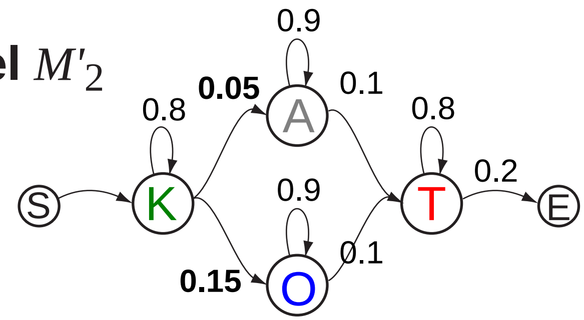


Model inference: Transition probabilities

Model M'_1



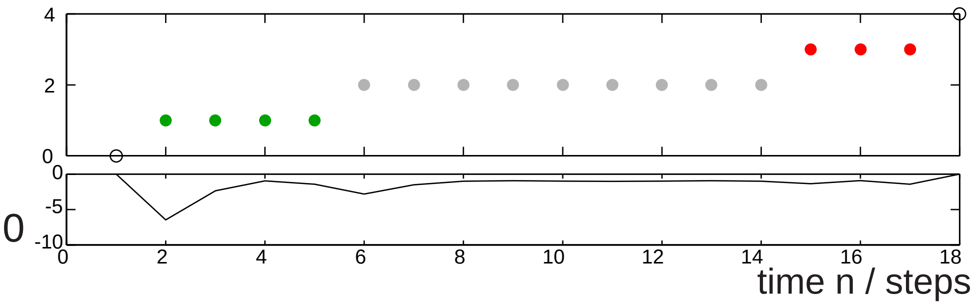
Model M'_2



state alignment

log obs.l'hood

$\log p(X \mid Q^*, M) = -26.0$

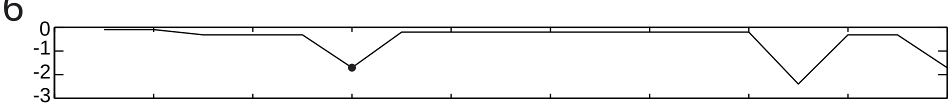


Model M'_1 $\log p(X \mid M) = -32.2$

$\log p(X, Q^* \mid M) = -33.6$

log trans.prob

$\log p(Q^* \mid M) = -7.6$

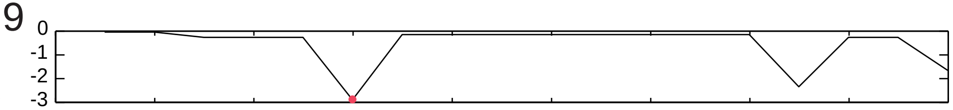


Model M'_2 $\log p(X \mid M) = -33.5$

$\log p(X, Q^* \mid M) = -34.9$

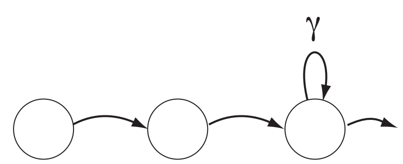
log trans.prob

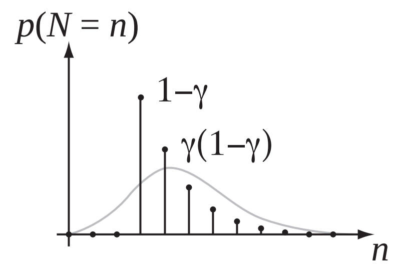
$\log p(Q^* \mid M) = -8.9$



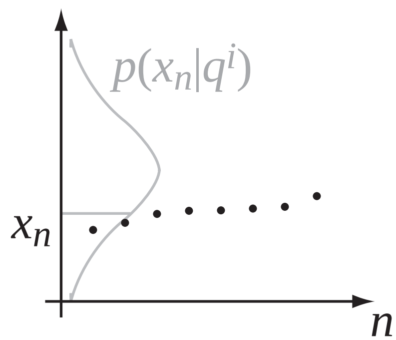
Validity of HMM assumptions

- **Key assumption is *conditional independence*:**
  Given q^i, future evolution & obs. distribution are independent of previous events
  - duration behavior: self-loops imply exponential distribution





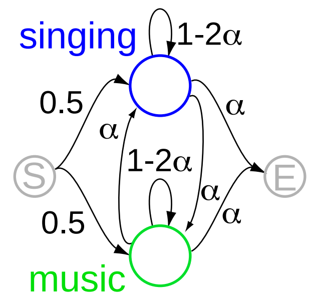
  - independence of successive x_ns



$$p(X) = \prod p(x_n|q^i) \text{ ?}$$

HMMs for Singing Detection

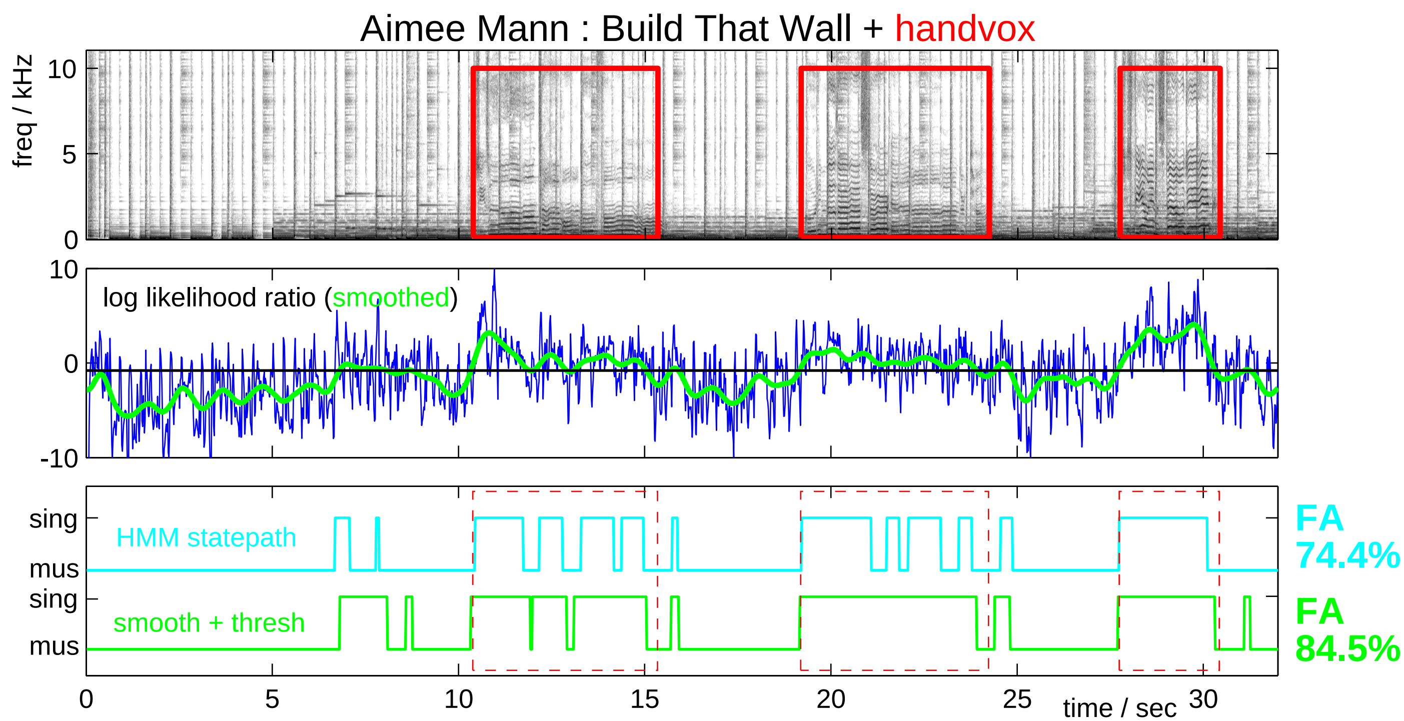
- **Previous singing detection system made noisy frame-level classifications**
  - smoothing greatly improved frame accuracy
  - but: boundary resolution blurred
- **HMM solves for best boundary location without blurring**



  - Viterbi path backtrace is best-l'hood state seq.
- **Improves:**
  - Frame accuracy?
  - boundary accuracy
  - number of segments

Singing Detection HMM results

- **GMMs from last session used as emission models**



- **HMM has lower frame accuracy...**
  - ... but ground-truth labels are suspect?
  - boundaries look sharper

HMM Model Training

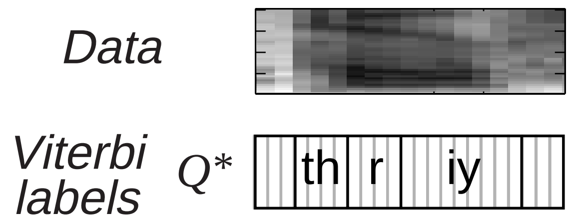
- **Probabilistic foundation allows us to *train* HMMs to 'fit' training data**
  - i.e. estimate a_{ij}, $b_i(x)$ given interpretations
  - better than DTW...
- **Algorithms to improve $p(M \mid X)$ are key to success of HMMs**
  - maximum-likelihood of models...
- **Problem arises because state alignments Q of training examples are generally unknown**

→ **Viterbi training**
  - choose 'best' labels (heuristic)

→ **EM training**
  - 'fuzzy labels' (guaranteed local convergence)

Viterbi training

- **"Fit models to data"**
  = Viterbi best-path alignment



- **"Re-estimate model parameters":**

  pdf e.g. 1D Gauss: $\mu_i = \dfrac{\sum_{n \in q^i} x_n}{\#(q_n^i)}$

  count transitions: $a_{ij} = \dfrac{\#(q_{n-1}^i \rightarrow q_n^j)}{\#(q_n^i)}$

- **And repeat...**

- **But: converges only if good initialization**

EM for HMMs

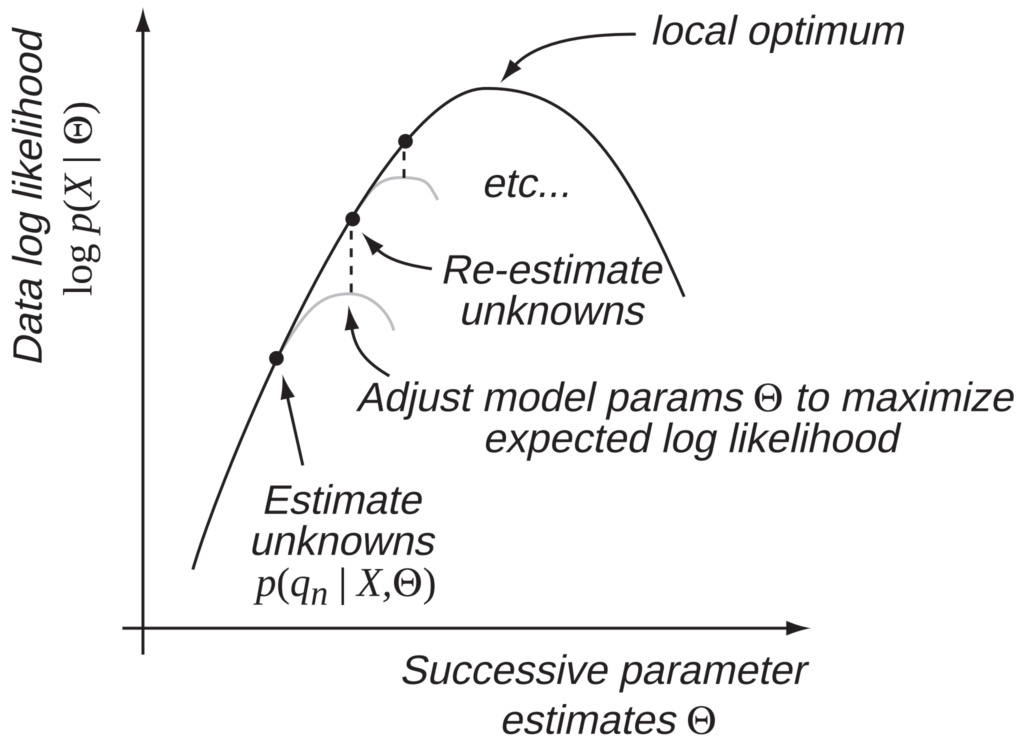
- **Expectation-Maximization (EM): optimizes models with unknown parameters**
  - finds locally-optimal parameters Θ to maximize data likelihood $p(x_{train}|\Theta)$
  - makes sense for decision rules like $p(x|M_j) \cdot p(M_j)$
- **Principle: adjust Θ to maximize expected log likelihood of known x & unknown u:**

$$E[\log p(x, u|\Theta)] = \sum_u p(u|x, \Theta_{old}) \log[p(x|u, \Theta) p(u|\Theta)]$$

  - for GMMs, unknowns = mix assignments k
  - for HMMs, unknowns = hidden state q_n (take Θ to include M_j)
- **Interpretation: "fuzzy" values for unknowns**

What EM does

- **Maximize data likelihood by repeatedly estimating unknowns and re-maximizing expected log likelihood:**



EM update equations

- **For acoustic model (e.g. 1-D Gauss):**

$$\mu_i = \frac{\sum_n p(q_n^i | X, \Theta_{\text{old}}) \cdot x_n}{\sum_n p(q_n^i | X, \Theta_{\text{old}})}$$

- **For transition probabilities:**

$$p(q_n^j | q_{n-1}^i) = a_{ij}^{\text{new}} = \frac{\sum_n p(q_{n-1}^i, q_n^j | X, \Theta_{\text{old}})}{\sum_n p(q_{n-1}^i | X, \Theta_{\text{old}})}$$

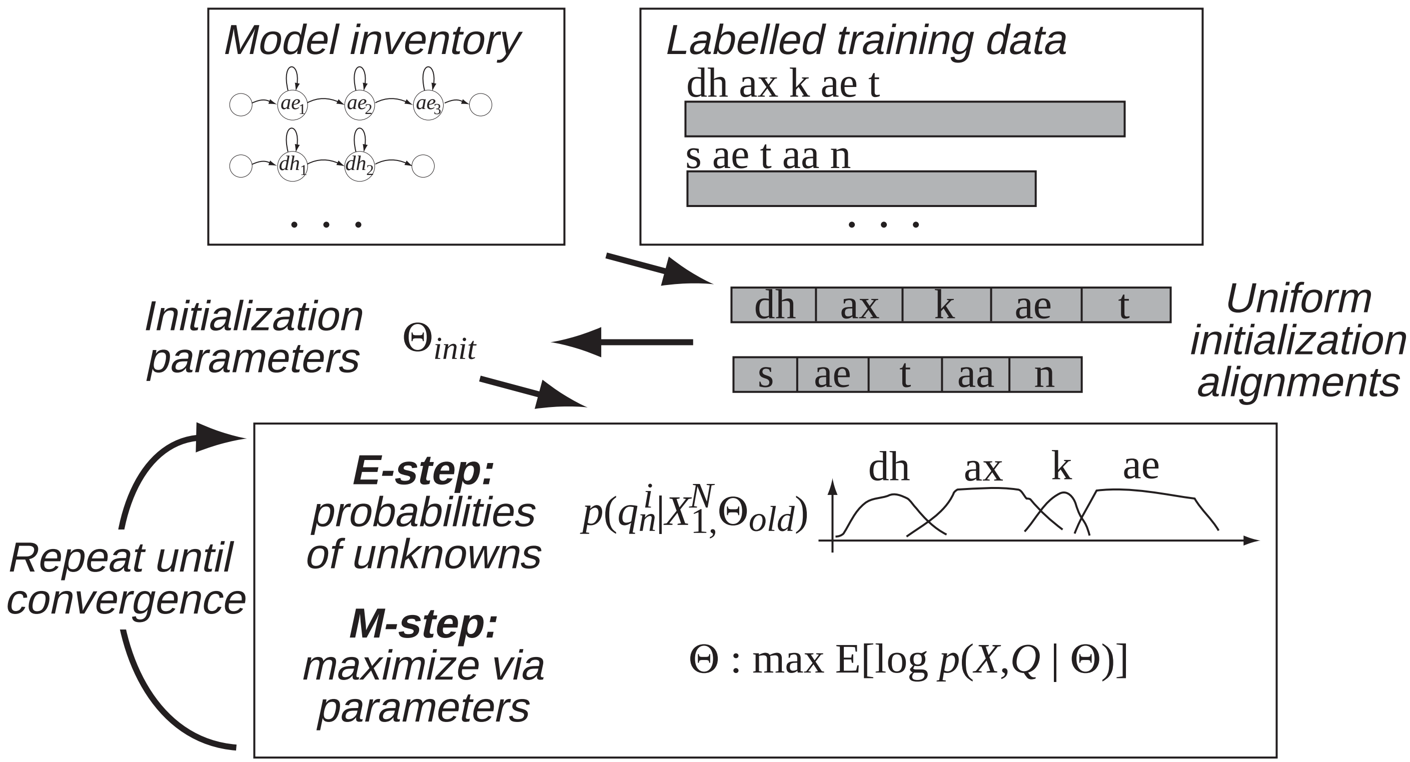
- **Fuzzy versions of Viterbi training**
  - reduce to Viterbi if $p(q|X) = 1/0$
- **Require 'state occupancy probabilities',**

$$p(q_n^i | X_1^N, \Theta_{\text{old}})$$

  - calculated by the "forward-backward" algorithm

HMM training in practice

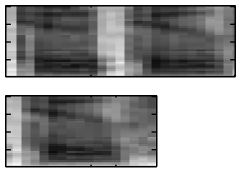
- **EM only finds local optimum**
  → critically dependent on initialization
  - approximate parameters / rough alignment



- **Applicable for more than just words...**

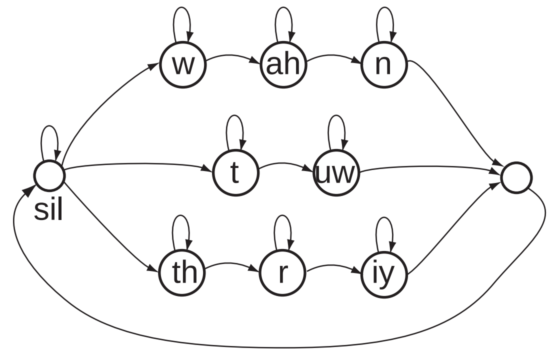
Training summary

- **Training data + basic model topologies**
  → derive fully-trained models



TWO ONE

FIVE



ONE = w ah n
TWO = t uw

  - alignment all handled implicitly

- **What do the states end up *meaning*?**
  - not necessarily what you intended;
    whatever locally maximizes data likelihood

- **What if the models or transcriptions are bad?**
  - slow convergence, poor discrimination in models

- **Other kinds of data, transcriptions**
  - less constrained initial models...

Outline

1. Speech Recognition
2. Sequence Modeling
3. Hidden Markov Models
4. **Chord Sequence Recognition**
    - Data and features
    - EM training
    - Results

4 Chord Sequence Recognition

(Alex Sheh)

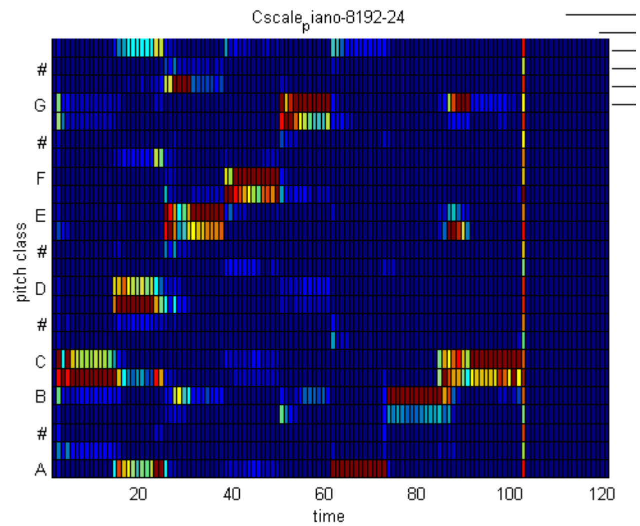
- **HMM technology assigns a single state to each time frame**
  - e.g. phone (or other subword)
  - solutions for multiple states exist, but are much harder to use
- **Analogs of word transcription in music?**
  - notes? because many overlap at onces
  - chords: unique, global property at each time
- **Chord transcription is an interesting task**
  - highly characteristic of a piece of music
  - chord labels are somewhat ambiguous
    ... but spoken words can be too

Chord Recognition: Features

- **Chord identity relies on fine pitch structure**
- **Cepstra are designed to hide pitch**
  - ... because pitch is not useful in speech recog.
  - ... so cepstra are unlikely to be useful here
- **"Pitch Class Profile" (Fujishima 1999) maps FFT into chroma equivalence bins**

$$b[i] = \sum_{k \in B_i} |X[k]|^2$$

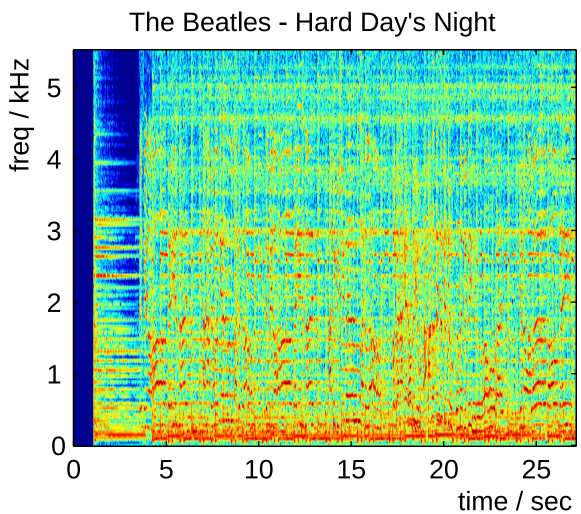
$$\langle \lfloor 24 \cdot \log 2(f_k) \rfloor \rangle_{24} = i$$



- **Or something better: e.g. subharmonics from autocorrelation?**

Chord Recognition: Data

- **Could train GMMs from hand-marked chords**
  - but: no data available, no fun to annotate...
- **The speech recognition training procedure:**
  - training data: utterances + word, no timings
  - use EM to figure out the best alignments
- **Try this for chords:**
  - start with audio + chord sequence only
  - EM defines both boundaries and class models




```
# The Beatles - A Hard Day's Night
#
G Cadd9 G F6 G Cadd9 G F6 G C D G C9 G
G Cadd9 G F6 G Cadd9 G F6 G C D G C9 G
Bm Em Bm G Em C D G Cadd9 G F6 G Cadd9 G
 F6 G C D G C9 G D
G C7 G F6 G C7 G F6 G C D G C9 G Bm Em Bm
 G Em C D
G Cadd9 G F6 G Cadd9 G F6 G C D G C9 G
C9 G Cadd9 Fadd9
```

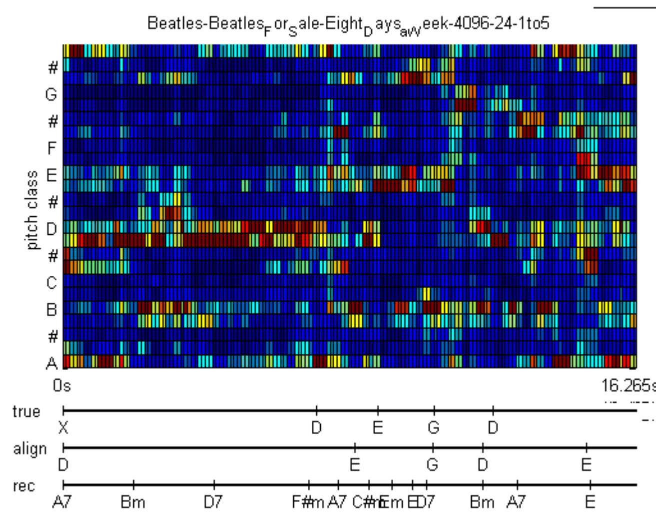
(Dataset: 20 early Beatles tracks)

Chord Recognition: Results

- **How well do the features model chords? Chord-label frame accuracies:**

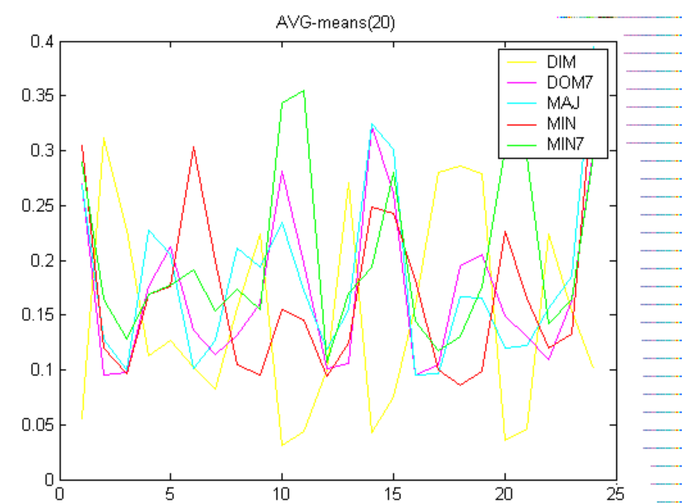
features	*Forced alignment frame accuracy*	*Recognition frame accuracy*
Cepstra + deltas	22%	18%
PCP-24	47%	25%
PCP-24-rot	75%	18%

- when forced, aligns chords OK
  but can't recognize them!



Chord Recognition: Models

- **Best feature was PCP-rot**
  - shared models for Major, Minor etc. chords (suitably rotated to align root)
    → better generalization, more data per model
- **Means of GMMs show what was learned:**



- **Future work:**
  - more training data
  - different chord classes?
  - better features

Summary

- **The time dimension complicates pattern recognition**
- **Hidden Markov Models provide a rigorous, trainable (if simplistic) foundation**
- **Advanced techniques from speech recognition can be applied to music**

What else can we do? How about real problems?

References

Takuya Fujishima (1999)
"Realtime Chord Recognition of Musical Sound: A System Using Common Lisp Music", Proc. ICMC, Beijing, Oct. 1999.

Ben Gold & Nelson Morgan (2000)
Speech and Audio Signal Processing: Processing and perception of speech and music,
Wiley, 2000.