INTERNATIONAL AUDIO LABORATORIES ERLANGEN

AUDIO LABS

Lecture

Music Processing

Chord Recognition

Meinard Müller and Christof Weiß

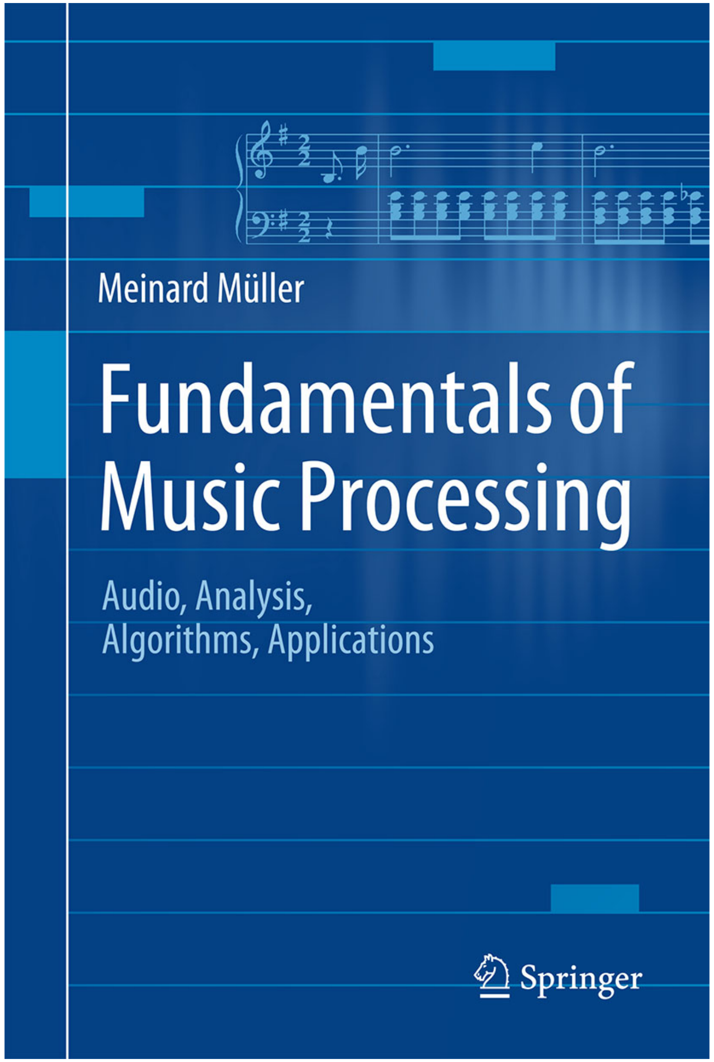
International Audio Laboratories Erlangen

meinard.mueller@audiolabs-erlangen.de

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Fraunhofer IIS

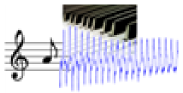
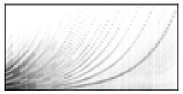
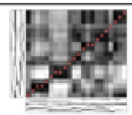
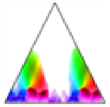
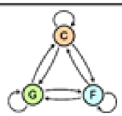
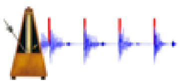
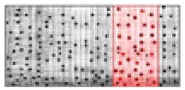
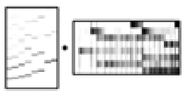
Book: Fundamentals of Music Processing



Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

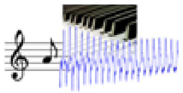
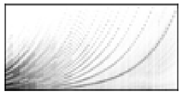
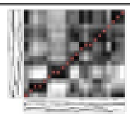
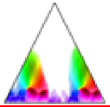
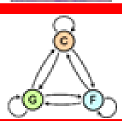
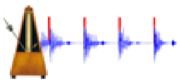
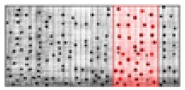
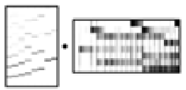
Book: Fundamentals of Music Processing

Chapter		Music Processing Scenario
1		Music Represenations
2		Fourier Analysis of Signals
3		Music Synchronization
4		Music Structure Analysis
5		Chord Recognition
6		Tempo and Beat Tracking
7		Content-Based Audio Retrieval
8		Musically Informed Audio Decomposition

Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

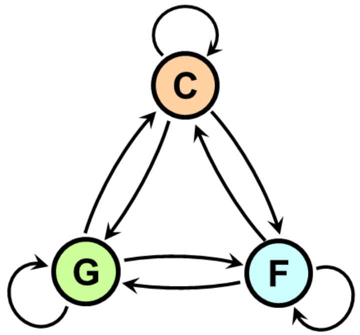
Book: Fundamentals of Music Processing

Chapter		Music Processing Scenario
1		Music Representations
2		Fourier Analysis of Signals
3		Music Synchronization
4		Music Structure Analysis
5		Chord Recognition
6		Tempo and Beat Tracking
7		Content-Based Audio Retrieval
8		Musically Informed Audio Decomposition

Meinard Müller
Fundamentals of Music Processing
Audio, Analysis, Algorithms, Applications
483 p., 249 illus., hardcover
ISBN: 978-3-319-21944-8
Springer, 2015

Accompanying website:
www.music-processing.de

Chapter 5: Chord Recognition

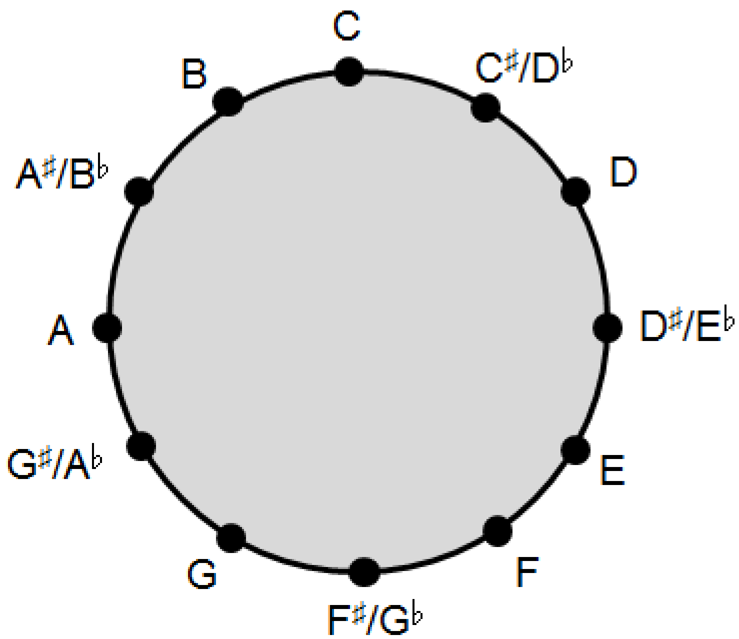


In Chapter 5, we consider the problem of analyzing harmonic properties of a piece of music by determining a descriptive progression of chords from a given audio recording. We take this opportunity to first discuss some basic theory of harmony including concepts such as intervals, chords, and scales. Then, motivated by the automated chord recognition scenario, we introduce template-based matching procedures and hidden Markov models—a concept of central importance for the analysis of temporal patterns in time-dependent data streams including speech, gestures, and music.

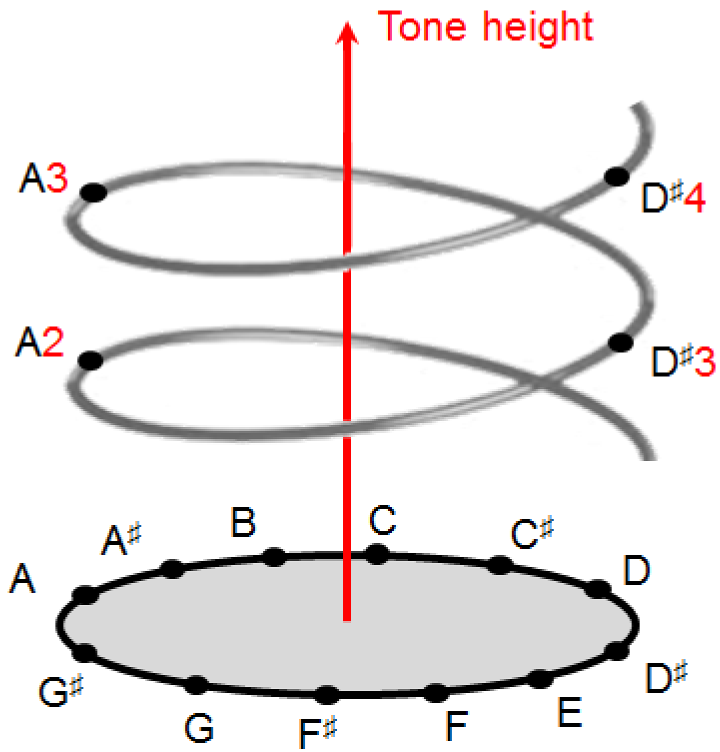
Recall: Chroma Features

- Human perception of pitch is periodic
- Two components: tone height (octave) and chroma (pitch class)

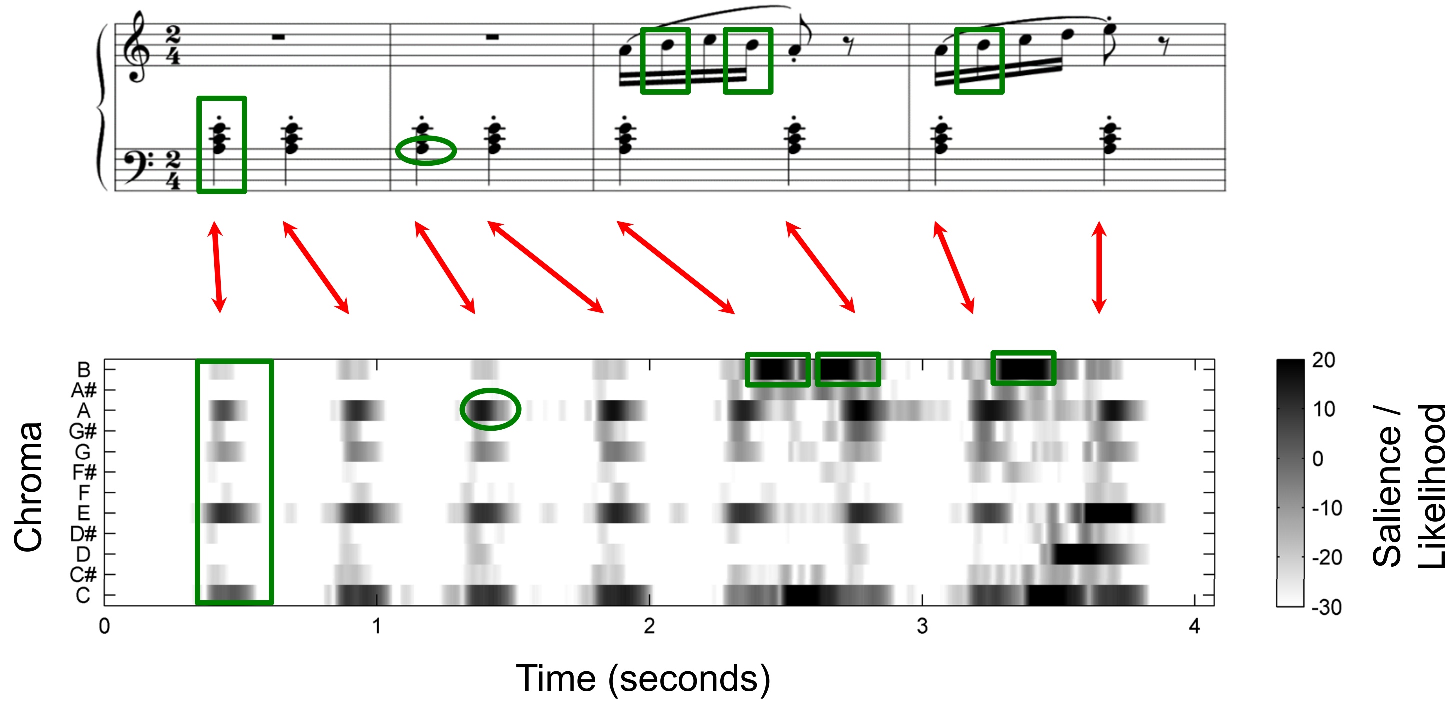
Chromatic circle



Shepard's helix of pitch



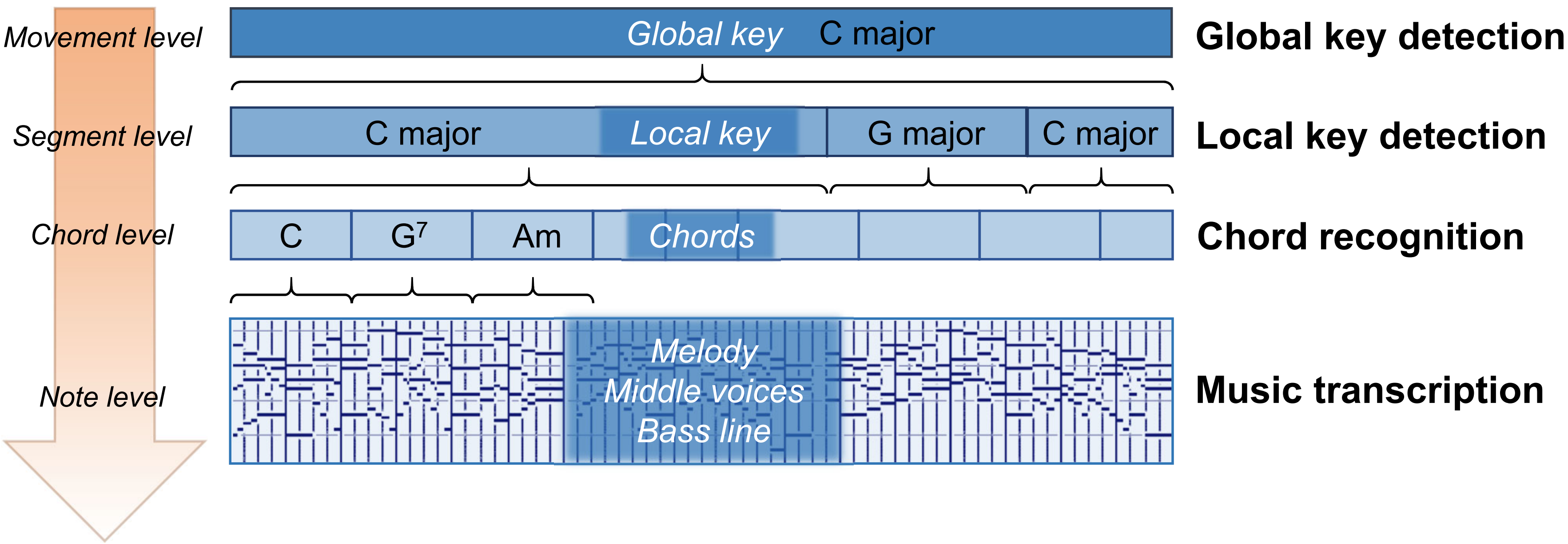
Recall: Chroma Features



→ capture harmonic progression

Harmony Analysis: Overview

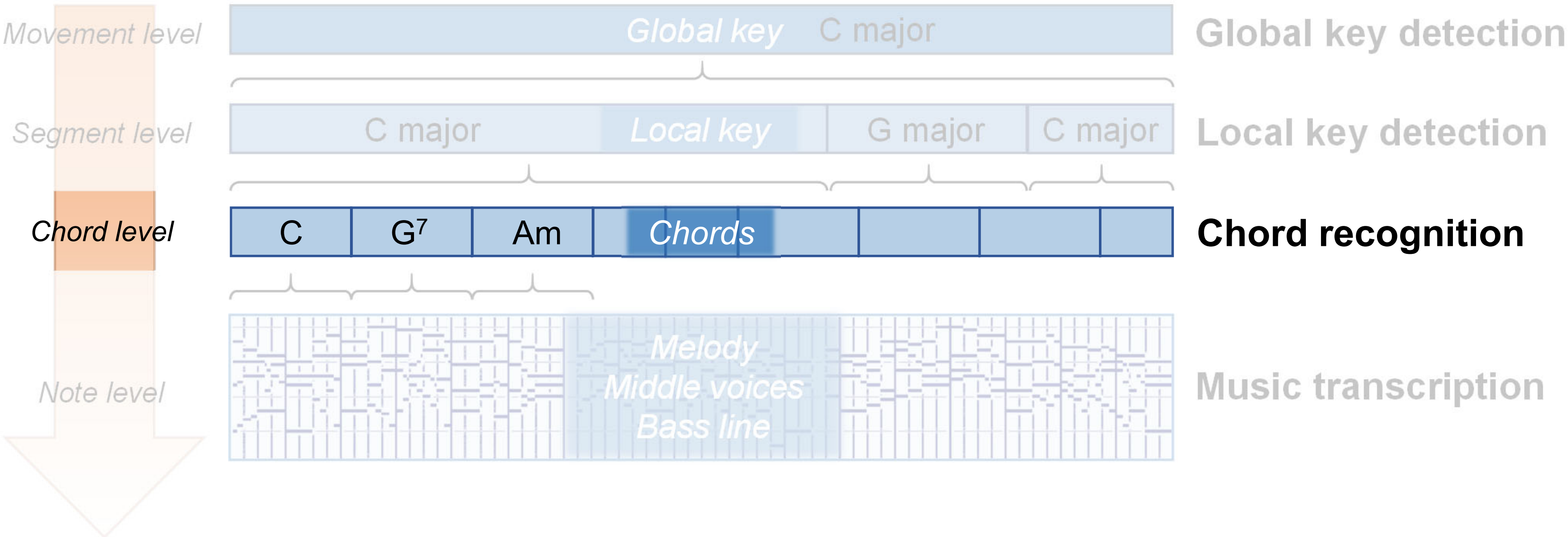
- Western music (and most other music): Different aspects of harmony
- Referring to different time scales



Christof Weiß: Computational Methods for Tonality-Based Style Analysis of Classical Music Audio Recordings, PhD thesis, Ilmenau University of Technology, 2017

Harmony Analysis: Overview

- Western music (and most other music): Different aspects of harmony
- Referring to different time scales



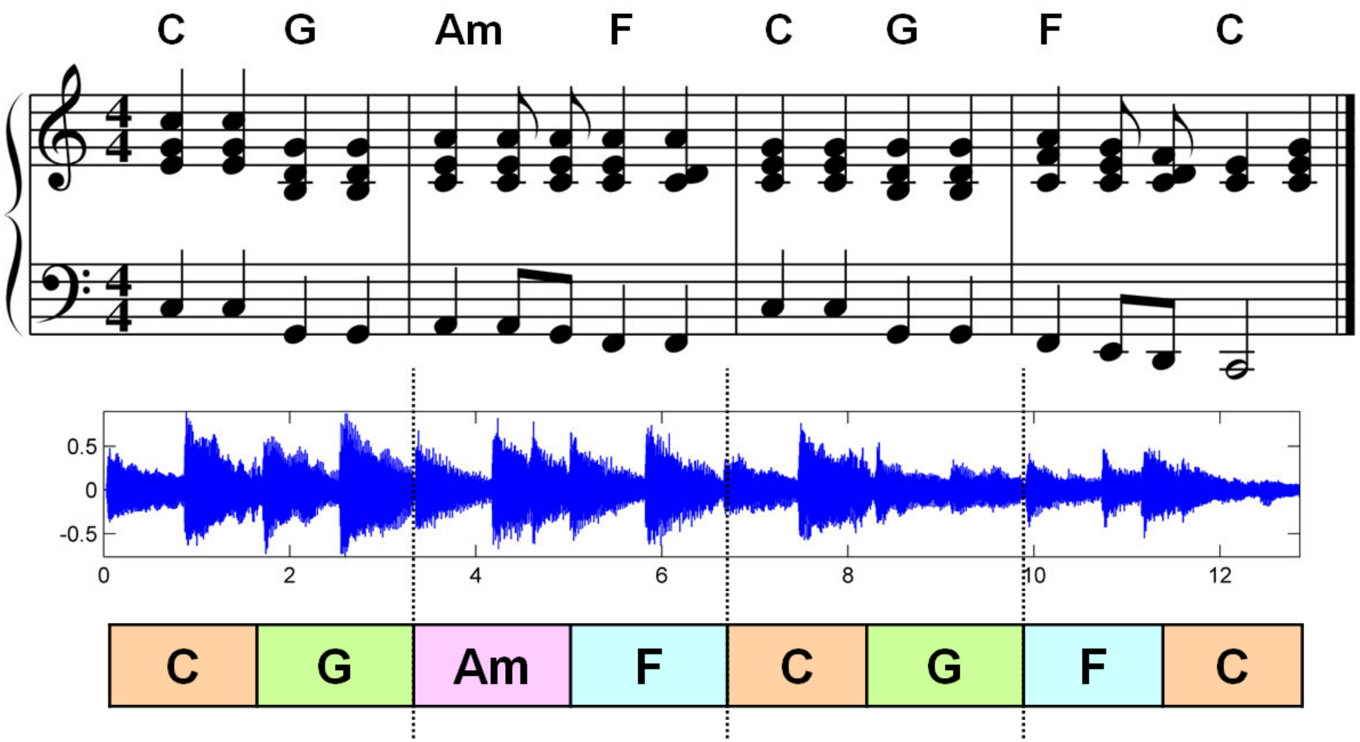
Christof Weiß: Computational Methods for Tonality-Based Style Analysis of Classical Music Audio Recordings, PhD thesis, Ilmenau University of Technology, 2017

Chord Recognition

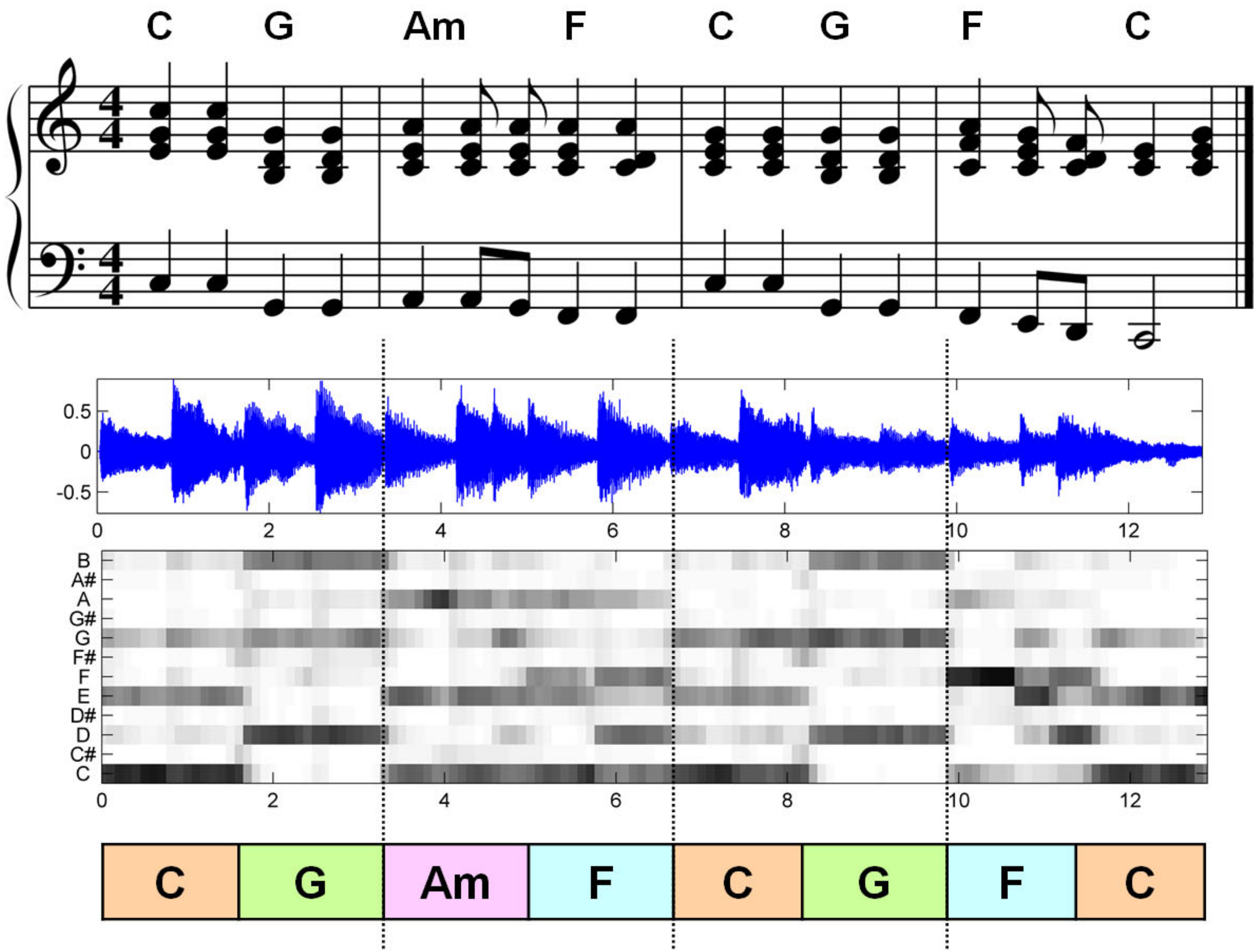
```
Let It Be chords
The Beatles 1970  (Let It Be)

[Intro]

C G Am F C G
F C Dm C

[Verse 1]

       C              G                 Am          F
When I find myself in times of trouble, Mother Mary comes to me
C                 G              F C Dm C
Speaking words of wisdom, let it be

    C             G                 Am                 F
And in my hour of darkness, she is standing right in front of me
C                 G              F C Dm C
Speaking words of wisdom, let it be

[Chorus]
```

Source: www.ultimate-guitar.com

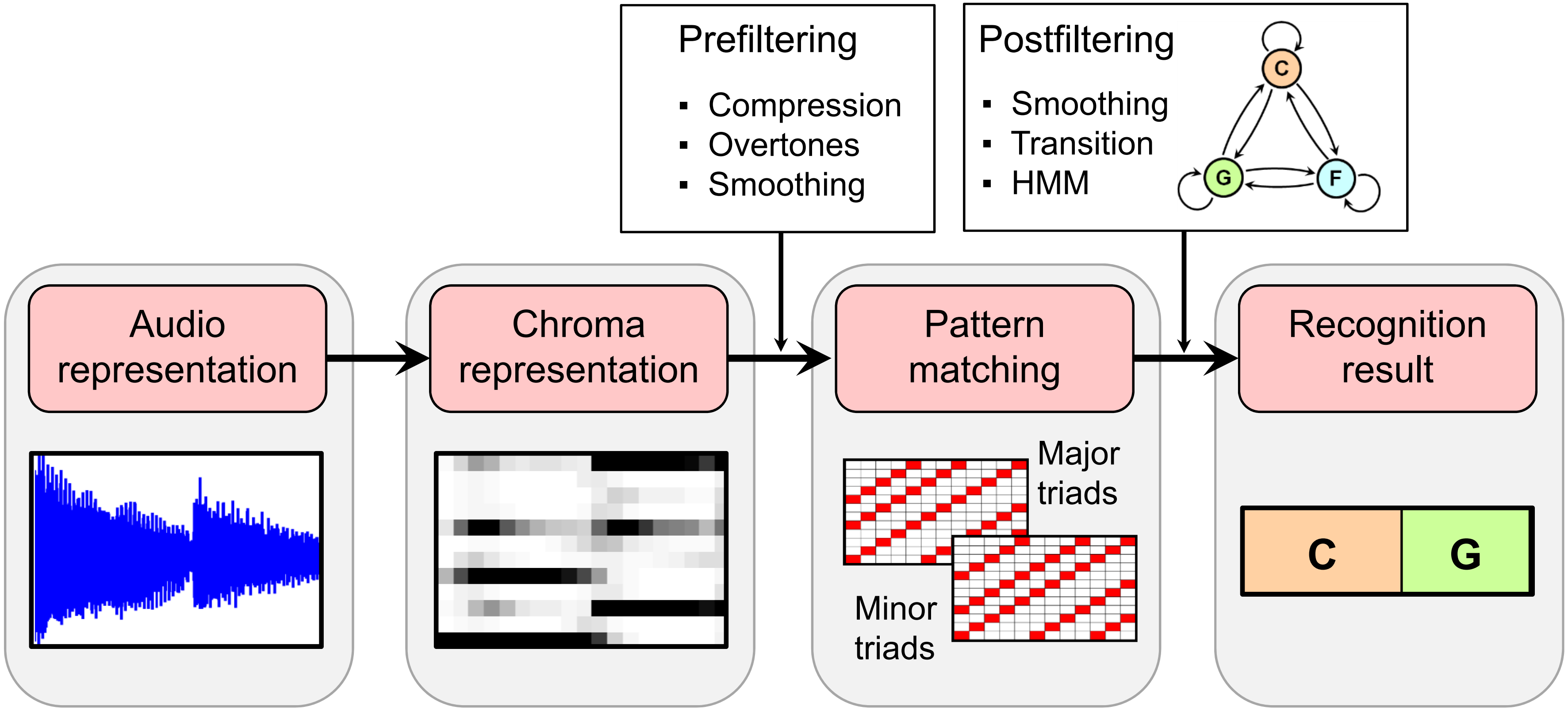
Chord Recognition



Chord Recognition



Chord Recognition

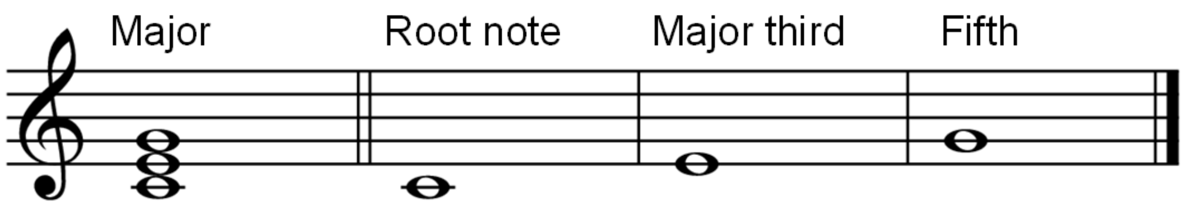


Chord Recognition: Basics

- Chord: Group of three or more **pitch classes** (sound simultaneously)
- Chord types: triads (3 pitch classes), seventh chords (4 pitch classes)…
- Chord classes: major, minor, diminished, augmented
- Here: focus on major and minor triads

Chord Recognition: Basics

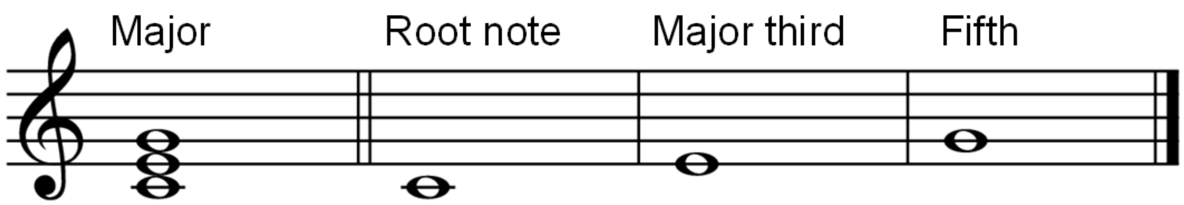
- Chord: Group of three or more **pitch classes** (sound simultaneously)
- Chord types: triads (3 pitch classes), seventh chords (4 pitch classes)…
- Chord classes: major, minor, diminished, augmented
- Here: focus on major and minor triads



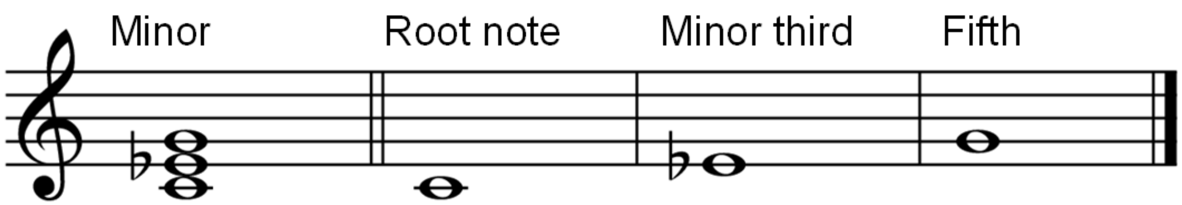
C Major (C)

Chord Recognition: Basics

- Chord: Group of three or more **pitch classes** (sound simultaneously)
- Chord types: triads (3 pitch classes), seventh chords (4 pitch classes)…
- Chord classes: major, minor, diminished, augmented
- Here: focus on major and minor triads



C Major (C)



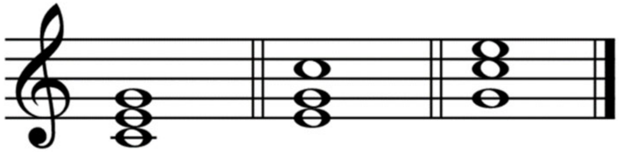
C Minor (Cm)

- Enharmonic equivalence: 12 root notes → 24 major/minor triads

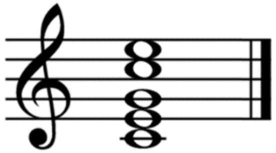
Chord Recognition: Basics

Chords appear in different forms:

- Inversions



- Different voicings

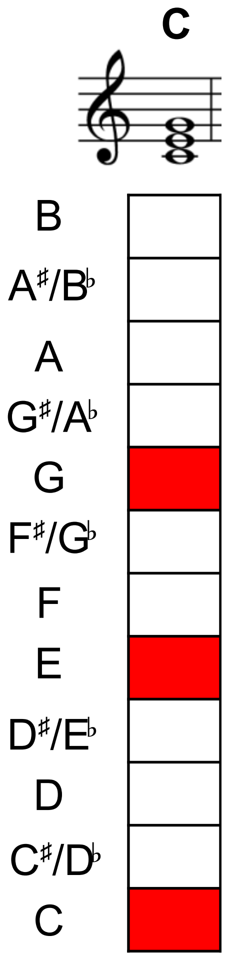


- Harmonic figuration: Broken chords (arpeggio)

- Melodic figuration: Different melody note (suspension, passing tone, …)
- Further: Additional notes, incomplete chords

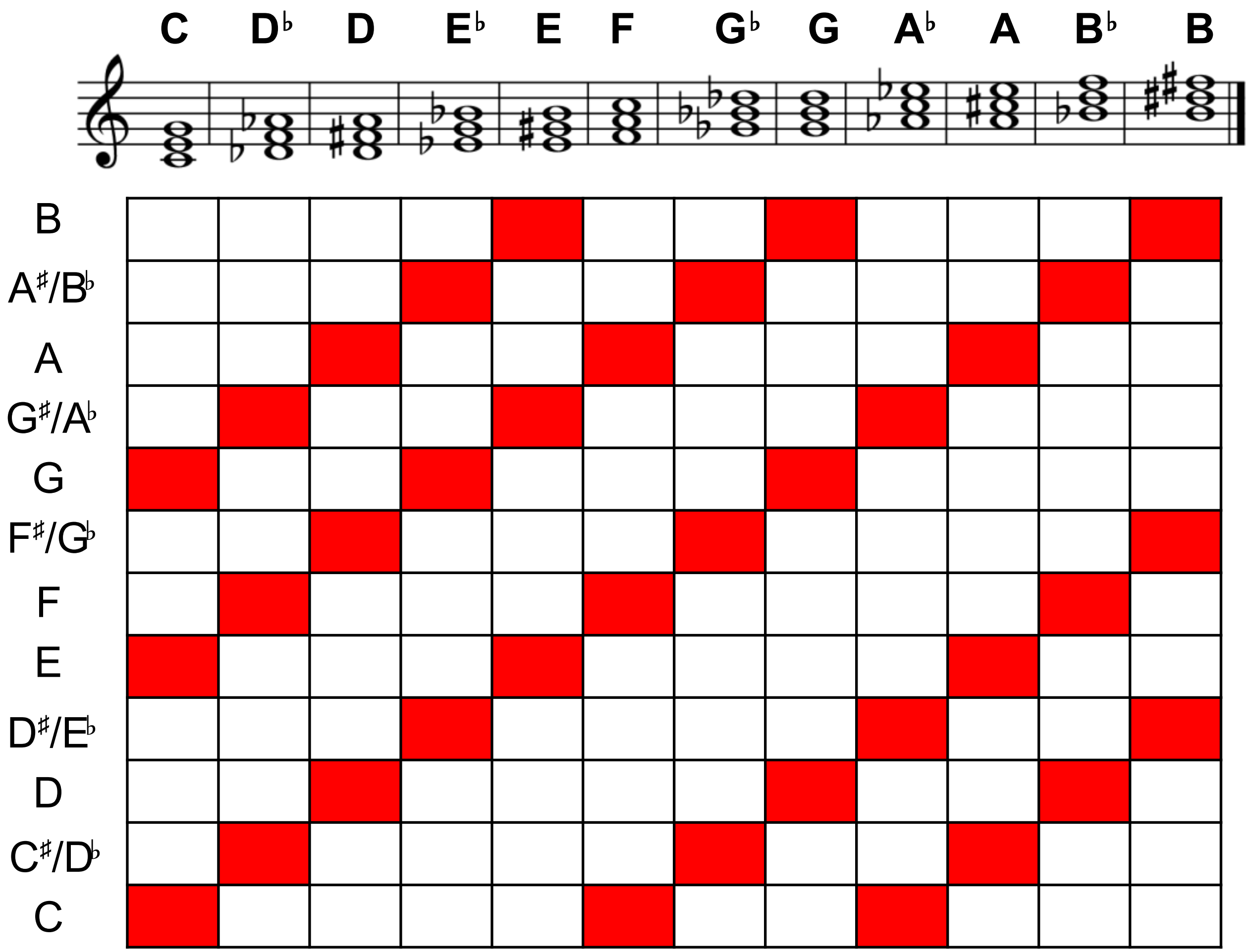
Chord Recognition: Basics

- Templates: **Major Triads**



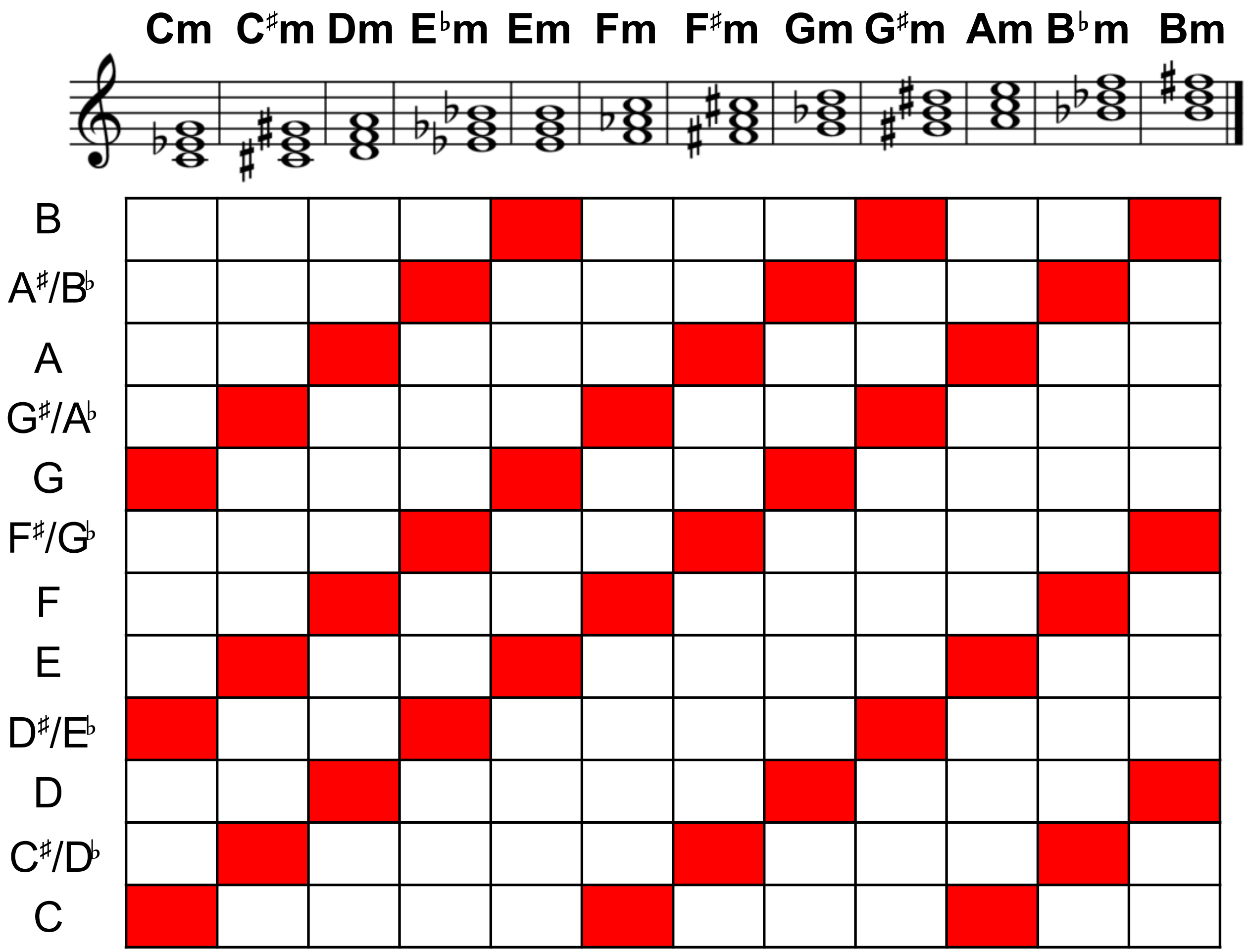
Chord Recognition: Basics

- Templates: **Major Triads**

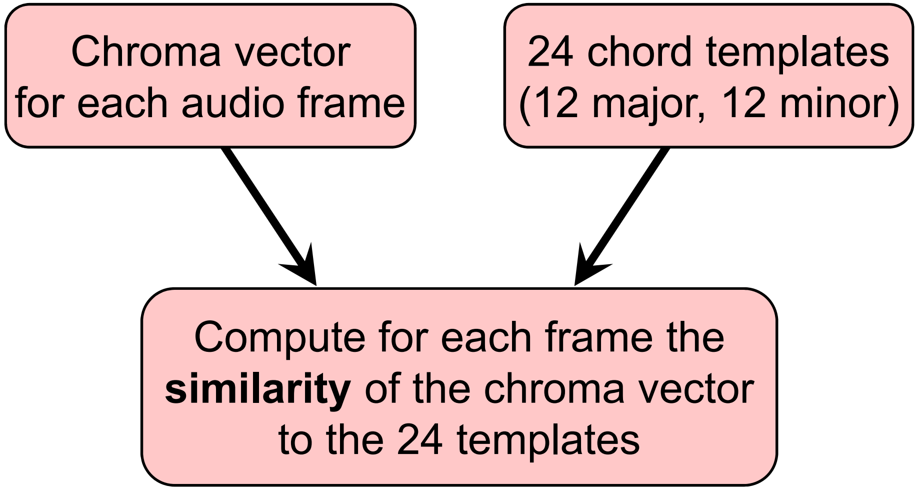


Chord Recognition: Basics

- Templates: **Minor Triads**

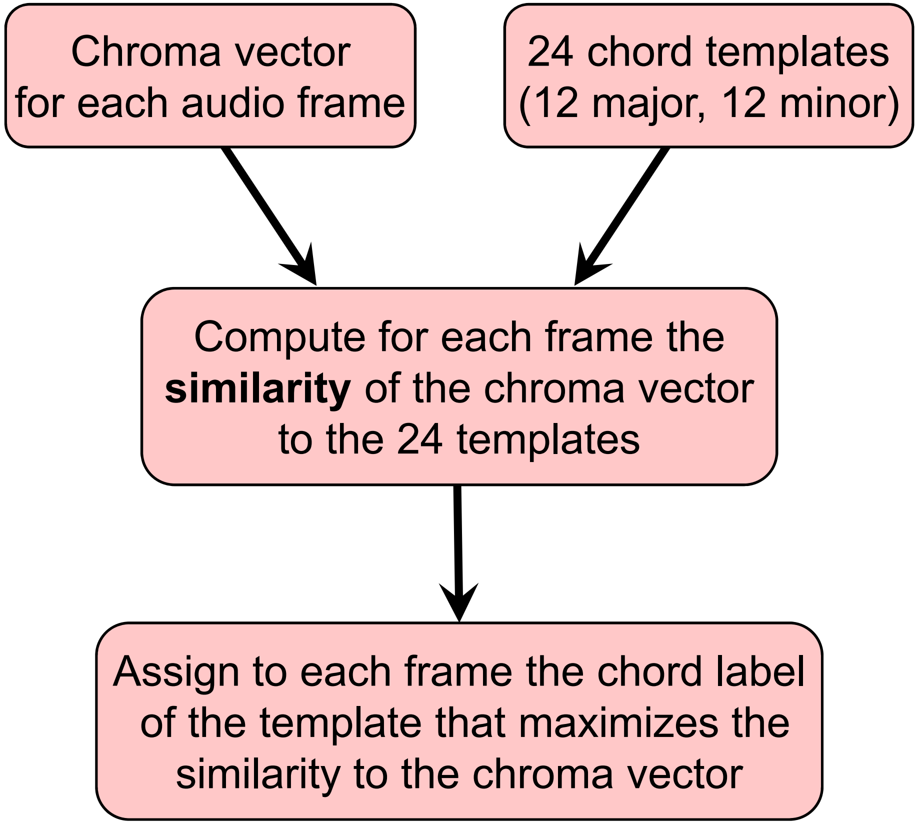


Chord Recognition: Template Matching



	C	**C♯**	**D**	**…**	**Cm**	**C♯m**	**Dm**	**…**
B	0	0	0	…	0	0	0	…
A♯	0	0	0	…	0	0	0	…
A	0	0	1	…	0	0	1	…
G♯	0	1	0	…	0	1	0	…
G	1	0	0	…	1	0	0	…
F♯	0	0	1	…	0	0	0	…
F	0	1	0	…	0	0	1	…
E	1	0	0	…	0	1	0	…
D♯	0	0	0	…	1	0	0	…
D	0	0	1	…	0	0	1	…
C♯	0	1	0	…	0	1	0	…
C	1	0	0	…	1	0	0	…

Chord Recognition: Label Assignment



	C	**C$^\sharp$**	**D**	**…**	**Cm**	**C$^\sharp$m**	**Dm**	**…**
B	0	0	0	…	0	0	0	…
A$^\sharp$	0	0	0	…	0	0	0	…
A	0	0	1	…	0	0	1	…
G$^\sharp$	0	1	0	…	0	1	0	…
G	1	0	0	…	1	0	0	…
F$^\sharp$	0	0	1	…	0	0	0	…
F	0	1	0	…	0	0	1	…
E	1	0	0	…	0	1	0	…
D$^\sharp$	0	0	0	…	1	0	0	…
D	0	0	1	…	0	0	1	…
C$^\sharp$	0	1	0	…	0	1	0	…
C	1	0	0	…	1	0	0	…

Chord Recognition: Template Matching

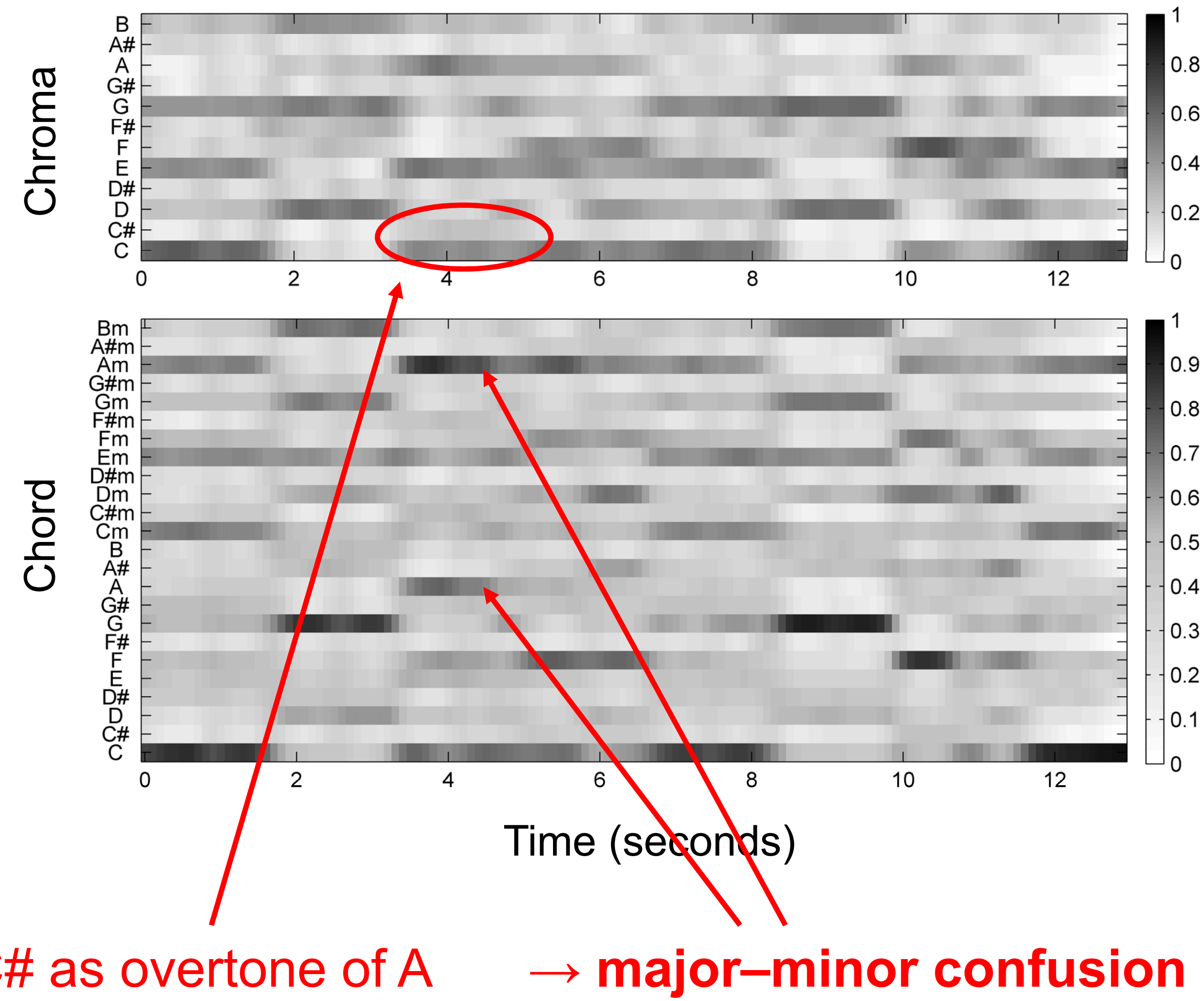
- Similarity measure: Cosine similarity (inner product of normalized vectors)

Chord template: $\boldsymbol{t} \in \mathbb{R}^{12}$

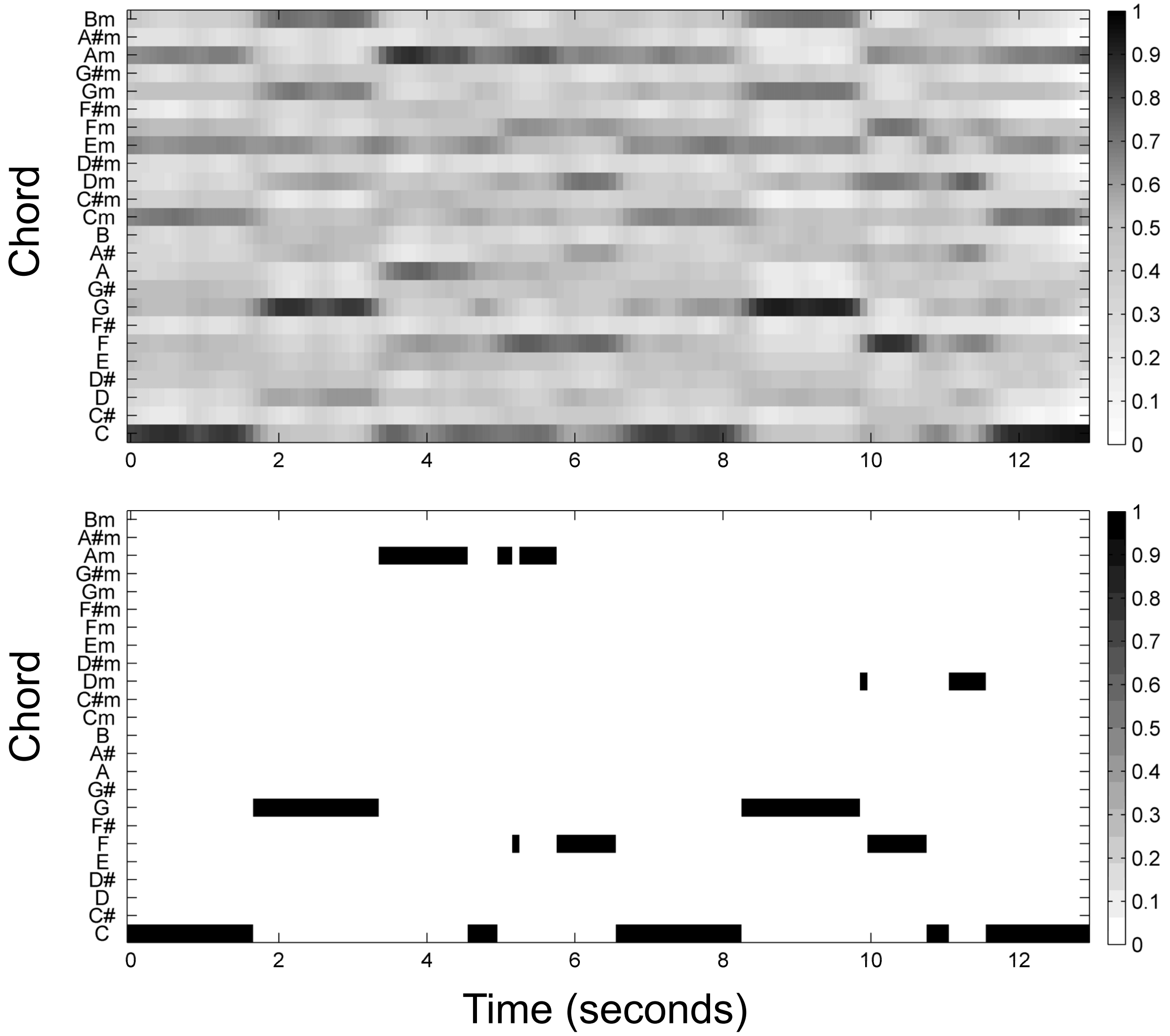
Chroma vector: $\boldsymbol{c} \in \mathbb{R}^{12}$

Similarity measure: $s(\boldsymbol{t}, \boldsymbol{c}) = \frac{\langle \boldsymbol{t} | \boldsymbol{c} \rangle}{||\boldsymbol{t}|| \cdot ||\boldsymbol{c}||}$

Chord Recognition: Template Matching



Chord Recognition: Label Assignment



Chord Recognition: Evaluation

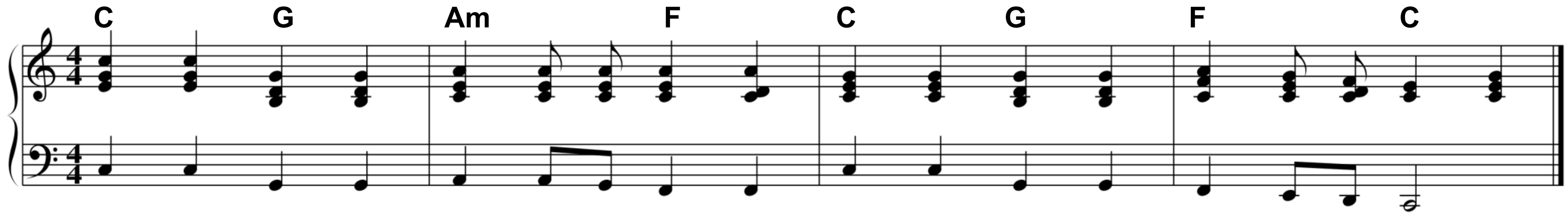
- Comparison of
  - reference labels (ground truth; relevant “items”)
  - estimated labels (computed)

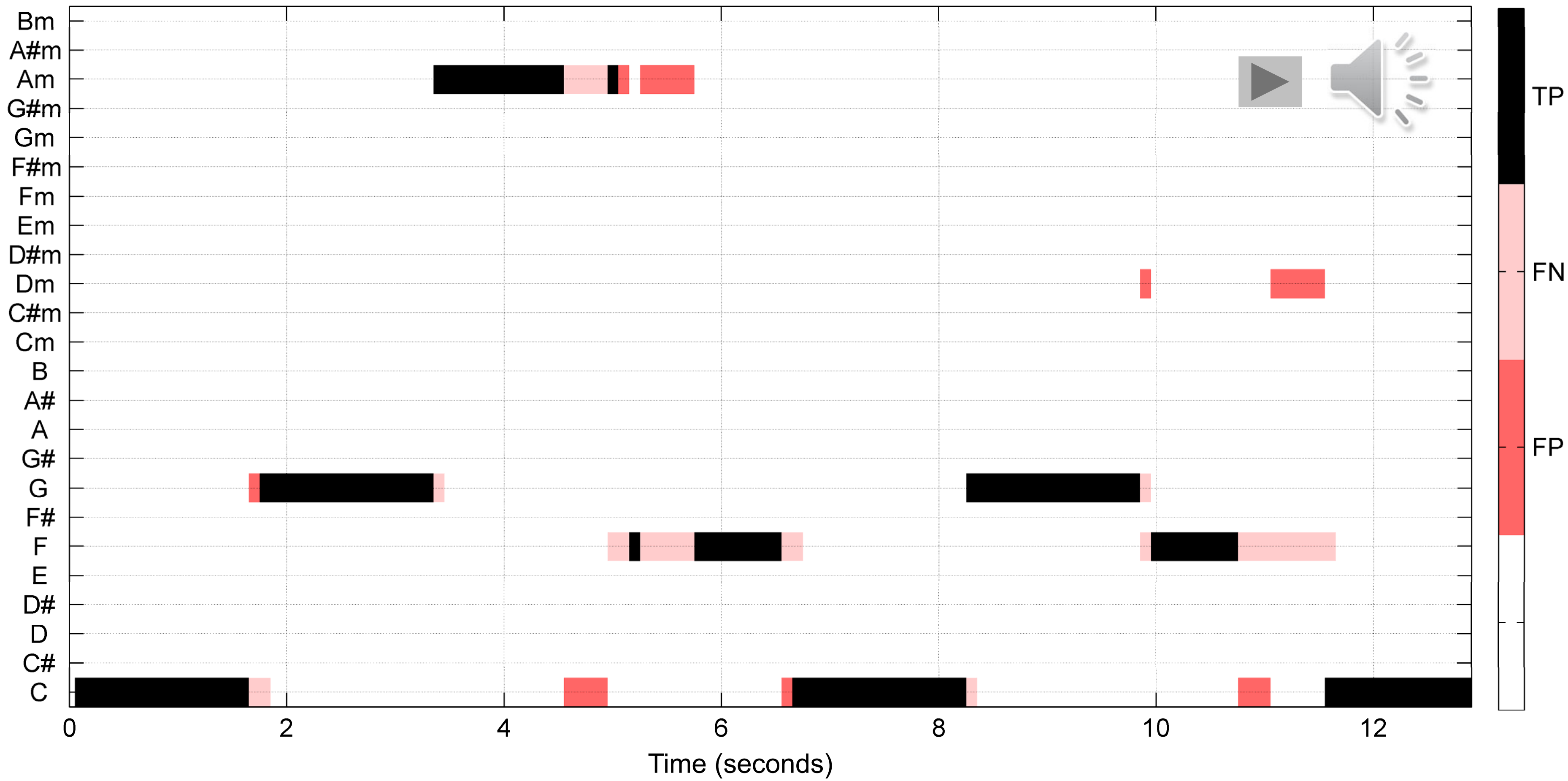
- TP (true positive):
  Reference label and estimated label agree

- FN (false negative):
  References label not detected

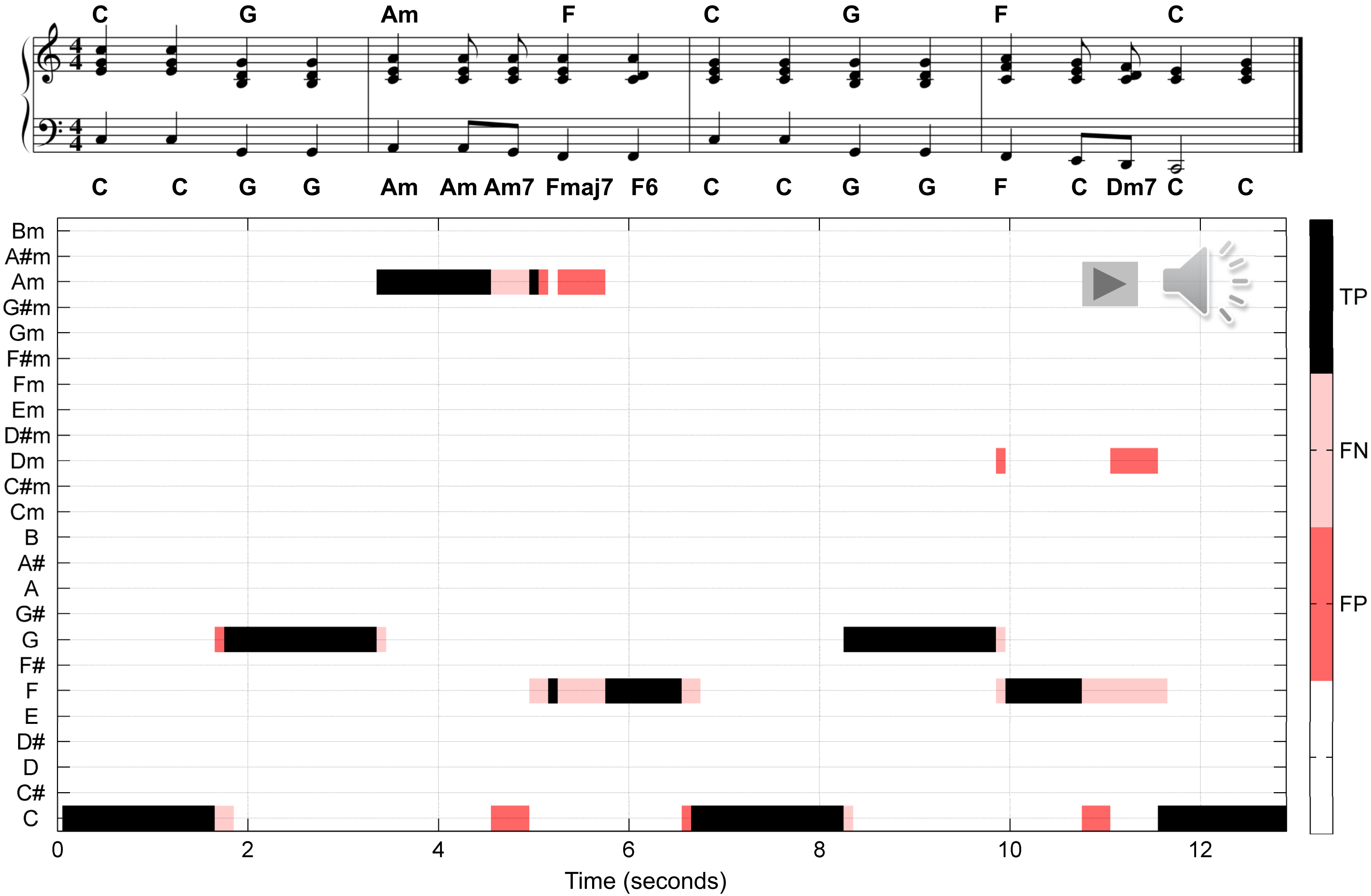
- FP (false positive):
  Estimated label not covered by reference label

Chord Recognition: Evaluation



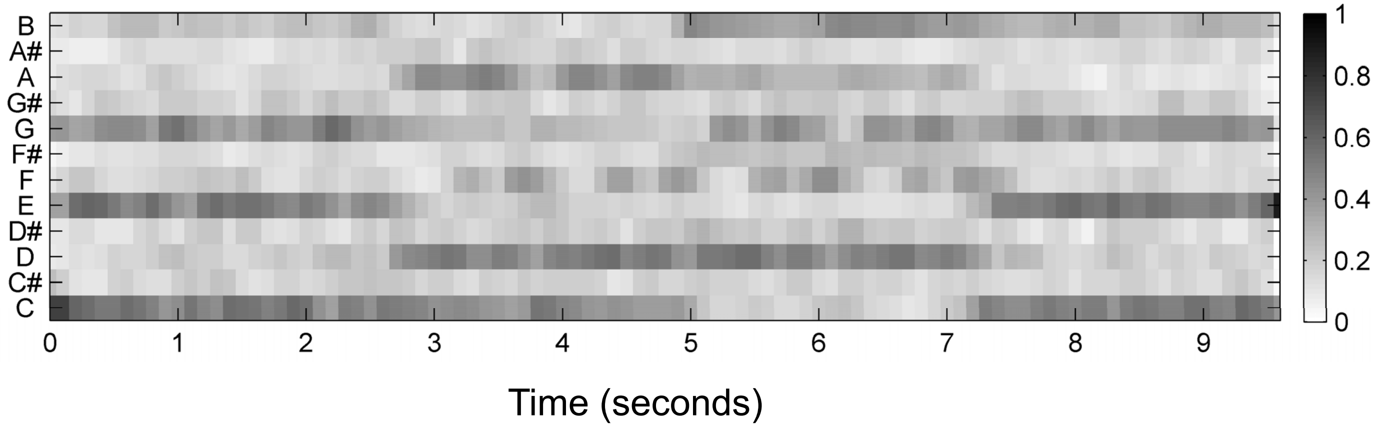


Chord Recognition: Evaluation



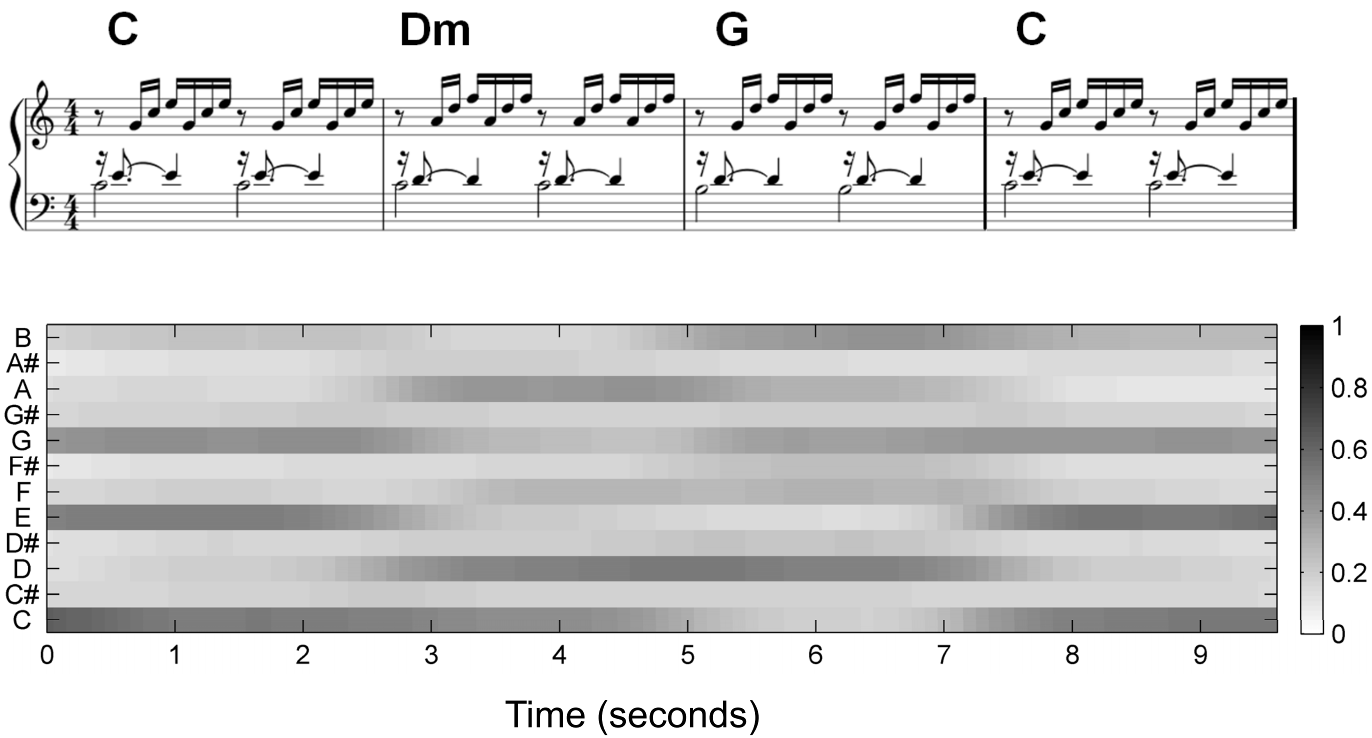
Chord Recognition: Smoothing

- Apply average filter of length $L \in \mathbb{N}$:



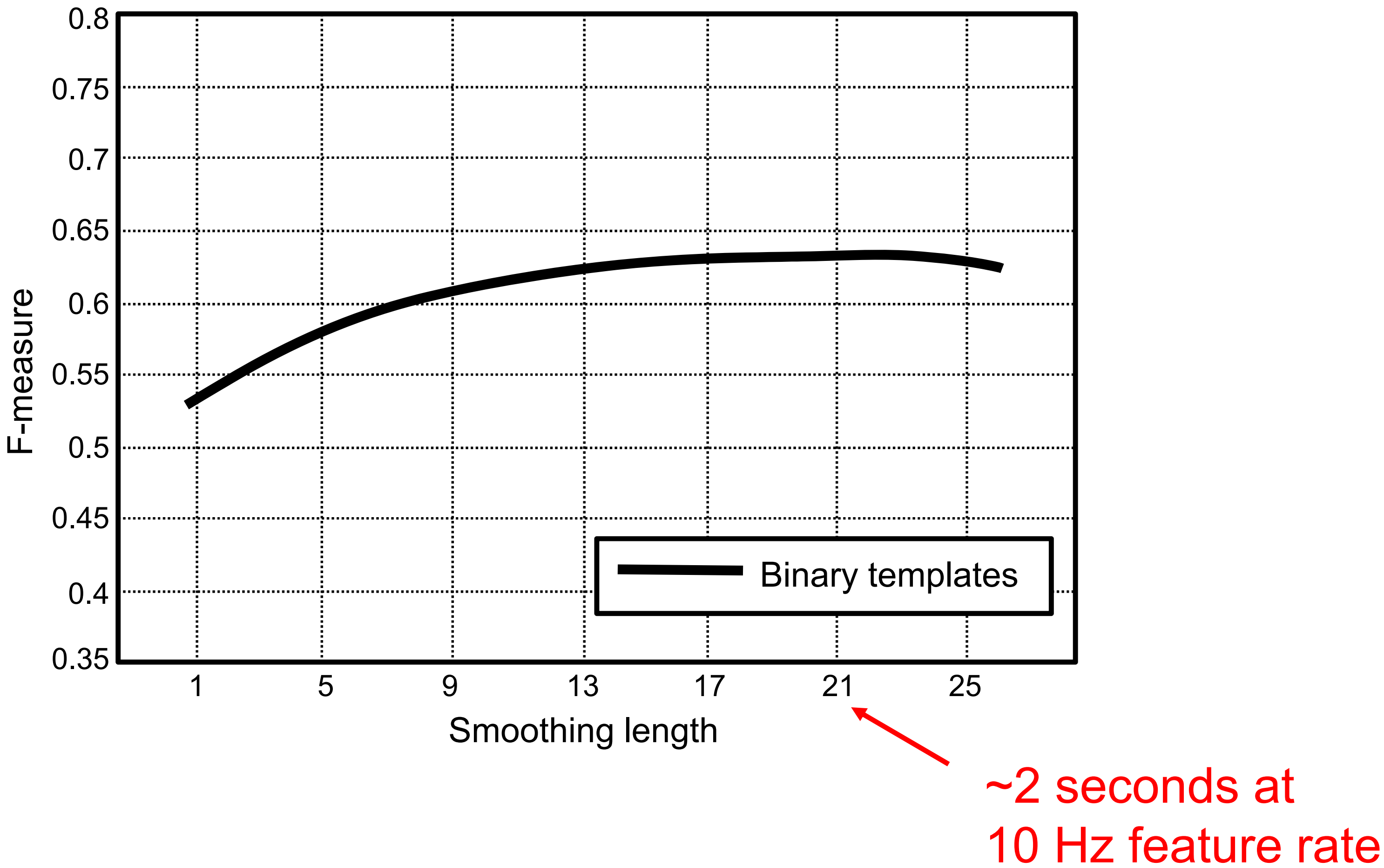
Chord Recognition: Smoothing

- Apply average filter of length $L \in \mathbb{N}$:



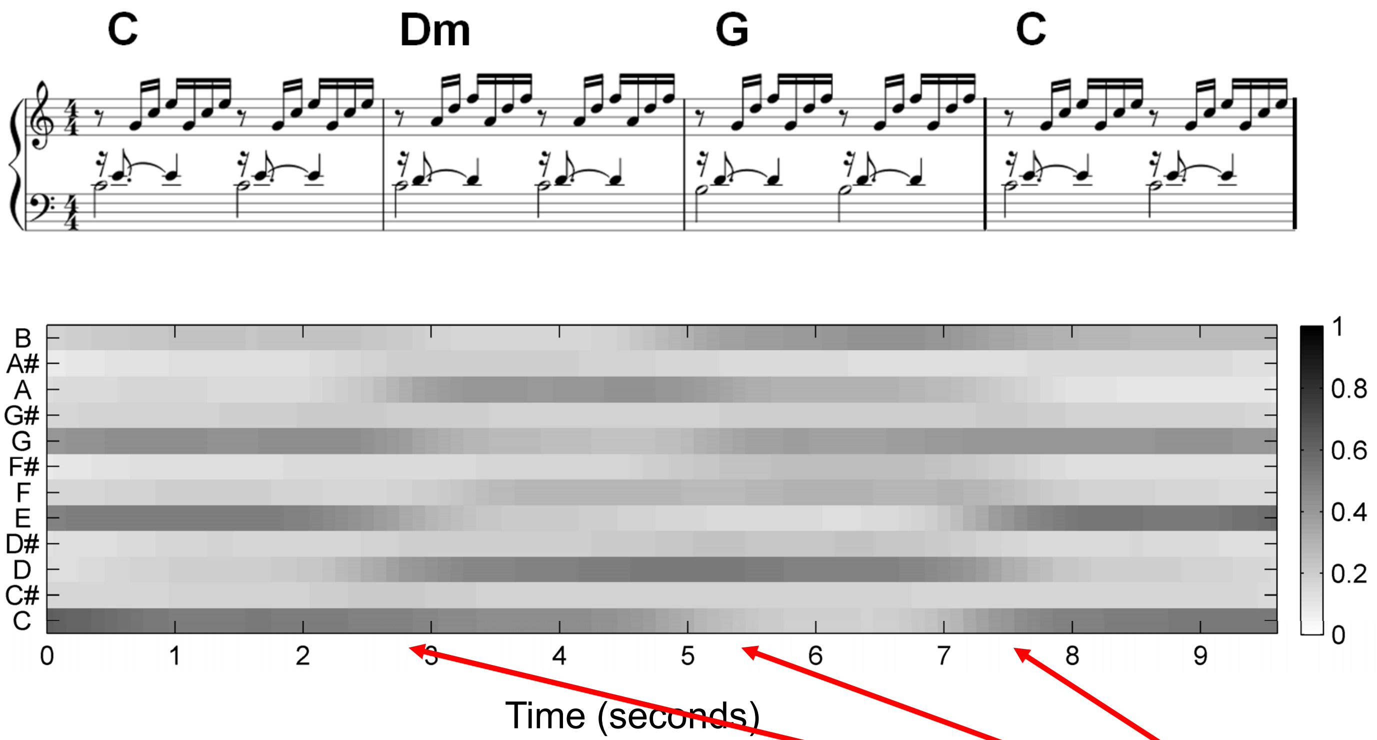
Chord Recognition: Smoothing

- Evaluation on all 180 Beatles songs (10 studio albums)



Chord Recognition: Smoothing

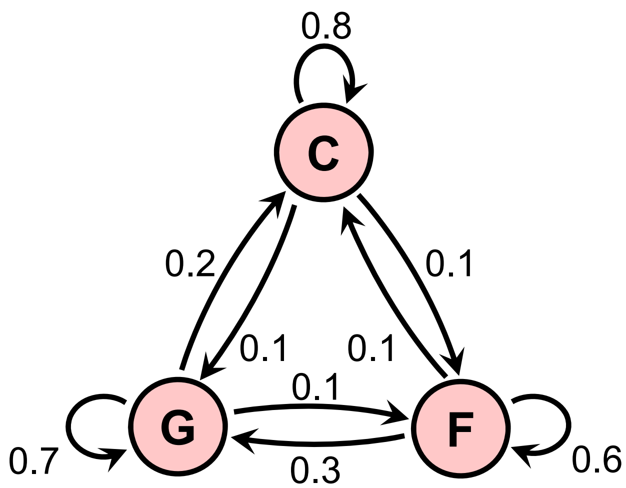
- Apply average filter of length $L \in \mathbb{N}$:



blurring of boundaries!

Markov Chains

- Probabilistic model for sequential data
- Markov property: Next state only depends on current state (transition model – time-invariant, no “memory”)
- Consist of:
  - Set of states
  - State transition probabilities
  - *Initial state probabilities*



Markov Chains

Notation:

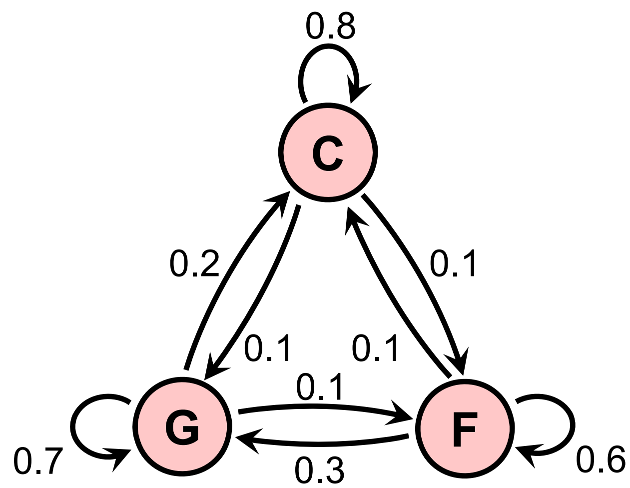
States α_i for $i \in [1:I]$

State transition probabilities a_{ij}

A	α₁	α₂	α₃
α₁	a_{11}	a_{12}	a_{13}
α₂	a_{21}	a_{22}	a_{23}
α₃	a_{31}	a_{32}	a_{33}

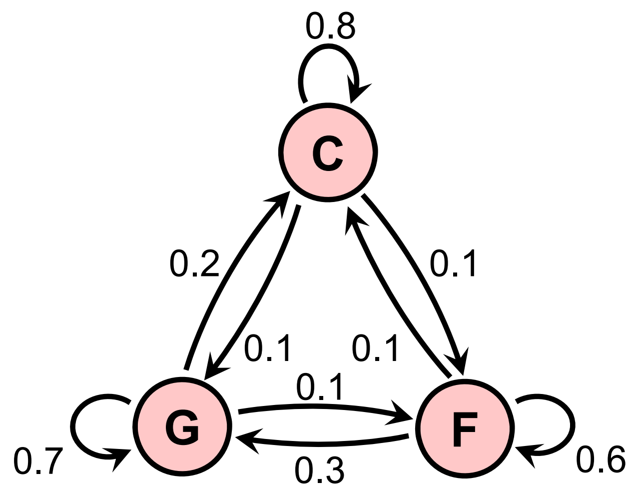
Initial state probabilities c_i

C	α₁	α₂	α₃
	c_1	c_2	c_3

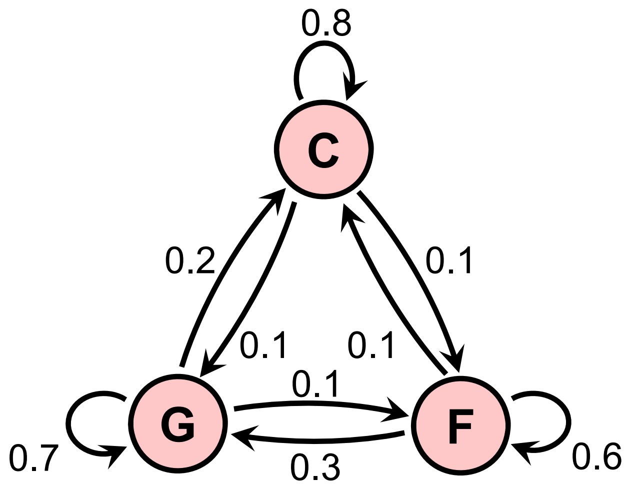


Markov Chains

- Application examples:
  - Compute probability of a sequence using given a model (evaluation)
  - Compare two sequences using a given model
  - Evaluate a sequence with two different models (classification)



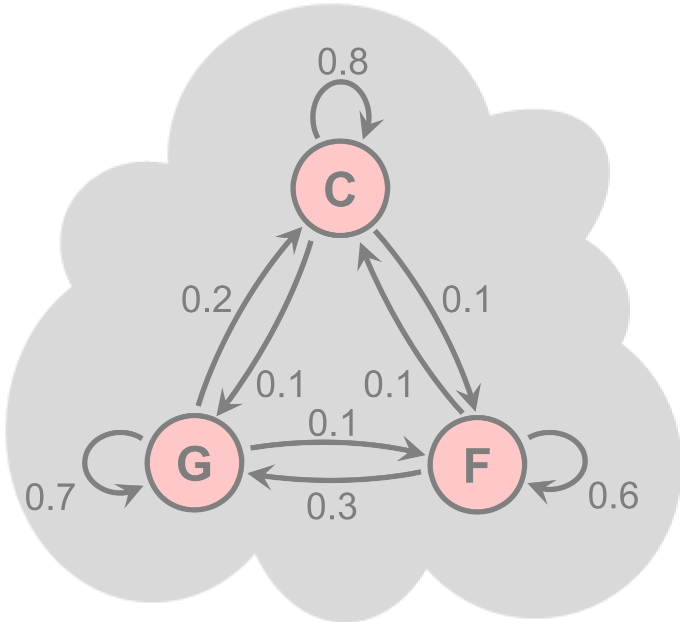
Hidden Markov Model



Hidden Markov Models

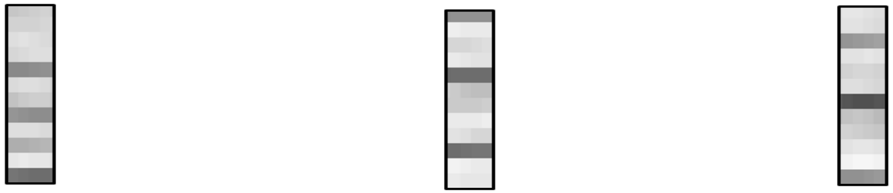
- States as **hidden** variables

- Consist of:
  - Set of states (hidden)
  - State transition probabilities
  - *Initial state probabilities*

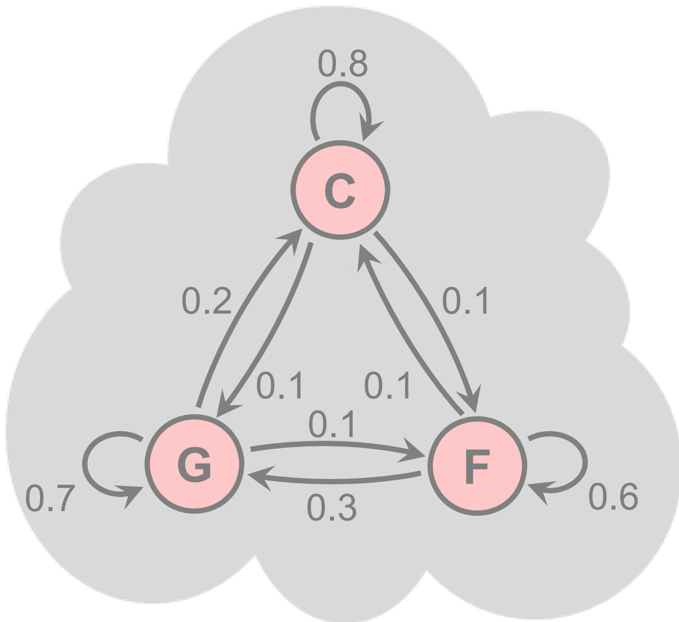


Hidden Markov Models

- States as **hidden** variables

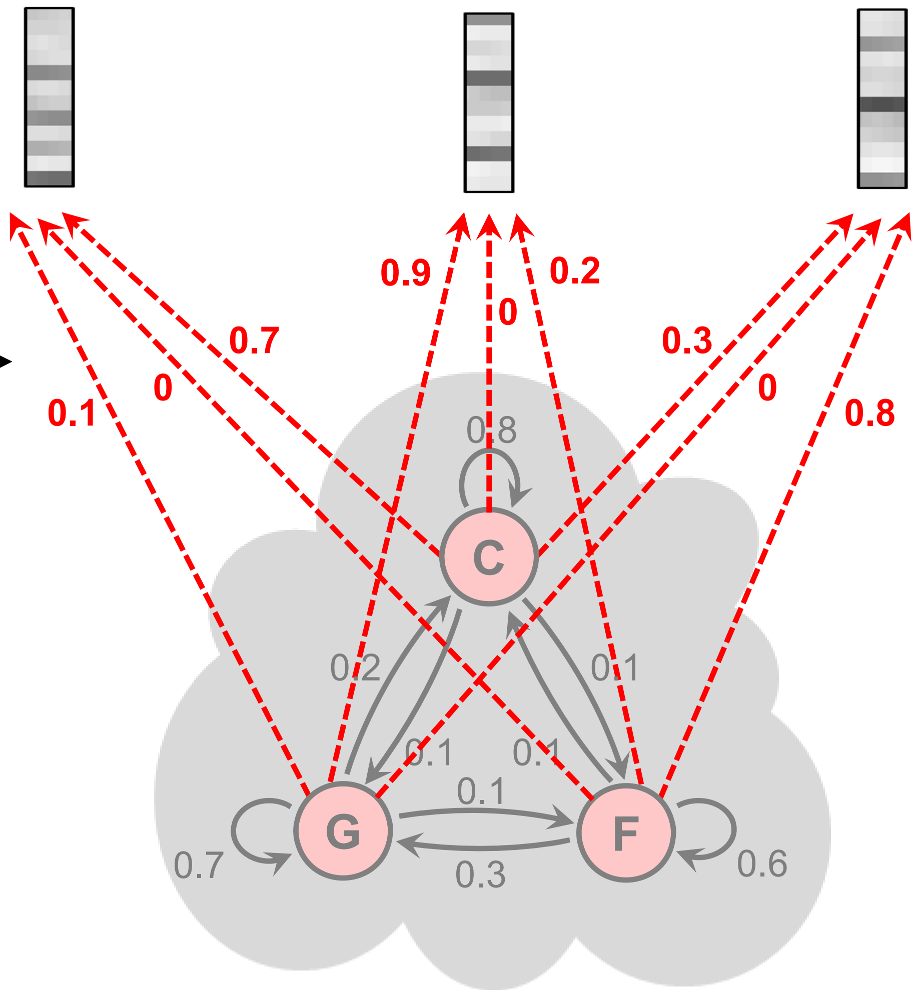


- Consist of:
  - Set of states (hidden)
  - State transition probabilities
  - *Initial state probabilities*
  - Observations (visible)



Hidden Markov Models

- States as **hidden** variables
- Consist of:
  - Set of states (hidden)
  - State transition probabilities
  - *Initial state probabilities*
  - Observations (visible)
  - Emission probabilities



Hidden Markov Models

Notation:

States — α_i for $i \in [1:I]$

State transition probabilities — a_{ij}

A	α1	α2	α3
α1	a_{11}	a_{12}	a_{13}
α2	a_{21}	a_{22}	a_{23}
α3	a_{31}	a_{32}	a_{33}

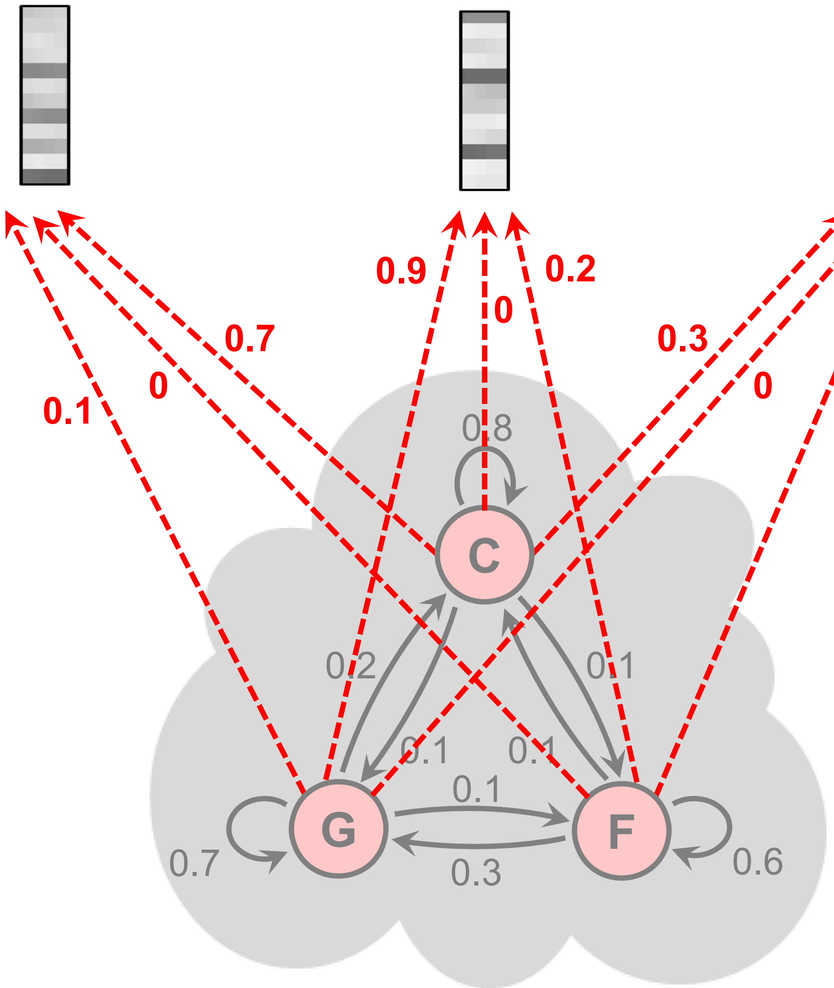
Initial state probabilities — c_i

C	α1	α2	α3
	c_1	c_2	c_3

Observation symbols — β_k for $k \in [1:K]$

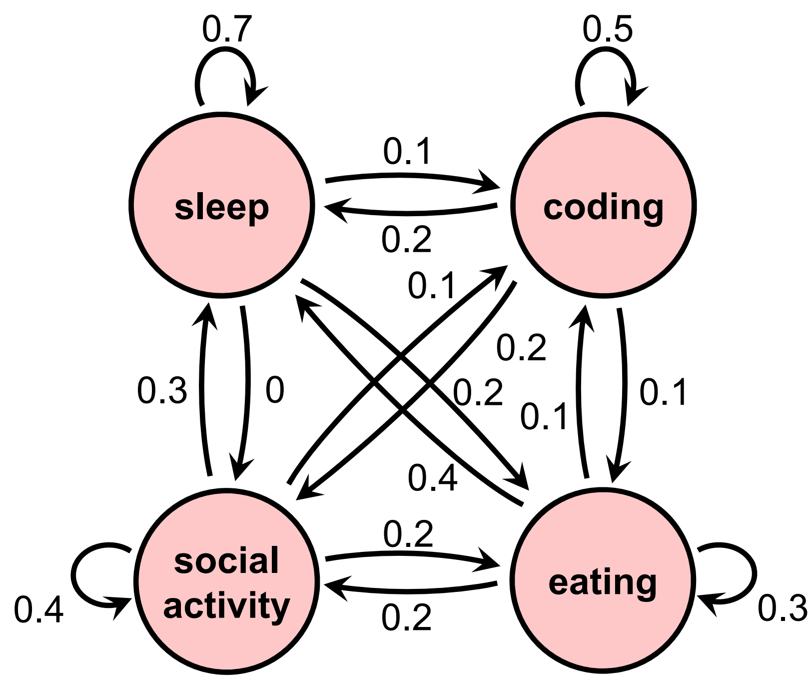
Emission probabilities — b_{ik}

B	β1	β2	β3
α1	b_{11}	b_{12}	b_{13}
α2	b_{21}	b_{22}	b_{23}
α3	b_{31}	b_{32}	b_{33}



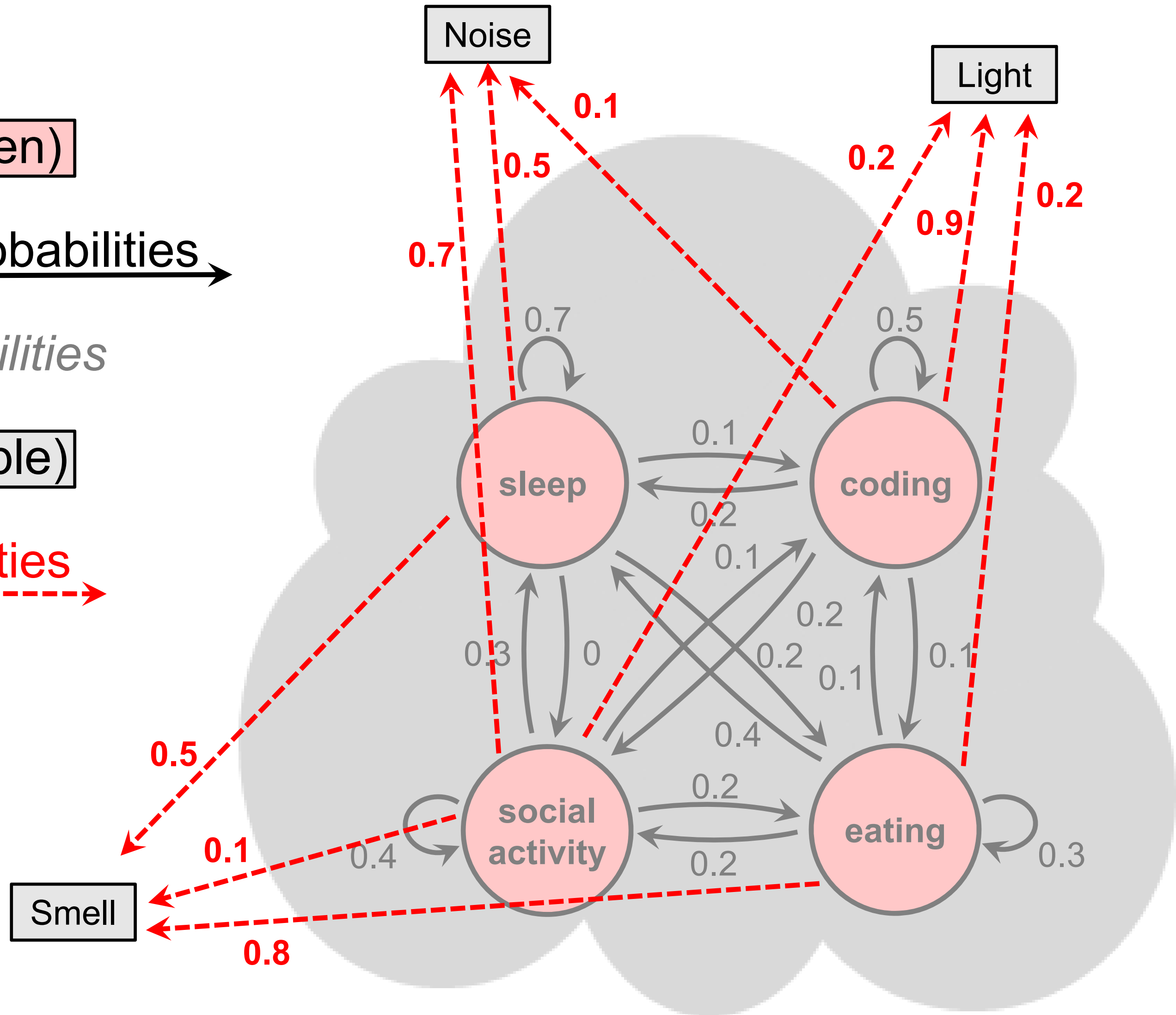
Markov Chains

- Analogon: the student's life
  - Set of states (hidden)
  - State transition probabilities
  - *Initial state probabilities*



Hidden Markov Models

- Analogon: the student's life
- Consists of:
  - Set of states (hidden)
  - State transition probabilities
  - *Initial state probabilities*
  - Observations (visible)
  - Emission probabilities



Hidden Markov Models

- Only observation sequence is visible!

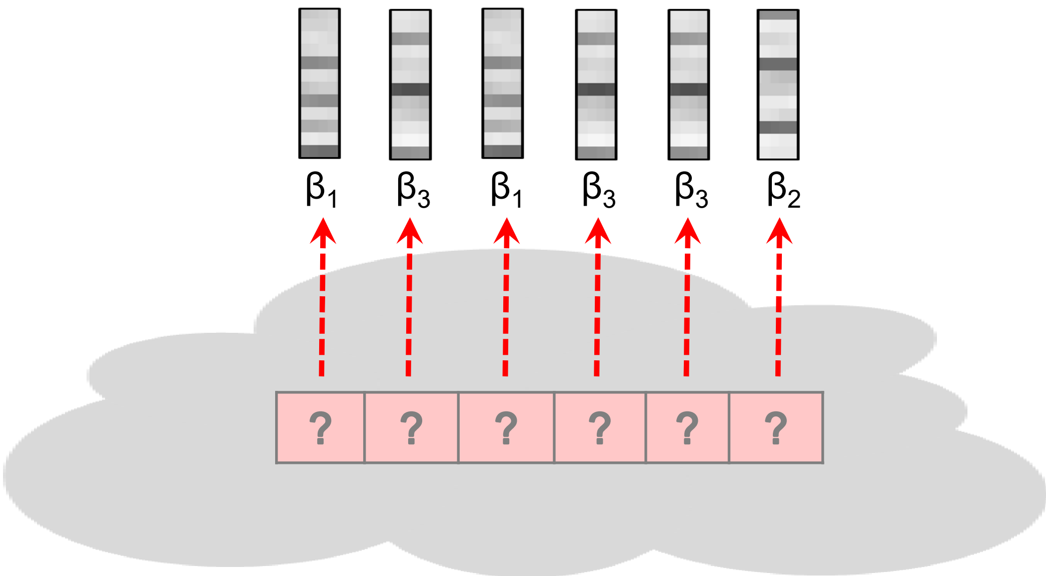
Different algorithmic problems:

- **Evaluation problem**
  - *Given:* observation sequence and model
  - *Find:* fitness (how well the model matches the sequence)
- **Uncovering problem:**
  - *Given:* observation sequence and model
  - *Find:* optimal hidden state sequence
- **Estimation problem** („training“ the HMM):
  - *Given:* observation sequence
  - *Find:* model parameters
  - Baum-Welch algorithm (Expectation-Maximization)

Uncovering problem

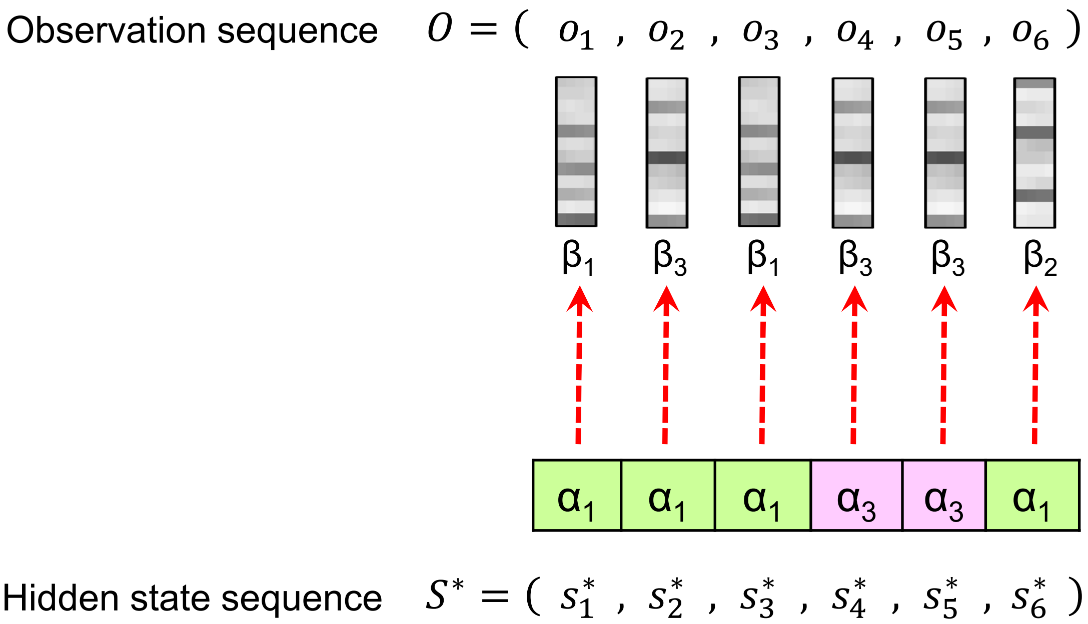
- *Given:* observation sequence $O = (o_1, \ldots, o_N)$ of length $N \in \mathbb{N}$ and HMM Θ (model parameters)
- *Find:* optimal hidden state sequence $S^* = (s_1^*, \ldots, s_N^*)$
- Corresponds to chord estimation task!

Observation sequence $O = (o_1 , o_2 , o_3 , o_4 , o_5 , o_6)$



Uncovering problem

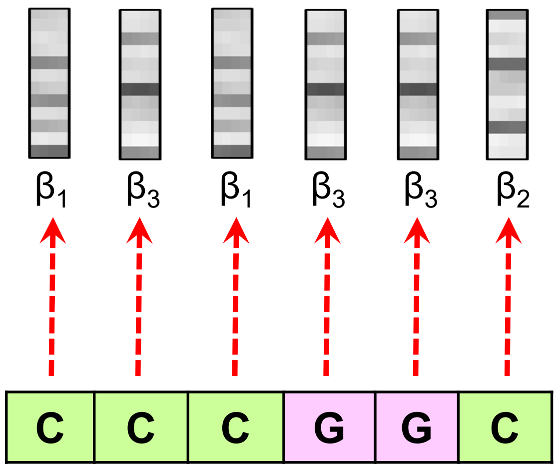
- *Given:* observation sequence $O = (o_1, \dots, o_N)$ of length $N \in \mathbb{N}$ and HMM Θ (model parameters)
- *Find:* optimal hidden state sequence $S^* = (s_1^*, \dots, s_N^*)$
- Corresponds to chord estimation task!



Uncovering problem

- *Given:* observation sequence $O = (o_1, \ldots, o_N)$ of length $N \in \mathbb{N}$ and HMM Θ (model parameters)
- *Find:* optimal hidden state sequence $S^* = (s_1^*, \ldots, s_N^*)$
- Corresponds to chord estimation task!

Observation sequence $O = (\ o_1\ ,\ o_2\ ,\ o_3\ ,\ o_4\ ,\ o_5\ ,\ o_6\ )$



Hidden state sequence $S^* = (\ s_1^*\ ,\ s_2^*\ ,\ s_3^*\ ,\ s_4^*\ ,\ s_5^*\ ,\ s_6^*\ )$

Uncovering problem

- **Optimal** hidden state sequence?
  - "Best explains" given observation sequence O
  - Maximizes probability $P[O, S \mid \Theta]$

$$\text{Prob}^* = \max_S P[O, S \mid \Theta]$$

$$S^* = \operatorname{argmax}_S P[O, S \mid \Theta]$$

- Straight-forward computation (naive approach):
  - Compute probability for each possible sequence S
  - Number of possible sequences of length N ($I =$ number of states):

$$\underbrace{I \cdot I \cdot \ldots \cdot I}_{N \text{ factors}} = I^N$$

computationally infeasible!

Viterbi Algorithm

- Based on dynamic programming (similar to DTW)
- Idea: Recursive computation from sub-problems
- Use truncated versions of observation sequence

$$O(1{:}n) := (o_1, \ldots, o_n), \text{ length } n \in [1{:}N]$$

- Define $\mathbf{D}(i, n)$ as the highest probability along a single state sequence $(s_1, \ldots, s_n)$ that ends in state $s_n = \alpha_i$

$$\mathbf{D}(i, n) = \max_{(s_1, \ldots, s_n)} P[O(1{:}n), (s_1, \ldots, s_{n-1}, s_n = \alpha_i) \mid \Theta]$$

- Then, our solution is the state sequence yielding

$$\text{Prob}^* = \max_{i \in [1:I]} \mathbf{D}(i, N)$$

Viterbi Algorithm

- $\mathbf{D}$: matrix of size $I \times N$
- Recursive computation of $\mathbf{D}(i,n)$ along the column index n
- Initialization:
  - $n = 1$
  - Truncated observation sequence: $O(1) = (o_1)$
  - Current observation: $o_1 = \beta_{k_1}$

$$\mathbf{D}(i,1) = c_i \cdot b_{ik_1} \quad \text{for some} \quad i \in [1:I]$$

Viterbi Algorithm

- $\mathbf{D}$: matrix of size $I \times N$
- Recursive computation of $\mathbf{D}(i, n)$ along the column index n
- Recursion:
  - $n \in [2:N]$
  - Truncated observation sequence: $O(1:n) = (o_1, \dots, o_n)$
  - Last observation: $o_n = \beta_{k_n}$

$$\mathbf{D}(i,n) = b_{ik_n} \cdot a_{j^*i} \cdot \underbrace{P\big[O(1:n-1), (s_1, \dots, s_{n-1} = \alpha_{j^*}) \,\big|\, \Theta\big]}_{\text{must be maximal!}} \qquad \text{for } i \in [1:I]$$

$$\mathbf{D}(i,n) = b_{ik_n} \cdot a_{j^*i} \cdot \mathbf{D}(j^*, n-1)$$

Viterbi Algorithm

- $\mathbf{D}$: matrix of size $I \times N$
- Recursive computation of $\mathbf{D}(i,n)$ along the column index n
- Recursion:
  - $n \in [2:N]$
  - Truncated observation sequence: $O(1:n) = (o_1, \ldots, o_n)$
  - Last observation: $o_n = \beta_{k_n}$

$$\mathbf{D}(i,n) = b_{ik_n} \cdot a_{j^*i} \cdot \underbrace{P\big[O(1:n-1), (s_1, \ldots, s_{n-1} = \alpha_{j^*}) \,\big|\, \Theta\big]}_{\text{must be maximal!}} \qquad \text{for } i \in [1:I]$$

$$\mathbf{D}(i,n) = b_{ik_n} \cdot \underbrace{a_{j^*i} \cdot \mathbf{D}(j^*, n-1)}_{\text{must be maximal (best index } j^*\text{)}}$$

$$\mathbf{D}(i,n) = b_{ik_n} \cdot \max_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1) \right)$$

Viterbi Algorithm

- $\mathbf{D}$ given – find optimal state sequence $S^* = (s_1^*, \ldots, s_N^*) := \left(\alpha_{i_1}, \ldots, \alpha_{i_N}\right)$
- Backtracking procedure (reverse order)
- Last element:
  - $n = N$
  - Optimal state: α_{i_N}

$$i_N = \underset{j \in [1:I]}{\operatorname{argmax}} \, \mathbf{D}(j, N)$$

Viterbi Algorithm

- $\mathbf{D}$ given – find optimal state sequence $S^* = (s_1^*, \ldots, s_N^*) := \left(\alpha_{i_1}, \ldots, \alpha_{i_N}\right)$
- Backtracking procedure (reverse order)
- Further elements:
  - $n = N-1, N-2, \ldots, 1$
  - Optimal state: α_{i_n}

$$i_n = \underset{j \in [1:I]}{\operatorname{argmax}} \left(a_{j i_{n+1}} \cdot \mathbf{D}(j, n)\right)$$

Viterbi Algorithm

- $\mathbf{D}$ given – find optimal state sequence $S^* = (s_1^*, \ldots, s_N^*) \coloneqq \left(\alpha_{i_1}, \ldots, \alpha_{i_N}\right)$
- Backtracking procedure (reverse order)
- Further elements:
  - $n = N-1, N-2, \ldots, 1$
  - Optimal state: α_{i_n}

$$i_n = \underset{j \in [1:I]}{\operatorname{argmax}} \left(a_{j i_{n+1}} \cdot \mathbf{D}(j, n)\right)$$

- Simplification of backtracking: Keep track of maximizing index j in

$$\mathbf{D}(i, n) = b_{i k_n} \cdot \max_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1)\right)$$

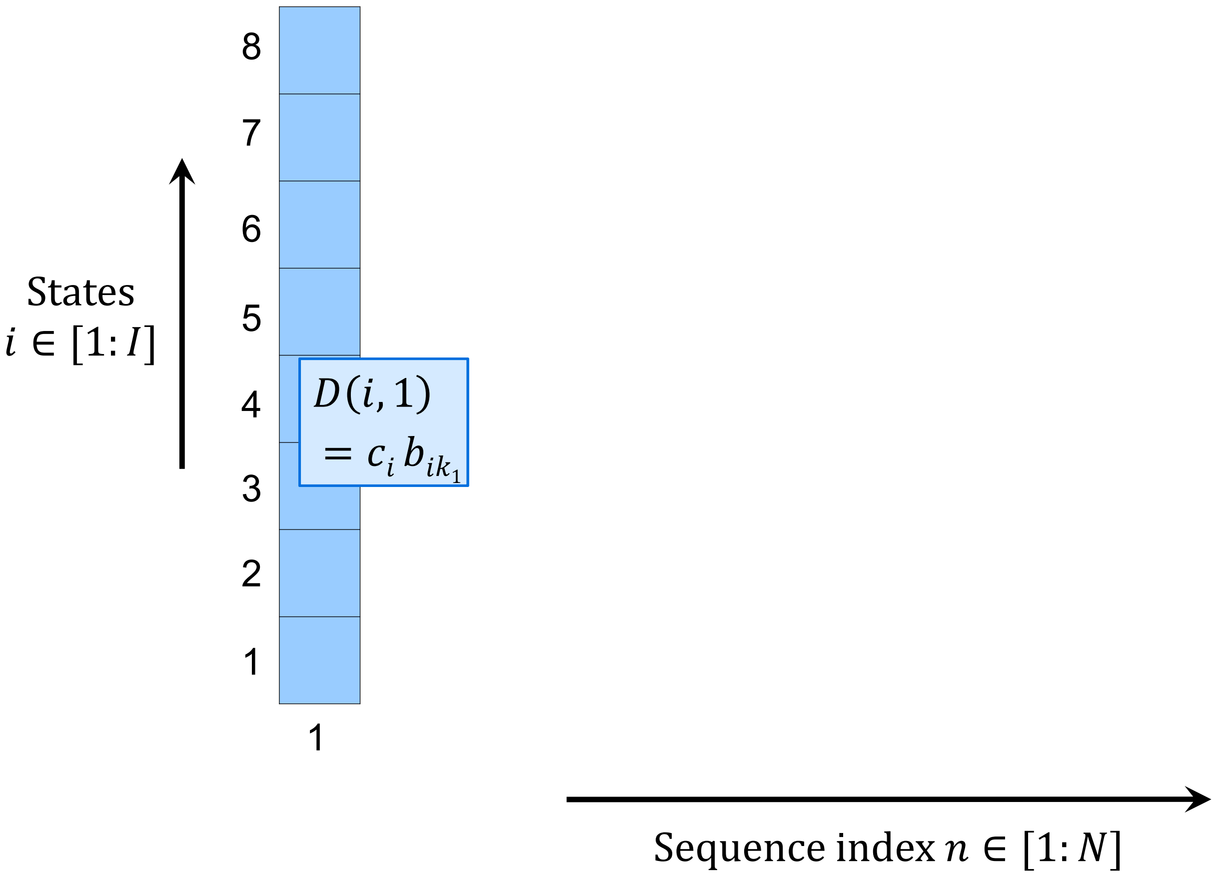
- Define $(I \times (N-1))$ matrix $\mathbf{E}$:

$$\mathbf{E}(i, n-1) = \underset{j \in [1:I]}{\operatorname{argmax}} \left(a_{ji} \cdot \mathbf{D}(j, n-1)\right)$$

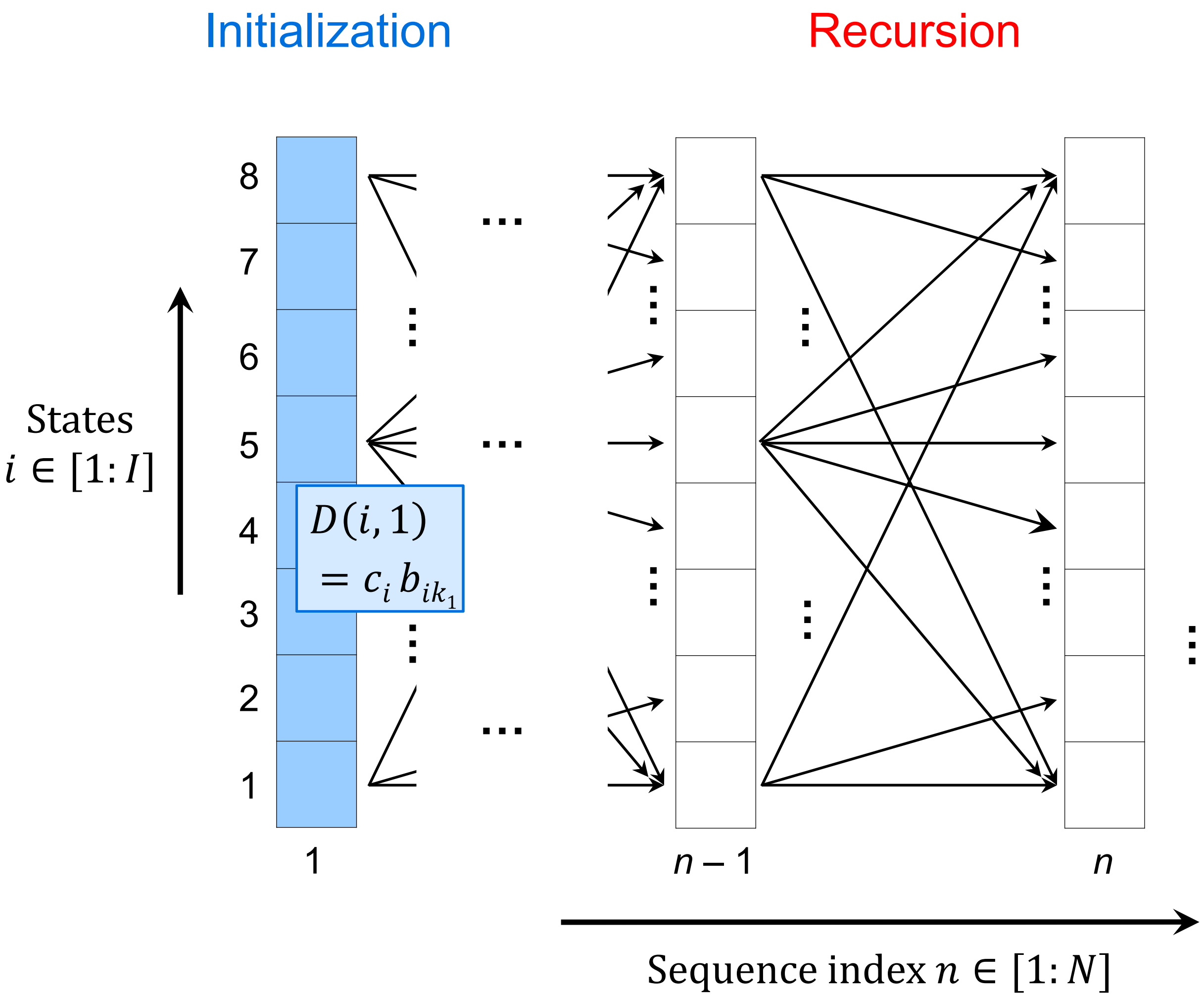
Viterbi Algorithm

$$o_1 = \beta_{k_1}$$

Initialization

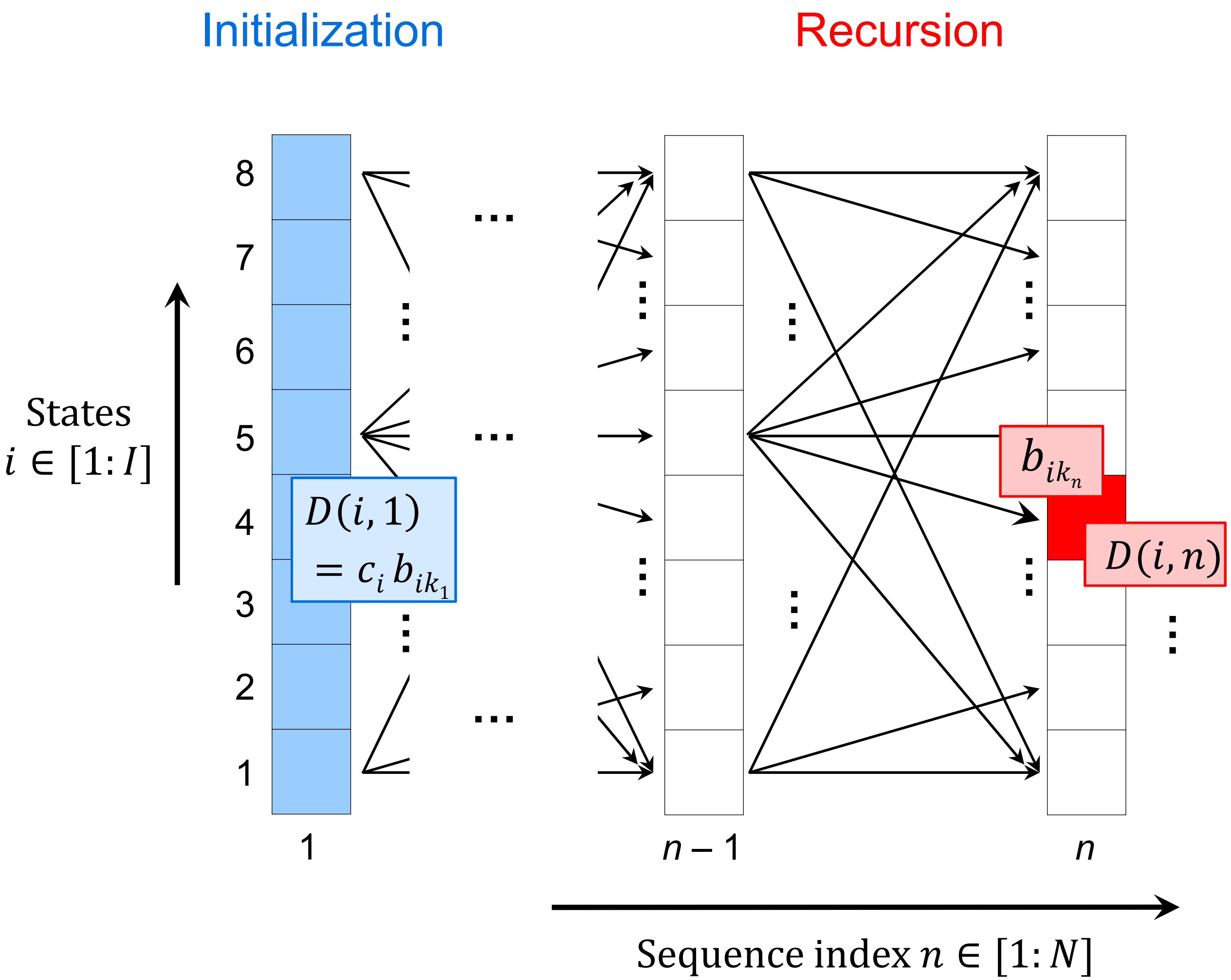


Viterbi Algorithm



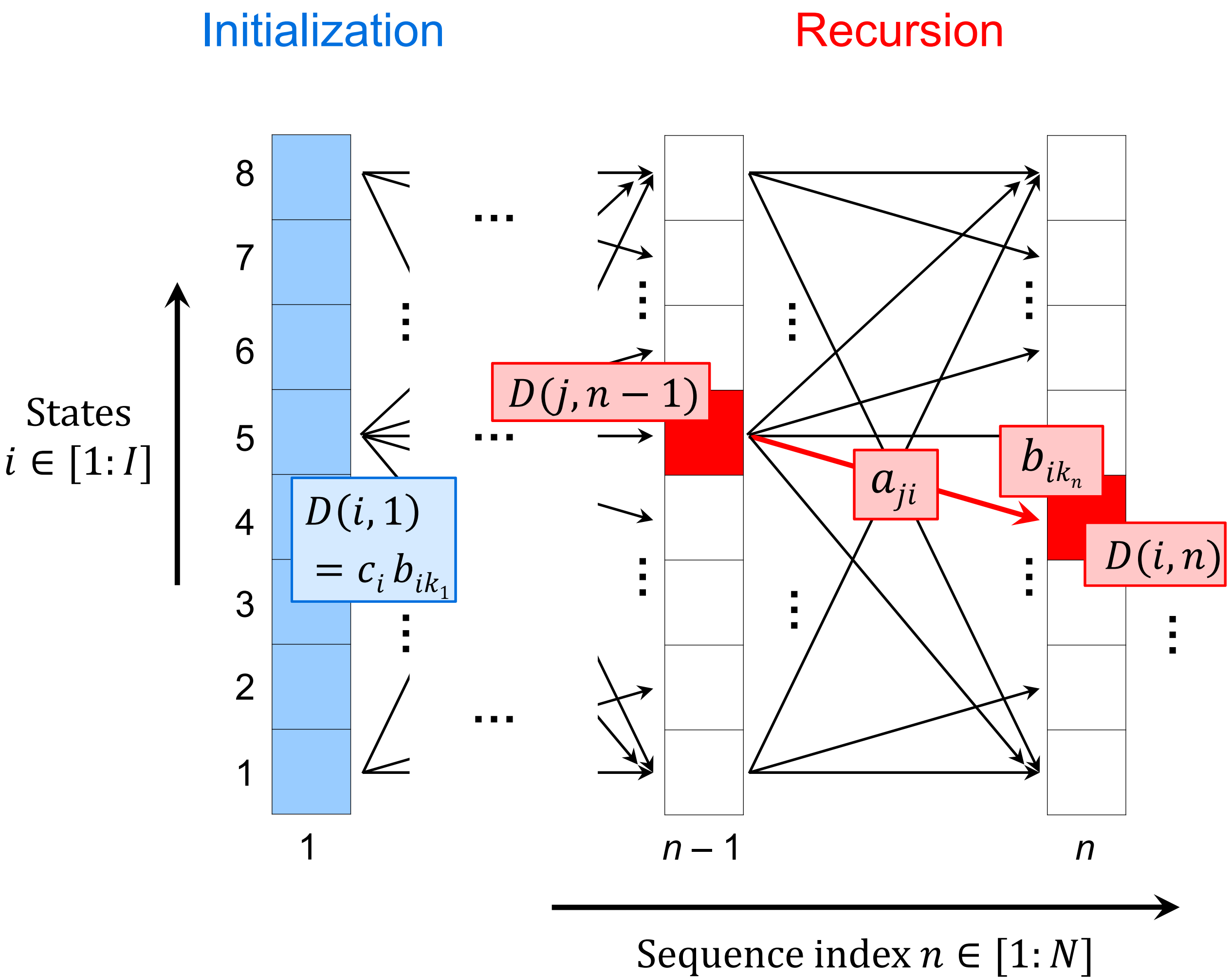
Viterbi Algorithm

$o_n = \beta_{k_n}$

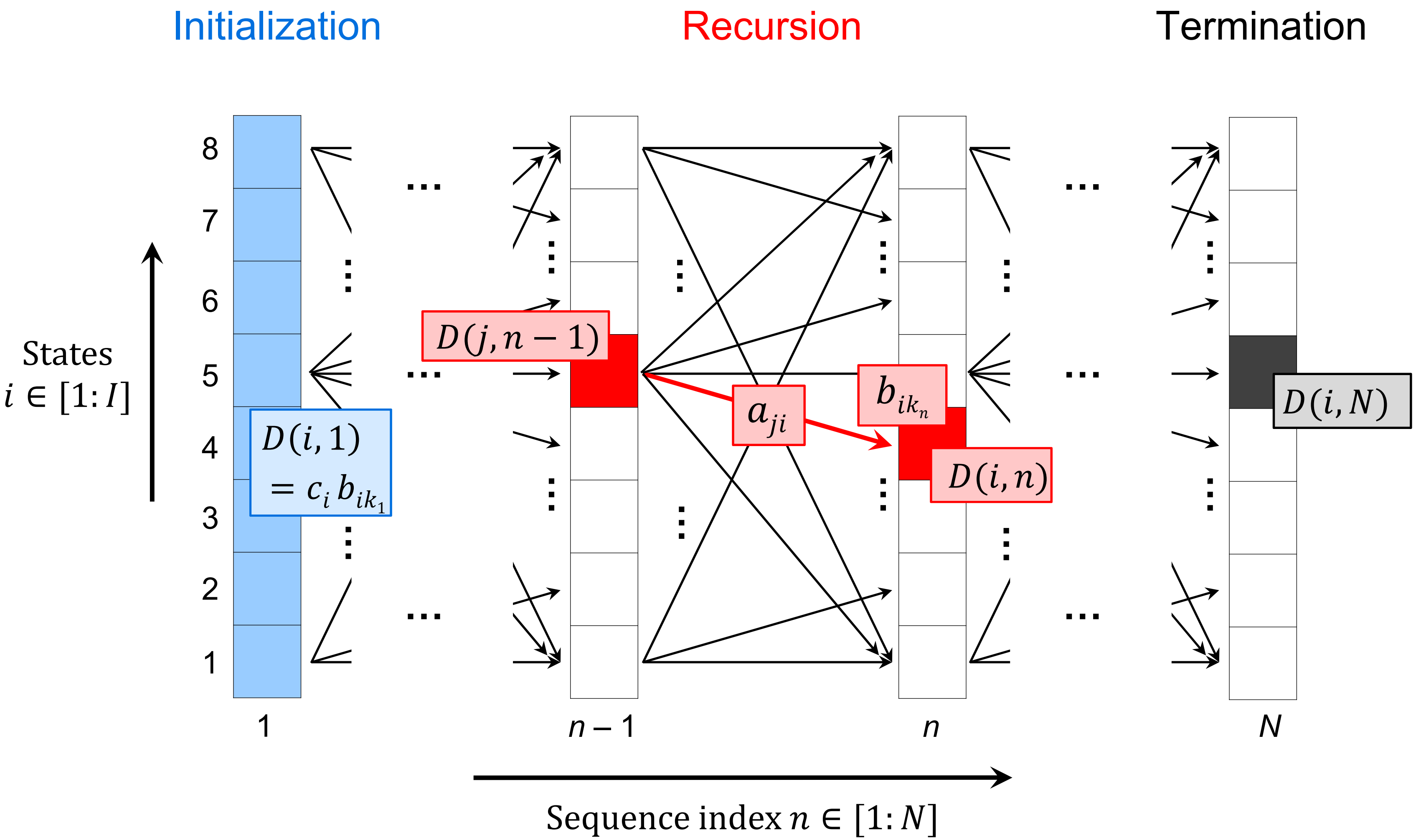


Viterbi Algorithm

$$\mathbf{D}(i,n) = b_{ik_n} \cdot \max_{j\in[1:I]} \left(a_{ji} \cdot \mathbf{D}(j,n-1)\right)$$



Viterbi Algorithm

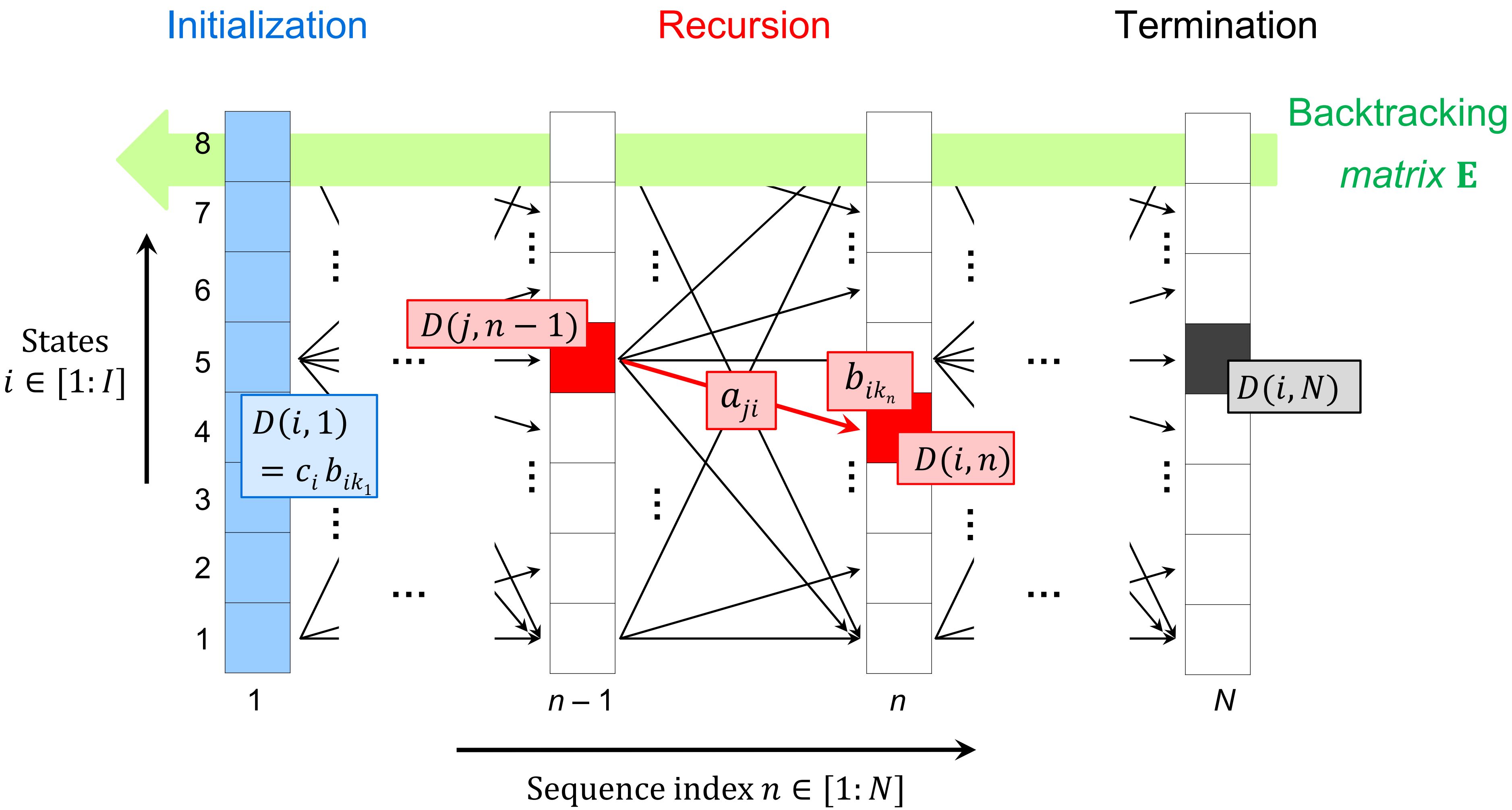


Viterbi Algorithm

$$S^* = (s_1^*, \dots, s_N^*) := (\alpha_{i_1}, \dots, \alpha_{i_N})$$

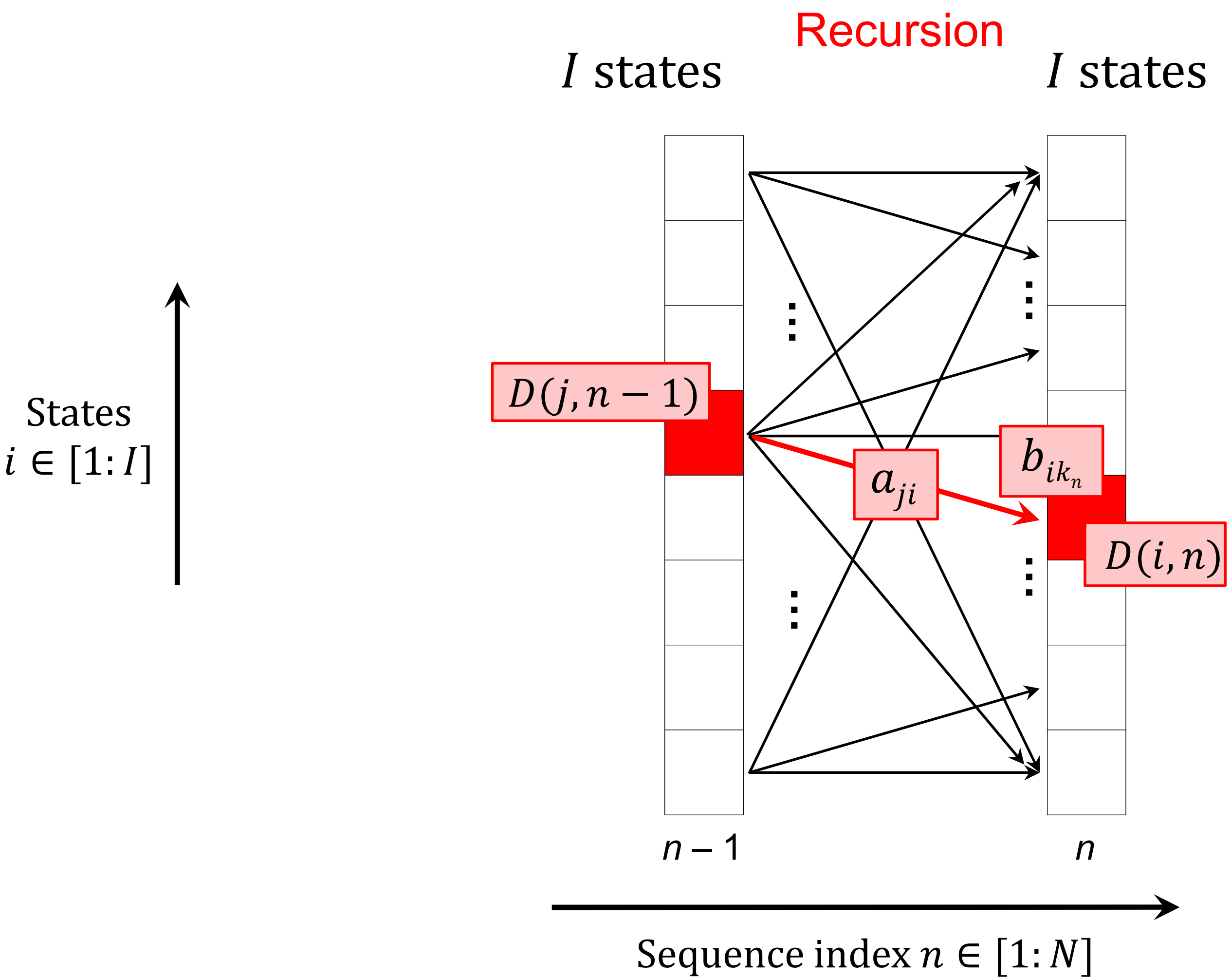
$$i_N = \underset{j \in [1:I]}{\operatorname{argmax}} \, \mathbf{D}(j, N)$$

$$i_n = \underset{j \in [1:I]}{\operatorname{argmax}} \left(a_{j i_{n+1}} \cdot \mathbf{D}(j, n) \right)$$



Viterbi Algorithm

Computational Complexity



Per recursion step:
$I \cdot I$

Total recursion:
$I^2 \cdot N$

Viterbi Algorithm

Summary

Algorithm: VITERBI

Input: HMM specified by $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$
Observation sequence $O = (o_1 = \beta_{k_1}, o_2 = \beta_{k_2}, \ldots, o_N = \beta_{k_N})$
Output: Optimal state sequence $S^* = (s_1^*, s_2^*, \ldots, s_N^*)$

Procedure: Initialize the $(I \times N)$ matrix $\mathbf{D}$ by $\mathbf{D}(i,1) = c_i b_{ik_1}$ for $i \in [1:I]$. Then compute in a nested loop for $n = 2, \ldots, N$ and $i = 1, \ldots, I$:

$$\begin{aligned} \mathbf{D}(i,n) &= \max_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1) \right) \cdot b_{ik_n} \\ \mathbf{E}(i,n-1) &= \operatorname{argmax}_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1) \right) \end{aligned}$$

Set $i_N = \operatorname{argmax}_{j \in [1:I]} \mathbf{D}(j,N)$ and compute for decreasing $n = N-1, \ldots, 1$ the maximizing indices

$$i_n = \operatorname{argmax}_{j \in [1:I]} \left(a_{ji_{n+1}} \cdot \mathbf{D}(j,n) \right) = \mathbf{E}(i_{n+1}, n).$$

The optimal state sequence $S^* = (s_1^*, \ldots, s_N^*)$ is defined by $s_n^* = \alpha_{i_n}$ for $n \in [1:N]$.

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α1	α2	α3
α1	a_{11}	a_{12}	a_{13}
α2	a_{21}	a_{22}	a_{23}
α3	a_{31}	a_{32}	a_{33}

Emission probabilities

b_{ik}

B	β1	β2	β3
α1	b_{11}	b_{12}	b_{13}
α2	b_{21}	b_{22}	b_{23}
α3	b_{31}	b_{32}	b_{33}

Initial state probabilities

c_i

C	α1	α2	α3
	c_1	c_2	c_3

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

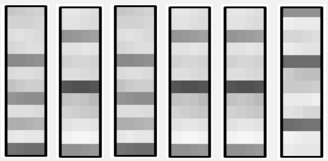
c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

Observation sequence

$O = (o_1, o_2, o_3, o_4, o_5, o_6)$



β_1 β_3 β_1 β_3 β_3 β_2

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

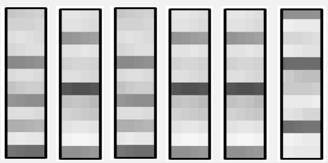
c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

Observation sequence

$O = (o_1,o_2,o_3,o_4,o_5,o_6)$



β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$	$o_6=\beta_2$
α_1						
α_2						
α_3						

E	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$
α_1					
α_2					
α_3					

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

o_1 o_2 o_3 o_4 o_5 o_6

β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$	$o_6 = \beta_2$
α_1						
α_2						
α_3						

E	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$
α_1					
α_2					
α_3					

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

o_1 o_2 o_3 o_4 o_5 o_6

β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$	$o_6 = \beta_2$
α_1	0.4200					
α_2	0.0200					
α_3	0					

E	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$
α_1					
α_2					
α_3					

Initialization

$$\mathbf{D}(i, 1) = c_i \cdot b_{ik_1}$$

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

o_1 o_2 o_3 o_4 o_5 o_6

β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$	$o_6 = \beta_2$
α_1	0.4200					
α_2	0.0200					
α_3	0					

E	$o_1 = \beta_1$	$o_2 = \beta_3$	$o_3 = \beta_1$	$o_4 = \beta_3$	$o_5 = \beta_3$
α_1					
α_2					
α_3					

Initialization

$$\mathbf{D}(i,1) = c_i \cdot b_{ik_1}$$

Recursion

$$\mathbf{D}(i,n) = b_{ik_n} \cdot \max_{j\in[1:I]} \left(a_{ji} \cdot \mathbf{D}(j,n-1)\right)$$

$$\mathbf{E}(i,n-1) = \operatorname{argmax}_{j\in[1:I]} \left(a_{ji} \cdot \mathbf{D}(j,n-1)\right)$$

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α1	α2	α3
α1	0.8	0.1	0.1
α2	0.2	0.7	0.1
α3	0.1	0.3	0.6

Emission probabilities

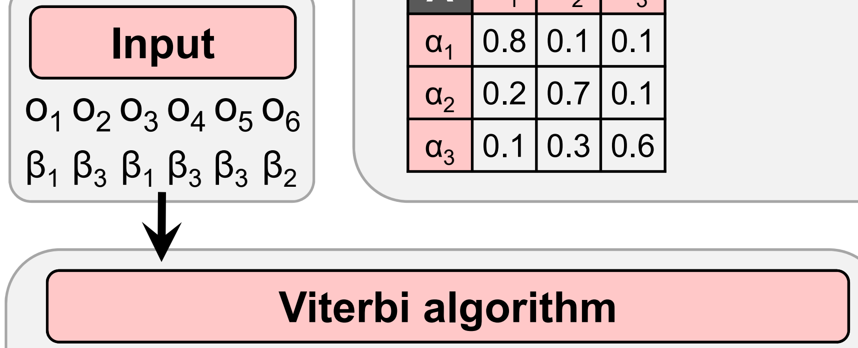
b_{ik}

B	β1	β2	β3
α1	0.7	0	0.3
α2	0.1	0.9	0
α3	0	0.2	0.8

Initial state probabilities

c_i

C	α1	α2	α3
	0.6	0.2	0.2



Input

o_1 o_2 o_3 o_4 o_5 o_6

β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$	$o_6=\beta_2$
α1	0.4200	0.1008				
α2	0.0200	0				
α3	0	0.0336				

E	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$
α1	1				
α2	1				
α3	1				

0.42 * 0.8 * 0.3 = 0.1008
0.42 * 0.2 * 0 = 0
0.42 * 0.1 * 0.8 = 0.0336

Initialization

$$\mathbf{D}(i,1) = c_i \cdot b_{ik_1}$$

Recursion

$$\mathbf{D}(i,n) = b_{ik_n} \cdot \max_{j\in[1:I]} \left(a_{ji} \cdot \mathbf{D}(j,n-1)\right)$$

$$\mathbf{E}(i,n-1) = \operatorname*{argmax}_{j\in[1:I]} \left(a_{ji} \cdot \mathbf{D}(j,n-1)\right)$$

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α1	α2	α3
α1	0.8	0.1	0.1
α2	0.2	0.7	0.1
α3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β1	β2	β3
α1	0.7	0	0.3
α2	0.1	0.9	0
α3	0	0.2	0.8

Initial state probabilities

c_i

C	α1	α2	α3
	0.6	0.2	0.2

Input

o_1 o_2 o_3 o_4 o_5 o_6

β_1 β_3 β_1 β_3 β_3 β_2

Viterbi algorithm

D	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$	$o_6=\beta_2$
α1	0.4200	0.1008	0.0564	0.0135	0.0033	0
α2	0.0200	0	0.0010	0	0	0.0006
α3	0	0.0336	0	0.0045	0.0022	0.0003

E	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$
α1	1	1	1	1	1
α2	1	1	1	1	3
α3	1	3	1	3	3

Backtracking

$$i_N = \underset{j\in[1:I]}{\operatorname{argmax}} \mathbf{D}(j,n)$$

$$i_n = \mathbf{E}(i_{n+1},n)$$

Viterbi Algorithm: Example

HMM:

States

α_i for $i \in [1:I]$

Observation symbols

β_k for $k \in [1:K]$

State transition probabilities

a_{ij}

A	α_1	α_2	α_3
α_1	0.8	0.1	0.1
α_2	0.2	0.7	0.1
α_3	0.1	0.3	0.6

Emission probabilities

b_{ik}

B	β_1	β_2	β_3
α_1	0.7	0	0.3
α_2	0.1	0.9	0
α_3	0	0.2	0.8

Initial state probabilities

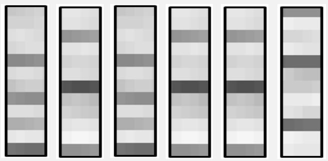
c_i

C	α_1	α_2	α_3
	0.6	0.2	0.2

Input

Observation sequence

$O = (o_1,o_2,o_3,o_4,o_5,o_6)$



Viterbi algorithm

D	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$	$o_6=\beta_2$
α_1	0.4200	0.1008	0.0564	0.0135	0.0033	0
α_2	0.0200	0	0.0010	0	0	0.0006
α_3	0	0.0336	0	0.0045	0.0022	0.0003

E	$o_1=\beta_1$	$o_2=\beta_3$	$o_3=\beta_1$	$o_4=\beta_3$	$o_5=\beta_3$
α_1	1	1	1	1	1
α_2	1	1	1	1	3
α_3	1	3	1	3	3

$i_6 = 2$

Output

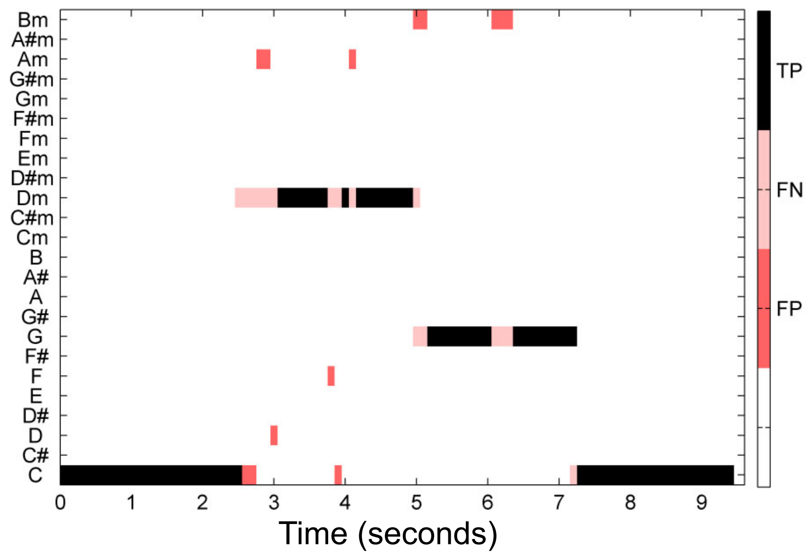
Optimal state sequence

$S^* = (\alpha_1,\alpha_1,\alpha_1,\alpha_3,\alpha_3,\alpha_2)$

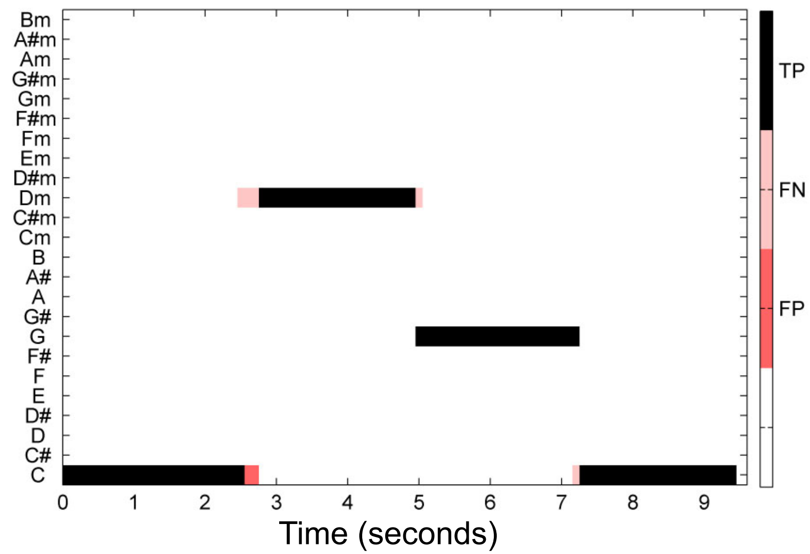
HMM: Application to Chord Recognition

- Effect of HMM-based chord estimation and smoothing:

(a) Template Matching (frame-wise)

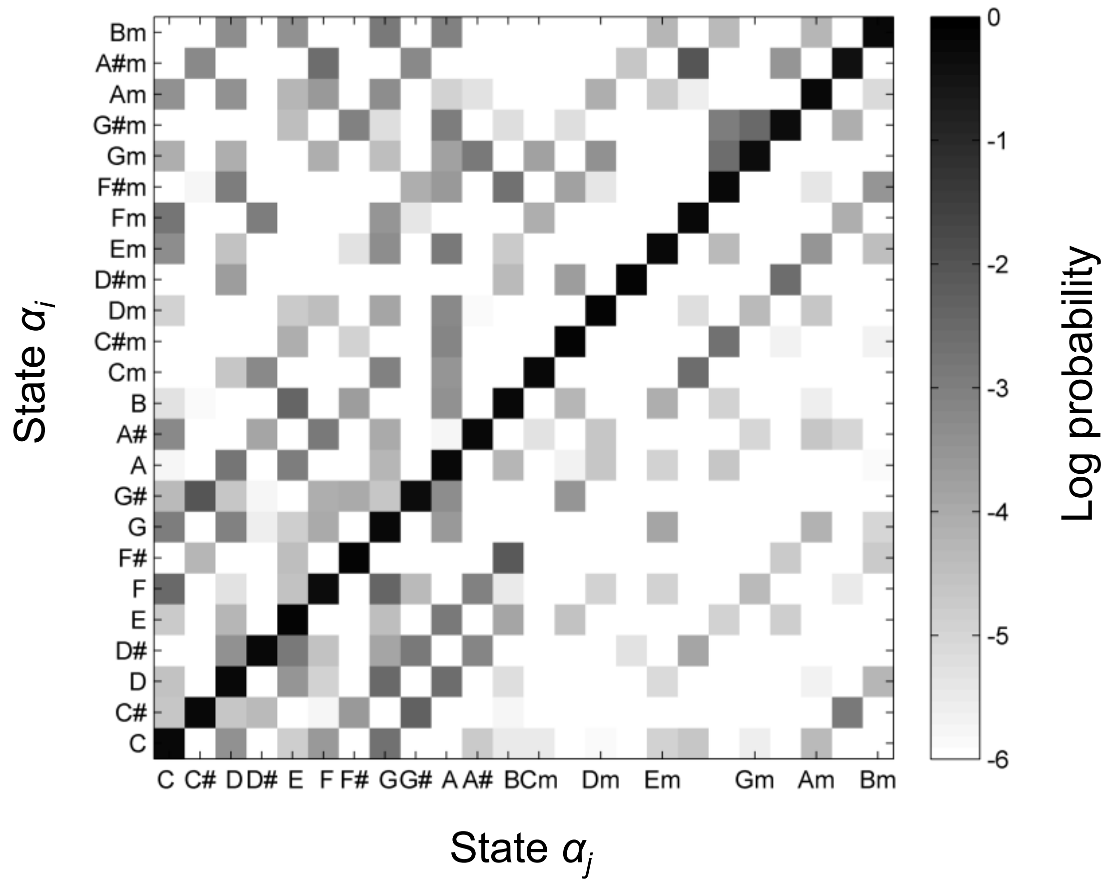


(b) HMM



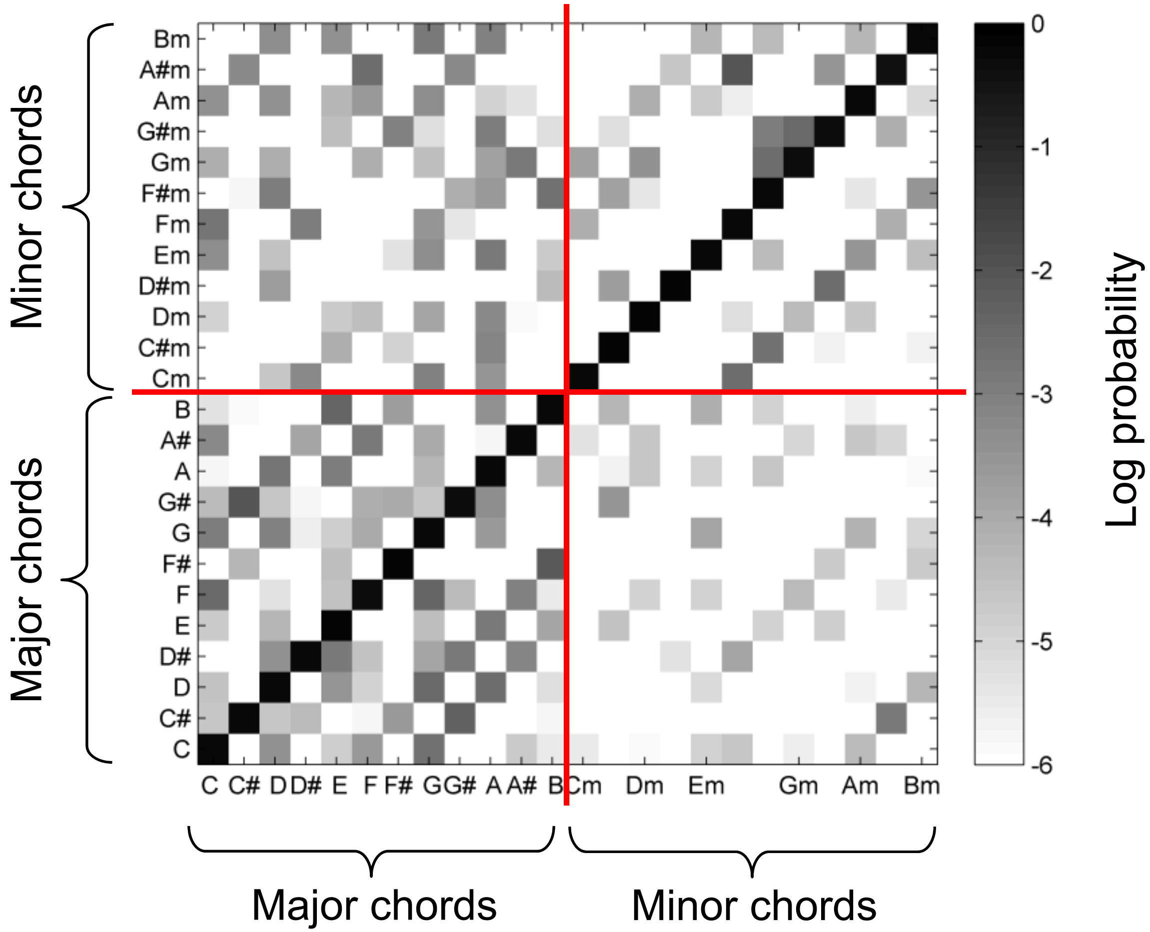
HMM: Application to Chord Recognition

- Parameters: **Transition probabilities**
- Estimated from data



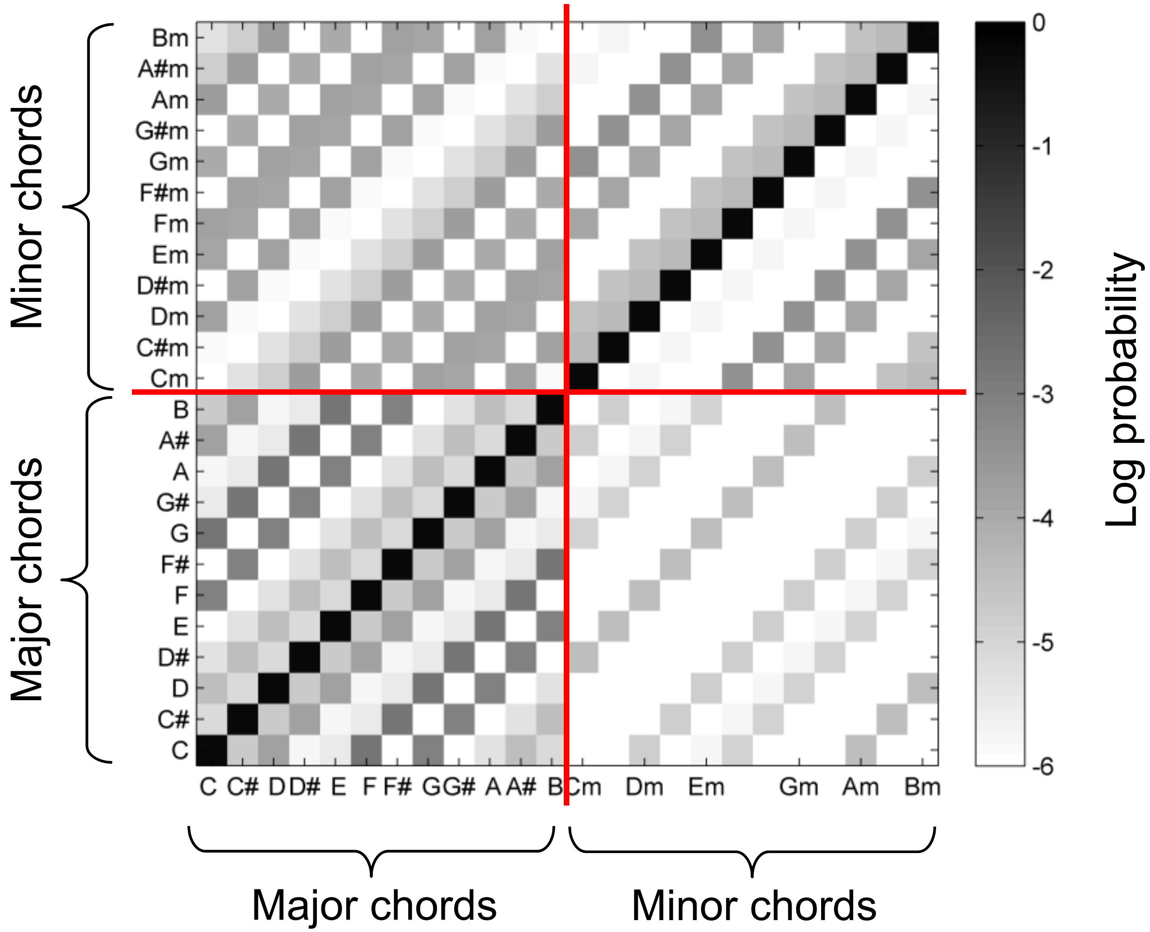
HMM: Application to Chord Recognition

- Parameters: **Transition probabilities**
- Estimated from data



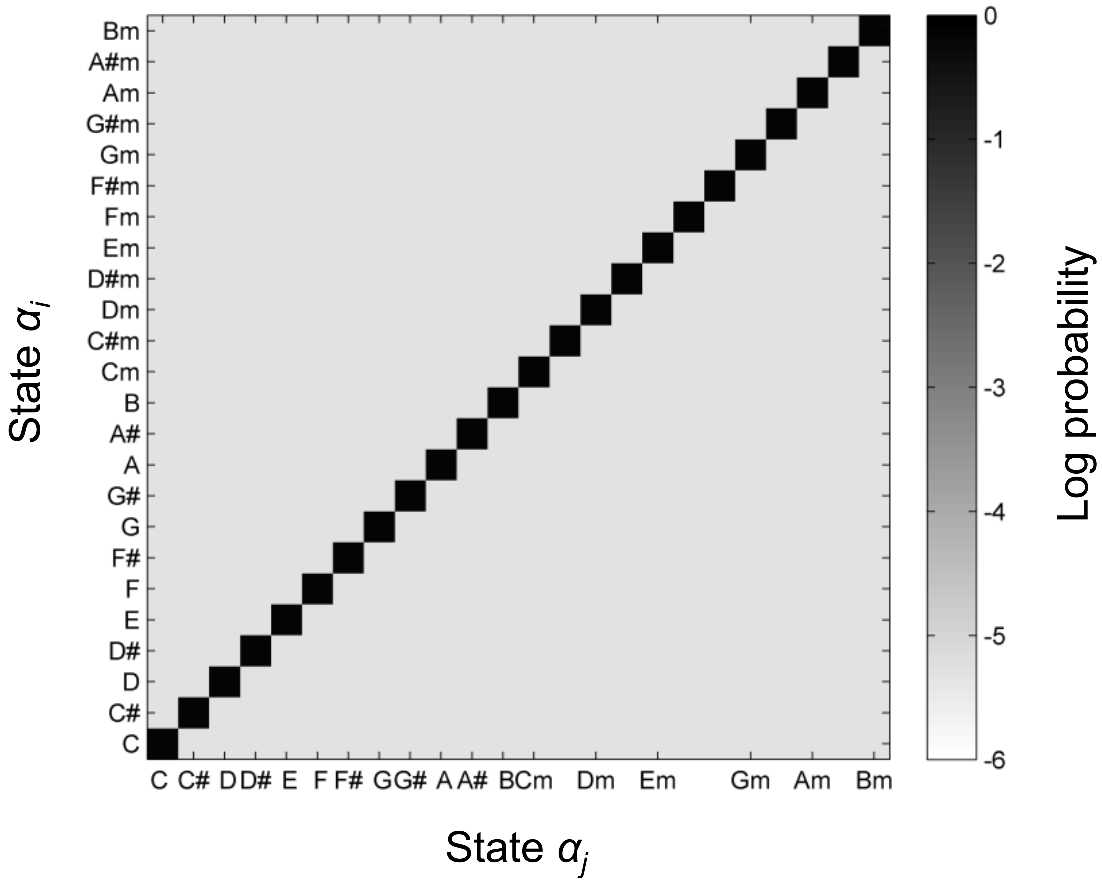
HMM: Application to Chord Recognition

- Parameters: **Transition probabilities**
- Transposition-invariant



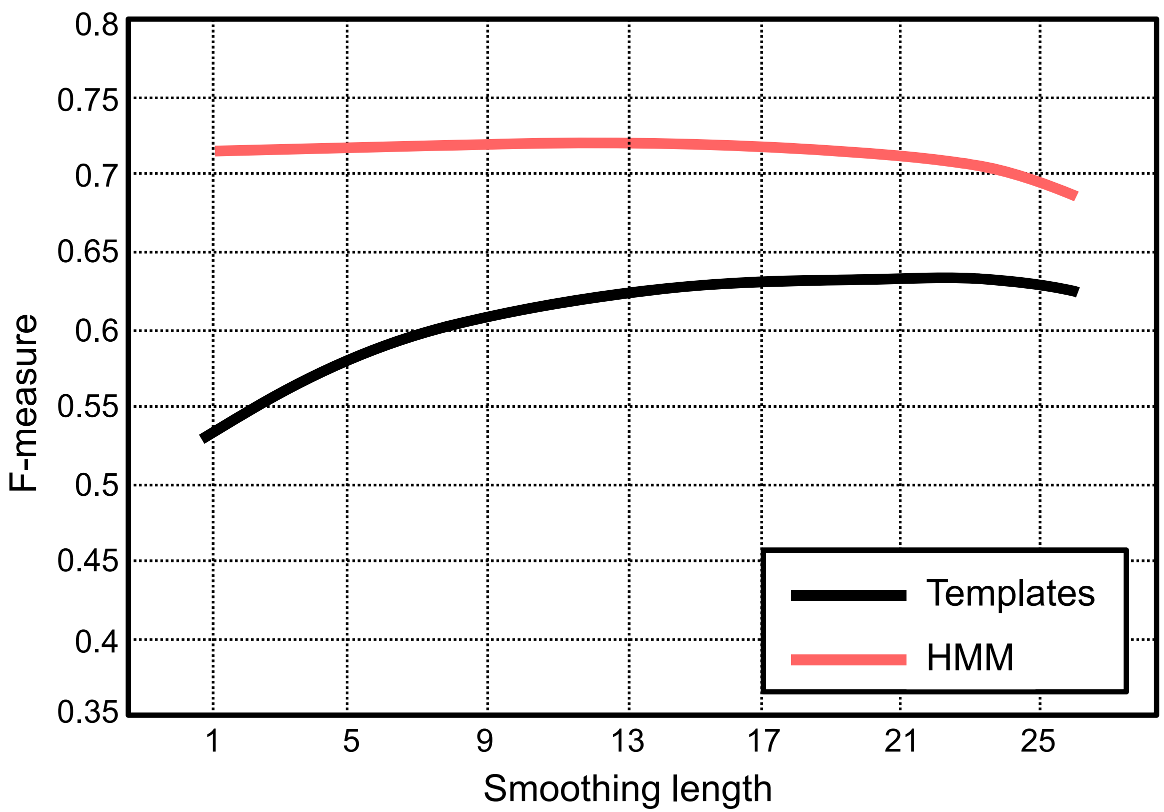
HMM: Application to Chord Recognition

- Parameters: **Transition probabilities**
- Uniform, diagonal-enhanced transition matrix (only smoothing)



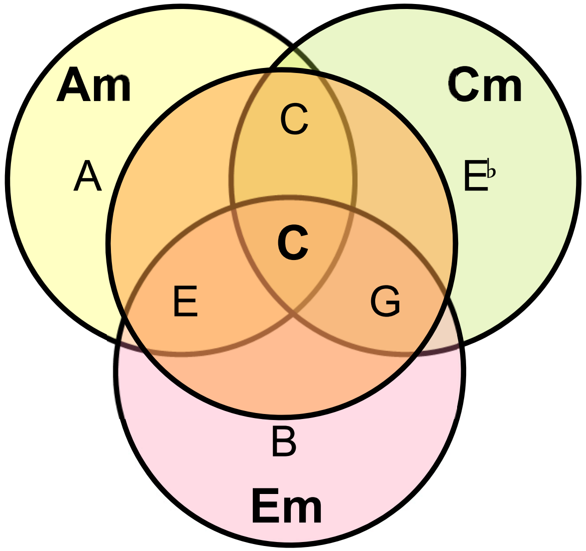
HMM: Application to Chord Recognition

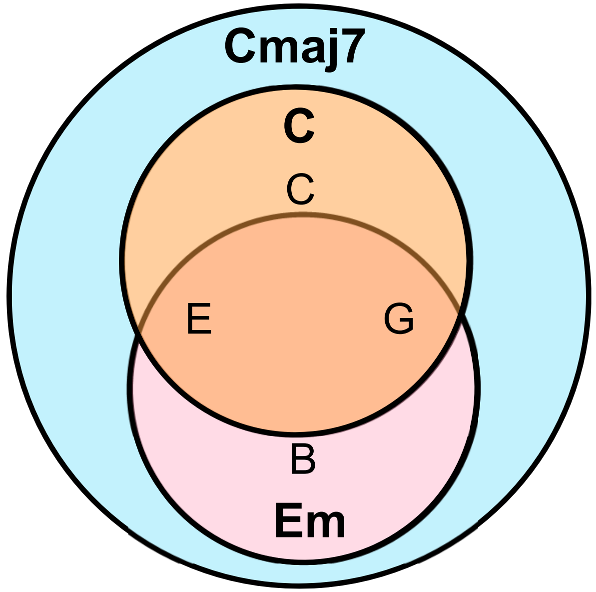
- Evaluation on all Beatles songs



Chord Recognition: Further Challenges

- Chord ambiguities





- Acoustic ambiguities (overtones)
  - Use advanced templates (model overtones, learned templates)
  - Enhanced chroma (logarithmic compression, overtone reduction)
- Tuning inconsistency