

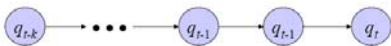
Hidden Markov Models

Chen Yu
Indiana University

Introduction

- Modeling dependencies in input.
- Sequences:
 - Temporal: In speech; phonemes in a word (dictionary), words in a sentence (syntax, semantics of the language). In handwriting, pen movements
 - Spatial: In a DNA sequence; base pairs

Markov Property



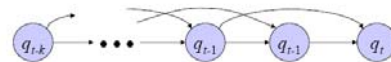
1st-order Markov model

q_t represents the state at time t

$$P(q_t | q_{t-1}, q_{t-2}, \dots) = P(q_t | q_{t-1})$$

$$P(q_t, q_{t-1}, q_{t-2}, \dots) = P(q_t | q_{t-1})P(q_{t-1} | q_{t-2}) \dots$$

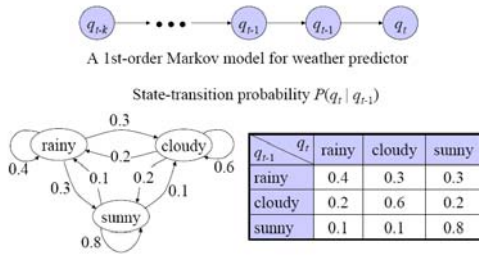
Markov Model



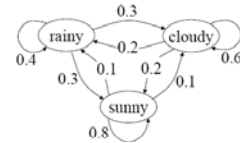
2nd-order Markov model

$$P(q_t | q_{t-1}, q_{t-2}, \dots) = P(q_t | q_{t-1}, q_{t-2})$$

An Example



P(Observation | Model)



Question: given the day 1 is sunny, what is the probability that the weather for the next 7 days will be "sun-sun-rain-rain-sun-cloudy-sun"?

$$\begin{aligned}
 P(O|Model) &= P[S_3, S_3, S_3, S_1, S_1, S_3, S_2, S_3|Model] \\
 &= P[S_3] \cdot P[S_3|S_3] \cdot P[S_3|S_3] \cdot P[S_1|S_3] \\
 &\quad \cdot P[S_1|S_1] \cdot P[S_3|S_1] \cdot P[S_2|S_3] \cdot P[S_3|S_2]
 \end{aligned}$$

Question: given the model is in a known state, what is the probability it stays in that state for exactly d days?

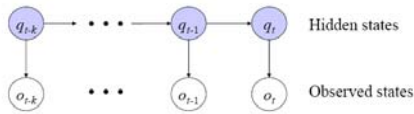
$$O = \{S_1, S_2, S_3, \dots, S_d, S_{d+1} \neq S_d\}$$

$$P(O|Model, q_1 = S_i) = (a_{ii})^{d-1}(1 - a_{ii})$$

From Markov To Hidden Markov

- **The previous model assumes that each state can be uniquely associated with an observable event**
 - Once an observation is made, the state of the system is then trivially retrieved
 - This model, however, is too restrictive to be of practical use for most realistic problems
- **To make the model more flexible, we will assume that the outcomes or observations of the model are a probabilistic function of each state**
 - Each state can produce a number of outputs according to a unique probability distribution, and each distinct output can potentially be generated at any state
 - These are known as **Hidden Markov Models (HMM)**, because the state sequence is not directly observable, it can only be approximated from the sequence of observations produced by the system

Hidden Markov Models



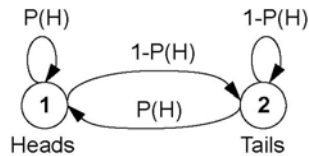
- Graphical Model
- Circles indicate states
- Arrows indicate probabilistic dependencies between states

The coin-toss problem

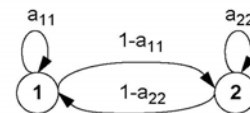
- To illustrate the concept of an HMM consider the following scenario
 - Assume that you are placed in a room with a curtain
 - Behind the curtain there is a person performing a coin-toss experiment
 - This person selects one of several coins, and tosses it: heads (H) or tails (T)
 - The person tells you the outcome (H,T), but not which coin was used each time
- Your goal is to build a probabilistic model that best explains a sequence of observations $O=\{o_1,o_2,o_3,o_4,\dots\}=\{H,T,H,\dots\}$
 - The coins represent the states; these are hidden because you do not know which coin was tossed each time
 - The outcome of each toss represents an observation
 - A “likely” sequence of coins may be inferred from the observations, but this state sequence will not be unique

The Coin Toss Example – 1 coin

- As a result, the Markov model is observable since there is only one state
- In fact, we may describe the system with a deterministic model where the states are the actual observations (see figure)
- the model parameter $P(H)$ may be found from the ratio of heads and tails
- $O = H H H T T H \dots$
- $S = 1 1 1 2 2 1 \dots$



The Coin Toss Example – 2 coins



$$P(H) = P_1 \quad P(H) = P_2$$

$$P(T) = 1-P_1 \quad P(T) = 1-P_2$$

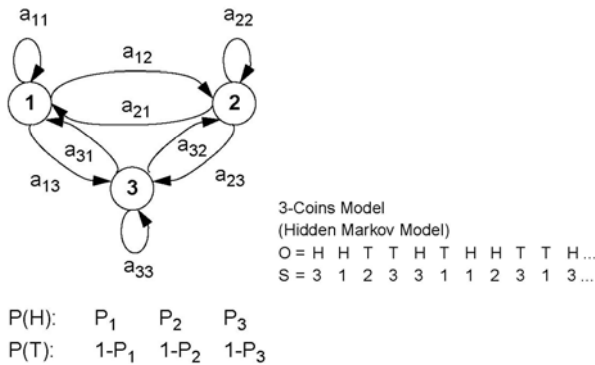
2-Coins Model

(Hidden Markov Model)

$O = H H T T H T H H T T H \dots$

$S = 1 1 2 2 1 2 1 1 2 2 1 \dots$

From Markov to Hidden Markov Model: The Coin Toss Example - 3 coins



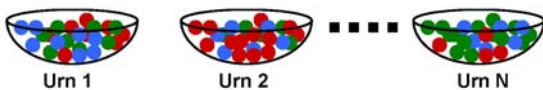
1, 2 or 3 coins?

• Which of these models is best?

- Since the states are not observable, the best we can do is select the model that best explains the data (e.g., Maximum Likelihood criterion)
- Whether the observation sequence is long and rich enough to warrant a more complex model is a different story, though

The urn-ball problem

- To further illustrate the concept of an HMM, consider this scenario
 - You are placed in the same room with a curtain
 - Behind the curtain there are N urns, each containing a large number of balls with M different colors
 - The person behind the curtain selects an urn according to an internal random process, then randomly grabs a ball from the selected urn
 - He shows you the ball, and places it back in the urn
 - This process is repeated over and over
- Questions?
 - How would you represent this experiment with an HMM?
 - What are the states?



Double Stochastic System

The Urn-and-Ball Model



$P(\text{red}) = b_1(1)$	$P(\text{red}) = b_2(1)$	$P(\text{red}) = b_3(1)$
$P(\text{green}) = b_1(2)$	$P(\text{green}) = b_2(2)$	$P(\text{green}) = b_3(2)$
$P(\text{blue}) = b_1(3)$	$P(\text{blue}) = b_2(3)$	$P(\text{blue}) = b_3(3)$
$P(\text{yellow}) = b_1(4)$	$P(\text{yellow}) = b_2(4)$	$P(\text{yellow}) = b_3(4)$

$O = \{ \text{green, blue, green, yellow, red, ..., blue} \}$

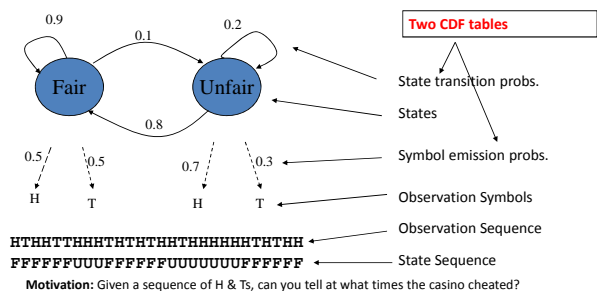
How can we determine the appropriate model for the observation sequence given the system above?

Elements in HMM

- $S = \{1, 2, \dots, N\}$ – all possible values of hidden states.
- o_t – The observation at time t
- $V = \{1, 2, \dots, M\}$ – all possible values of observed states.
- $A = [a_{ij}]_{i,j}$ – state transition probabilities.
 $a_{ij} = P(q_{t+1} = j | q_t = i), a_{ij} \geq 0, 1 \leq i, j \leq N, \sum_j a_{ij} = 1.$
- $B = [b_{ik}]_{i,k}$ – emission probabilities.
 $b_{ik} = P(o_t = k | q_t = i), b_{ik} \geq 0, 1 \leq i \leq N \& 1 \leq k \leq M, \sum_k b_{ik} = 1.$
- $\pi = \{\pi_i\}$ – initial state probabilities. $\pi_i = P(q_1 = i), \sum_i \pi_i = 1.$

$$\lambda = (A, B, \pi)$$

HMM Example - Casino Coin



Problem 1

Problem 1: Given the observation sequence $O = O_1 O_2 \dots O_T$, and a model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

How well a given model matches a given observation sequence. If we consider the case in which we are trying to choose among several competing models, the solution to Problem 1 allows us to choose the model which best matches the observations.

Example: Balls and Urns (HMM): Learning I

- Three urns each full of balls of different colors:

S_1 : state 1, S_2 : state 2, S_3 : state 3: start at urn

1. red green blue urn1 urn2 urn3

$$B = \begin{bmatrix} \text{urn 1} & 0.5, 0.2, 0.3 \\ \text{urn 2} & 0.2, 0.3, 0.5 \\ \text{urn 3} & 0.2, 0.5, 0.3 \end{bmatrix} \quad B = \begin{bmatrix} \text{urn 1} & 0.4 & 0.3 & 0.3 \\ \text{urn 2} & 0.2 & 0.6 & 0.2 \\ \text{urn 3} & 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$S = \{S_0 = 1; S_1 = 1, S_2 = 2, S_3 = 3\}; b = [b_0 = 1; b_1 = 3; b_2 = 2; b_3 = 1]$$

$$P(O, S | A, B) =$$

Problem 2

Problem 2: Given the observation sequence $O = O_1 O_2 \dots O_T$, and the model λ , how do we choose a corresponding state sequence $Q = q_1 q_2 \dots q_T$ which is optimal in some meaningful sense (i.e., best “explains” the observations)?

Problem 3

Problem 3: How do we adjust the model parameters $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$?

Solution to Problem 1

Given $O = o_1 o_2 \dots o_T$ and λ , compute $P(O|\lambda)$.

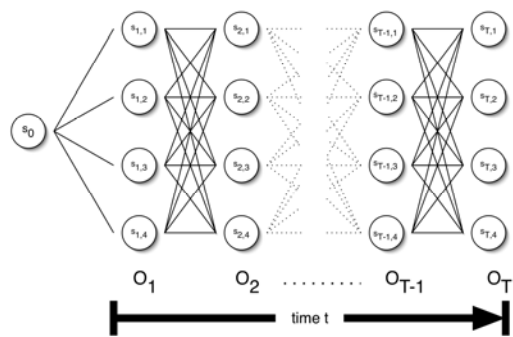
$$P(O|\lambda) = \sum_Q P(O, Q|\lambda) = \sum_Q P(O|Q, \lambda) P(Q|\lambda)$$

$$P(O|Q, \lambda) = \prod_{t=1}^T P(o_t | q_t, \lambda) \quad P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$$

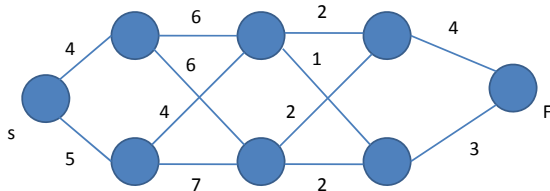
However, there are N^T possible hidden state sequences!

There is an efficient way – dynamic programming.

The Trellis



Dynamic Programming

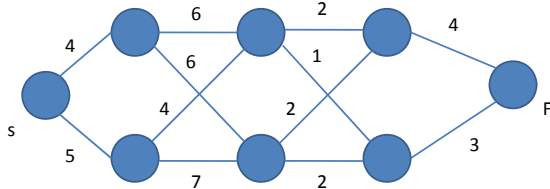


From S to F, 4 steps, two states in each step.

Principle of Optimality (Bellman 1965)

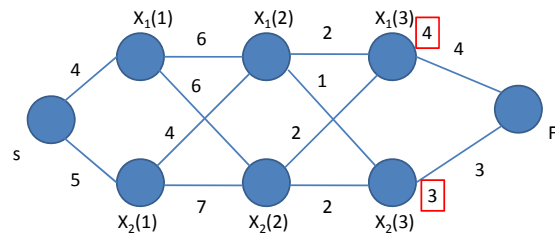
- From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point.
 - Problem 1: from 0 to T
 - Problem 2: from t1 to T
 - Problem 3: from 0 to t1
- Any intermediate point in the optimal path must be the optimal point linking the optimal **partial paths before and after** that point.

Dynamic Programming



- At each step we need to make a decision, up or down.
- The basic idea of principle optimality is that we proceed backward and at each step, we find the best path at this time, from the current state to the destination.
- At each step, we go back one more step back and deal with the sub-problem based on previous solutions.

Step 1



Step 1: $t = 3$

$$J(x_1(3), 3) = 4$$

$$J(x_2(3), 3) = 3$$

Step 2

$$J(x_1(2),2) = \min \begin{pmatrix} J(x_1(3),3) + D(x_1(2), x_1(3)) \\ J(x_2(3),3) + D(x_1(2), x_2(3)) \end{pmatrix}$$

$$J(x_2(2),2) = \min \begin{pmatrix} J(x_1(3),3) + D(x_2(2), x_1(3)) \\ J(x_2(3),3) + D(x_2(2), x_2(3)) \end{pmatrix}$$

Step 3

$$J(x_1(1),1) = \min \begin{pmatrix} J(x_1(2),2) + D(x_1(1), x_1(2)) \\ J(x_2(2),2) + D(x_1(1), x_2(2)) \end{pmatrix}$$

$$J(x_2(1),1) = \min \begin{pmatrix} J(x_1(2),2) + D(x_2(1), x_1(2)) \\ J(x_2(2),2) + D(x_2(1), x_2(2)) \end{pmatrix}$$

Step 4

$$J(x_1(1),1) = \min \begin{pmatrix} J(x_1(2),2) + D(x_1(1), x_1(2)) \\ J(x_2(2),2) + D(x_1(1), x_2(2)) \end{pmatrix}$$

$$J(x_2(1),1) = \min \begin{pmatrix} J(x_1(2),2) + D(x_2(1), x_1(2)) \\ J(x_2(2),2) + D(x_2(1), x_2(2)) \end{pmatrix}$$

The forward Procedure

$\alpha_t(i) = P(O_1 O_2 \cdots O_t, q_t = S_i | \lambda)$

The probability of the partial observation sequence and state S_i at time t given the model.

(1) Initialize

$$\alpha_1(i) = \pi_i b_{i o_1}, \quad 1 \leq i \leq N$$

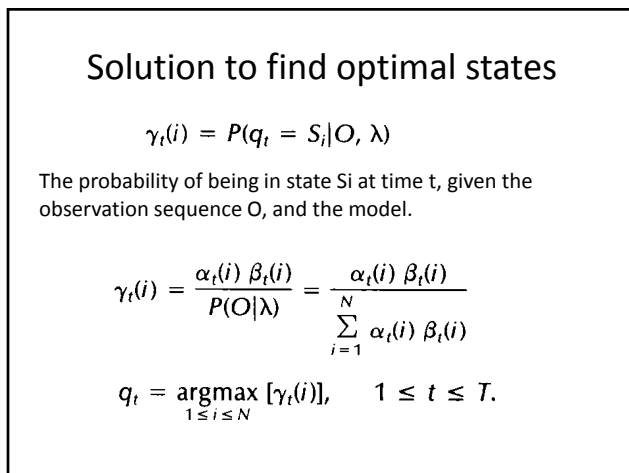
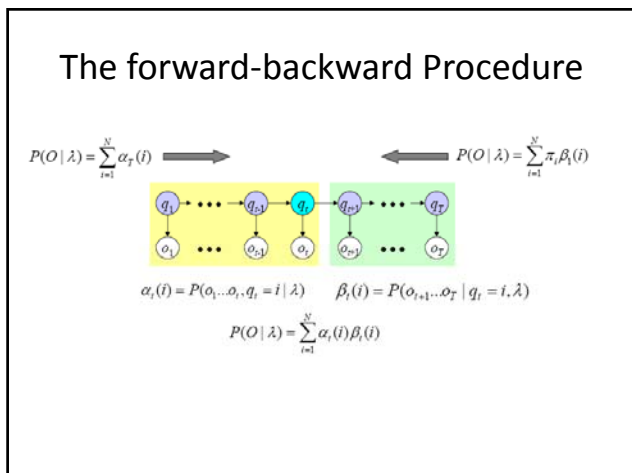
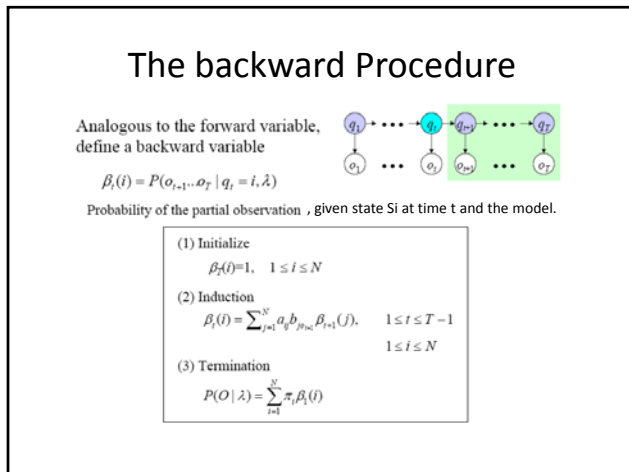
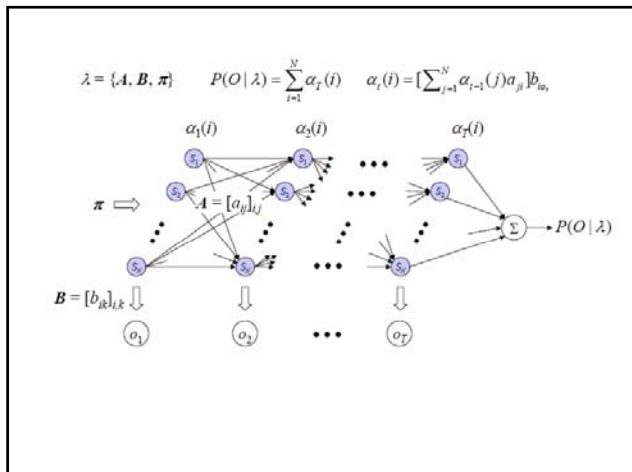
(2) Induction

$$\alpha_t(i) = [\sum_{j=1}^N \alpha_{t-1}(j) a_{ji}] b_{i o_t}, \quad 2 \leq t \leq T$$

$1 \leq i \leq N$

(3) Termination

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i)$$



A better Solution

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \dots q_t = i, O_1 O_2 \dots O_t | \lambda]$$

The highest probability along a state sequence that accounts for the first t observations and ends at state S_i .

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}).$$

The Viterbi Algorithm

1. Initialization

$$\delta_1(i) = \pi_i b_{i1} \text{ and } \psi_1(i) = 0, \quad 1 \leq i \leq N$$

2. for $t = 1$ to $T-1$

$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] b_{j,t+1}$$

$$\psi_{t+1}(j) = \arg \max_i \delta_t(i) a_{ij} b_{j,t+1}$$

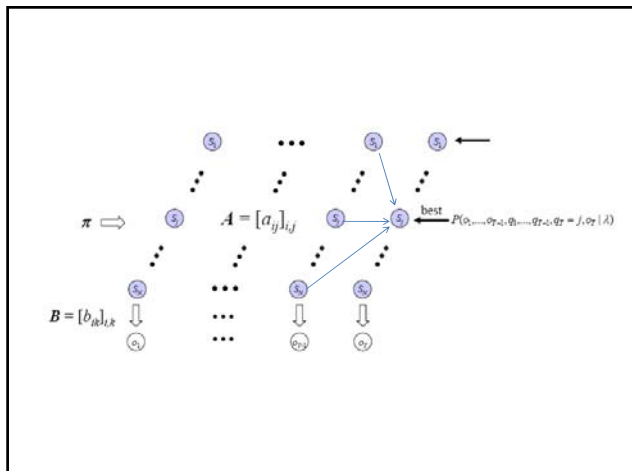
endfor

3. Termination

$$P^* = \max_{1 \leq i \leq N} \delta_T(i) \quad q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$$

4. State sequence backtracking

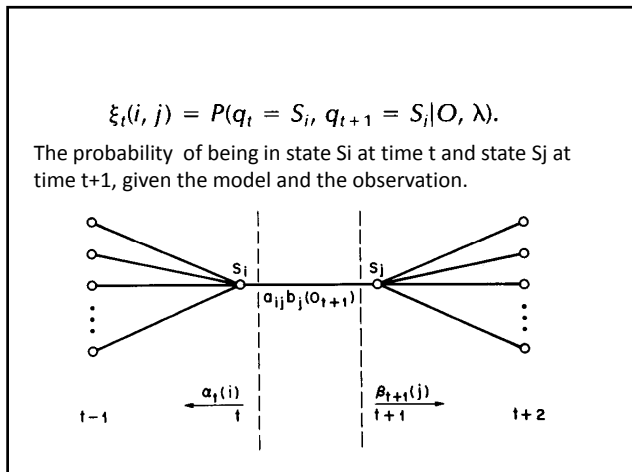
$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$



Solution to Problem 3

$$\lambda^* = \arg \max_{\lambda} P(O | \lambda)$$

Easy if the hidden states are known.



Baum-Welch Method

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}$$

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j). \quad \gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^N \alpha_t(i) \beta_t(i)}$$

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i, j) = \text{expected number of transitions from } S_i \text{ to } S_j$$

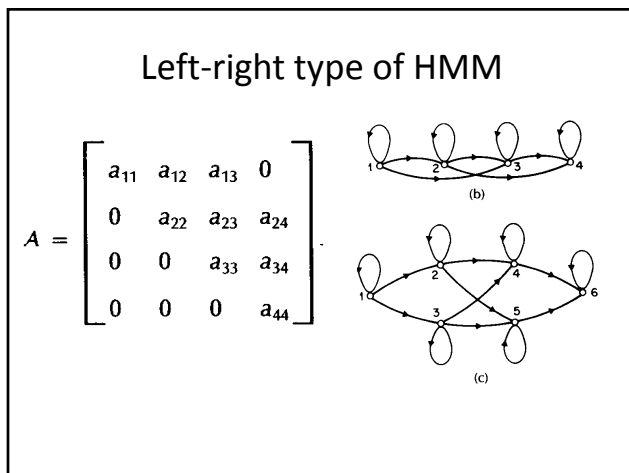
$\bar{\pi}_i$ = expected frequency (number of times) in state S_i at time $(t = 1) = \gamma_1(i)$

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$= \frac{\sum_{t=1}^T \gamma_t(j) \delta_{jk}}{\sum_{t=1}^T \gamma_t(j)}$$



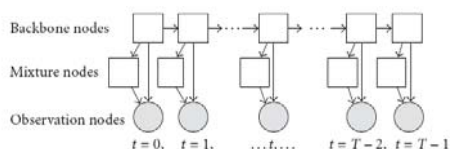
Continuous Observation Densities in HMMs

The observations are continuous values. A common representation is a mixture of Gaussian.

$$b_j(\mathbf{O}) = \sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{O}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm}), \quad 1 \leq j \leq N$$

$$\sum_{m=1}^M c_{jm} = 1, \quad 1 \leq j \leq N$$

$$c_{jm} \geq 0, \quad 1 \leq j \leq N, 1 \leq m \leq M$$



Training

$$\bar{c}_{jk} = \frac{\sum_{t=1}^T \gamma_t(j, k)}{\sum_{t=1}^T \sum_{k=1}^M \gamma_t(j, k)}$$

$$\bar{\boldsymbol{\mu}}_k = \frac{\sum_{t=1}^T \gamma_t(j, k) \cdot \mathbf{O}_t}{\sum_{t=1}^T \gamma_t(j, k)}$$

$$\bar{\mathbf{U}}_k = \frac{\sum_{t=1}^T \gamma_t(j, k) \cdot (\mathbf{O}_t - \bar{\boldsymbol{\mu}}_k)(\mathbf{O}_t - \bar{\boldsymbol{\mu}}_k)'}{\sum_{t=1}^T \gamma_t(j, k)}$$

$$\gamma_t(j, k) = \left[\frac{\alpha_t(j) \beta_t(j)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)} \right] \left[\frac{c_{jk} \mathcal{N}(\mathbf{O}_t, \boldsymbol{\mu}_{jk}, \mathbf{U}_{jk})}{\sum_{m=1}^M c_{jm} \mathcal{N}(\mathbf{O}_t, \boldsymbol{\mu}_{jm}, \mathbf{U}_{jm})} \right]$$

The probability of being in state j at time t with the k th mixture component account for \mathbf{O}_t .

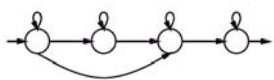
Action representation and recognition

- Core difficulties
 - The configuration of the body remains difficult to understand. This may not be essential to understand what is going on (appearance-based approach, etc.).
 - There is no natural taxonomy of activity.
 - Composition creates fearsome complexity.

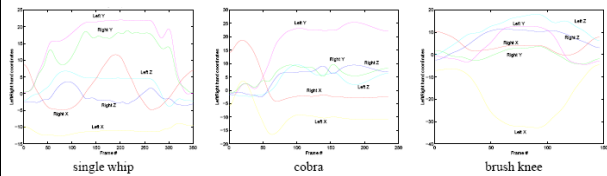
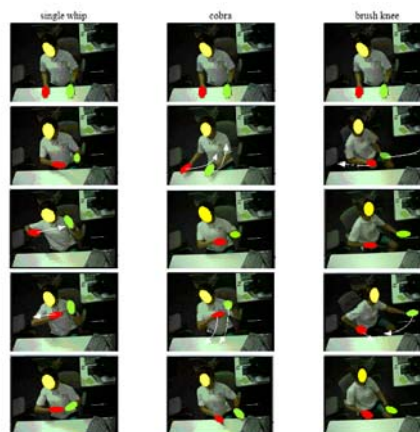
Temporal scale

- Very short timescales
 - no much happens
- Medium timescales
 - running, walking, jogging, jumping, punching, kicking, reaching
- Long timescales: motions are complex composites
 - visiting an ATM, reading a book, cooking a meal

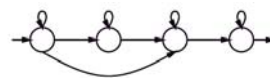
Action Recognition Using HMMs



- Feature extraction
- Training
- Testing

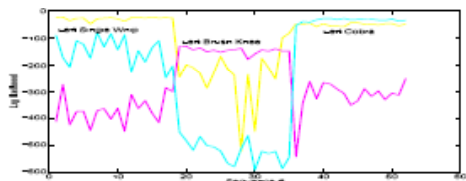


Action Recognition Using HMMs



- Feature extraction
- Training
- Testing

Results

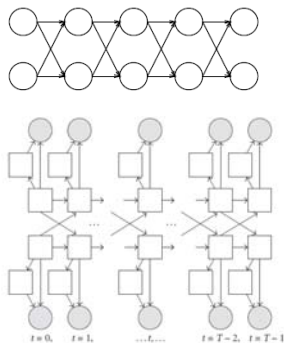


Coupled Hidden Markov Model

- HMMs are usually formulated with multivariate pdfs on the output variables.
- If there are multiple processes generating those channels, one must hope that these processes evolve in lockstep since any variation between them is modeled as noise.
- Two processes may interact without wholly determining each other. E.g. Players in a tennis game, two hand movements.

Coupled HMM

(Nefrian, Liang, Pi, Liu, Mao and Murphy, 2002)



The parameters in CHMM

$$\begin{aligned}\pi_0^c(i) &= P(q_t^c = i) \\ b_i^c(i) &= P(\mathbf{O}_t^c | q_t^c = i) \\ a_{i|j,k}^c &= P(q_t^c = i | q_{t-1}^0 = j, q_{t-1}^1 = k)\end{aligned}$$

$$b_i^c(i) = \sum_{m=1}^{M_i^c} w_{i,m}^c N(\mathbf{O}_t^c, \mu_{i,m}^c, \mathbf{U}_{i,m}^c)$$

Training

- Initialization

$$\begin{aligned}\delta_0(i, j) &= \pi_0^a(i)\pi_0^v(j)b_i^a(i)b_j^v(j) \\ \psi_0(i, j) &= 0\end{aligned}$$

- Recursion

$$\begin{aligned}\delta_t(i, j) &= \max_{k, l} \{\delta_{t-1}(k, l) a_{i|k, l}^a a_{j|k, l}^v\} b_i^a(k) b_j^v(l) \\ \psi_t(i, j) &= \arg \max_{k, l} \{\delta_{t-1}(k, l) a_{i|k, l}^a a_{j|k, l}^v\}\end{aligned}$$

Optimal state sequence using the Viterbi algorithm

- Termination

$$\begin{aligned}P &= \max_{i, j} \{\delta_T(i, j)\} \\ \{q_T^a, q_T^v\} &= \arg \max_{i, j} \{\delta_T(i, j)\}\end{aligned}$$

- Backtracking

$$\{q_t^a, q_t^v\} = \psi_{t+1}(q_{t+1}^a, q_{t+1}^v)$$

The results so far (r – a training sequence)

The sequence of states

$$\mathbf{Q} = q_{r,0}^{a,v}, \dots, q_{r,t}^{a,v}, \dots, q_{r,T-1}^{a,v}$$

The sequence of mixture components

$$\mathbf{S} = s_{0,r}^{a,v}, \dots, s_{r,t}^{a,v}, \dots, s_{r,T-1}^{a,v}$$

$$s_{r,t}^{a,v} = \max_{m=1, \dots, M_t^{a,v}} P(\mathbf{O}_t^{a,v} | q_{r,t}^{a,v} = i, m)$$

Two Latent Variables

$$\gamma_{r,t}^{a,v}(i, m) = \begin{cases} 1, & \text{if } q_{r,t}^{a,v} = i, s_{r,t}^{a,v} = m, \\ 0, & \text{otherwise} \end{cases}$$

$$\epsilon_{r,t}^{a,v}(i, k, l) = \begin{cases} 1, & \text{if } q_{r,t}^{a,v} = i, \\ & q_{r,t-1}^a = k, q_{r,t-1}^v = l \\ 0, & \text{otherwise} \end{cases}$$

$$\mu_{i,m}^{a,v} = \frac{\sum_{r,t} \gamma_{r,t}^{a,v}(i,m) \mathbf{O}_t^{a,v}}{\sum_{r,t} \gamma_{r,t}^{a,v}(i,m)}$$

$$\sigma_{i,m}^{a,v} = \frac{\sum_{r,t} \gamma_{r,t}^{a,v}(i,m) (\mathbf{O}_t^{a,v} - \mu_{i,m}^{a,v})(\mathbf{O}_t^{a,v} - \mu_{i,m}^{a,v})^T}{\sum_{r,t} \gamma_{r,t}^{a,v}(i,m)}$$

$$w_{i,m}^{a,v} = \frac{\sum_{r,t} \gamma_{r,t}^{a,v}(i,m)}{\sum_{r,t} \sum_m \gamma_{r,t}^{a,v}(i,m)}$$

$$a_{i|k,l}^{a,v} = \frac{\sum_{r,t} \epsilon_{r,t}^{a,v}(i,k,l)}{\sum_{r,t} \sum_k \sum_l \epsilon_{r,t}^{a,v}(i,k,l)}$$

Asynchronous HMM (Bengio, NIPS, 2003)

- Model the joint probability of asynchronous sequences describing the same event.
- Early integration
- Later integration
- The key idea is to temporarily stretch one stream in order to obtain a better match between the corresponding frames.

The model

For the sake of simplicity, let us present here the case where one is interested in modeling the joint probability of 2 asynchronous sequences, denoted x_1^T and y_1^S with $S \leq T$ without loss of generality¹.

$$\epsilon(i, t) = P(\tau_t = s | \tau_{t-1} = s - 1, q_t = i, x_1^t, y_1^s)$$

the probability that the system emits the next observation of sequence y at time t while in state i . The additional hidden variable $\tau_t = s$ can be seen as the alignment between y and q (and x which is aligned with q). Hence, we model $p(x_1^T, y_1^S, q_1^T, \tau_1^T)$.

$$\alpha(i, s, t) = p(q_t = i, \tau_t = s, x_1^t, y_1^s)$$

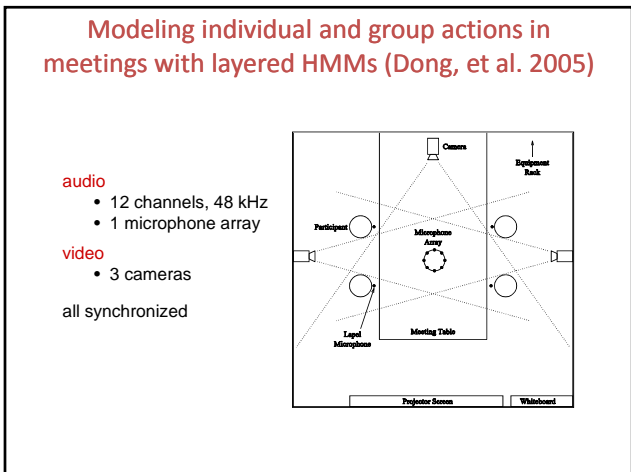
$$\alpha(i, s, t) = \epsilon(i, t) p(x_t, y_s | q_t = i) \sum_{j=1}^N P(q_t = j | q_{t-1} = i) \alpha(j, s - 1, t - 1) + (1 - \epsilon(i, t)) p(x_t | q_t = i) \sum_{j=1}^N P(q_t = j | q_{t-1} = i) \alpha(j, s, t - 1)$$

$$p(x_1^T, y_1^S) = \sum_{i=1}^N p(q_T = i, \tau_T = S, x_1^T, y_1^S) = \sum_{i=1}^N \alpha(i, S, T)$$

$$P(q_t=i, \tau_t=s | \tau_{t-1}=s-1, x_1^T, y_1^S) = \frac{\alpha^1(i, s, t)\beta(i, s, t)}{P(x_1^T, y_1^S)}$$

$$P(q_t=i, \tau_t=s | \tau_{t-1}=s, x_1^T, y_1^S) = \frac{\alpha^0(i, s, t)\beta(i, s, t)}{P(x_1^T, y_1^S)}$$

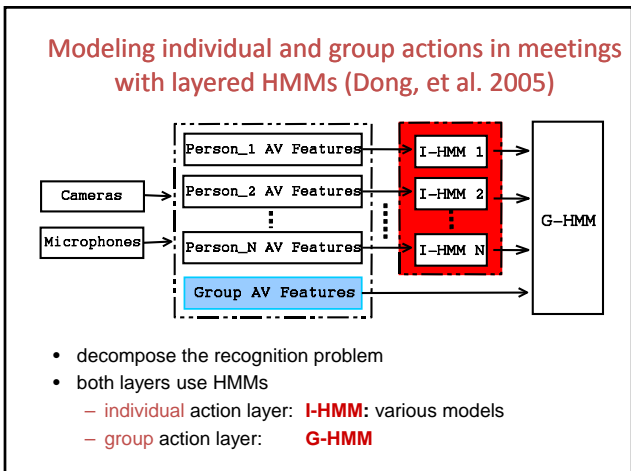
$$P(q_t=i, q_{t-1}=j | x_1^T, y_1^S) = \frac{P(q_t=i | q_{t-1}=j)}{P(x_1^T, y_1^S)} \cdot \left(\sum_{s=1}^S \alpha(j, s-1, t-1) p(x_t, y_t | q_t=i) \epsilon(i, t) \beta(i, s, t) + \sum_{s=0}^S \alpha(j, s, t-1) p(x_t | q_t=i) (1 - \epsilon(i, t)) \beta(i, s, t) \right)$$



Experiment setup

- 59 meetings (30/29 train/test)
- four-people, five-minute
- scripts
 - schedule of actions
 - natural behavior
- features: 5 f/s

mmm.idiap.ch



multi-modal turn-taking

- describes the **group discussion state**

A = { 'discussion',
 'monologue' (x4),
 'white-board',
 'presentation',
 'note-taking',
 'monologue + note-taking' (x4),
 'white-board + note-taking',
 'presentation + note-taking' }

- individual actions

I = { 'speaking',
 'writing',
 'idle' }

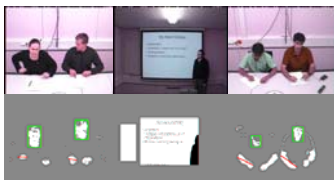
- actions are **multi-modal** in nature

example



Person 1	S	S	W	W
Person 2	W	S	W	W
Person 3	W	S	S	W
Person 4	S	W	S	
Presentation		Used		
Whiteboard			Used	
Group Action	Monologue1 + Note-taking	Discussion	Presentation + Note-taking	Whiteboard + Note-taking

multimodal feature extraction: video



- head + hands blobs**
 - skin colour models (GMM)
 - head position
 - hands position + features (eccentricity, size, orientation)
 - head + hands blob motion
- moving blobs** from background subtraction

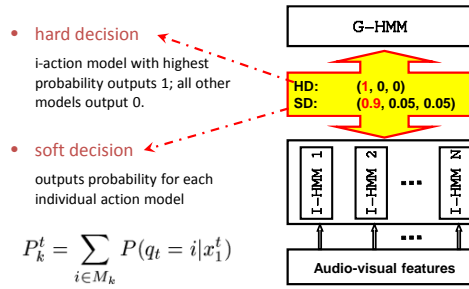
recognition with two-layer HMM

- each layer → trained independently
- compared with single-layer HMM
 - smaller observation spaces
 - I-HMM → trained with **much more data**
 - G-HMM → **less sensitive** to feature variations
 - combinations can be explored

models for I-HMM

- **early integration**
 - all observations concatenated
 - correlation between streams
 - frame-synchronous streams
- **multi-stream (Dupont, TMM 2000)**
 - HMM per stream (a or v), trained independently
 - decoding: weighted likelihoods combined at each frame
 - little inter-stream asynchrony
 - multi-band and a-v ASR
- **asynchronous (Bengio, NIPS 2002)**
 - a and v streams with single state sequence
 - states emit on one or both streams, given a **sync variable**
 - inter-stream asynchrony

linking the two layers



experiments: data + setup

- 59 meetings (30/29 train/test)
- four-people, five-minute
- scripts
 - schedule of actions
 - natural behavior
- features: 5 f/s



mmm.idiap.ch

results: individual actions

Table 2: Results of individual action recognition 43000 frames

Method	Features	FER (%)	STD
Early Int.	Visual	34.17	3.64
	Audio	23.48	2.70
	Audio-Visual	9.98	2.65
MS-HMM	Audio-Visual	8.58	1.76
A-HMM	Audio-Visual	7.42	1.13

- visual-only → audio-only → audio-visual
- asynchronous effects between modalities
- accuracy: speaking: 96.6 %, writing: 90.8%, idle: 81.5%

results: group actions

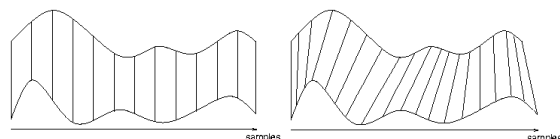
- multi-modality outperforms single modalities
- two-layer HMM outperforms single-layer HMM for a-only, v-only and a-v
- best model: A-HMM
- soft decision slightly better than hard decision

Table 3: Results of group action recognition

Method		AER (%)	STD	
Single-layer	Visual	48.20	3.78	
	Audio	36.70	4.12	
	Audio-visual	23.74	2.97	
Two-layer	Visual	42.45	2.85	
	Audio	32.37	2.10	
	Early Int.	hard	17.98	2.75
		soft	16.55	1.40
	MS-HMM	hard	17.27	2.01
		soft	15.83	1.61
	A-HMM	hard	17.85	2.87
		soft	15.11	1.48

8% improvement

How to compare two sequences



Kovar & Gleicher (2004, Siggraph)

- A direct numerical comparison can be used to determine similarity between two sequences.
- With existing metrics, a **large** distance may reflect either that motions are **unrelated** or that they are different **variations** of the same actions.
- There is no easy to distinguish these two cases.

Variation

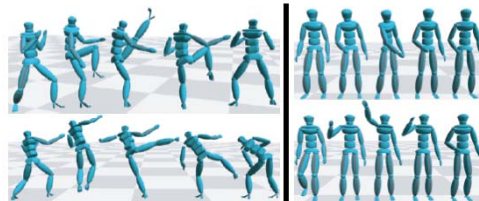
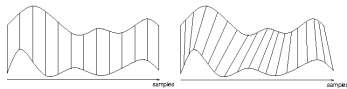


Figure 1: Logically similar motions may be numerically dissimilar. **Left:** A standing front kick vs. a leaping side kick. Note the differences in the arms, torso posture, and kick trajectory. **Right:** While these two reaching motions have somewhat similar skeletal postures, the changes in posture are in completely opposite directions.

Criteria for numerical similarity

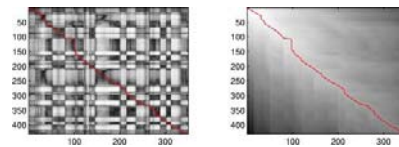
- First, corresponding frames should have **similar** skeleton poses.
- Second, **frame correspondences** should be easy to identify. That is, related events in the motions should be clearly recognizable.
- How to find frame correspondences?

Time alignment



Dynamic time warping

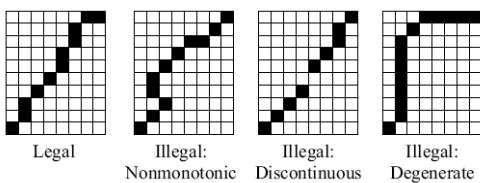
- We start with computing for every pair of frames, forming a **grid** of distances.
- The time alignment is a **path** on this grid from one corner to the corner that minimizes the total cost of its cells.



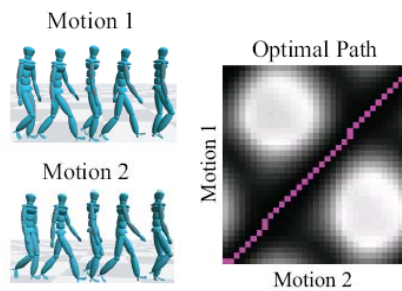
Time alignment

- Continuous, monotonic and non-degenerate

Time Alignment Restrictions



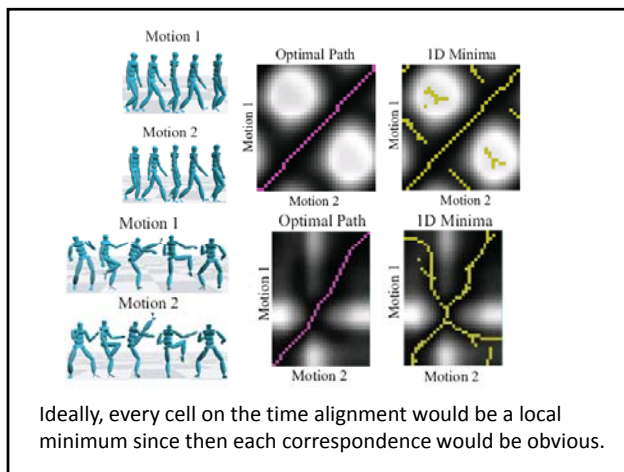
Optimal Path



Local optimality

Second, frame correspondences should be easy to identify. That is, related events in the motions should be clearly recognizable.

- If a cell on the time alignment is a horizontal or vertical **1D local minimum**, then the frame correspondence is strong in the sense that holding one frame fixed and varying the other only yields more dissimilar skeletal poses.



Valid region

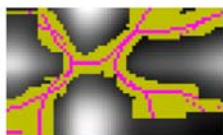


Figure 5: Local minima (magenta) are extended to form the valid region (yellow).

- We compute all 1D local minima and extend each one along the directions in which it is a minimum (horizontal, vertical, or both) until a cell is encountered whose value is larger than a **threshold**.
- We call the resulting region on the grid the valid region. The time alignment is restricted to the valid region.

Similarity Measurement

- With existing metrics, a **large** distance may reflect either that motions are unrelated or that they are different variations of the same actions.
- A **small** distance is a reasonable indicator of similarity.
- We can find close motions and then use them as new queries to find more distant motion.

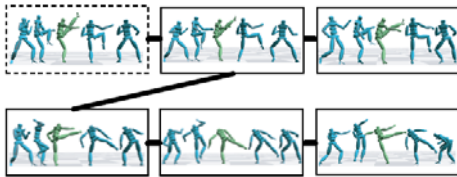


Figure 10: An example match graph. The query is in the dotted box and the other nodes are matches. Edges indicate numerical similarity.

- With existing metrics, a **large** distance may reflect either that motions are unrelated or that they are different variations of the same actions.
- A **small** distance is a reasonable indicator of similarity .
- We can find close motions and then use them as new queries to find more distant motion.