Machine-Learning for SEQUENTIAL data

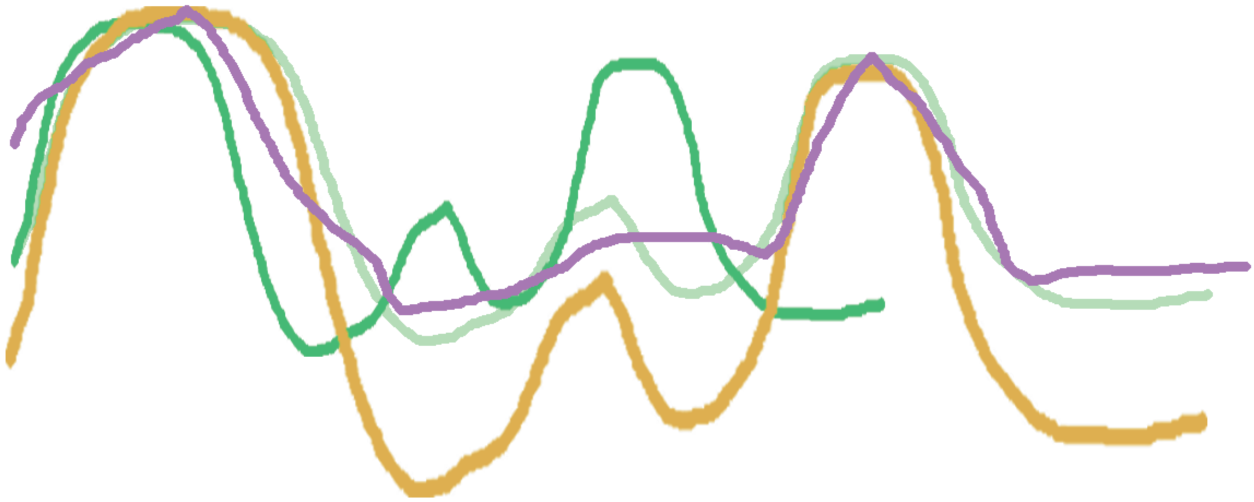
Pr. Fabien Moutarde
Center for Robotics
MINES ParisTech
PSL Université Paris

Fabien.Moutarde@mines-paristech.fr
http://people.mines-paristech.fr/fabien.moutarde

Outline

- Specificities of SEQUENTIAL data
- Alignment of sequences by DTW
- Model sequential data with HMM

Specificities of SEQUENTIAL data



- 2 specific problems:
  - How to compare sequences?
  - Length often VARIABLE!

Handling COMPARISON of sequences

- 2 main types of approaches:
  - Alignment of sequences
    → Dynamic Time Warping (DTW)
  - Model-based method
    (e.g. Hidden Markov Model, HMM)

Handling VARIABLE LENGTH of sequences

- 2 main types of approaches:
  - Time Resampling or Padding
    (but unapplicable for “stream” inline recognition)
  - Model-based methods: streaming successive inputs into a fixed-size model
    - Hidden Markov Model (HMM)
    - Recurrent Neural Network (RNN)

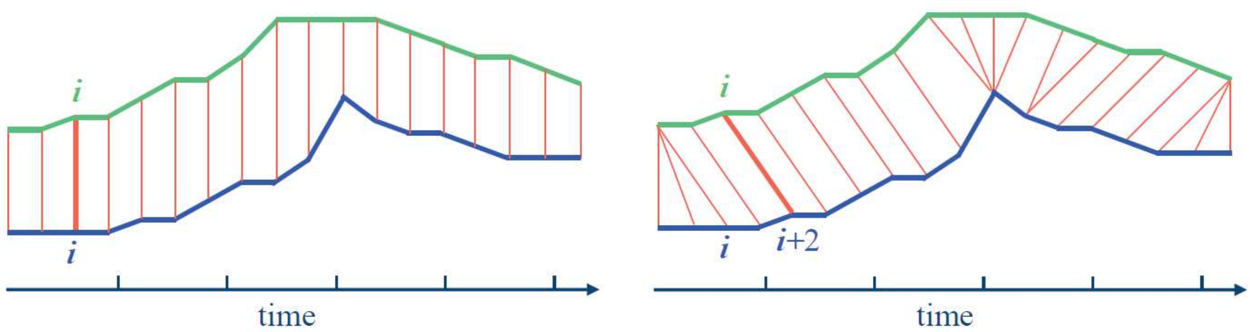
Outline

- Specificities of SEQUENTIAL data
- **Alignment of sequence by DTW**
- Model sequential data with HMM

Dynamic Time Warping

- **Principle of DTW:**
  1. **Align sequences and compute an adapted similarity measure**
  2. **Perform recognition by template-matching with k Nearest Neighbors (using DTW similarity)**

Alignment of sequences

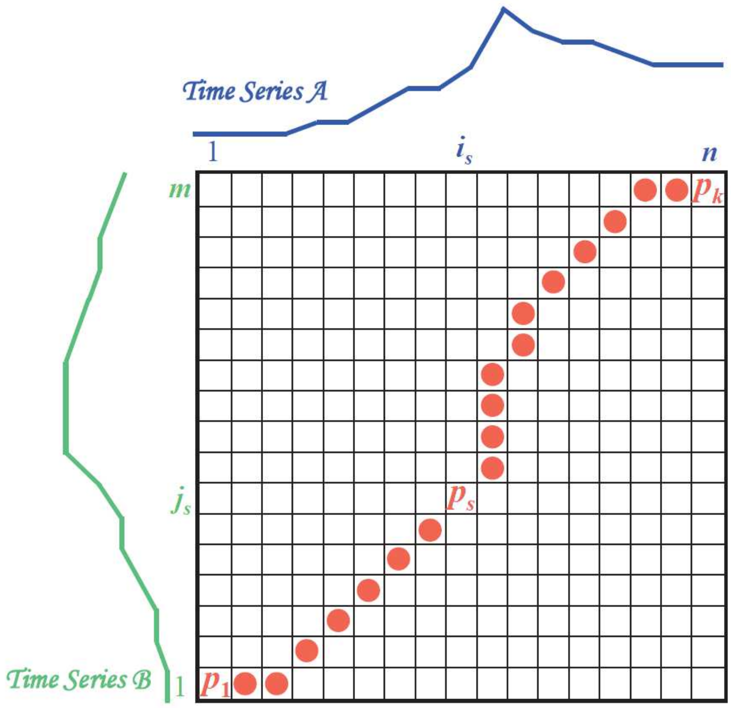


Any distance (Euclidean, Manhattan, …) which aligns the *i*-th point on one time series with the *i*-th point on the other will produce a poor similarity score.

A non-linear (elastic) alignment produces a more intuitive similarity measure, allowing similar shapes to match even if they are out of phase in the time axis.

[Slide from Elena Tsiporkova]

Warping function



To find the *best alignment* between $\mathcal{A}$ and $\mathcal{B}$ one needs to find the path through the grid

$$P = p_1, \ldots, p_s, \ldots, p_k$$

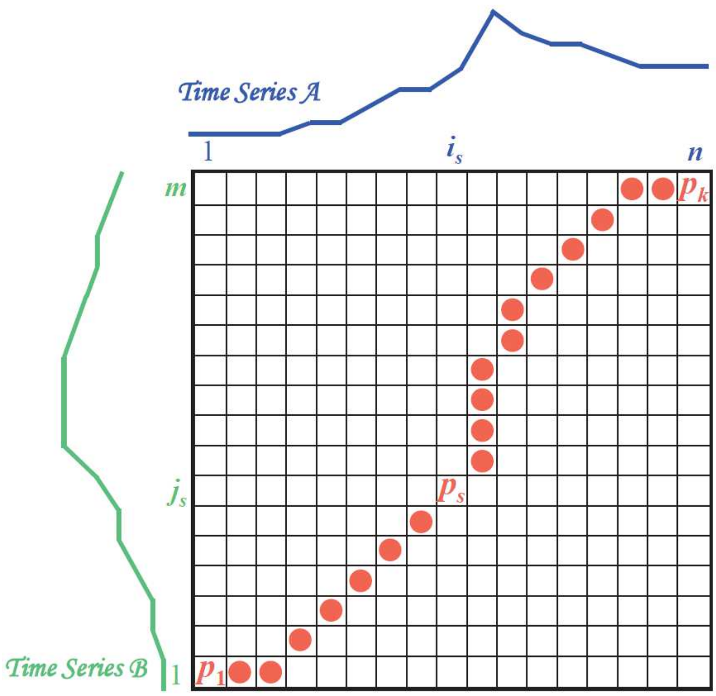
$$p_s = (i_s, j_s)$$

which *minimizes* the total distance between them.

P is called a *warping function*.

[Slide from Elena Tsiporkova]

Time-Normalized Distance Measure



Time-normalized distance between $\mathcal{A}$ and $\mathcal{B}$:

$$D(\mathcal{A}, \mathcal{B}) = \left[\frac{\sum_{s=1}^{k} d(p_s) \cdot w_s}{\sum_{s=1}^{k} w_s} \right]$$

$d(p_s)$: distance between i_s and j_s

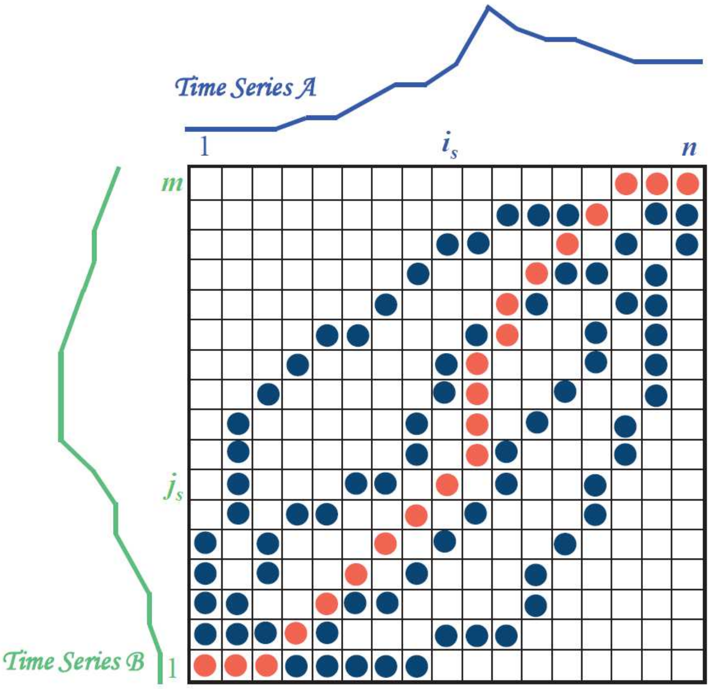
$w_s > 0$: weighting coefficient.

Best alignment path between $\mathcal{A}$ and $\mathcal{B}$:

$$P_0 = \arg\min_P (D(\mathcal{A}, \mathcal{B})).$$

[Slide from Elena Tsiporkova]

Optimizing DTW algorithm



The number of possible warping paths through the grid is exponentially explosive!

reduction of the search space

Restrictions on the warping function:

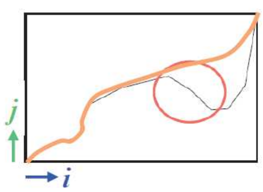
- monotonicity
- continuity
- boundary conditions
- warping window
- slope constraint.

[Slide from Elena Tsiporkova]

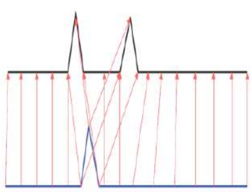
Usual restrictions on Warping function

Monotonicity: $i_{s-1} \le i_s$ and $j_{s-1} \le j_s$.

The alignment path does not go back in "*time*" index.

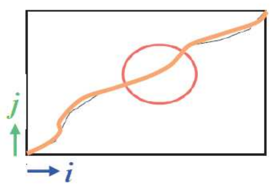


Guarantees that features are not repeated in the alignment.

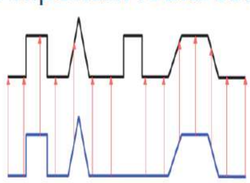


Continuity: $i_s - i_{s-1} \le 1$ and $j_s - j_{s-1} \le 1$.

The alignment path does not jump in "*time*" index.



Guarantees that the alignment does not omit important features.

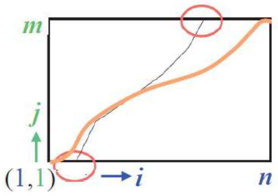


[Slide from Elena Tsiporkova]

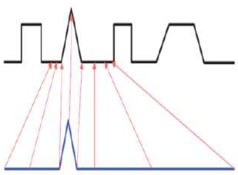
Other restrictions on Warping function

Boundary Conditions: $i_1 = 1$, $i_k = n$ and $j_1 = 1$, $j_k = m$.

The alignment path starts at the bottom left and ends at the top right.

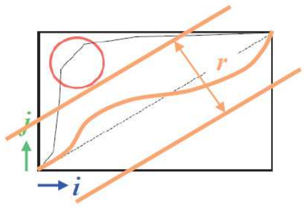


Guarantees that the alignment does not consider partially one of the sequences.

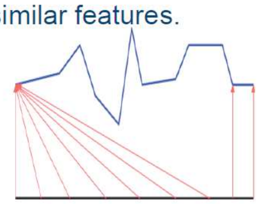


Warping Window: $|i_s - j_s| \leq r$, where $r > 0$ is the window length.

A good alignment path is unlikely to wander too far from the diagonal.



Guarantees that the alignment does not try to skip different features and gets stuck at similar features.

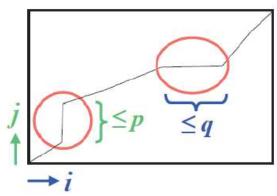


[Slide from Elena Tsiporkova]

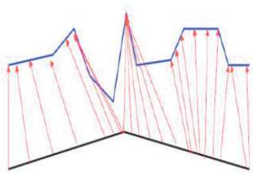
Slope constraints on Warping function

Slope Constraint: $(j_{s_p} - j_{s_0}) / (i_{s_p} - i_{s_0}) \leq p$ and $(i_{s_q} - i_{s_0}) / (j_{s_q} - j_{s_0}) \leq q$, where $q \geq 0$ is the number of steps in the x-direction and $p \geq 0$ is the number of steps in the y-direction. After q steps in x one must step in y and vice versa: $S = p / q \in [0, \infty]$.

The alignment path should not be too steep or too shallow.



Prevents that very short parts of the sequences are matched to very long ones.



[Slide from Elena Tsiporkova]

Choice of weighting coefficients

Time-normalized distance between $\mathcal{A}$ and $\mathcal{B}$:

$$D(\mathcal{A},\mathcal{B}) = \min_{P}\left[\frac{\sum_{s=1}^{k} d(p_s)\cdot w_s}{\sum_{s=1}^{k} w_s}\right].$$

complicates optimisation

Seeking a weighting coefficient function which guarantees that:

$$C = \sum_{s=1}^{k} w_s$$

is independent of the warping function. Thus

$$D(\mathcal{A},\mathcal{B}) = \frac{1}{C}\min_{P}\left[\sum_{s=1}^{k} d(p_s)\cdot w_s\right]$$

can be solved by use of dynamic programming.

Weighting Coefficient Definitions

- Symmetric form

  $w_s = (i_s - i_{s-1}) + (j_s - j_{s-1}),$

  then $C = n + m$.
- Asymmetric form

  $w_s = (i_s - i_{s-1}),$

  then $C = n$.

  Or equivalently,

  $w_s = (j_s - j_{s-1}),$

  then $C = m$.

[Slide from Elena Tsiporkova]

Pros and Cons of DTW

- **Pros**
  - **Allows speed-insensitive and flexible alignment**
- **Cons**
  - **Computationally expansive (especially for multi-variate time-series)**
  - **Vanilla version is OFFLINE (i.e. after gesture) BUT "STREAM DTW" version solves this issue**

Outline

- Specificities of SEQUENTIAL data
- Alignment of sequence by DTW
- **Model sequential data with HMM**

What is a HMM?

HMM = Hidden Markov Model

<u>Stochastic (probabilistic)</u> model
obtained by statistical analysis of
sequences of many examples of same class

Markovian??

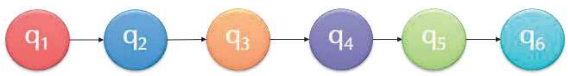
« The future is independent of the past, given the present »

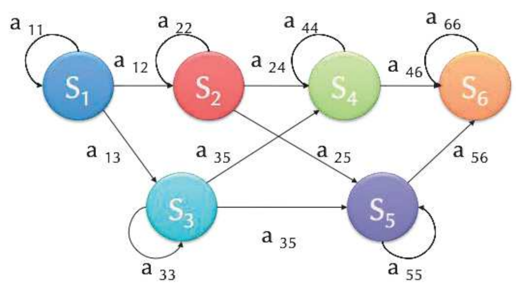
Andreï Andreïevitch Markov
Андрей Андреевич Марков
2 June 1856 - 20 July 1921

Markov chains

Model definition

- Set of N States, $\{S_1, S_2, \ldots S_N\}$
- Sequence of states $Q = \{q_1, q_2, \ldots\}$
- Initial probabilities $\pi = \{\pi_1, \pi_2, \ldots \pi_N\}$
  - $\pi_i = P(q_1 = S_i)$





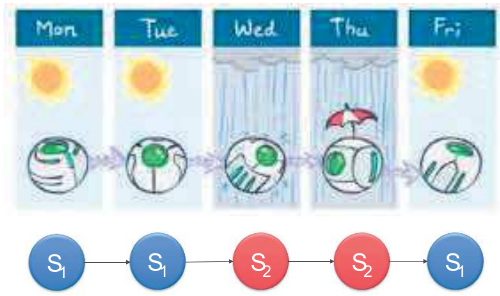
- Transition matrix A NxN
  - $a_{ij} = P(q_{t+1} = S_j \mid q_t = S_i)$

Example of Markov chain

Example in weather forecasting

Weather model:

- 3 states {sunny, rainy, cloudy}

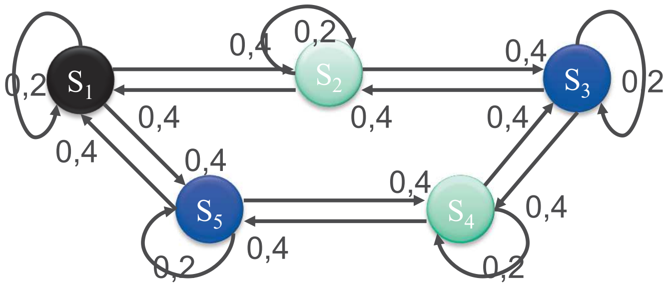


Problem:

- Forecast weather state, based on the current weather state

Markov chain in action

Let's pick arbitrarily some numbers for $P(q_i|q_{i-1})$ and draw a probabilistic finite state automaton



<u>Question</u>

Given that now the state is S_2, what's the probability that next state will be S_3 AND the state after will be S_4?

Answer to Question

This translates into:

$$P(q_2 = S_3, q_3 = S_4 | q_1 = S_2) = P(q_3 = S_4 | q_2 = S_3, q_1 = S_2)* P(q_2 = S_3 | q_1 = S_2)$$
$$= P(q_3 = S_4 | q_2 = S_3) * P(q_2 = S_3 | q_1 = S_2)$$
$$= 0,4 * 0,4$$
$$= 0,16$$

You can also think this as moving through the automaton, multiplying the probabilities

Hidden Markov Model

<u>λ=(A, B, π): Hidden Markov Model</u>

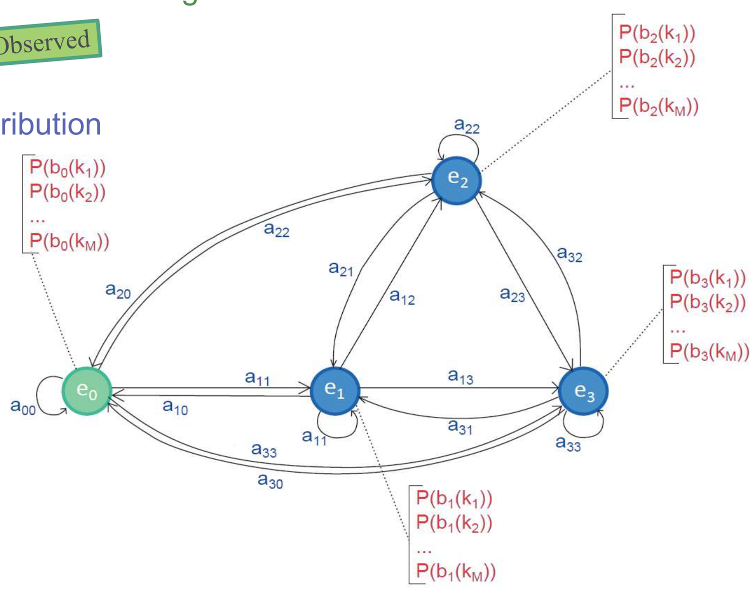
- A={a_{ij}}: Transition probabilities between HIDDEN states
  - $a_{ij}=P(q_{t+1}=S_j \mid q_t=S_i)$

Hidden

- B={$b_i(x)$}: Emission probabilities for observation given hidden state
  - $b_i(O_t)=P(O_t=x \mid q_t=S_i)$

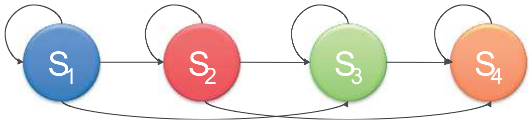
Observed

- π={π_i}: Initial state probabilistic distribution
  - $\pi_i=P(q_1=S_i)$

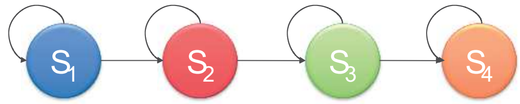


HMM topologies

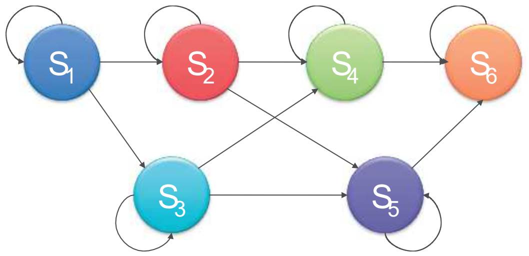
Left to right (A)



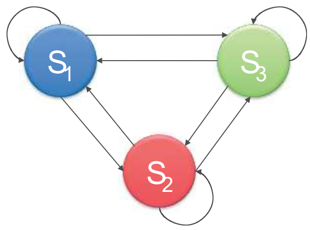
Left to right (B)



Left to right (C)



Ergodic



HMM algorithms

- **Evaluation**
  – O, λ → P(O|λ)

- Uncover the hidden part
  – O, λ → Q that P(Q|O, λ) is maximum

- **Learning**
  – {O} → λ such that P(O|λ) is maximum

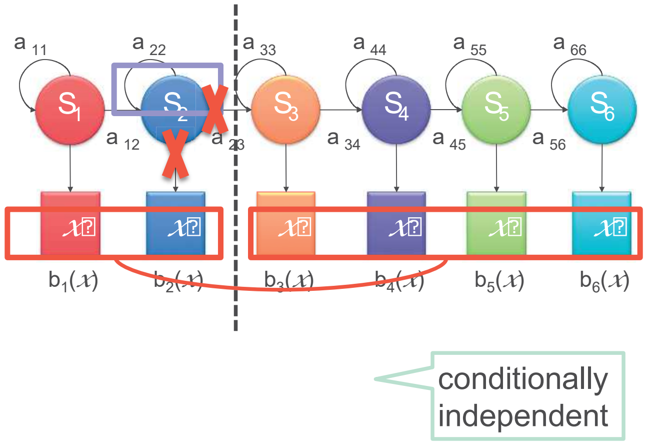
HMM: Evaluation

O, λ → P(O|λ) ?

- Solved by the **Forward** algorithm

Applications

- Find some likely samples
- Evaluation of a sequence of observations
- Change detection



Initialisation

$$\alpha_1(i) = \pi_i * b_i(o_1)$$

Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] * b_j(o_{t+1})$$

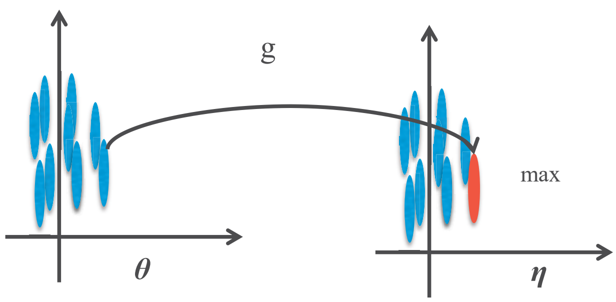
Termination

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

HMM learning

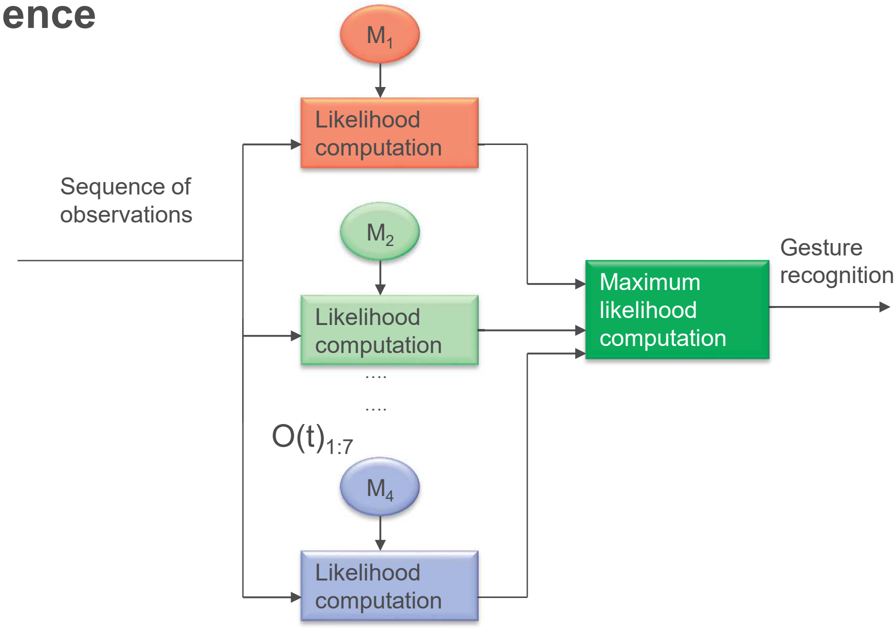
- {O} → λ such that P(O|λ) is maximum
- No analytic solution
- Solved by **Baum-Welch algorithm** (which is ***particular case of Expectation Maximization [EM] algo***) when some data is missing (the states)

- Applications
  - Unsupervised Learning (single HMM)
  - Supervised Learning (multiple HMM)

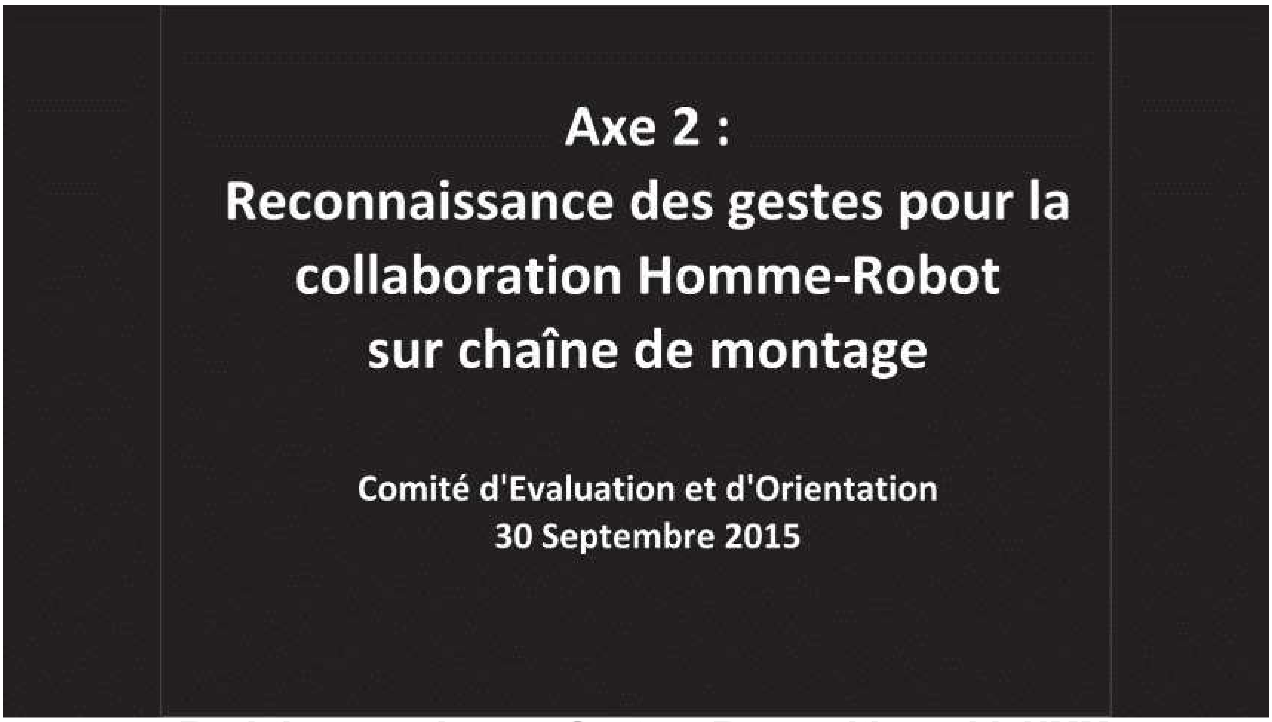


Recognition with HMM

- Typically, learn ONE HMM per class, and then sequentially feed data in <u>all</u> HMM, so each one updates likelihood of sequence



Application example



Real-time *continuous* Gesture Recognition *<u>with HMM</u>* for Human-Robot Collaboration

Pros and Cons of HMM

- **Pros**
  - **Natural handling of variable length**

- **Cons**
  - **Many hyper-parameters (ARCHITECTURE and # of hidden states)**

Synthesis

- **Sequential data raise specific problems:**
  - **what similarity measure should be used? (cf alignment problem)**
  - **Often *variable* length input**

- **Two main shallow ML approaches adapted to this specificities:**
  - **Dynamic Time Warping (DTW)**
  - **Hidden Markov Model (HMM)**

Deep-Learning → Deep RECURRENT Neural Nets (LSTM, GRU) or 1D ConvNet over time

Any QUESTIONS ?