Università di Verona

A.Y. 2021-22

Machine Learning & Artificial Intelligence

Hidden Markov Models

Vittorio Murino

Summary

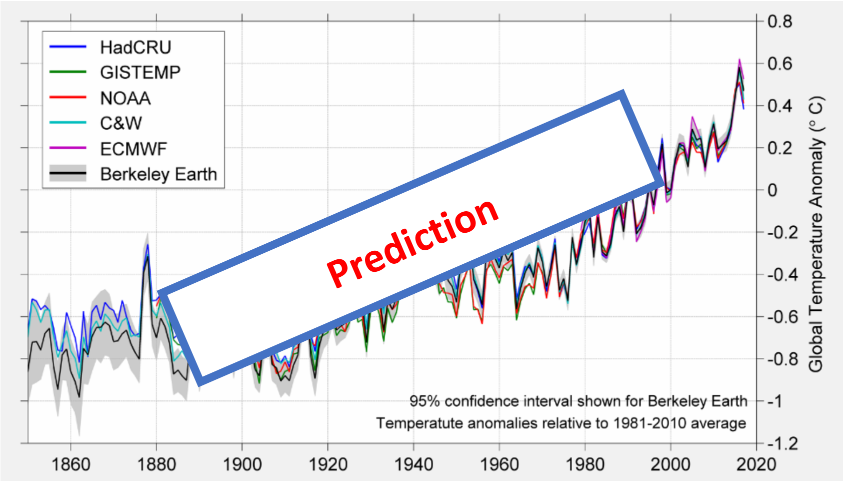
1. Markov processes and models;

2. Hidden Markov Model (HMM);

3. Research and applications on HMM.

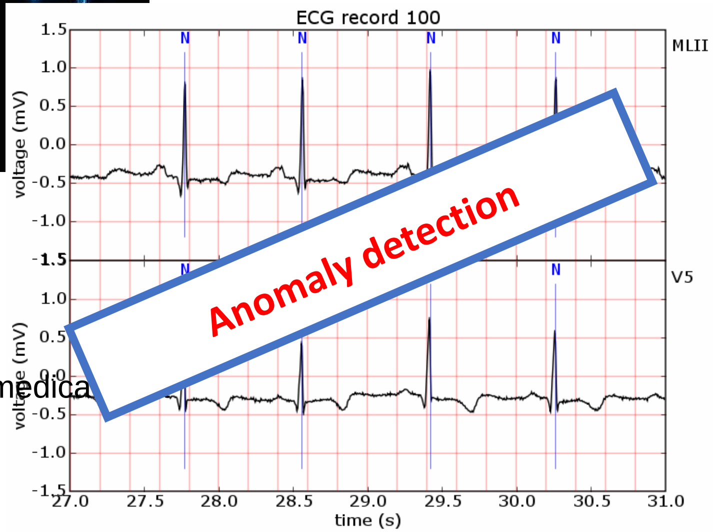
Time series analysis

Financial series



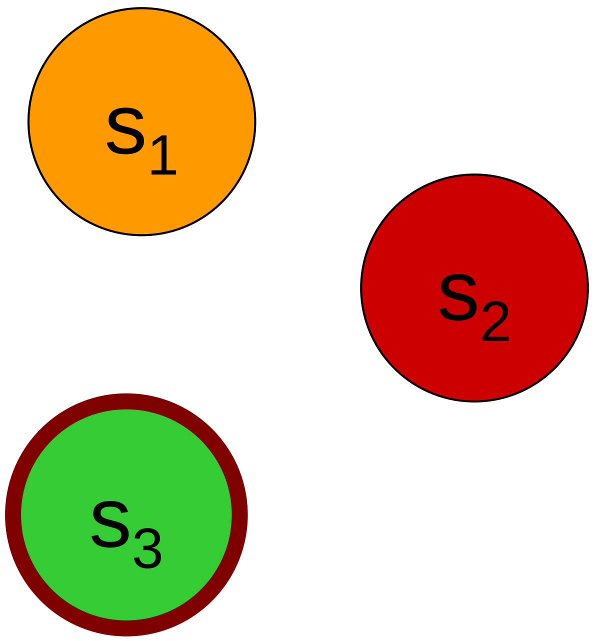
Audio signals

Gestures



Biomedica

Markov process (order 1)

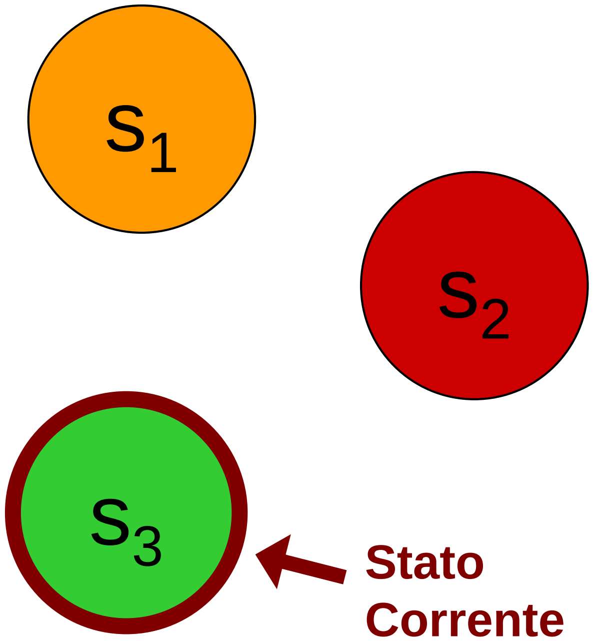


N=3
t=1

- Has N states , $s_1, s_2, \ldots, s_N$
- It is characterized by discrete steps, t=1,t=2,…
- The probability of starting from a certain state is dictated by the distribution:
- $=\{\pi_i\}$: $\pi_i = P(q_1 = s_i)$ with

$$1 \le i \le N, \;\; \pi_i \ge 0 \quad \text{and} \quad \sum_{i=1}^{N} \pi_i = 1$$

Markov process

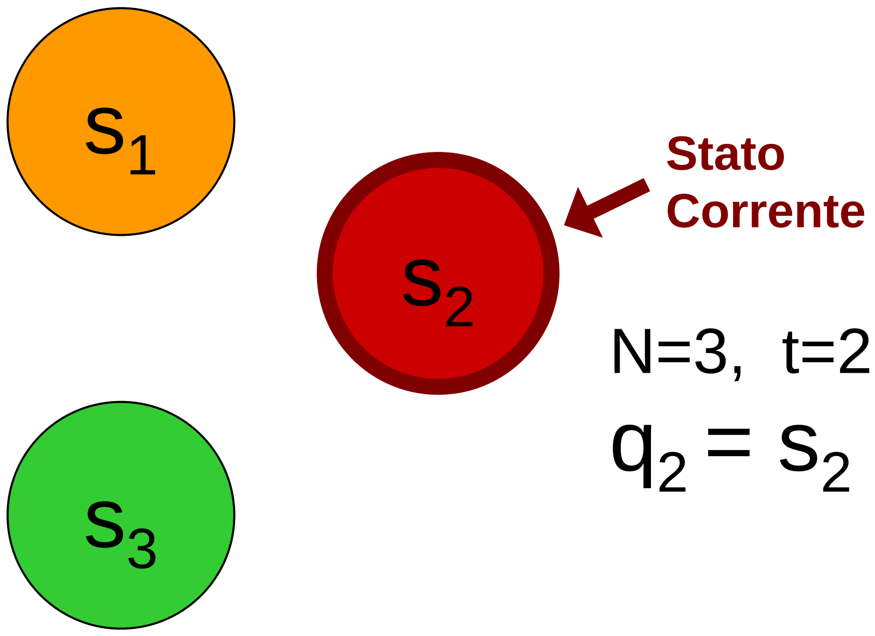


N=3, t=1

$q_1 = s_3$

- At the t-th instant the process is exactly in one of the available states, indicated by the variable q_t
- Note: $q_t \in \{s_1, s_2, \dots, s_N\}$
- At each iteration, the next state is chosen with a certain probability
-

Markov process



$P(q_{t+1}=s_1|q_t=s_1) = 0$
$P(q_{t+1}=s_2|q_t=s_1) = 0$
$P(q_{t+1}=s_3|q_t=s_1) = 1$

$P(q_{t+1}=s_1|q_t=s_2) = 1/2$
$P(q_{t+1}=s_2|q_t=s_2) = 1/2$
$P(q_{t+1}=s_3|q_t=s_2) = 0$

$P(q_{t+1}=s_1|q_t=s_3) = 1/3$
$P(q_{t+1}=s_2|q_t=s_3) = 2/3$
$P(q_{t+1}=s_3|q_t=s_3) = 0$

Stato Corrente

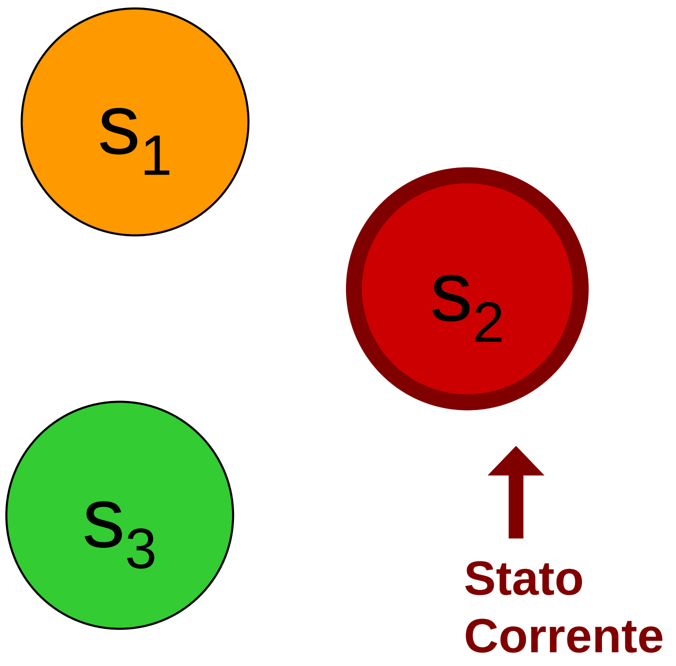
- This probability is only determined by the previous state (*first-order markovianity*):
- $P(q_{t+1}= s_j|q_t= s_i,q_{t-1}=s_k,\ldots,q_1=s_l) = P(q_{t+1}= s_j|q_t= s_i)$

Markov hypothesis

The probability of moving to a given state depends only on the current state.

$$P(q_t = S^* | q_{t-1}, \dots, q_1) = P(q_t = S^* | q_{t-1})$$

Markov process



N=3, t=2

$q_2 = s_2$

- Defining:

$a_{i,j} = P(q_{t+1} = s_j \mid q_t = s_i)$

I get the matrix NxN

A *transition between states, invariant over time:*

$$A = \begin{array}{|c|c|c|} \hline a_{1,1} & a_{1,2} & a_{1,3} \\ \hline a_{2,1} & a_{2,2} & a_{2,3} \\ \hline a_{3,1} & a_{3,1} & a_{3,3} \\ \hline \end{array}$$

Markov Models

- A set of N **states** $\mathcal{S} = \{S_1, S_2, \dots, S_N\}$

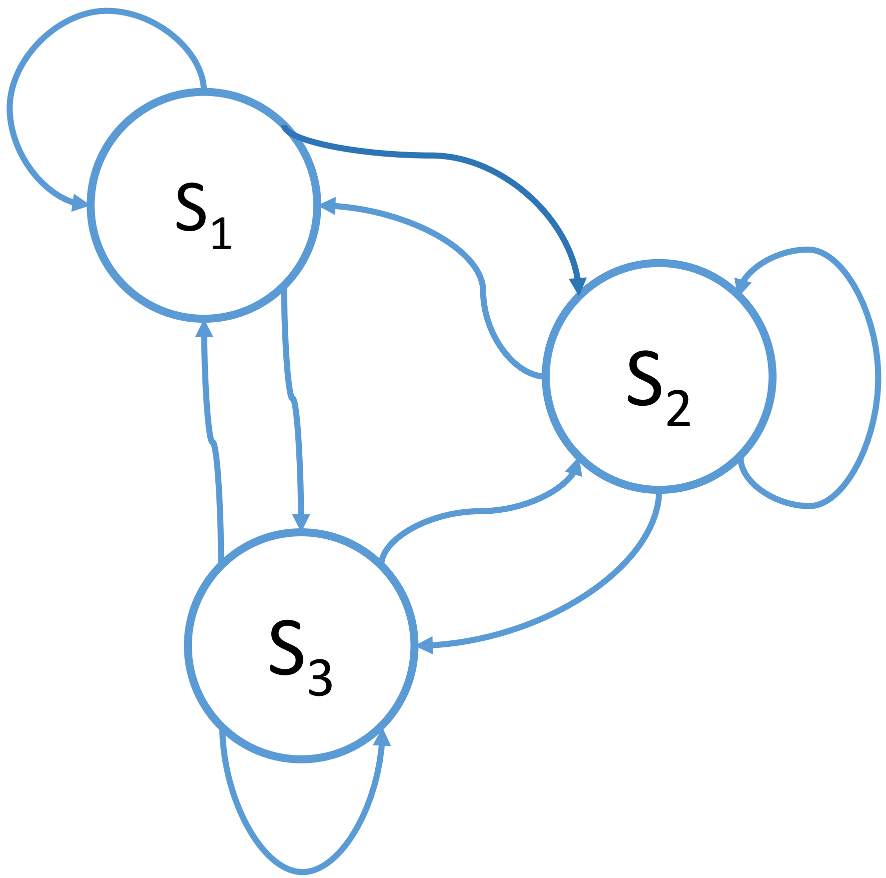
- A sequence of states $\mathcal{Q} = \{q_1, q_2, \dots, q_T\}$

- A **transition probability matrix**

$$A = \{a_{ij} = P(q_t = S_j | q_{t-1} = S_i)\}$$

- An **initial probability distribution** over states

$$\Pi = \{\pi_i = P(q_1 = S_i)\}$$



Markov Models

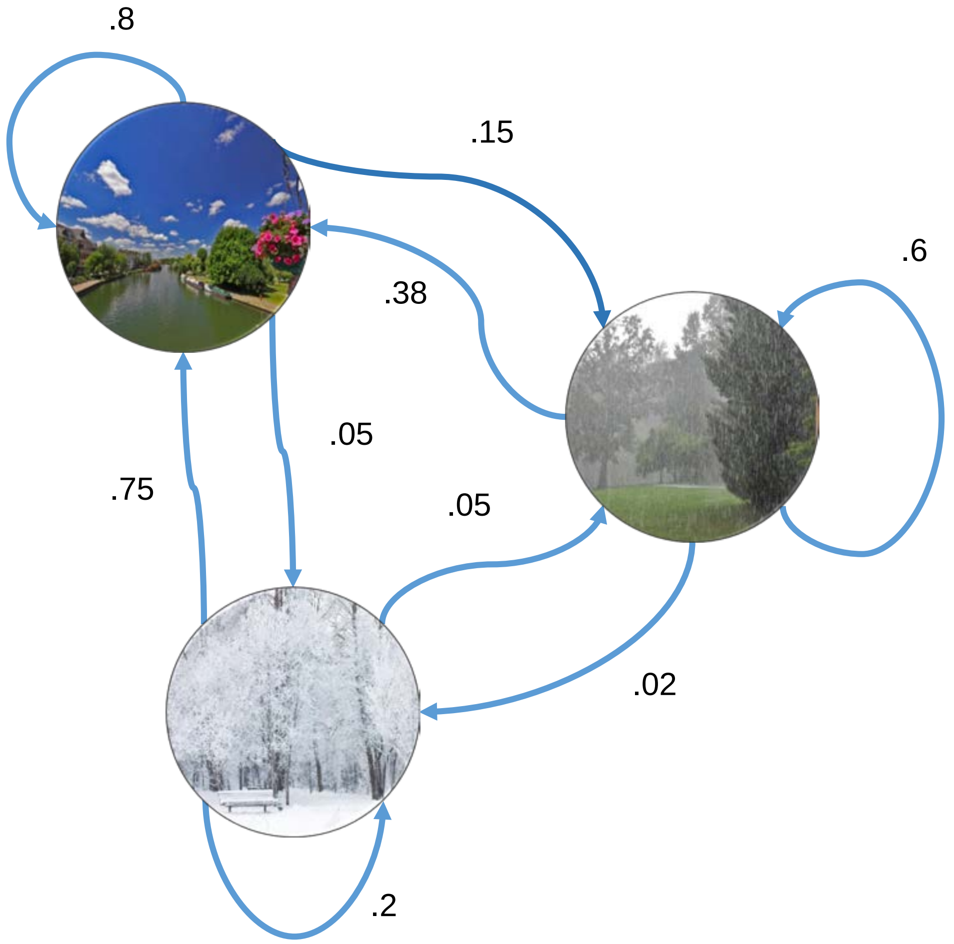
- **States:** $\{sunny, rainy, snowy\}$

- **Transition probability matrix:**

$$A = \begin{bmatrix} .8 & .15 & .05 \\ .38 & .6 & .02 \\ .75 & .05 & .2 \end{bmatrix}$$

- **Initial probability distribution**:

$$\Pi = [.7 \quad .25 \quad .05]$$



Exercise

- Compute the probability for the sequence:

$$P = P(Su) * P(R|Su) * P(R|R) * P(R|R) * P(Sn|R) * P(Sn|Sn)$$

$$P = 0.0001512$$

Features of Markovian processes

- They are (discrete) processes characterized by:
  - Markovianity of the first order
  - stationary
  - have an initial distribution

- Knowing the above features, one can exhibit a **(probabilistic) model of Markov (MM)** as

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

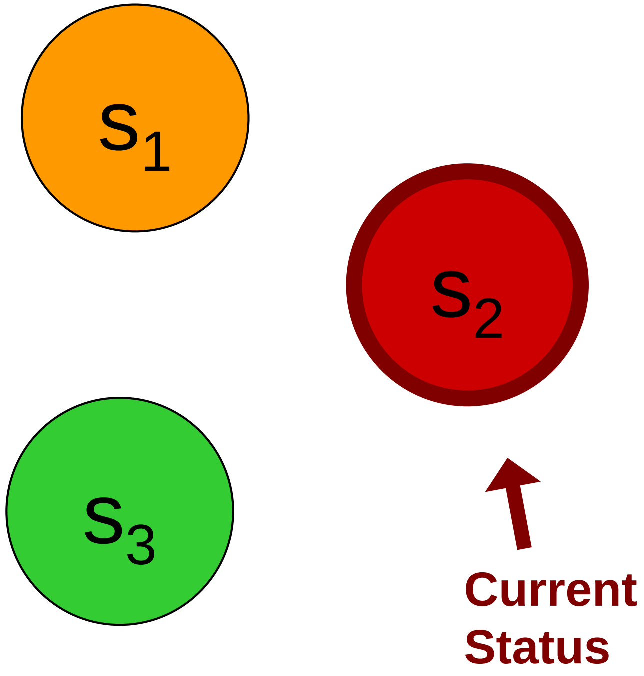
What is a stochastic or probabilistic model for?

- Models and reproduces **stochastic processes**
- Describes by probability the **causes that lead from one state of the system to another**
- In other words, the more likely you are to move from state A to state B, the more likely it is that **A will cause B**

What can be done on a probabilistic model?

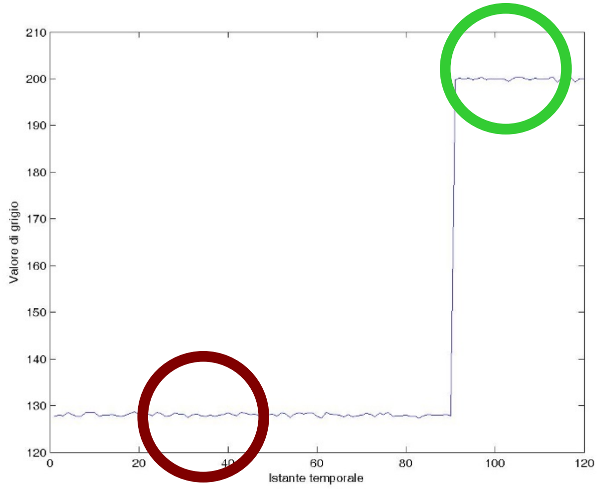
- **Training**
  - The constituent elements of the model are estimated
- **Inferences of various kinds (I question the model):**
  - Probability of a sequence of states, given the model
  - Invariant properties etc.

What is a Markov model for?



- Model Markovian *stochastic behaviors (of order N)* of a system in which the states are:
  - **Explicit** (I can give them a name)
  - **Observable** (I have observations that uniquely identify the state)

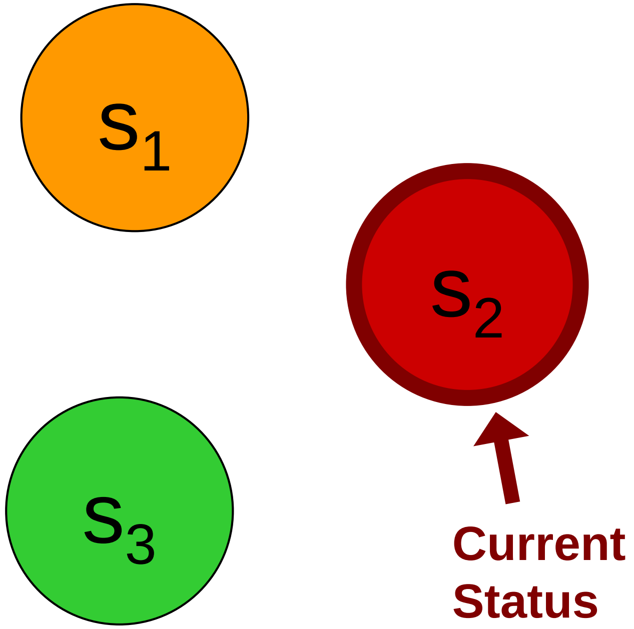
Example: Traffic light

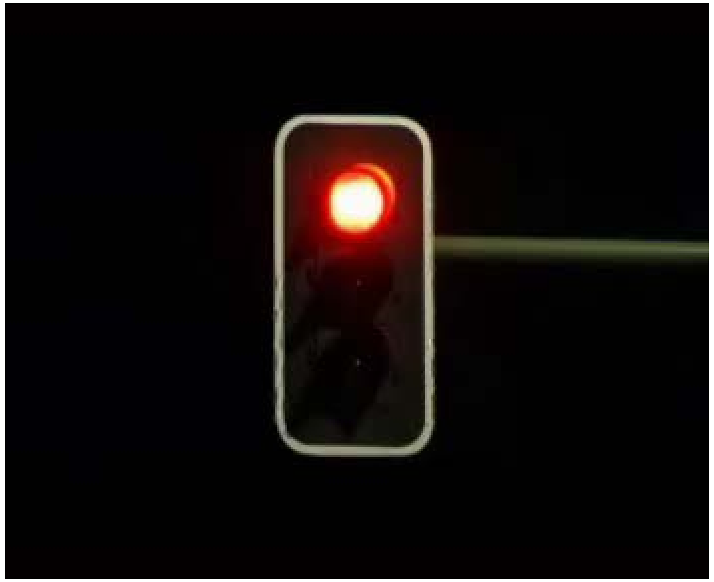


t=1 t=2 ...

- It is a system whose states are:
  - **Explicit** (the different lamps lit)
  - **Observable** (the colors of the lamps I observe)

Traffic light – trained model





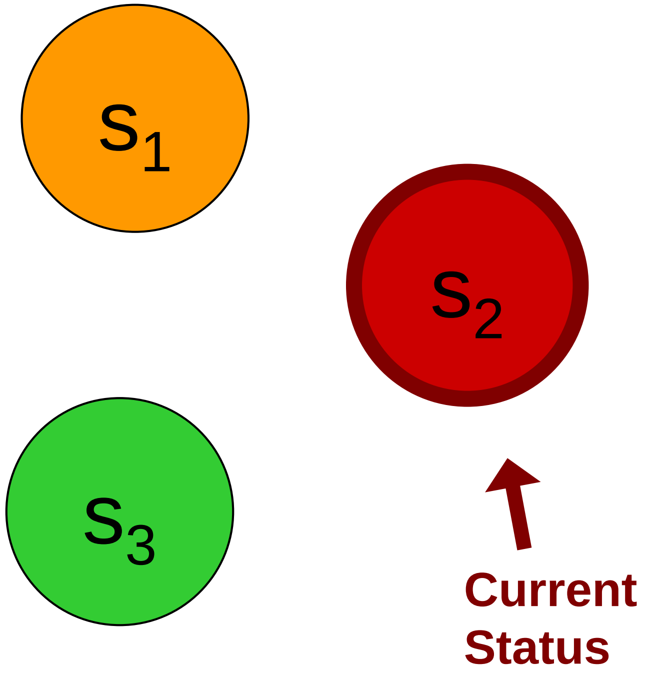
$\pi =$

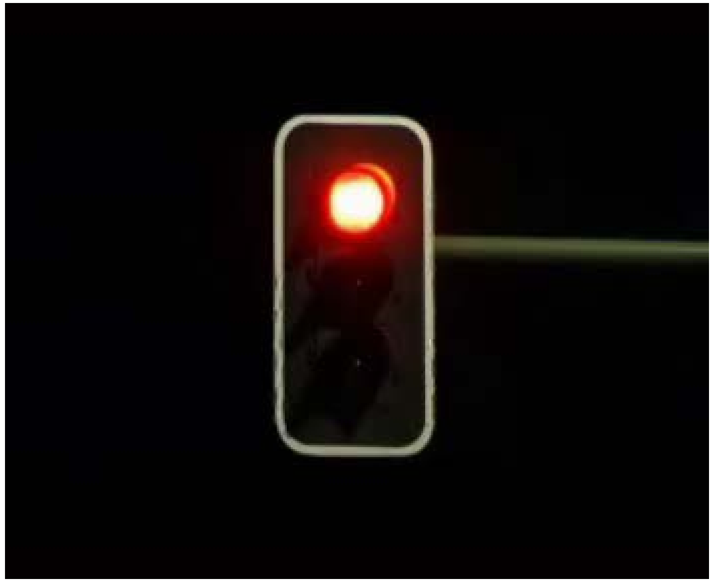
$\pi_1 = 0.33$	$\pi_2 = 0.33$	$\pi_3 = 0.33$

$\boldsymbol{A} =$

$a_{11} = 0.1$	$a_{12} = 0.9$	$a_{13} = 0$
$a_{21} = 0.01$	$a_{22} = 0$	$a_{23} = 0.99$
$a_{31} = 1$	$a_{32} = 0$	$a_{33} = 0$

Traffic light – inference





$$O_2 = \langle q_2 = s_3,\ q_1 = s_2 \rangle$$

Inference:

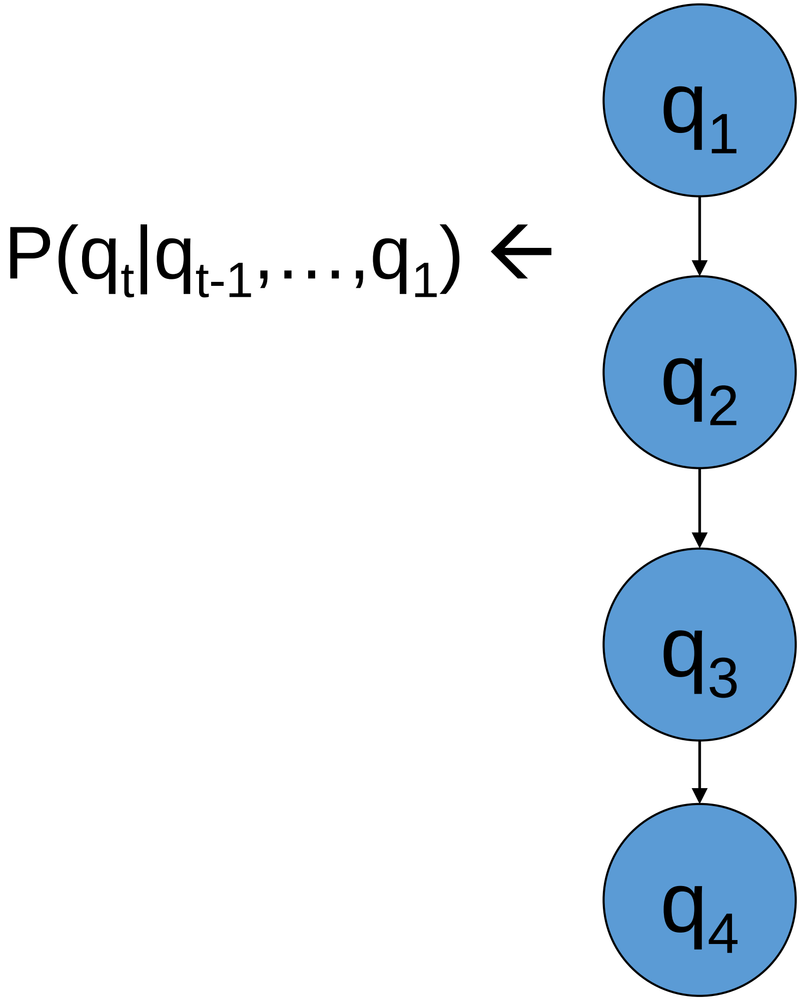
$$P(O|\ \lambda) = P(O) =$$

$$= P(q_2 = s_3, q_1 = s_2) = P(q_2, q_1)$$

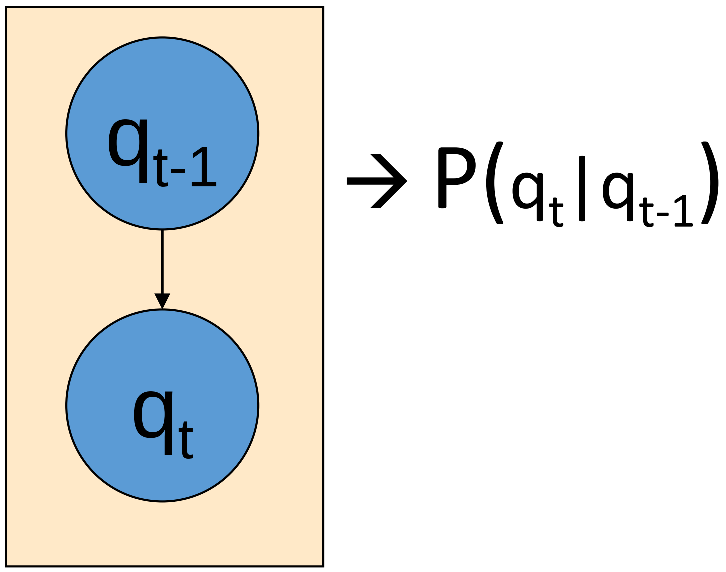
Important inference

$$
\begin{aligned}
P(q_t,q_{t-1},\dots,q_1) &= P(q_t|q_{t-1},\dots,q_1)\, P(q_{t-1},\dots,q_1) \\
&= P(q_t|q_{t-1})\, P(q_{t-1},q_{t-2},\dots,q_1) \\
&= P(q_t|q_{t-1})\, P(q_{t-1}|q_{t-2})\, P(q_{t-2},\dots,q_1) \\
&\dots \\
&= P(q_t|q_{t-1})\, P(q_{t-1}|q_{t-2})\dots P(q_2|q_1)P(q_1)
\end{aligned}
$$

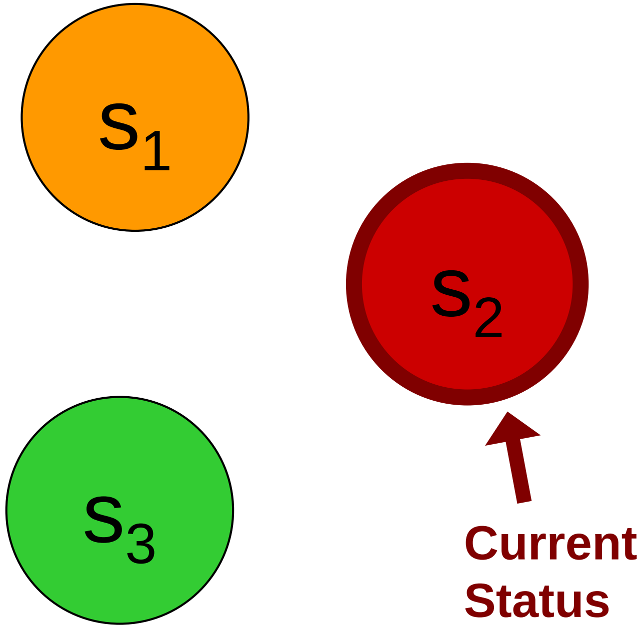
Graphic representation

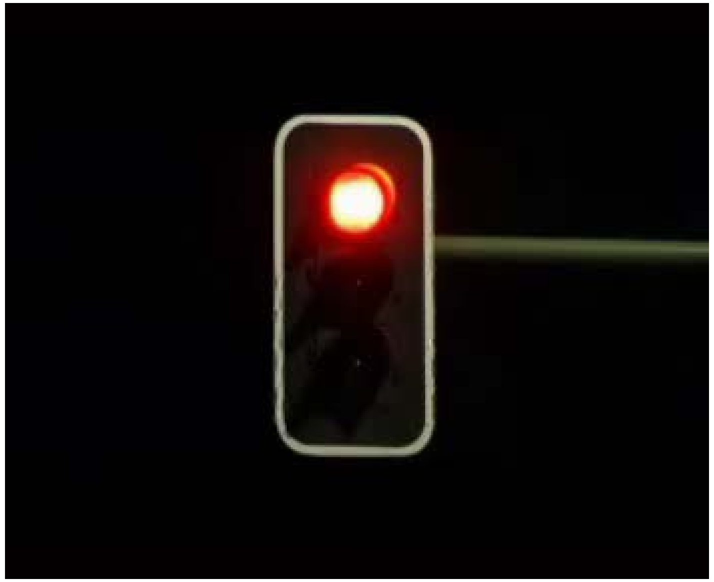


The graphic structure of such a joint probability is written in this form, where



Traffic light – inferences, response





$$P(O|\lambda) = P(O)$$

$$= P(q_2 = s_3, q_1 = s_2)$$

$$= P(q_2 = s_3 | q_1 = s_2)\ P(q_1 = s_2)$$

$$= \quad 0.99 \quad * \quad 0.33 \quad = 0.326$$

Second important inference

- Probability calculation $P(q_T = s_j)$
- <u>STEP 1</u>:

  I evaluate how to calculate P(Q) for each path of states

  $\mathbf{Q}=\{q_1,q_2,\ldots,q_T=s_j\}$, that is

  $$P(q_T, q_{T-1}, \ldots, q_1)$$

- <u>STEP 2</u>:

  I use this method to calculate $P(q_T = s_j)$, that is:

  o $$P(q_T = s_j) = \sum_{\mathbf{Q} \in \text{paths of length T ending in } s_j} P(\mathbf{Q})$$

  Onerous calculation: EXPONENTIAL in T ($O(N^T)$)!

Second important inference (2)

- Idea: for each state s_j we call
  $p_T(j)$= prob. to be in the state s_j at the time T → $P(q_T= s_j)$;
  - It can be defined by induction:

$$\forall i \quad p_1(i) = \begin{cases} 1 & \text{if } \mathrm{s_i} \text{ is the current state} \\ 0 & \text{otherwise} \end{cases}$$

$$\forall j \quad p_{t+1}(j) = P(q_{t+1} = s_j) = \sum_{\mathrm{i=1}}^{\mathrm{N}} P(q_{t+1} = s_j, q_t = s_i)$$

Second important inference (3)

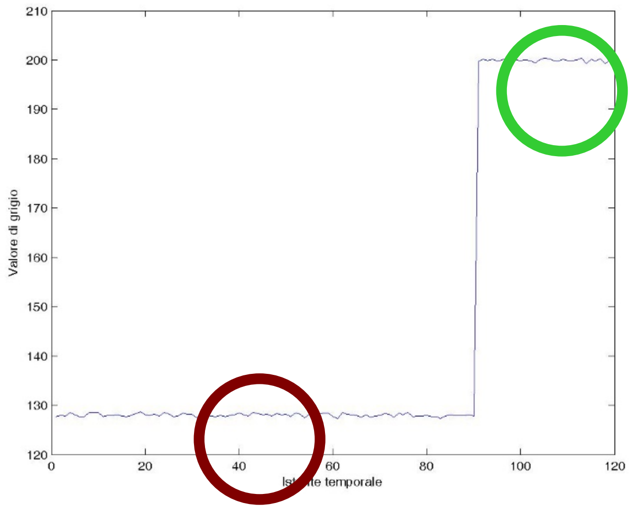
$$p_{t+1}(j) = \sum_{i=1}^{N} P(q_{t+1} = s_j, q_t = s_i) =$$

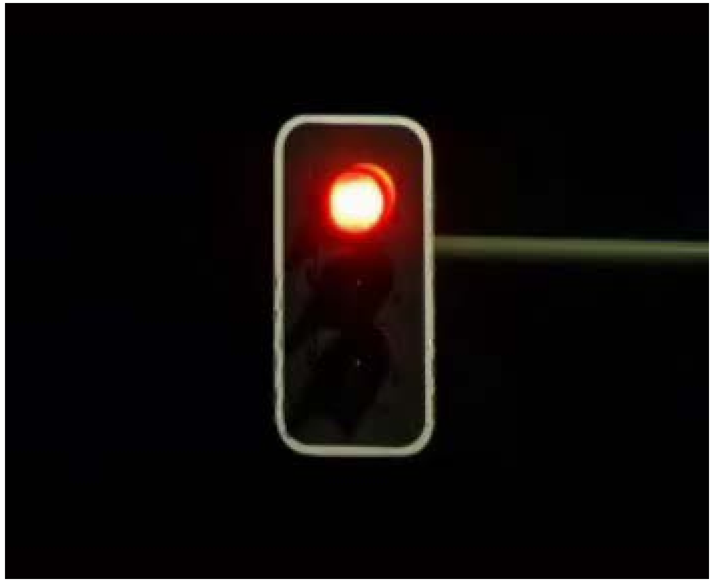
$$= \sum_{i=1}^{N} P(q_{t+1} = s_j \mid q_t = s_i) P(q_t = s_i) = \sum_{i=1}^{N} a_{ij}\, p_t(i)$$

- I use this method starting from $P(q_T = s_j)$ and proceeding backwards
- The cost of computation in this case is $O(TN^2)$

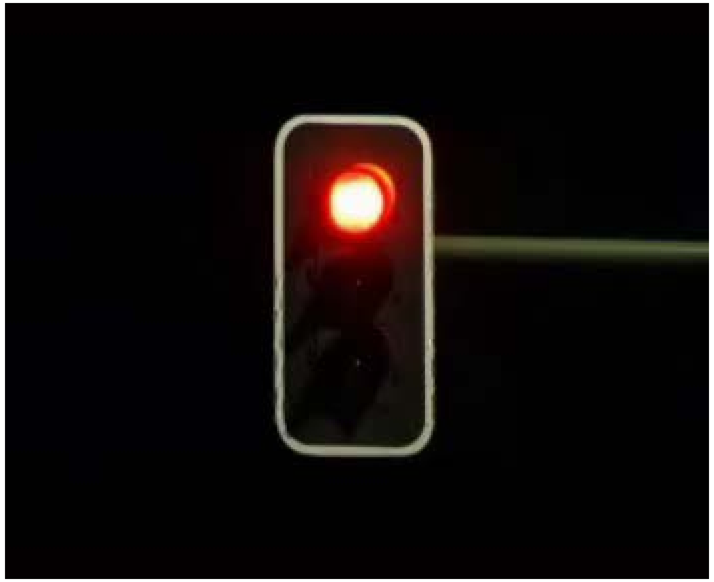
Limits of Markovian models

1. The state should always be ***observable deterministically***, observations have no noise.

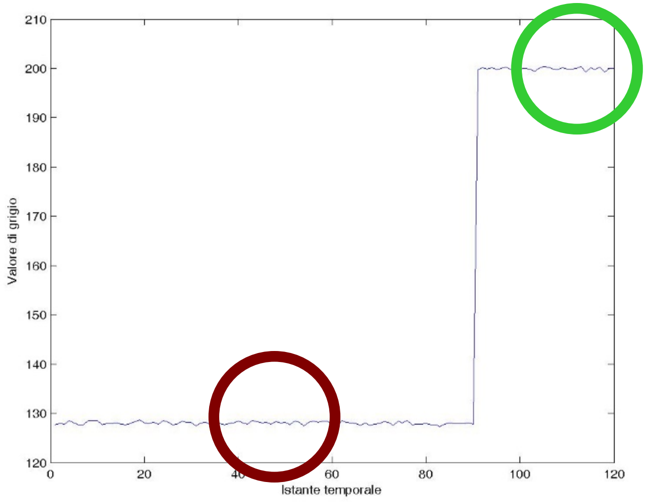




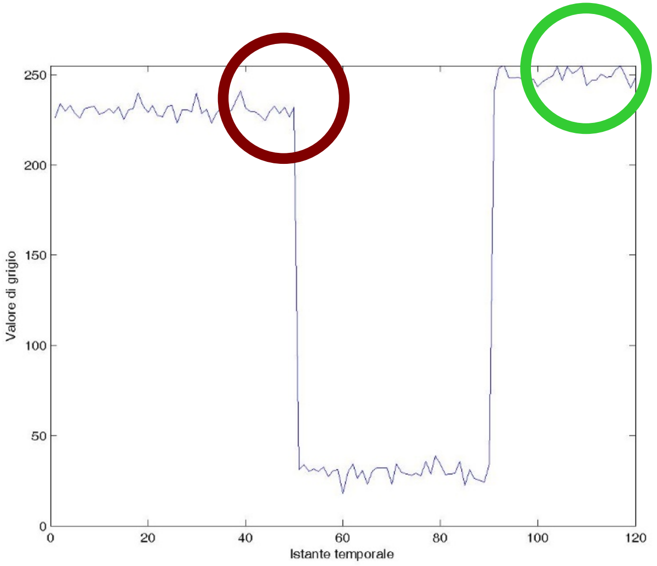
Limits of Markovian models



OK



NO

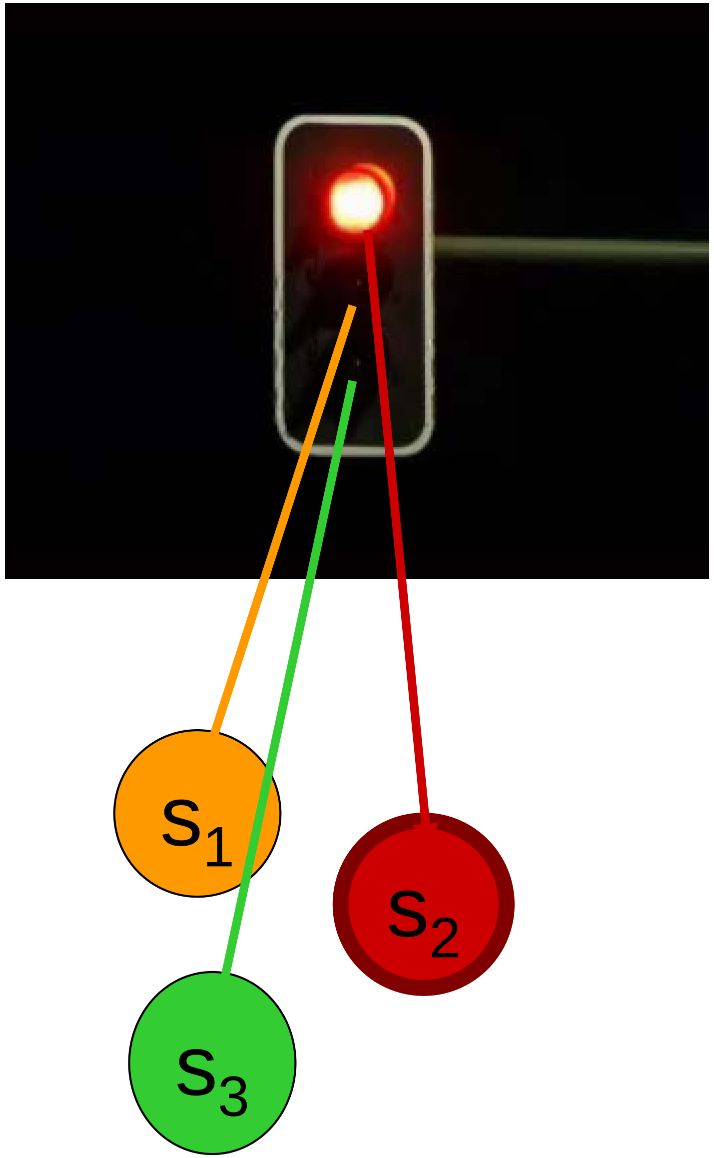


Limits of Markovian models

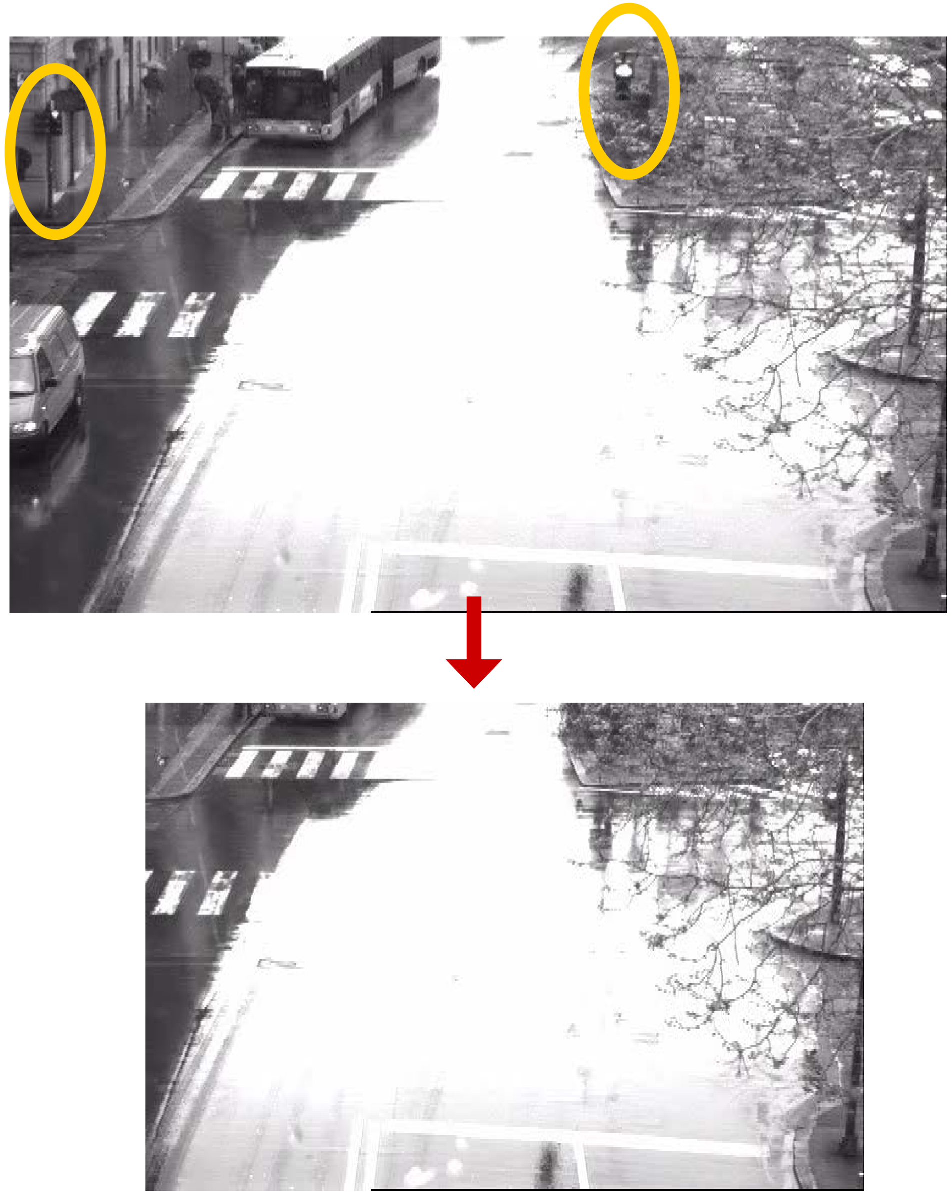
2. **(and more important!)** In the case of the traffic light the state is **explicit**, (a particular traffic light configuration), and **can be assessed directly through observation**

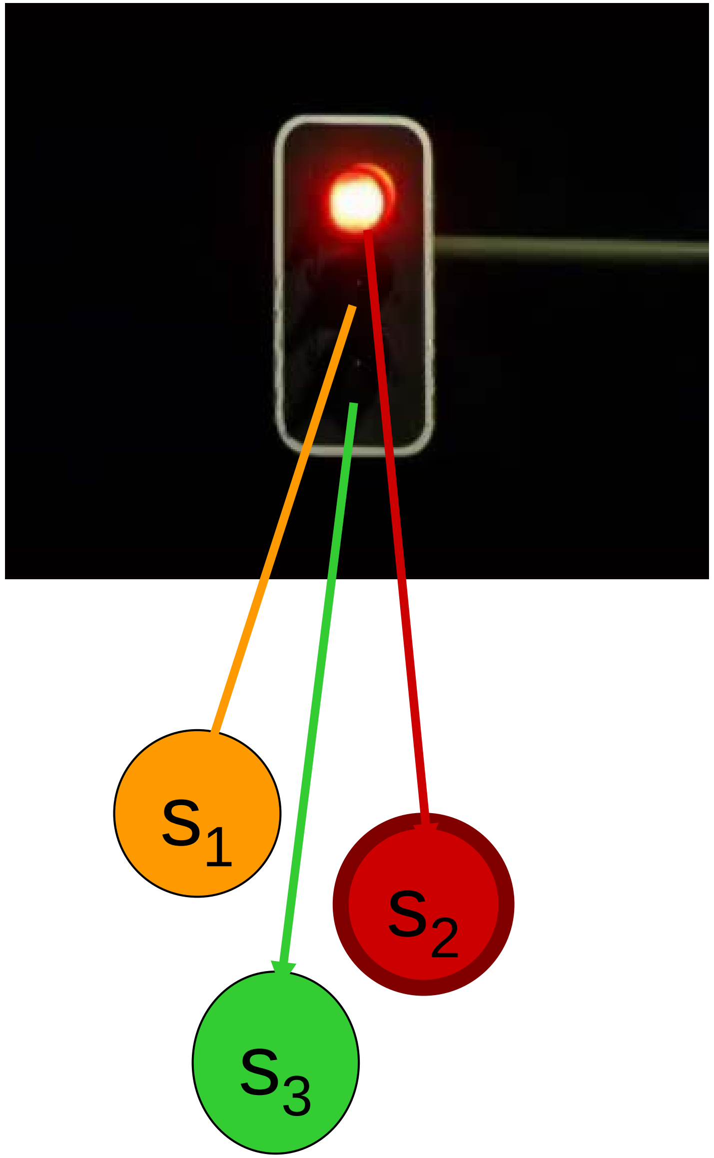
   (the status corresponds to the color of the traffic light)

- This is not always the case!

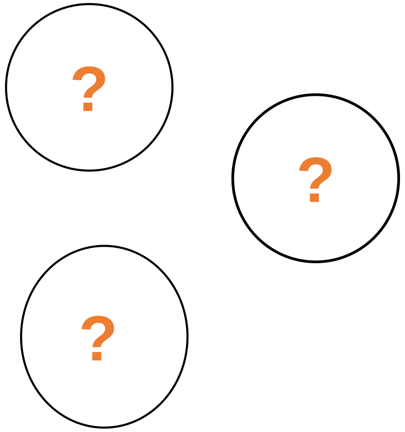


Limits of Markovian models



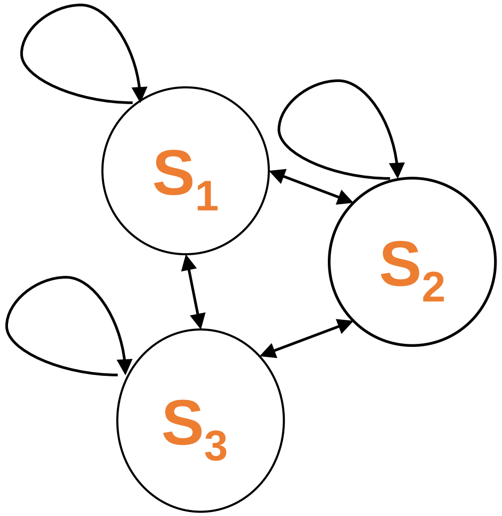


Limits of Markovian models



- *I watch* the video sequence: *I observe* that there is a **system that evolves**, but I cannot understand which the regulatory states are.

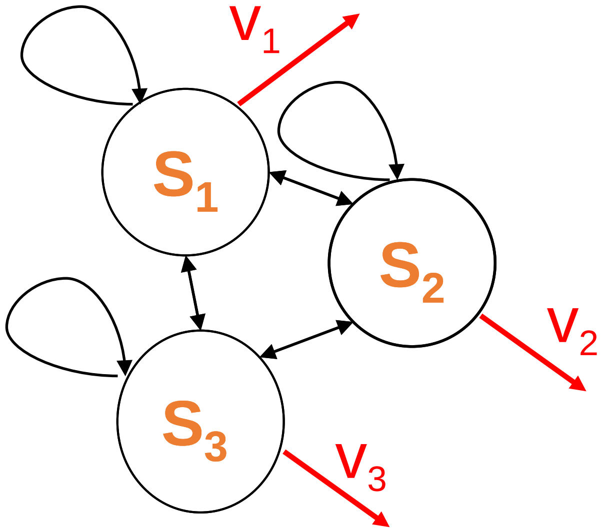
Limits of Markovian models



- *I watch* the video sequence: *I observe* that there is a **system that evolves**, but I cannot understand which the regulatory states are.

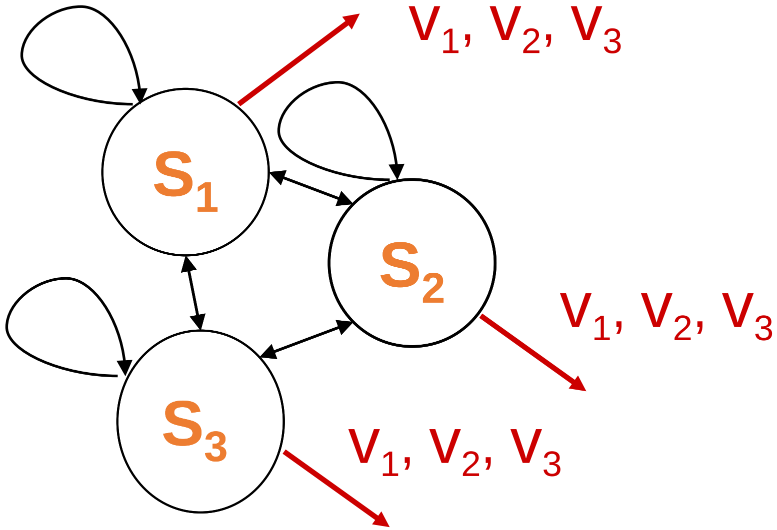
- The system, however, evolves in **states**, which I understand by *observing* the phenomenon

Limits of Markovian models



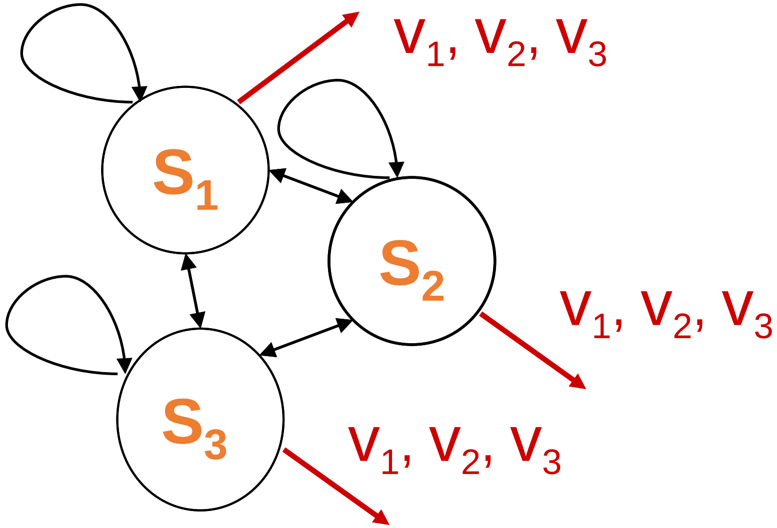
- Better: the system evolves thanks to **"hidden" states**, the states of the traffic light, which I do not see and I do not know even the existence.
- I don’t observe the states, but I can only ***observe*** the probable "consequences« of such states, i.e. the flows of cars

Limits of Markovian models



- I don’t name the states, I just consider as hidden entities and *identifiable only through observations* (cars’ flows)
- I can establish a **relationship between *observation* and *hidden state***

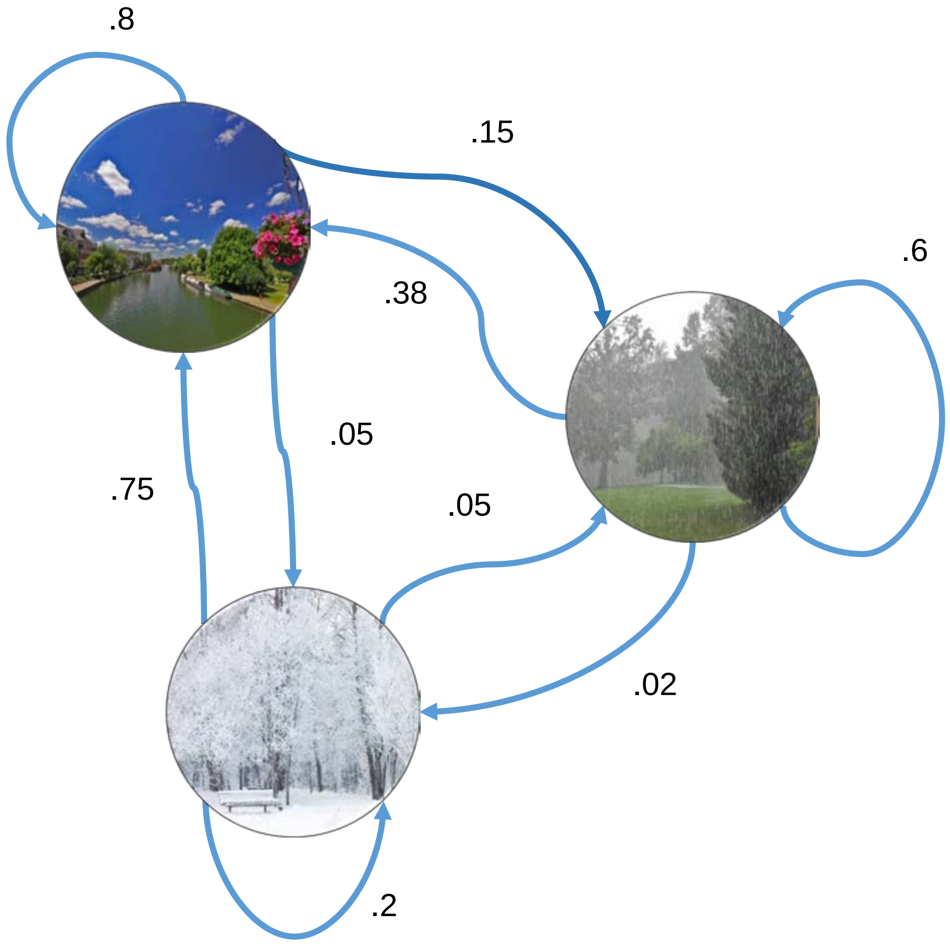
Markov Models with Hidden states or Hidden Markov Models (HMMs)



- Hidden Markov Model fits into this context
- They probabilistically describe the *system dynamics* avoiding to directly identify its causes, or rather, seeking to estimate them
- States are identifiable only by *observations*, in a probabilistic manner.

Hidden Markov Models

States are not observable!



Hidden Markov Model (HMM)

- Statistical sequence classifier, widely used in different contexts.
- Such a model can be understood as an extension of the Markov model from which it differs for the **unobservability of its states**.
- Each state has associated a probability function that describes the probability that a certain symbol (output) is emitted by that state.

HMM: a formal definition

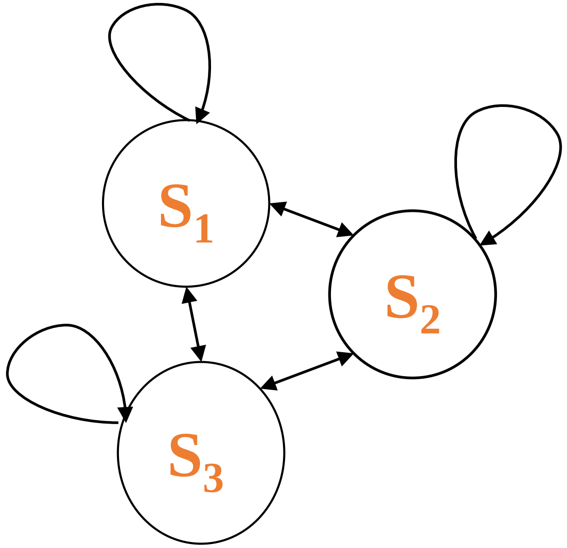
- From [Rabiner 89]:

“The Hidden Markov Model is a doubly embedded stochastic process with an underlying stochastic process that is *not* observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations”

L.R. Rabiner. **A tutorial on hidden Markov models and selected applications in speech recognition.** *Proceedings of the IEEE,* Vol. 77, Issue 2, Feb. 1989.

HMM: formal definition

- An HMM (discrete) consists of:
  - A set $S = \{s_1, s_2, \ldots, s_N\}$ of hidden states
  - A transition matrix $A = \{a_{ij}\}$ between hidden states 1=<i,j=<N
  - An initial distribution over hidden states $\pi=\{\pi_i\}$



π =

$\pi_1 = 0.33$	$\pi_2 = 0.33$	$\pi_3 = 0.33$

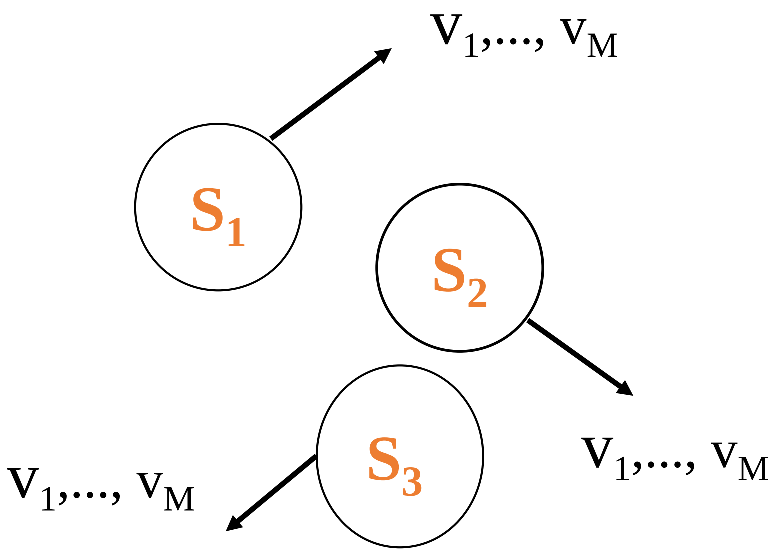
A =

$a_{11} = 0.1$	$a_{12} = 0.9$	$a_{13} = 0$
$a_{21} = 0.01$	$a_{22} = 0.2$	$a_{23} = 0.79$
$a_{31} = 1$	$a_{32} = 0$	$a_{33} = 0$

HMM: formal definition

- A set $V = \{v_1, v_2, \ldots, v_M\}$ *of observation symbols*

- A probability distribution on observation symbols $B = \{b_{jk}\}$*,* which *indicates the probability of emission of the symbol* v_k when the system state is s_j.



$\boldsymbol{B} =$

$b_{11}=0.8$	$b_{21}=0.1$	$b_{31}=0.1$
$b_{12}= 0.1$	$b_{22}= 0.8$	$b_{32}= 0.1$
$\vdots$	$\vdots$	$\vdots$
$b_{1M}= 0.1$	$b_{2M}= 0.1$	$b_{3M}=0.8$

HMM: formal definition

- We denote an HMM with a triple $\lambda = (A, B, \pi)$

$\boldsymbol{\pi} =$

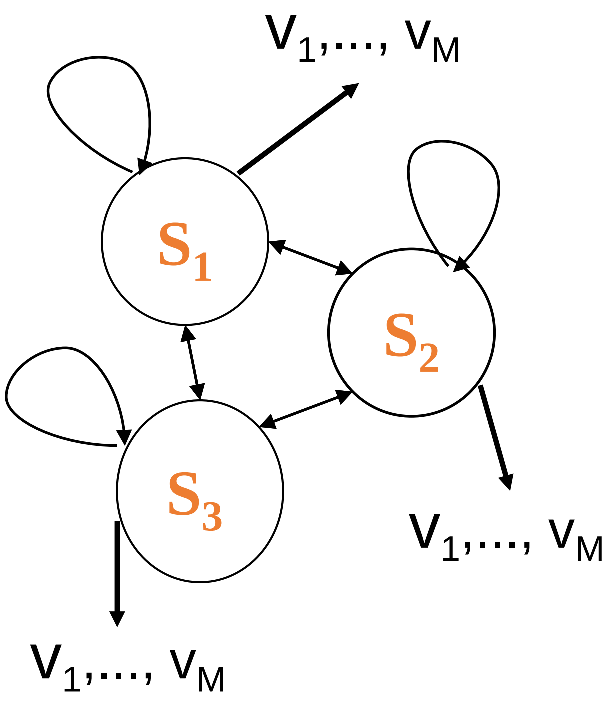
π_1= 0.33	π_2= 0.33	π_3= 0.33

$\boldsymbol{A} =$

a_{11}= 0.1	a_{12}= 0.9	a_{13}=0
a_{21}= 0.01	a_{22}= 0.2	a_{23}= 0.79
a_{31}= 1	a_{32}= 0	a_{33}= 0

$\boldsymbol{B} =$

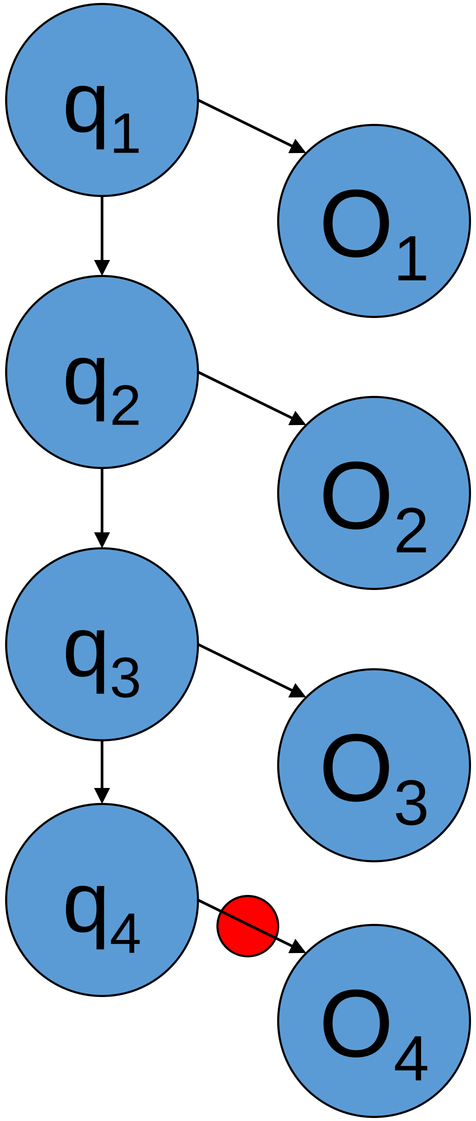
b_{11}=0.8	b_{21}=0.1	b_{31}=0.1
b_{12}= 0.1	b_{22}= 0.8	b_{32}= 0.1
⋮	⋮	⋮
b_{1M}= 0.1	b_{2M}= 0.1	b_{3M}=0.8



Assumptions on observations

- Conditional independence

$$P(O_t = X \mid q_t = s_j, q_{t-1}, q_{t-2}, \ldots, q_2, q_1, O_{t-1}, O_{t-2}, \ldots, O_2, O_1)$$
$$= P(O_t = X \mid q_t = s_j)$$

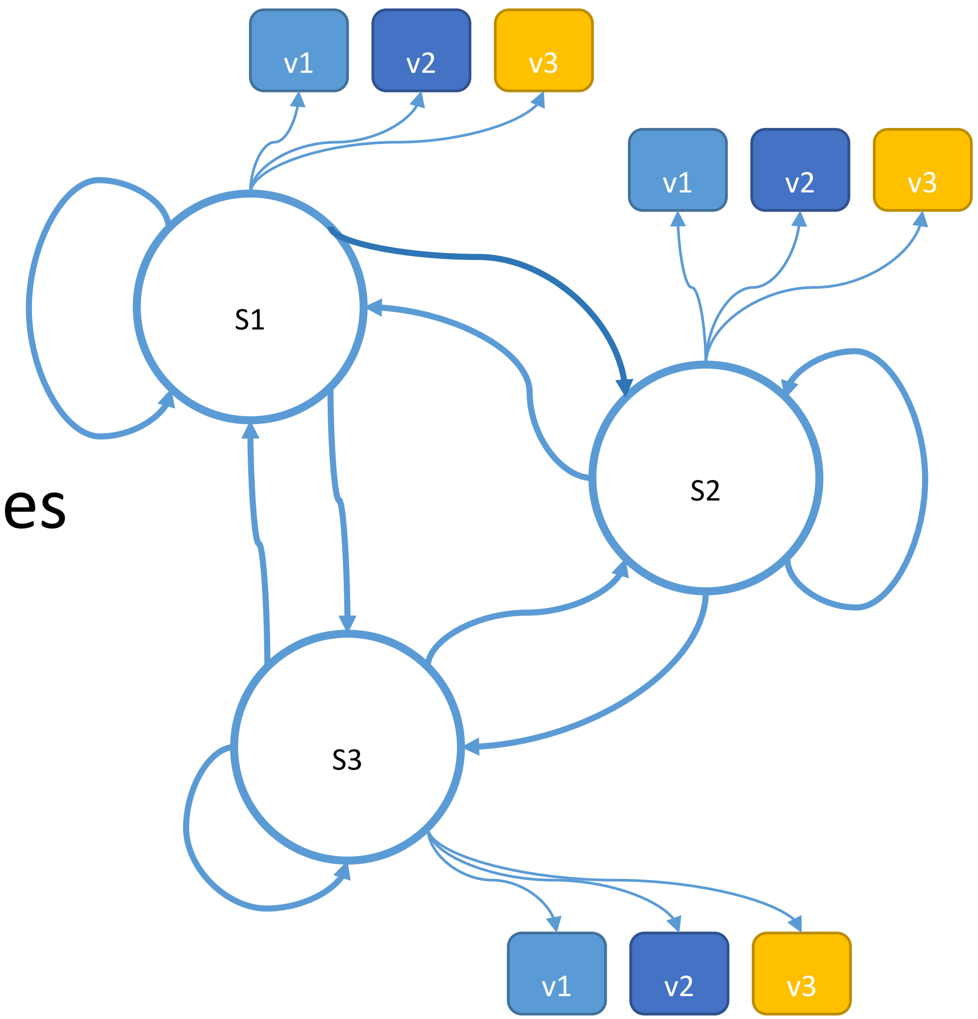


Hidden Markov Models

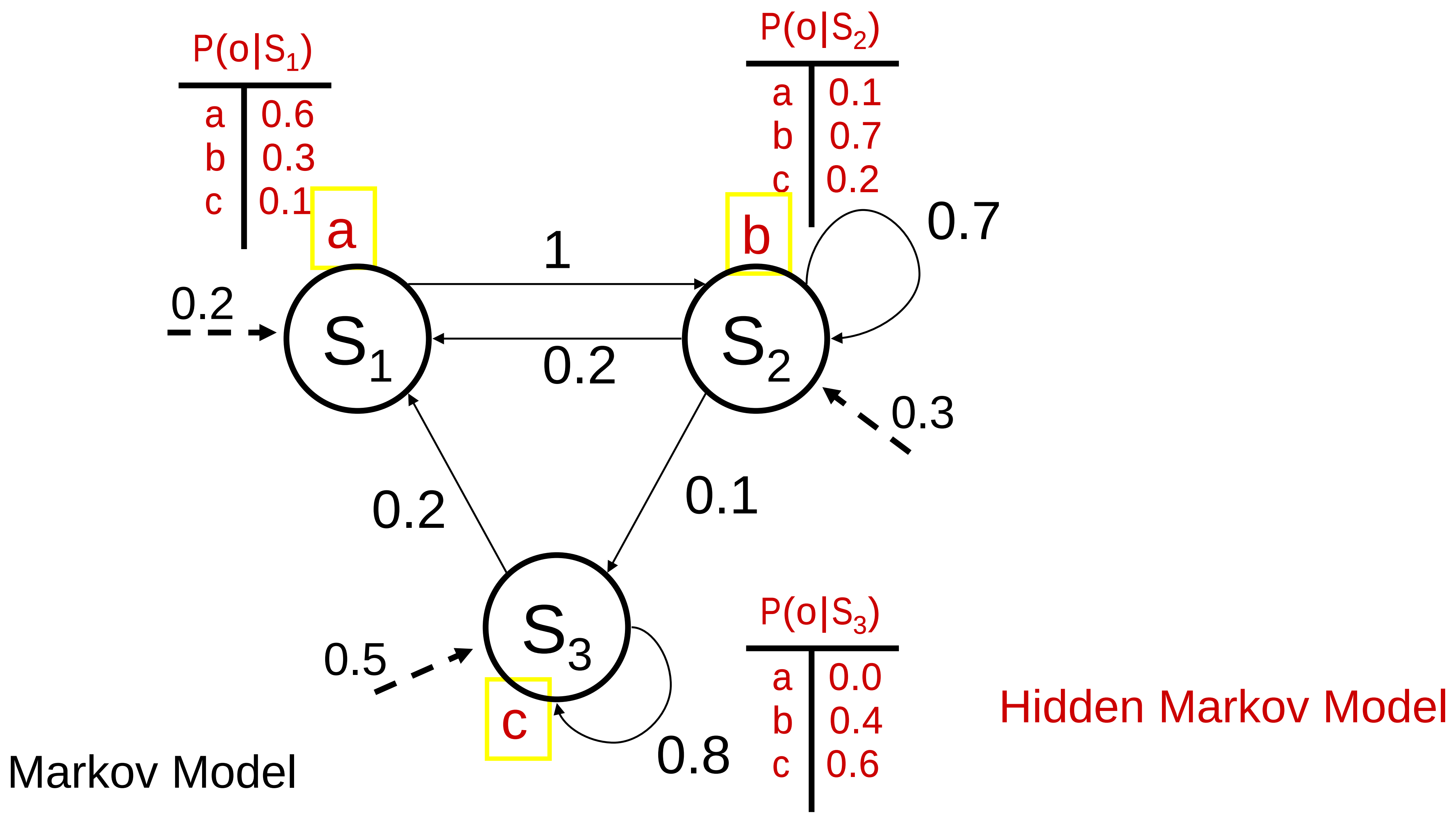
- A set of N **states** $\mathcal{S} = \{S_1, \dots, S_N\}$
- A sequence of states $Q = q_1, \dots, q_T$

- An **initial probability distribution** over states
$$\Pi = \{\pi_i = P(q_1 = S_i)\}$$
- A **transition probability matrix**
$$A = \{a_{ij} = P(q_t = S_j | q_{t-1} = S_i)\}$$
- A set of **emission probabilities**
$$\mathrm{B} = b_i(v_k) = P(o_t = v_k | q_t = S_i)$$

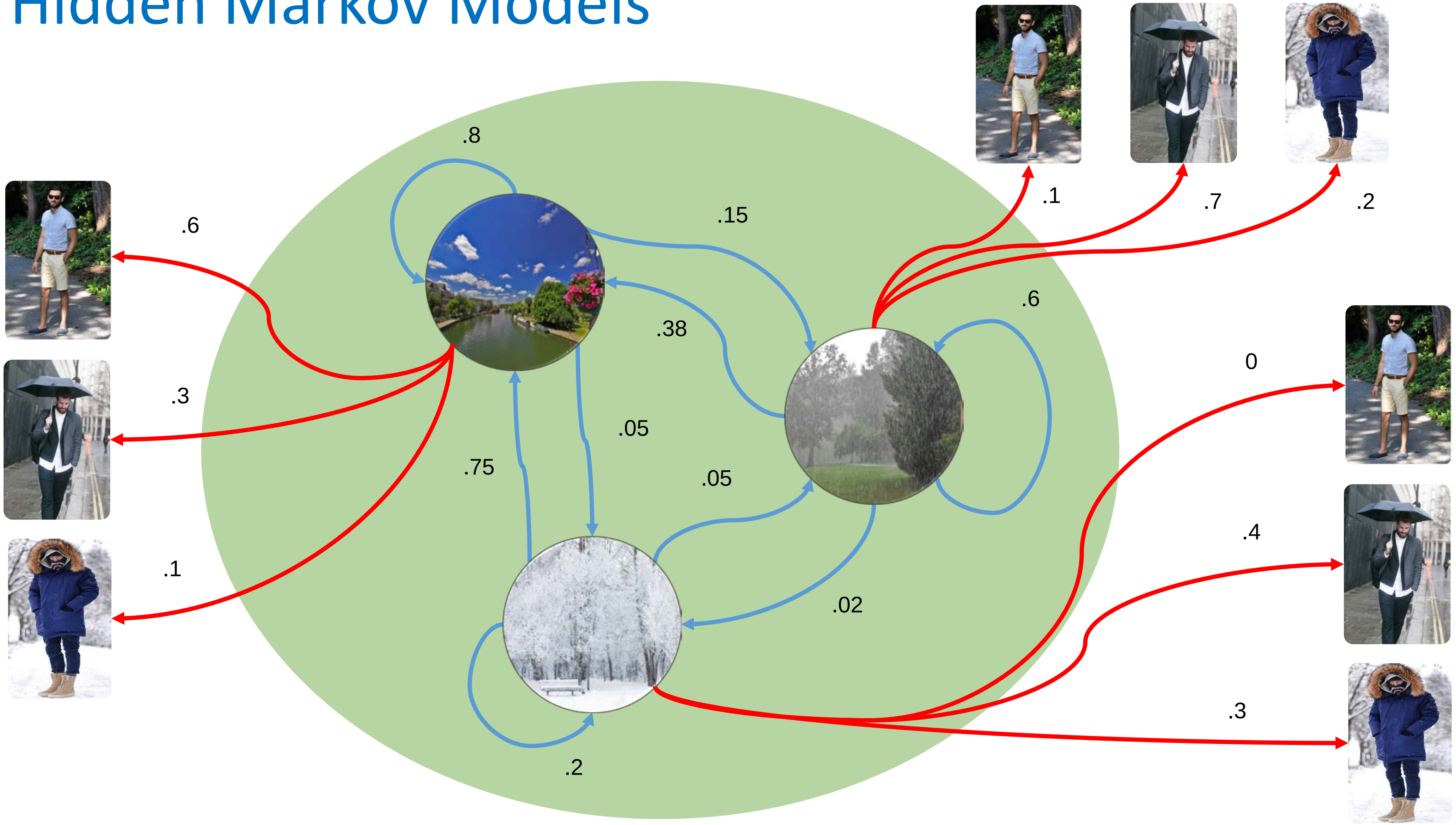
- An **observation vocabulary** $\mathcal{V} = \{v_1, \dots, v_M\}$
- A sequence of **observations** $\mathcal{O} = o_1, \dots, o_T$



From a Markov Model to a Hidden Markov Model



Hidden Markov Models



Key problems for HMMs

<u>Problem 1:</u> ***Evaluation o Likelihood***

Given an HMM λ model and an observation string $\mathbf{O}=(O_1,O_2,\ldots,O_t,\ldots,O_T)$ calculate $P(\mathbf{O}|\lambda)$
→ *forward* procedure

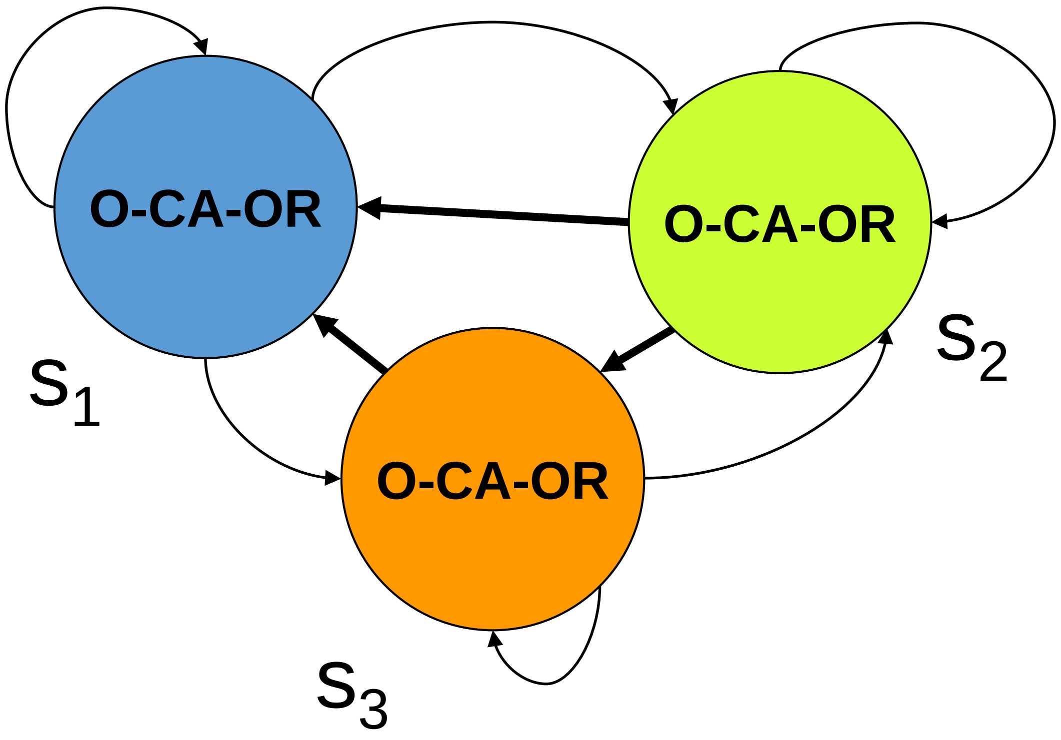
<u>Problem 2:</u> ***Decoding***

Given an observation string $\mathbf{O}$ and an HMM λ model, calculate the most likely sequence of states $S_1\ldots S_T$ that generated $\mathbf{O}$
→ *Viterbi* procedure

<u>Problem 3:</u> ***Training***

Given a set of observations $\{\mathbf{O}\}$, determine the best HMM model $\lambda=(\pi, A, B)$, i.e. the model for which $P(\mathbf{O}|\lambda)$ is maximized
→ *Baum Welch* procedure *(forward-backword)*

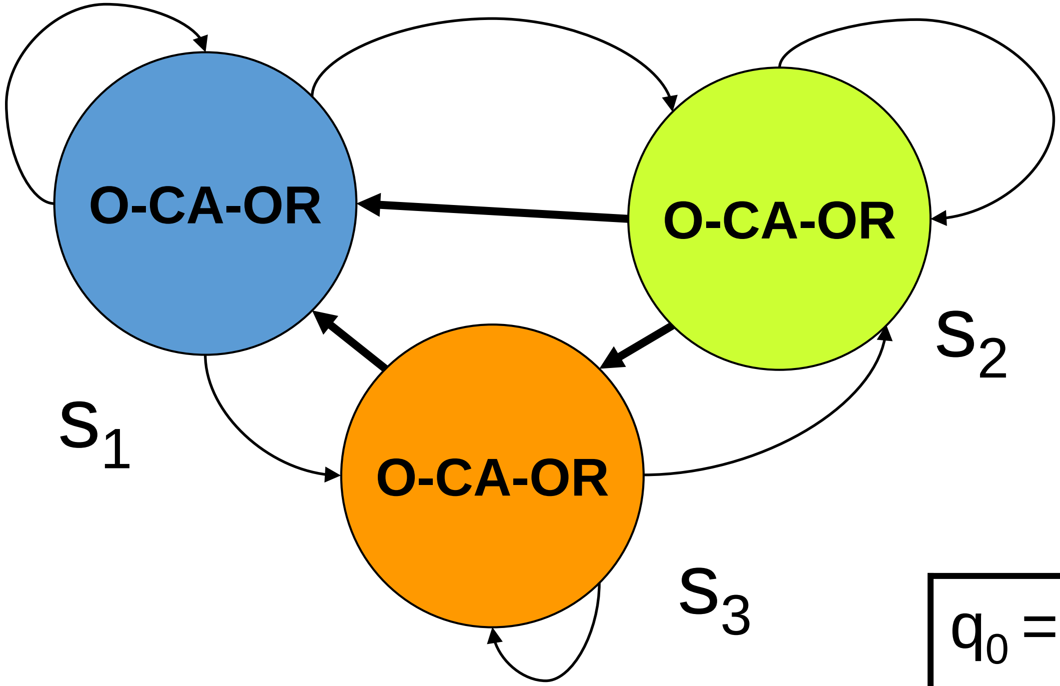
HMM – string generator



- 3 states: s_1, s_2, s_3
- 3 symbols: O,CA,OR

π_1= 0.33	π_2= 0.33	π_3= 0.33
b_1(O)=0.8	b_2(O)=0.1	b_3(O)=0.1
b_1(OR)= 0.1	b_2(OR)= 0.0	b_3(OR)= 0.8
b_1(CA)= 0.1	b_2(CA)= 0.9	b_3(CA)= 0.1
a_{11}= 0	a_{12}= 1	a_{13}= 0
a_{21}= 1/3	a_{22}= 2/3	a_{23}= 0
a_{31}= 1/2	a_{32}= 1/2	a_{33}= 0

HMM – string generator



Our problem is that the states are not directly observable!

$q_0 =$	S_2	$O_1 =$	CA
$q_1 =$	S_2	$O_2 =$	CA
$q_2 =$	S_1	$O_3 =$	O

$q_0 =$	?	$O_1 =$	CA
$q_1 =$	?	$O_2 =$	CA
$q_2 =$	?	$O_3 =$	O

Problem 1: Probability of a series of observations

- $P(\mathbf{O})=P(O_1,O_2,O_3)=P(O_1=CA,O_2=CA,O_3=O)$?
- Brute force strategy:

$$P(\mathbf{O}) = \sum_{\mathbf{Q} \in \text{paths of length 3}} P(\mathbf{O},\mathbf{Q})$$

$$= \sum_{\mathbf{Q} \in \text{paths of length 3}} P(\mathbf{O}\,|\,\mathbf{Q})P(\mathbf{Q})$$

? ?

Problem 1: Probability of a series of observations

- $P(\mathbf{O})=P(O_1,O_2,O_3)=P(O_1=X,O_2=X,O_3=Z)$?
- Brute force strategy:

$$P(\mathbf{O}) = \sum P(\mathbf{O},\mathbf{Q})$$

$$= \sum P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q})$$

$$P(\mathbf{Q}) = P(q_1,q_2,q_3) =$$

$$= P(q_1)P(q_2,q_3|q_1)$$

$$= P(q_1)P(q_2|q_1)P(q_3|q_2)$$

For example, in the case

$$\mathbf{Q} = S_2\, S_2\, S_1 = \pi_2\, a_{22}\, a_{21}$$

$$= 1/3*2/3*1/3 = 2/27$$

Problem 1: Probability of a series of observations

- $P(\mathbf{O})=P(O_1,O_2,O_3) = P(O_1=X,O_2=X,O_3=Z)$?
- Brute force strategy:
  - $P(\mathbf{O}) = \sum P(\mathbf{O},\mathbf{Q})$

    $= \sum P(\mathbf{O}|\mathbf{Q})P(\mathbf{Q})$

$P(\mathbf{O}|\mathbf{Q}) =$

$= P(O_1,O_2,O_3|q_1,q_2,q_3)$

$= P(O_1|q_1)P(O_2|q_2)P(O_3|q_3)$

<u>For example</u>, in the case

$\mathbf{Q} = S_2 S_2 S_1 =$

$= 9/10*9/10*8/10 = 0.648$

Considerations

- Previous calculations solve **only one term of the summation**: for the calculation of P(**O**) are required 27 P(Q) and 27 P(**O**|**Q**)
- For a sequence of 20 observations we need 3^{20} P(Q) and 3^{20} P(O|Q)
- There is a more effective way, which is based on the definition of a particular probability
- Generally:

$$P(O \mid \lambda) = \sum_{\text{All sequences } Q_1,\dots,Q_T} \pi_{Q_1} b_{Q_1}(O_1) a_{Q_1 Q_2} b_{Q_2}(O_2) a_{Q_2 Q_3} \cdots$$

is of high complexity, $O(N^T T)$, where N is the number of states, T length of the sequence

Forward Procedure

- Given the observations $O_1, O_2, \ldots, O_T$ we define

  $$\alpha_t(i) = P(O_1, O_2, \ldots, O_t,\ q_t = s_i \mid \lambda),\ \text{where } 1 \le t \le T$$

  that is:
  - *we have seen the first t observations*
  - *we ended up in* s_i *at the t-th visited state*
- This probability can be defined recursively:

  $$\alpha_1(i) = P(O_1, q_1 = s_i) = P(q_1 = s_i) P(O_1 | q_1 = s_i) = \pi_i\, b_i(O_1)$$

  - By inductive hypothesis $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = s_i \mid \lambda)$
  - I want to calculate:

  $$\alpha_{t+1}(j) = P(O_1, O_2, \ldots, O_t,\ O_{t+1},\ q_{t+1} = s_j \mid \lambda)$$

$$\alpha_{t+1}(j) = P(O_1,O_2,\ldots,O_t, O_{t+1}, q_{t+1}=s_j)$$

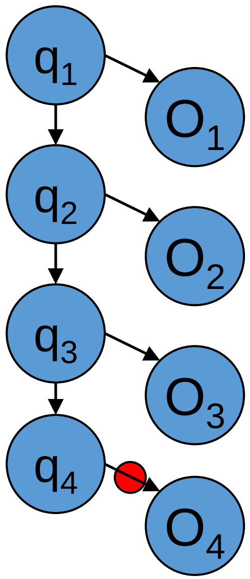
$$= \sum_{i=1}^{N} P(O_1,O_2,\ldots,O_t,q_t=s_i,\underbrace{O_{t+1}, q_{t+1}=s_j})$$

$$= \sum_{i=1}^{N} P(O_{t+1},q_{t+1}=s_j|\underbrace{O_1,O_2,\ldots,O_t,q_t=s_i})\underbrace{P(O_1,O_2,\ldots,O_t,q_t=s_i)}$$

$$= \sum_{i=1}^{N} \underbrace{P(O_{t+1}, q_{t+1}=s_j|q_t=s_i)}\,\alpha_t(i) \quad \text{p.i.i.}$$

$$= \sum_{i=1}^{N} \underbrace{P(q_{t+1}=s_j|q_t=s_i)}\,\underbrace{P(O_{t+1}|q_{t+1}=s_j)}\,\alpha_t(i)$$

$$= \sum_{i=1}^{N} [a_{ij}\,\alpha_t(i)]\, b_j(O_{t+1})$$



Response to Problem 1: Evaluation

- Given $O_1,O_2,\ldots, O_t, \ldots,O_T$ and knowing $\alpha_t(i) = P(O_1,O_2,\ldots,O_t,q_t=s_i|\lambda)$, we can calculate:

$$\boxed{P(O|\lambda) = P(O_1,O_2,\ldots,O_T|\lambda) = \sum_{i=1}^{N} \alpha_T(i)}$$

having complexity $O(N^2T)$

- But also other useful quantities, for example:

$$P(q_t=s_i \mid O_1,O_2,\ldots,O_t) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}$$

Response to Problem 1: Evaluation

- α is called a *forward* variable
- Alternatively, it can be calculated recursively by introducing another variable, the so-called *backward variable*

$$\beta_t(j) = P(O_{t+1}\ldots O_T \mid q_t{=}s_j, \lambda) = \sum_{i=1}^{N} P(O_{t+1}\ldots O_T, q_{t+1}{=}s_i \mid q_t{=}s_j, \lambda)$$

$$= \sum_{i=1}^{N} \beta_{t+1}(i)\, a_{ji}\, b_i(O_{t+1})$$

Hence

$$P(O \mid \lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j) \qquad \forall t$$

$$= \sum_{j=1}^{N} \beta_1(j) \qquad \text{Please, verify!}$$

Problem 2: Decoding (more likely path)

- What is the most probable (state) path (MPP) that generated $O_1,O_2,...,O_T$? That is, how to compute:

$$\underset{\mathbf{Q}}{\operatorname{argmax}}\, P(\mathbf{Q} \,|\, \mathrm{O}_1\mathrm{O}_2...\mathrm{O}_\mathrm{T})\,?$$

- Brute force strategy:

$$\underset{\mathbf{Q}}{\operatorname{argmax}} \frac{P(\mathrm{O}_1\mathrm{O}_2...\mathrm{O}_\mathrm{T} \,|\, \mathbf{Q})P(\mathbf{Q})}{P(\mathrm{O}_1\mathrm{O}_2...\mathrm{O}_\mathrm{T})}$$

$$\propto \underset{\mathbf{Q}}{\operatorname{argmax}}\, P(\mathrm{O}_1\mathrm{O}_2...\mathrm{O}_\mathrm{T} \,|\, \mathbf{Q})P(\mathbf{Q})$$

Efficient computation of the MPP

- Let's define the following probability:

$$\delta_t(i) = \max_{q_1 q_2 \ldots q_{t-1}} P(q_1 q_2 \ldots q_{t-1}, q_t = s_i, O_1 O_2 \ldots O_t)$$

i.e., the maximum probability of paths of length t-1 which:

  - occur,
  - end up in the state s_i at time t,
  - produce as output $O_1, O_2, \ldots, O_t$

- You look for the single best sequence of single states (path) maximizing $P(\mathbf{Q}|\mathbf{O}, \lambda)$
- The solution to this problem is a dynamic programming technique called Viterbi's algorithm.
  - We look for the most likely single state at the i-th position given the previous observations and states

Viterbi algorithm

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N$$

$$\psi_1(i) = 0.$$

By induction we have

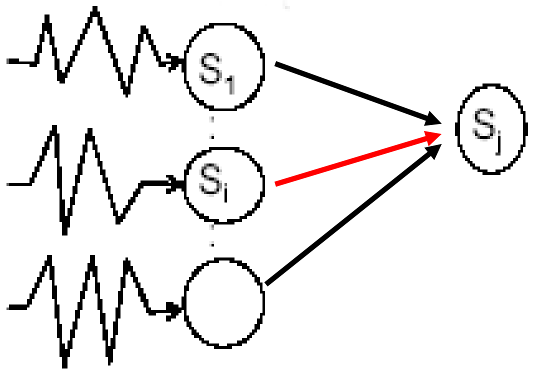
$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}).$$

Viterbi algorithm

2) Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(O_t), \qquad 2 \le t \le T$$
$$1 \le j \le N$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], \qquad 2 \le t \le T$$
$$1 \le j \le N.$$



ATTENTION:
calculated for each j !!!

Viterbi algorithm

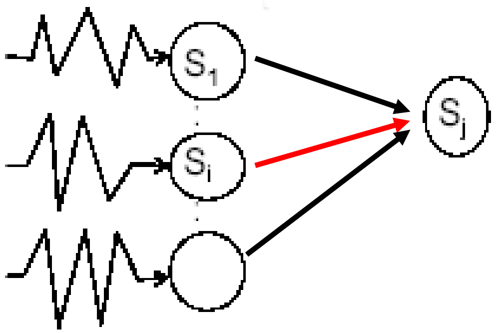
3) Termination:

$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = \underset{1 \le i \le N}{\operatorname{argmax}} [\delta_T(i)].$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \cdots, 1.$$



Problem 3: HMM Training

- We talk about HMM training process or parameter estimation phase, in which the parameters of $\lambda=(A,B,\pi)$, are estimated from *training observations*

- Usually, the Maximum Likelihood estimate is used

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \quad P(O_1O_2 \ldots O_T \mid \lambda)$$

- But other estimates can also be used

$$\max_{\lambda} \quad P(\lambda \mid O_1O_2 \ldots O_T)$$

ML estimation of HMM: Baum Welch's re-estimation procedure

Let's define

- $\gamma_t(i) = P(\mathrm{q_t} = \mathrm{s_i} \mid \mathrm{O_1 O_2 ... O_T}, \lambda)$
- $\xi_t(i, j) = P(\mathrm{q_t} = \mathrm{s_i}, \mathrm{q_{t+1}} = \mathrm{s_j} \mid \mathrm{O_1 O_2 ... O_T}, \lambda)$

These quantities can be calculated efficiently (cf. Rabiner)

$$\sum_{j=1}^{N} \xi_t(i, j) = \gamma_t(i)$$

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

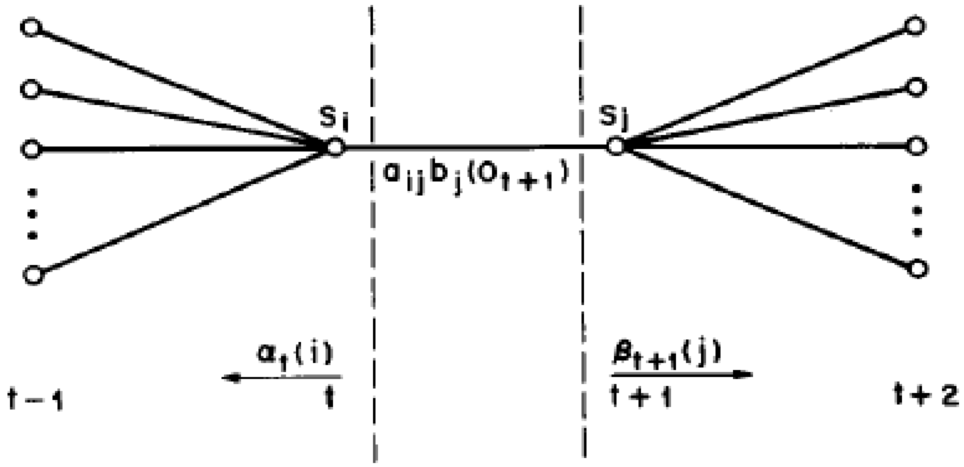
$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{ expected number of transitions passing through state } i \text{ along the way}$$

- Using forward and backward variables, ξ is also calculable as

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

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

(E step)



ML estimation of HMM: Baum Welch's re-estimation procedure

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

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

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

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

$$= \frac{\sum_{\substack{t=1 \\ \text{s.t. } O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}.$$

Parameter re-estimation formulas (M step)

Baum-Welch algorithm

- These quantities are used in the process of estimating HMM parameters iteratively

- A variation of the Expectation-Maximization (EM) algorithm is used
    - that performs a local optimization
    - maximizing the log-likelihood of the model with respect to the data

$$\lambda_{opt} = \text{argmax} \log P(\{\mathbf{O}_l\} \mid \lambda)$$

EM - BAUM WELCH (2)

- Knowing the quantities such as:
  - expected number of transitions leaving state *i* along the way,
  - expected number of transitions from state *i* to state *j* along the path,
  - we could calculate the current ML estimates of λ (= $\overline{\lambda}$), that is

$$\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$$

- These considerations give rise to the Baum-Welch algorithm

- Algorithm:
  1) I initialize the model $\overline{\lambda} = (A_0, B_0, \pi_0)$
  2) the current model is $\lambda = \overline{\lambda}$
  3) I use the model λ to calculate the right part of the re-estimation formulas, i.e., the statistics (E step)
  4) I use such statistics for the re-estimation of parameters obtaining the new model $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ (M step)
  5) go to step 2, until termination occurs

- Baum showed that at every step:

$$P(O_1, O_2, \ldots, O_T \mid \overline{\lambda}) > P(O_1, O_2, \ldots, O_T \mid \lambda)$$

- Usual termination conditions:
  - after a fixed number of cycles
  - convergence of the likelihood value

HMM training

Fundamental issue:

- Baum-Welch is a gradient-descent optimization technique (local optimizer)
- the log-likelihood is highly multimodal

$$\Downarrow$$

- initialization of parameters can crucially affect the convergence of the algorithm

Some open issues/research trends

1. Model selection
   - how many states?
   - which topology?

2. Extending standard models
   - modifying dependencies or components

3. Injecting discriminative skills into HMM

Some open issues/research trends

1. Model selection
   - how many states?
   - which topology?

2. Extending standard models
   - modifying dependencies or components

3. Injecting discriminative skills into HMM

Model selection

- The problem of determining the HMM structure:
  - not a new problem, but still a not completely solved issue

1. Choosing the number of states: the “standard” model selection problem

2. Choosing the topology: forcing the absence or the presence of connections

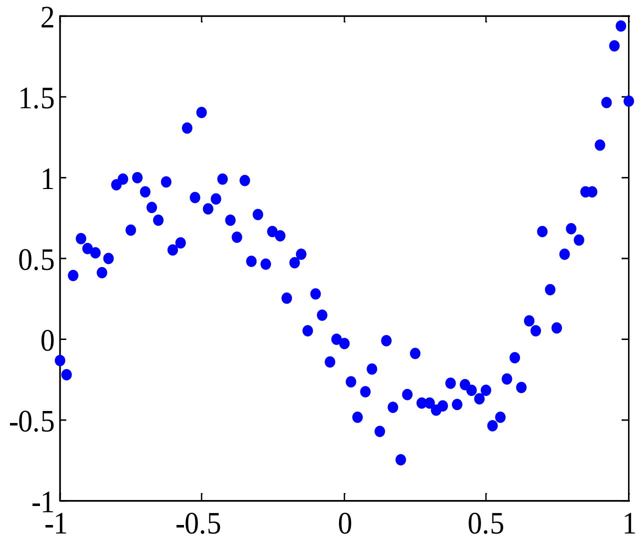
Model selection problem 1:
selecting the number of states

- Number of states: prevents overtraining
- The problem could be addressed using standard model selection approaches

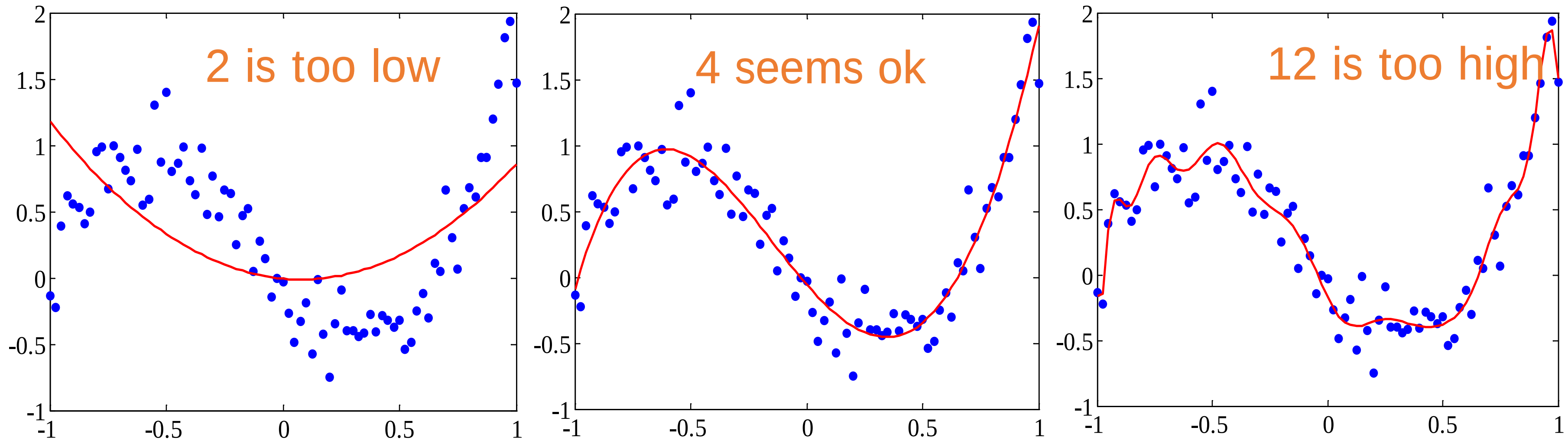
… let’s understand the concept with a toy example

What is model selection?

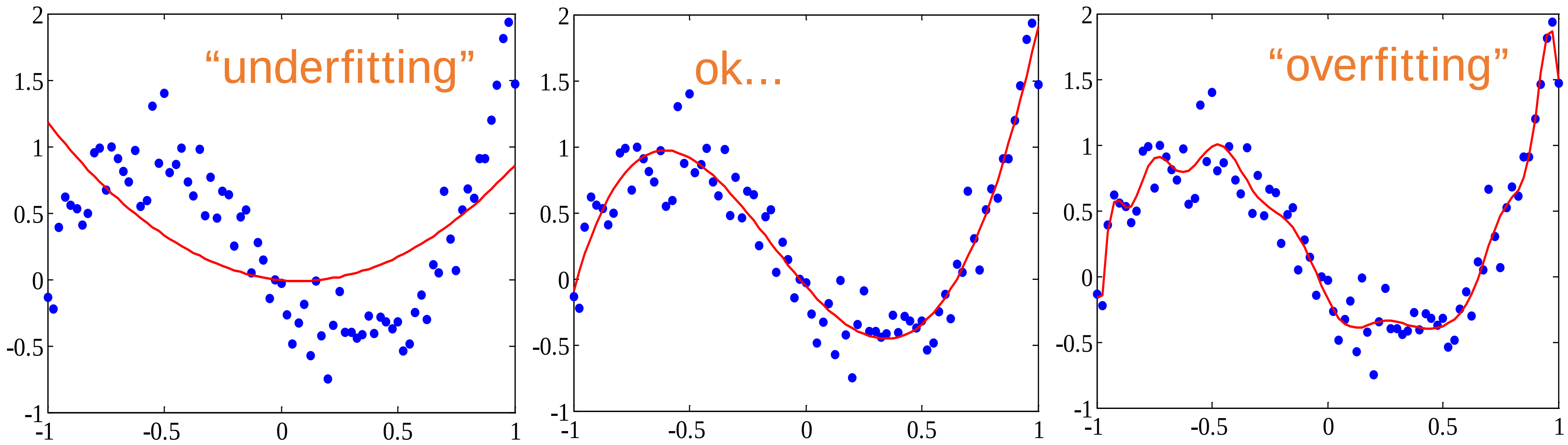
Toy example: some experimental data to which we want to fit a polynomial.



The model selection question is: which order?



What is model selection?



2 is too low

Model selection goal:

how to identify the underlying trend of the data, ignoring the noise?

Model selection: solutions

- Typical solution (usable for many probabilistic models)
  - train several models with different orders k
  - choose the one maximizing an “optimality” criterion

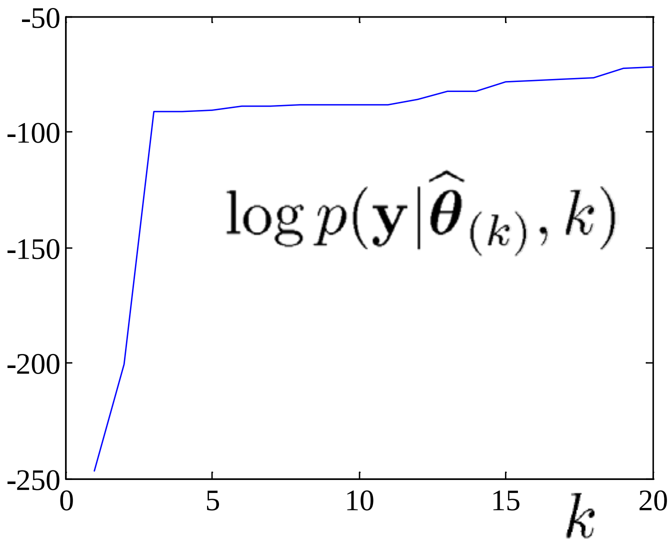
Which “optimality” criterion?

- First naive solution: maximizing likelihood of data w.r.t. model

Maximizing Log Likelihood

- Problem: Log Likelihood is <u>not</u> decreasing when augmenting the order

Not applicable criterion!



Alternative: penalized likelihood

- Idea: find a compromise between fitting accuracy and simplicity of the model
- Insert a “penalty term” which grows with the order of the model and discourages highly complex models

$$K_{best} = \operatorname{argmax}_k \left(LL(y|\theta_k) - \underbrace{C(k)}_{\text{complexity penalty}} \right)$$

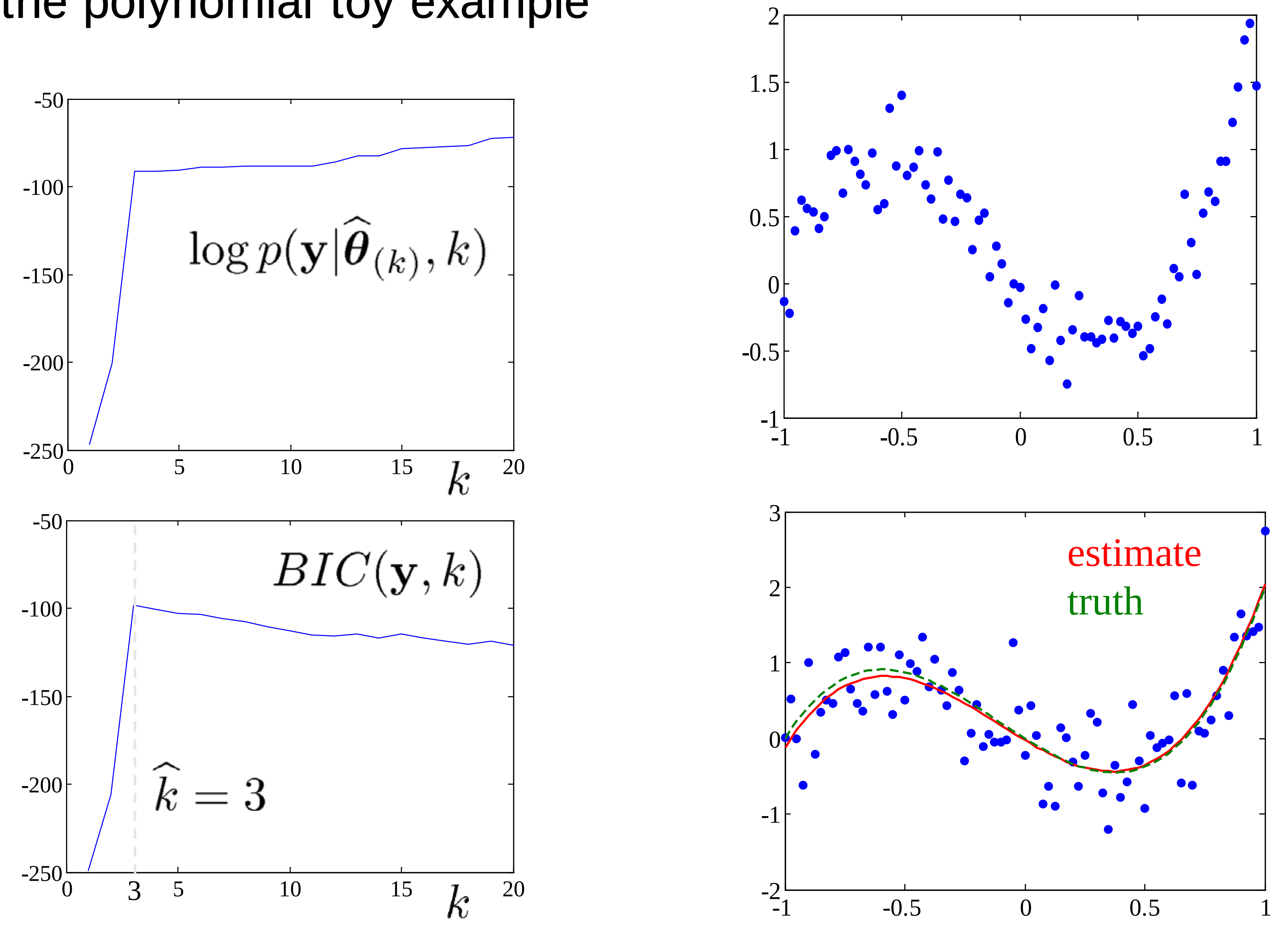
Examples: BIC, MDL, MML, AIC, ...

Alternative: penalized likelihood

- Example: Bayesian information criterion (BIC) [Schwartz, 1978]

$$k_{best} = \arg\max_{k} \left\{ \underbrace{LL(y \mid \theta_k)}_{\text{increases with k}} - \underbrace{\frac{k}{2}\log(n)}_{\text{decreases with k (penalizes larger k)}} \right\}$$

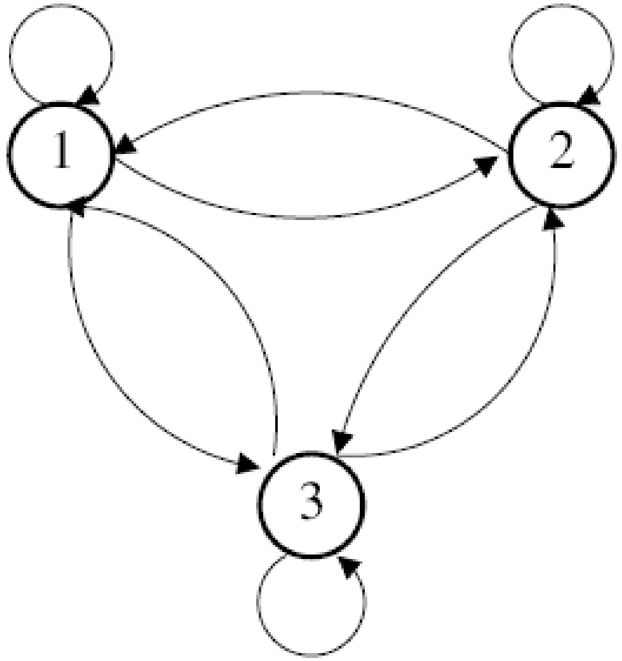
Back to the polynomial toy example



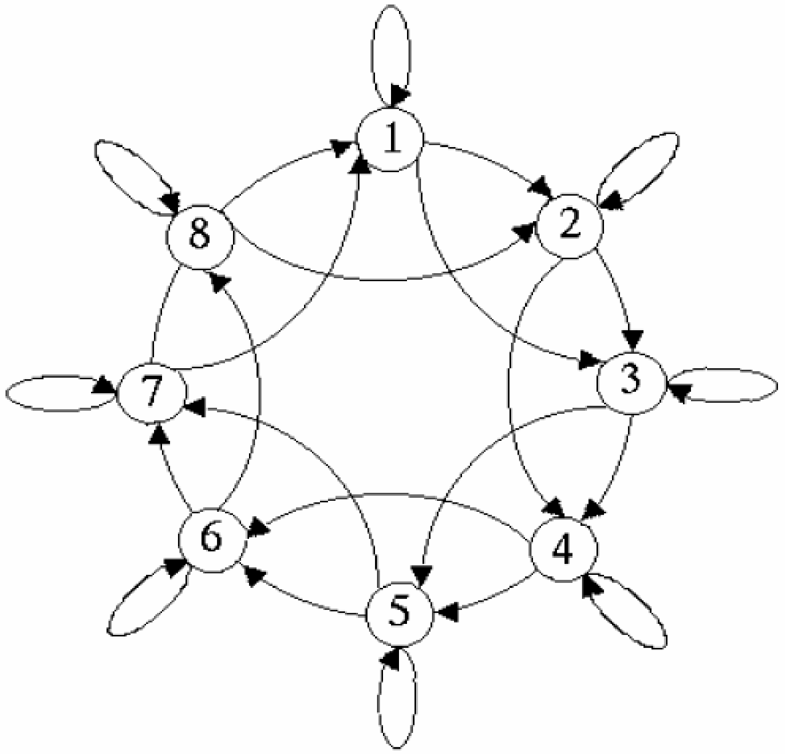
Model selection problem 2: selecting the best topology

- Problem: forcing the absence or the presence of connections

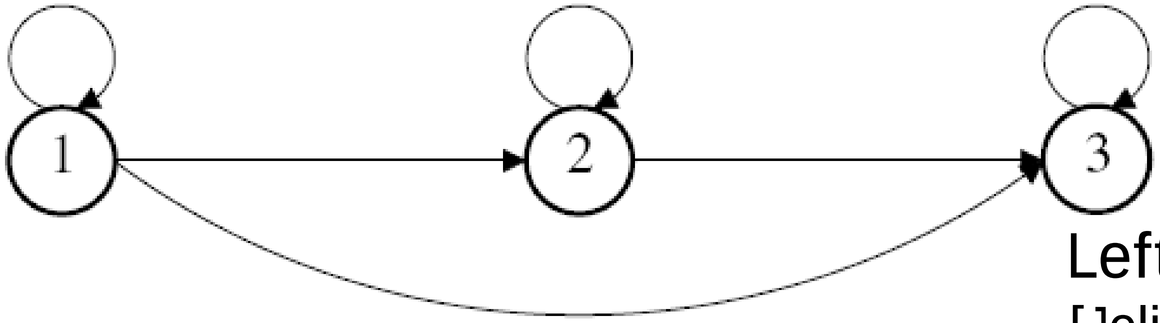
- Typical ad-hoc solutions
    - ergodic HMM (no contraints)
    - left to right HMM (for speech)
    - circular HMM (for shape recognition)



standard ergodic HMM



circular HMM [Arica,Yarman-Vural ICPR00]

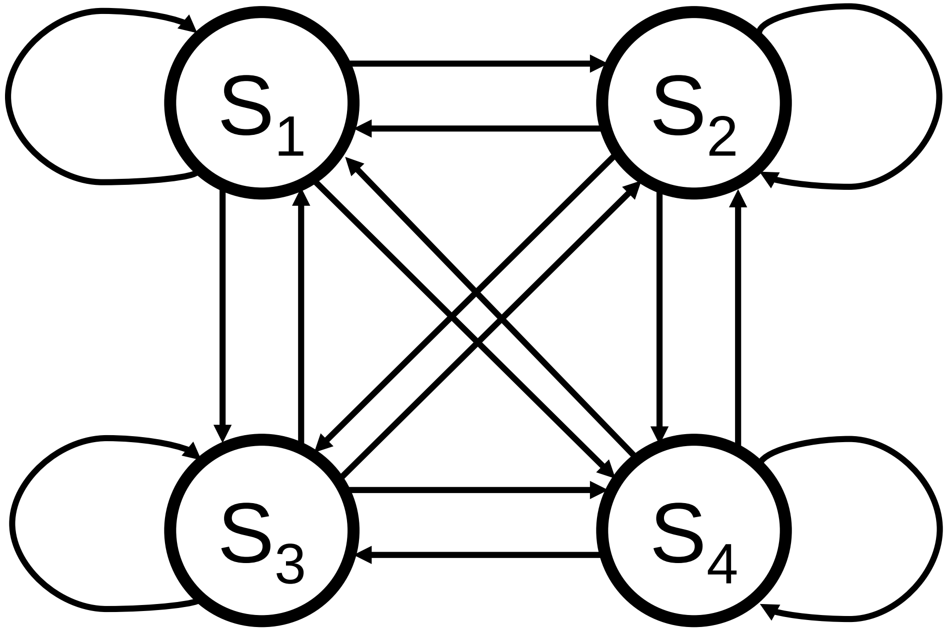


Left to right HMM
[Jelinek, Proc. IEEE 1976]

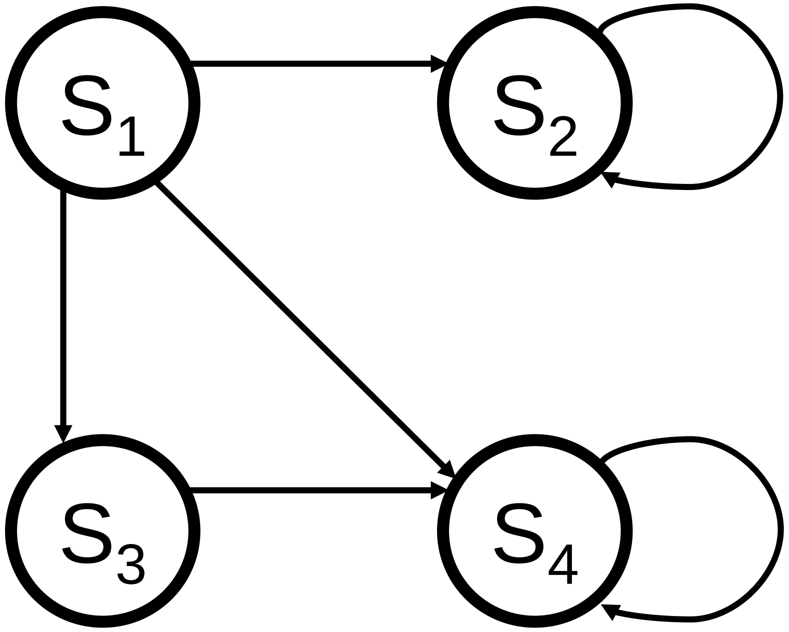
One data-driven solution

[Bicego, Cristani, Murino, ICIAP07]

Sparse HMM: a HMM with a sparse topology (irrelevant or redundant components are *exactly* 0)



Fully connected model: all transitions are present



Sparse model: many transition probabilities are zero (no connections)

Sparse HMM

Sparseness is highly desirable:

- It produces a structural simplification of the model, disregarding unimportant parameters
- A sparse model distills the information of all the training data providing only high representative parameters.
- Sparseness is related to generalization ability (Support Vector Machines)

Some open issues/research trends

1. Model selection
   - how many states?
   - which topology?

2. Extending standard models
   - modifying dependencies or components

3. Injecting discriminative skills into HMM

Extending standard models (1)

First extension:

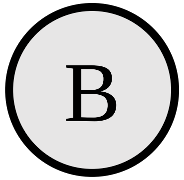
adding novel dependencies between components, in order to model different behaviours

Examples:

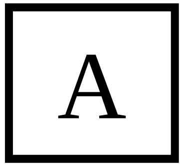
- o Input/Output HMM
- o Factorial HMM
- o Coupled HMM
- o ...

Preliminary note: the Bayesian Network formalism

Bayes Net: graph where nodes represent variables and edges represent causality



hidden variable



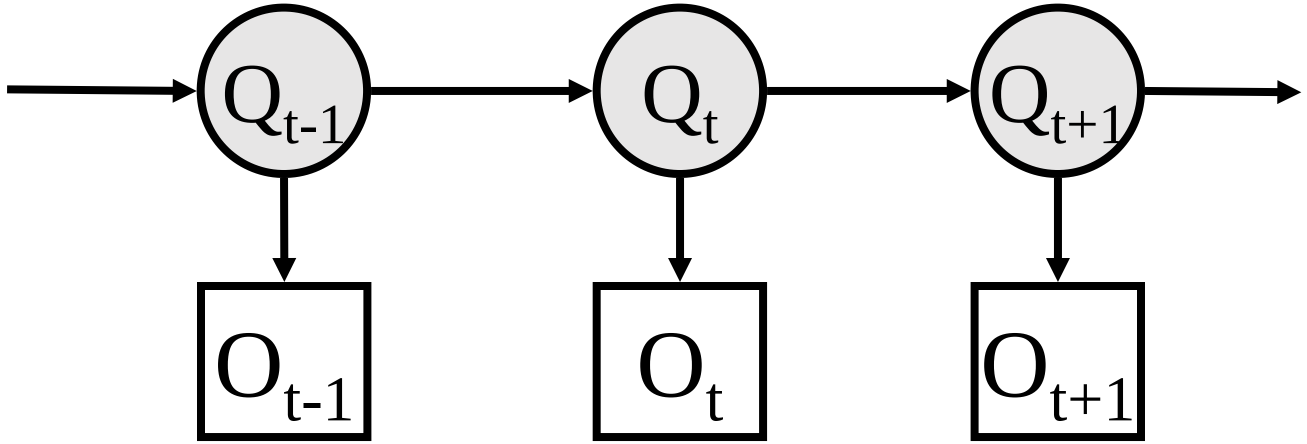
observable variable

causal dependency

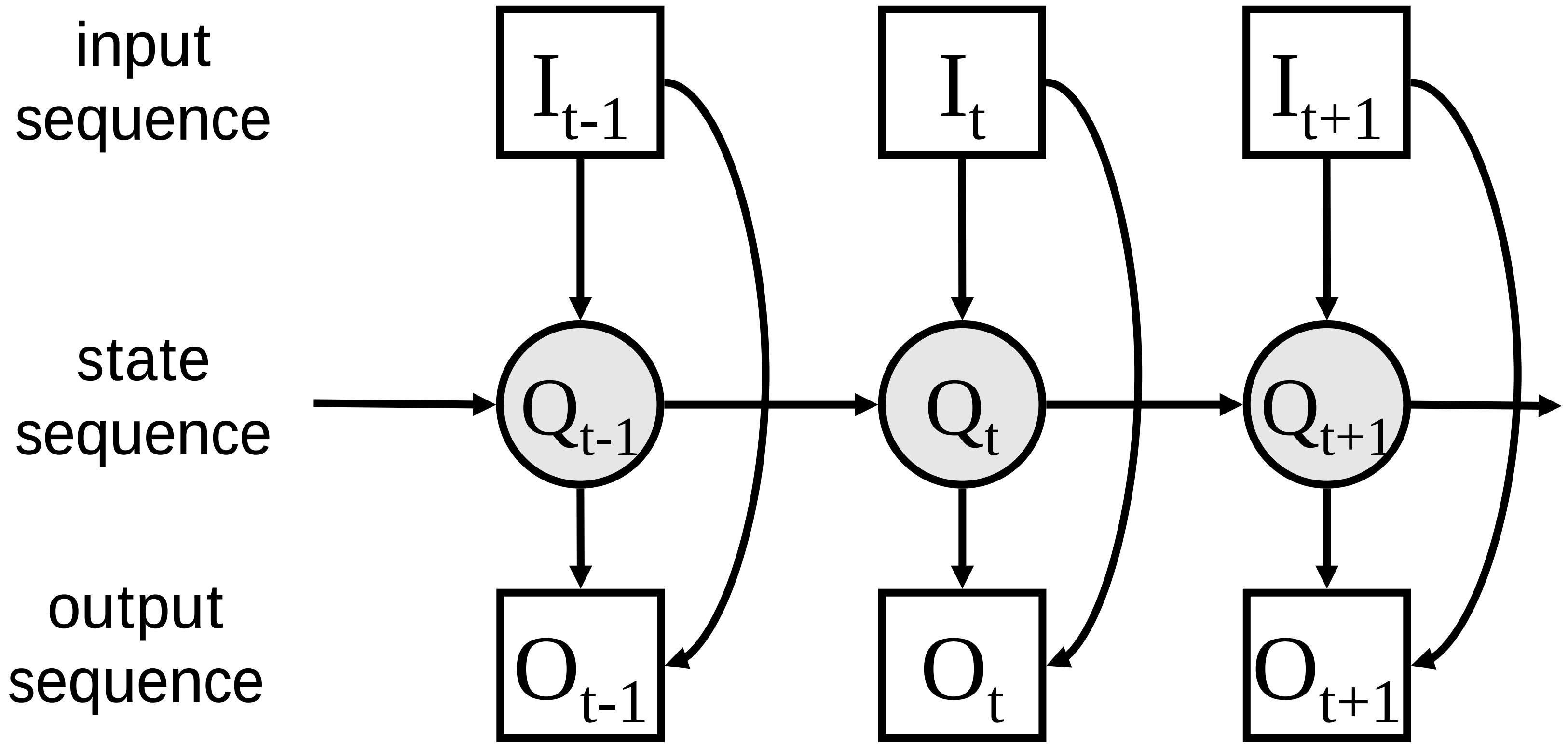
EX.: HMM

state sequence

output sequence

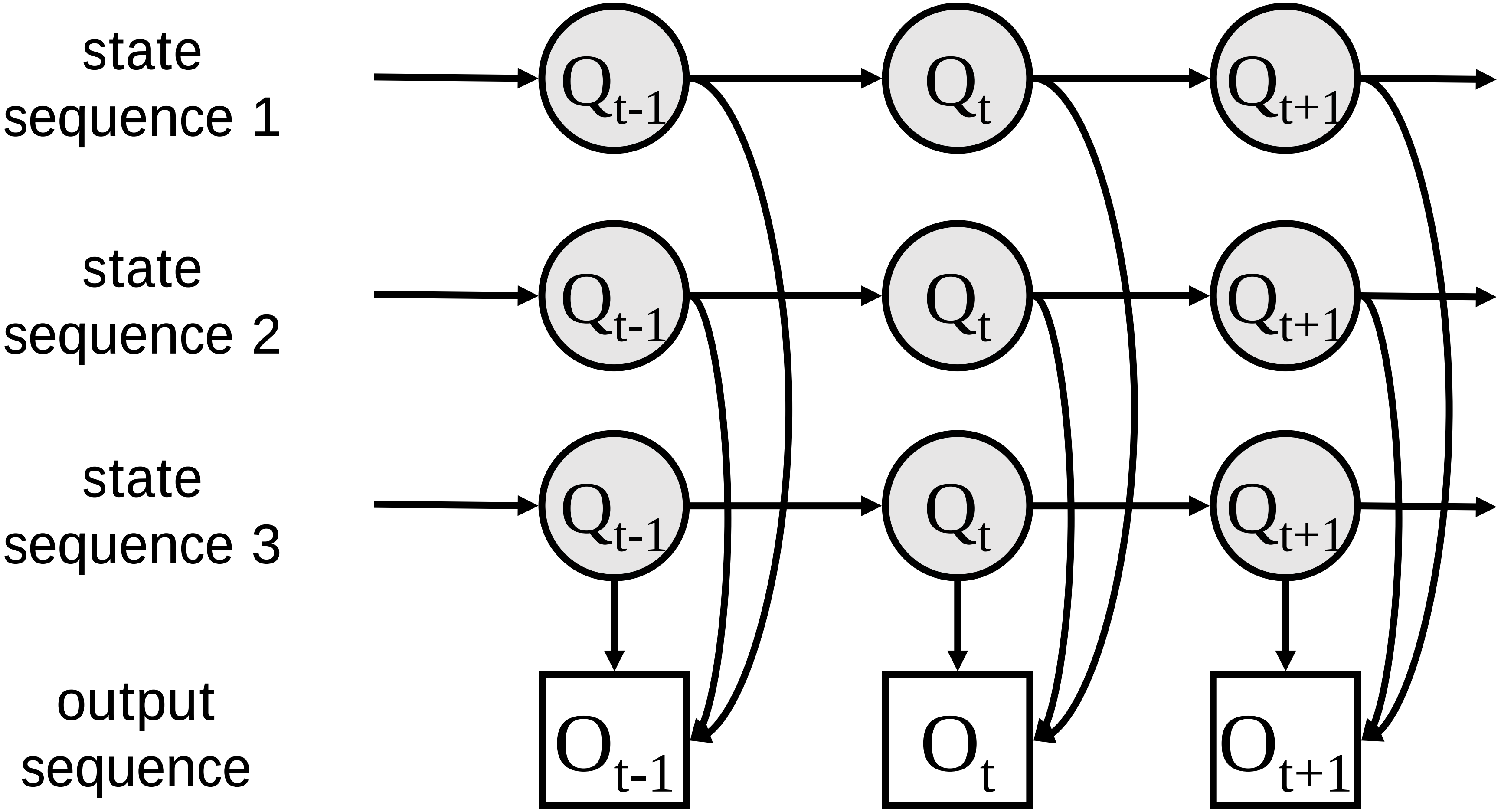


Input-Output HMM: HMM where transitions and emissions are conditional on another sequence (the input sequence)



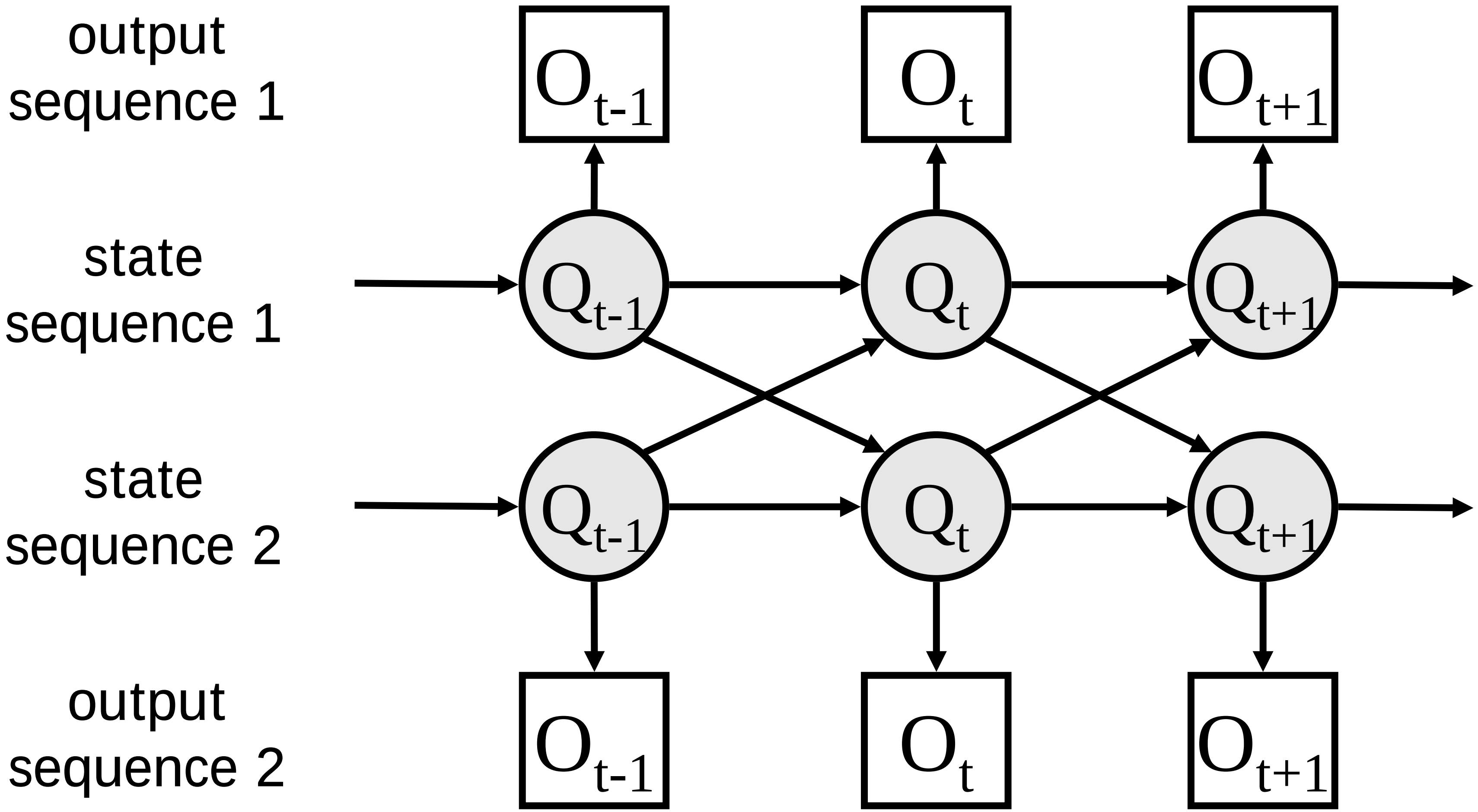
EX.: finance, the input sequence is a leading market index

Factorial HMM: more than one state-chain influencing the output



Ex.: speech recognition, where time series generated from several independent sources.

Coupled HMMs: two interacting HMMs



Ex.: video surveillance, for modelling complex actions like interacting processes

Extending standard models (2)

Second extension:

- employing as emission probabilities (namely functions modelling output symbols) complex and effective techniques (classifier, distributions,...)

Examples:

- Neural Networks
  [Bourlard, Wellekens, TPAMI 90],...
- Another HMM (to compose Hierarchical HMMs)
  [Fine, Singer, Tishby, ML 98] [Bicego, Grosso, Tistarelli, IVC 09]
- Kernel Machines, such as SVM
- Factor analysis
  [Rosti, Gales, ICASSP 02]
- Generalized Gaussian Distributions
  [Bicego, Gonzalez-Jimenez, Alba-Castro, Grosso, ICPR 08]
- ...

Extending standard models (2)

- Problems to be faced:
  - o full integration of the training of each technique inside the HMM framework
    - "naive" solution: segment data and train separately emissions and other parameters
    - challenging solution: simultaneous training of all parameters
  - o in case of Neural Networks or Kernel Machines, it is needed to cast the output of the classifier into a probability value

HMM application

2D shape classification

2D shape classification

- Addressed topic in Computer Vision, often basic for three dimensional object recognition
- Fundamental: contour representation
  - Fourier Descriptor
  - chain code
  - curvature based techniques
  - invariants
  - auto-regressive coefficients
  - Hough-based transforms
  - associative memories

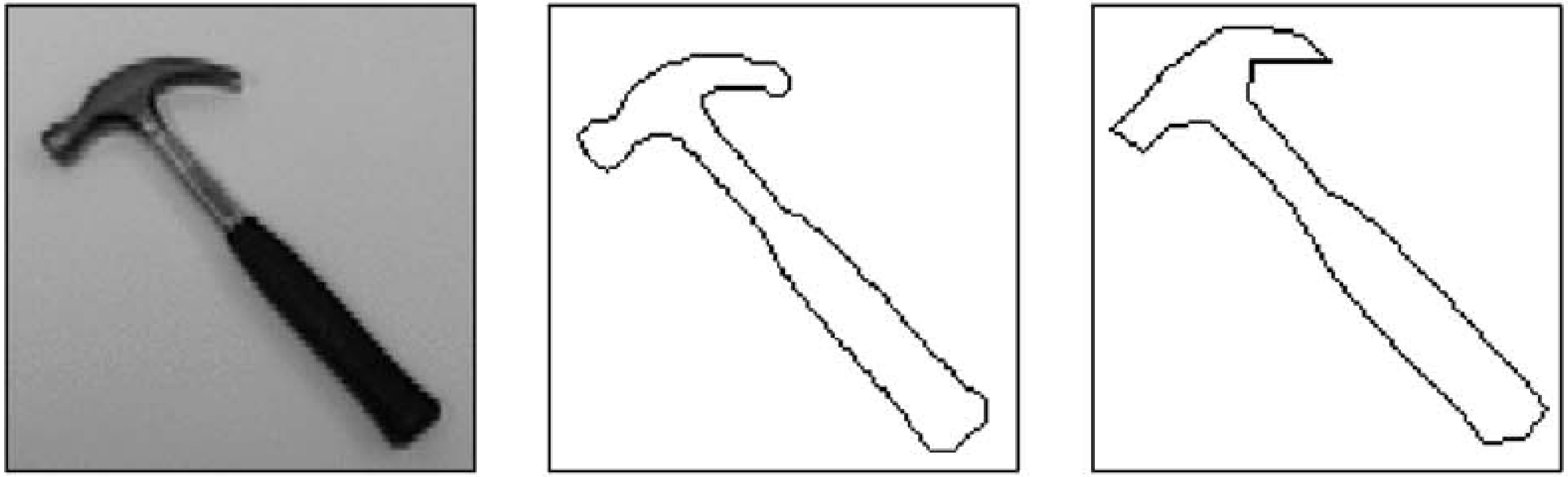
Motivations

- The use of HMM for shape analysis is very poorly addressed
- Previous works:
  - He Kundu (PAMI - 91) using AR coefficients
  - Fred Marques Jorge 1997 (ICIP 97) using chain code
  - Arica Yarman Vural (ICPR 2000) using circular HMM
- Very low entity occlusion
- Closed contours
- Noise sensitivity not analysed

Objectives

- Investigate the capability of HMM in discriminating object classes, with respect to object translation, rotation, occlusion, noise, and affine projections.
- We use curvature representation for object contour.
- No assumption about HMM topologies or closeness of boundaries.

Curvature representation



Curvature representation

- Advantages
  - invariant to object translation
  - rotation of object is equal to phase translation of the curvature signal;
  - can be calculated for open contours

- Disadvantages
  - noise sensitivity

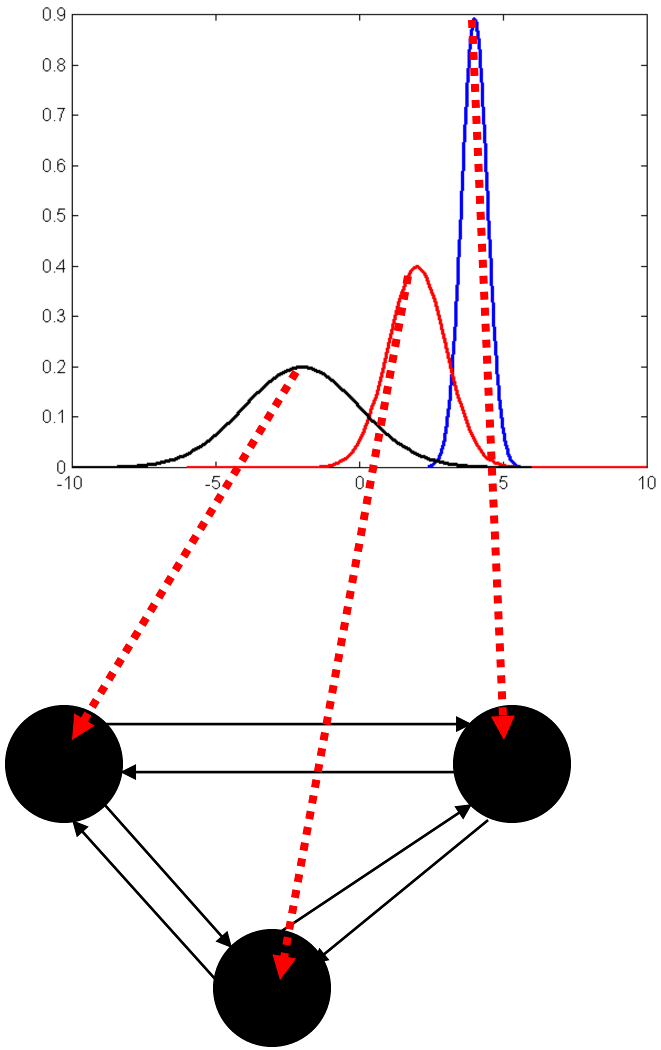
Hidden Markov Model

- Use of Continuous Hidden Markov Model: the emission probability of each state is a Gaussian distribution

- Crucial Issues:
  - Initialisation of training algorithm
  - Model Selection

HMM Initialisation

- Gaussian Mixture Model clustering of the curvature coefficients: each cluster centroid is used for initialising the parameters of each state.



HMM model selection

- Bayesian Information Criterion on the initialization
  - 1 HMM model per shape
  - Using BIC on the Gaussian mixture model clustering in order to choose the optimal number of states
  - Advantage: only one HMM training session

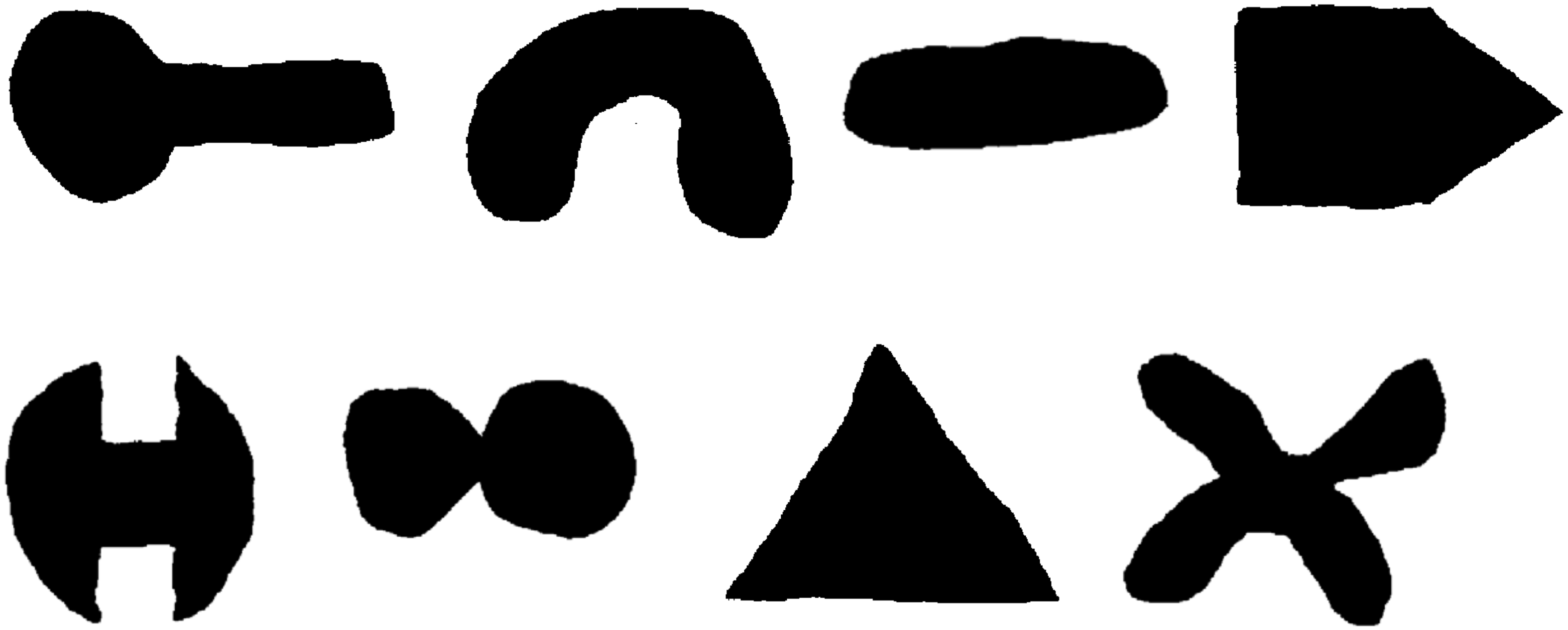
Strategy

- Training: for any object we perform these steps
  - extract edges with Canny edge detector
  - calculate the related curvature signature;
  - train an HMM on it:
    - the HMM was initialised with GMM clustering;
    - the number of HMM states is estimated using the BIC criterion;
    - each HMM was trained using Baum-Welch algorithm
  - at the end of training session we have one HMM λ_i for each object.

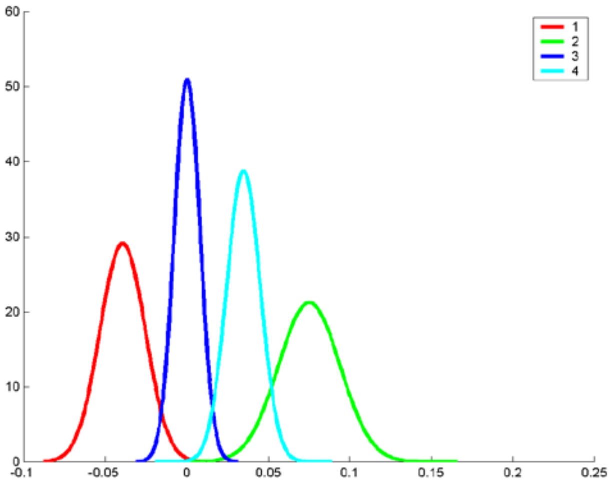
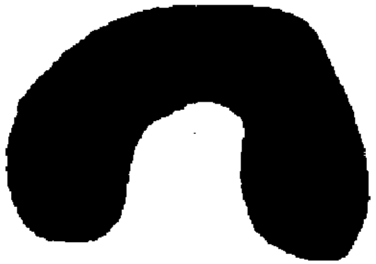
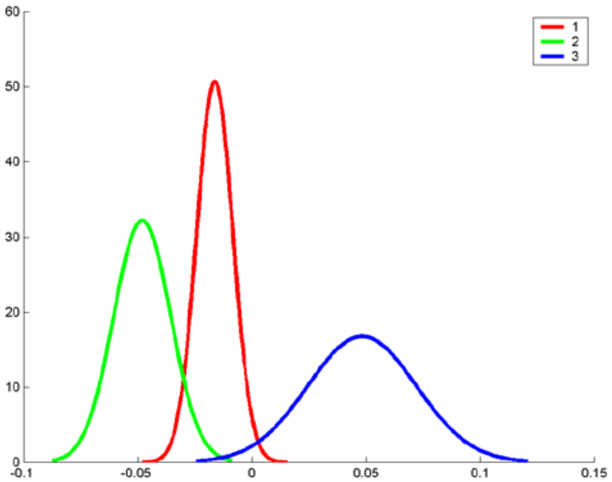
Strategy (cont.)

- Classification: given an unknown sequence O
  - compute, for each model λ_i, the probability $P(O|\lambda_i)$ of generating the sequence O
  - classify O as belonging to the class whose model shows the highest probability $P(O|\lambda_i)$.

Experimental: The test set

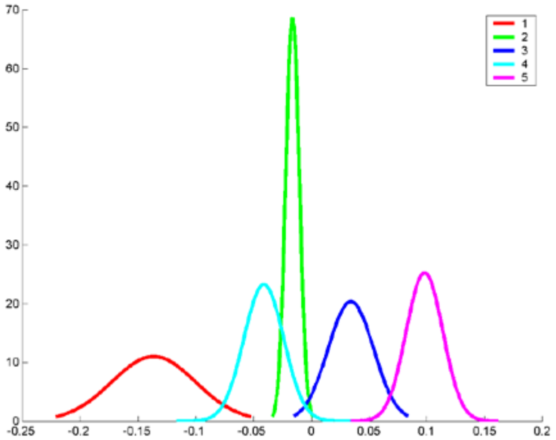


The models

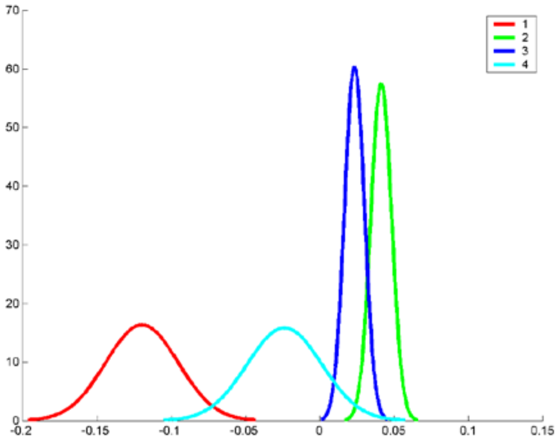
<table>
<tr><th>Shape</th><th>Emission Probability</th><th>Transition Probability</th></tr>
<tr><td></td><td>
</td><td>
<table>
<tr><td>0.94</td><td>0.00</td><td>0.06</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.96</td><td>0.00</td><td>0.04</td></tr>
<tr><td>0.02</td><td>0.00</td><td>0.96</td><td>0.02</td></tr>
<tr><td>0.00</td><td>0.02</td><td>0.02</td><td>0.96</td></tr>
</table></td></tr>
<tr><td></td><td>
</td><td>
<table>
<tr><td>0.98</td><td>0.01</td><td>0.01</td></tr>
<tr><td>0.03</td><td>0.97</td><td>0.00</td></tr>
<tr><td>0.02</td><td>0.00</td><td>0.98</td></tr>
</table></td></tr>
</table>

The models (2)

Shape	Emission Probability	Transition Probability



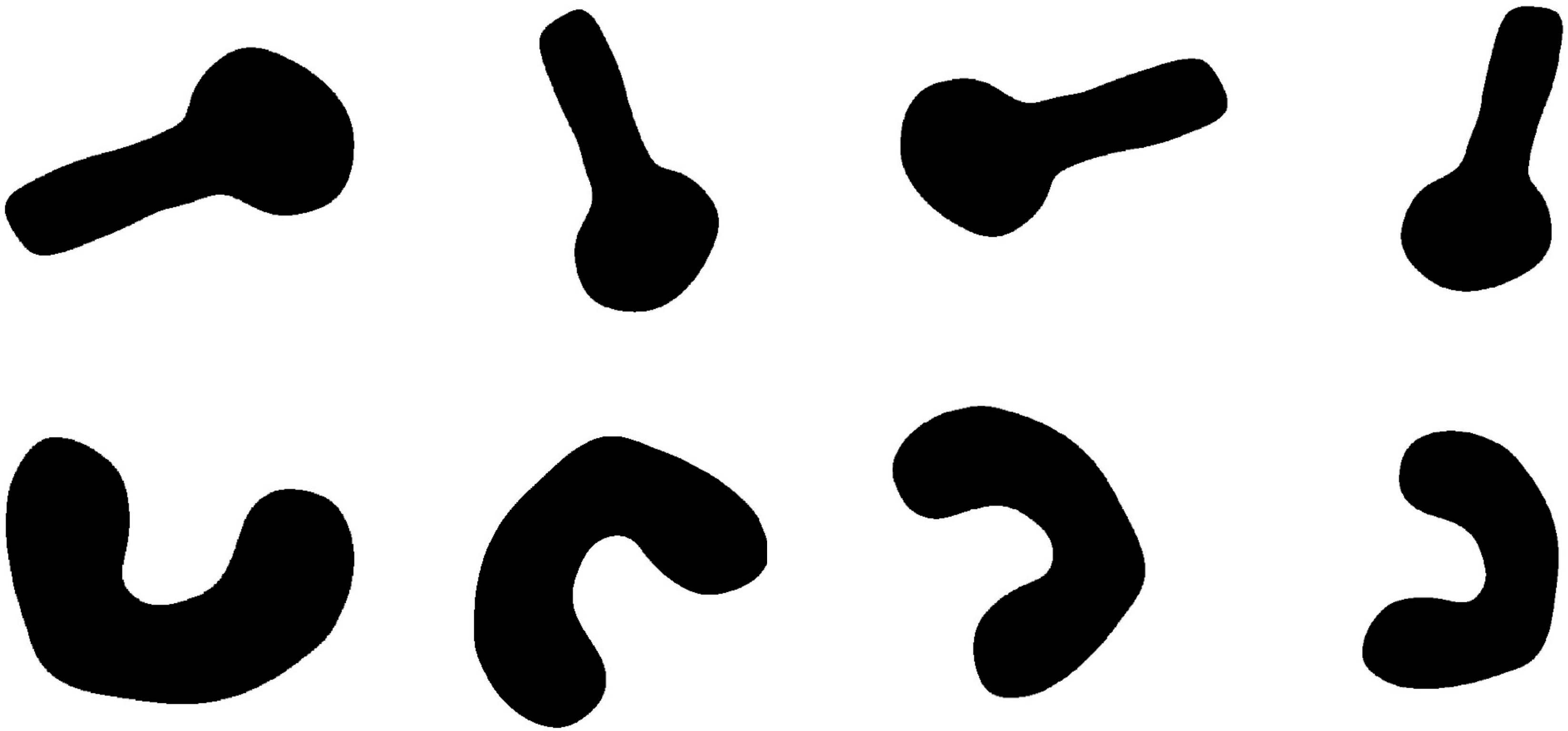
0.92	0.00	0.00	0.08	0.00
0.00	0.97	0.01	0.02	0.00
0.00	0.00	0.89	0.04	0.07
0.09	0.11	0.00	0.80	0.00
0.00	0.00	0.09	0.00	0.91



0.91	0.00	0.00	0.09
0.00	0.95	0.05	0.00
0.00	0.06	0.92	0.02
0.08	0.00	0.08	0.83

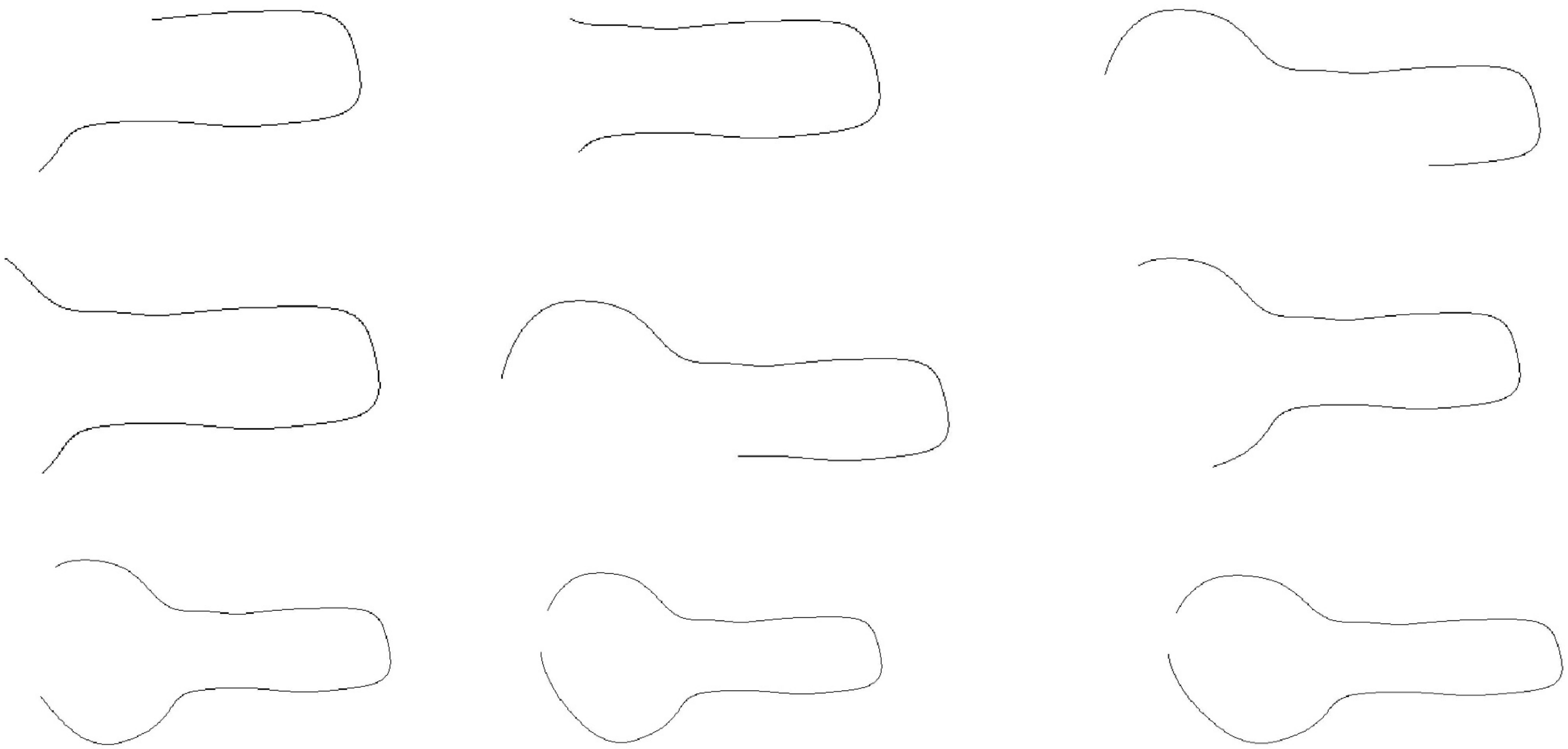
Rotations

- Test set is obtained by rotating 10 times each object by a random angle from 0 to 2π.
- Results: Accuracy 100%



Occlusions

- Each object is occluded: occlusion vary from 5% to 50% (only an half of the whole object is visible)



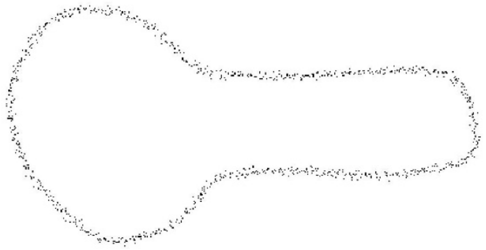
Occlusions: results

Occlusion percentage level	Classification Accuracy
5%	100%
10%	100%
15%	100%
20%	100%
25%	100%
30%	100%
35%	100%
40%	97.5%
45%	96.25%
50%	95%

Noise

- A Gaussian Noise (with mean 0 and variance σ^2) is added to the X Y coordinates of the object

- σ^2 varies from 1 to 5: Accuracy 100%. The gaussian filter applied before calculating the curvature is able to remove completely this kind of noise



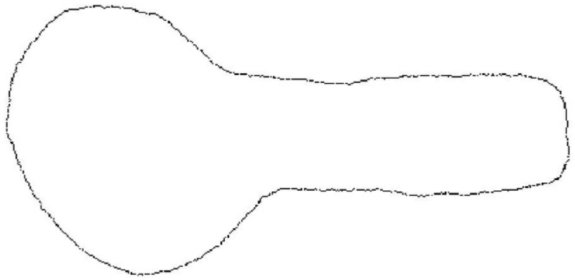
$\sigma^2=1$

$\sigma^2=4$

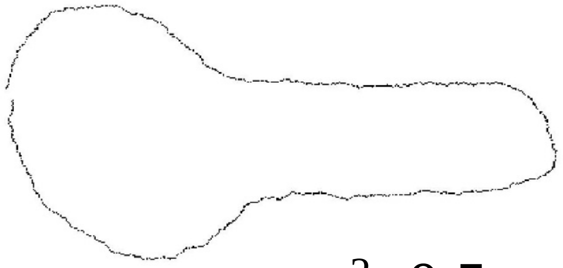
$\sigma^2=5$

Alternative Noise

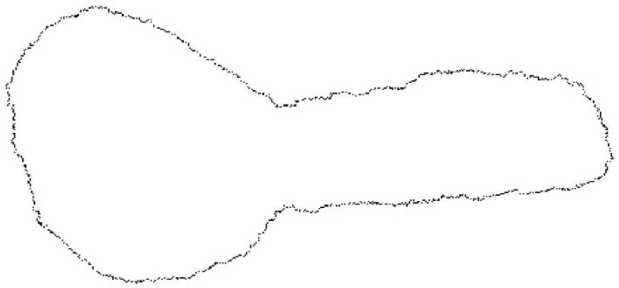
- Adding noise to the first derivative



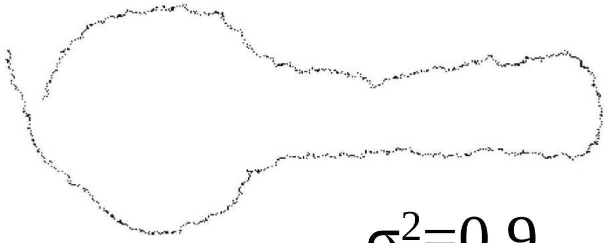
$\sigma^2=0.3$



$\sigma^2=0.5$



$\sigma^2=0.7$



$\sigma^2=0.9$

Noise: results

Noise variance σ^2	Classification Accuracy
0.1	100.00%
0.3	97.50%
0.5	88.75%
0.7	82.50%
0.9	73.75%

Occlusions and Rotations: results

Occlusion percentage level	Classification Accuracy
5%	100%
10%	100%
15%	100%
20%	100%
25%	96.25%
30%	96.25%
35%	95%
40%	91.25%
45%	85%
50%	87.5%

Occlusions, Rotations and Noise: Results

Occlusion Percentage level	Classification Accuracy		
	Noise σ^2=0.1	Noise σ^2=0.3	Noise σ^2=0.5
50%	86.25%	83.75%	75.00%
40%	93.75%	87.50%	77.50%
30%	98.75%	90.00%	80.00%
20%	98.75%	93.75%	80.00%
10%	100.00%	97.50%	87.50%

Slant and Tilt Projections

Angoli proiezione	Tilt = 10	Tilt = 20	Tilt = 30	Tilt = 40	Tilt = 50
Slant = 10					
Slant = 20					
Slant = 30					
Slant = 40					
Slant = 50					

Slant and Tilt Projections: results

Angoli proiezione	Tilt = 10	Tilt = 20	Tilt = 30	Tilt = 40	Tilt = 50
Slant = 10	8/8	8/8	8/8	7/8	4/8
Slant = 20	8/8	8/8	8/8	7/8	4/8
Slant = 30	8/8	8/8	8/8	7/8	4/8
Slant = 40	8/8	8/8	7/8	5/8	4/8
Slant = 50	8/8	8/8	6/8	4/8	4/8

Conclusions

- System is able to recognize object that could be translated, rotated and occluded, also in presence of noise.
- Translation invariance: due to Curvature
- Rotation invariance: due to Curvature and HMM
- Occlusion invariance: due to HMM
- Robustness to noise: due to HMM

HMM application

Video Analysis

Use of the HMMs: main idea

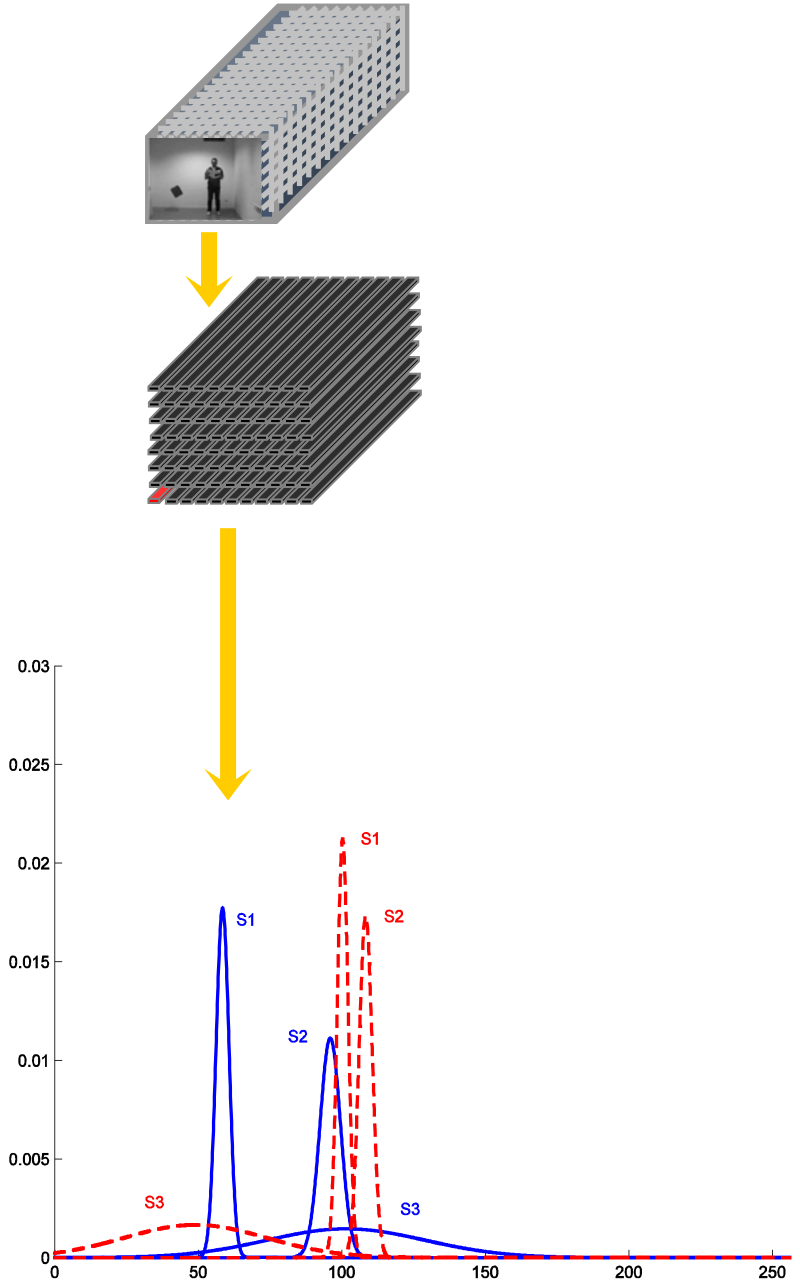
- Each pixel (signal) *v* of the sequence is modeled with an HMM $\lambda_v=(A,B,\pi)$

- B = $\{\mu_i,\sigma_i^2\}$ represents gray level ranges assumed by the v-th pixel signal, and

$$b_i(O_v) = N(O_v;\mu_i,\sigma_i^2)$$

- The larger the σ_i^2, the more irregular the corresponding signal

- A := Markov chain that mirrors the evolution of the gray levels



The idea

- Define the distances between locations on the basis of the distances between the trained Hidden Markov Models
- The segmentation process is obtained using a spatial clustering of HMMs
- We need to define a similarity measure
  - decide when a group (at least, a couple) of neighboring pixels must be labelled as belonging to the same region
- Using this measure the segmentation is obtained as a standard region growing algorithm

The similarity measure

- The used similarity measure is:

$$D(i,j) = \frac{1}{2}\left\{\frac{L_{ij} - L_{jj}}{L_{jj}} + \frac{L_{ji} - L_{ii}}{L_{ii}}\right\}$$

where

$$L_{ij} = P(O_i \mid \lambda_j)$$

- We use a similar distance, *more robust*, which weighs more the states in which the model stands more time

Results (real)

[Corridoio.avi]

Image based segmentation

HMM based segmentation