Università di Verona

A.A. 2020-21

Machine Learning & Artificial Intelligence

HMM
Modelli di Markov a stati nascosti

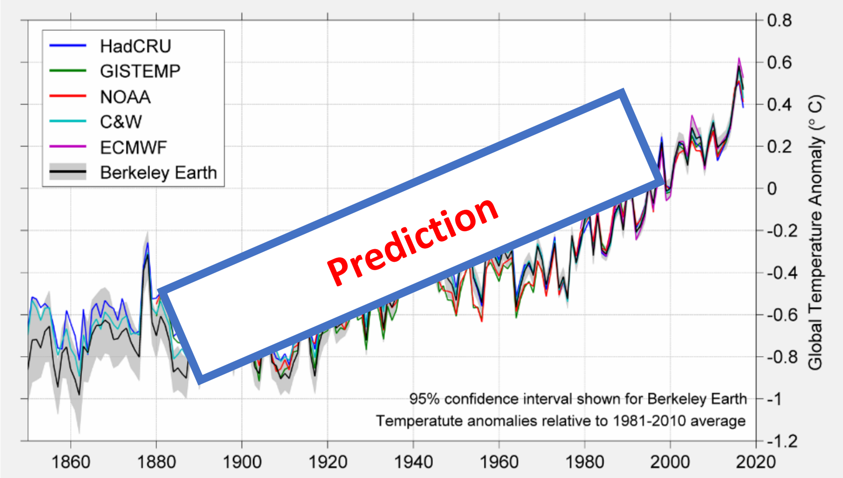
Vittorio Murino

Sommario

1. Processi e modelli di Markov;
2. Processi e modelli di Markov a stati nascosti (Hidden Markov Model, HMM);
3. Attività di ricerca e applicazioni su HMM;

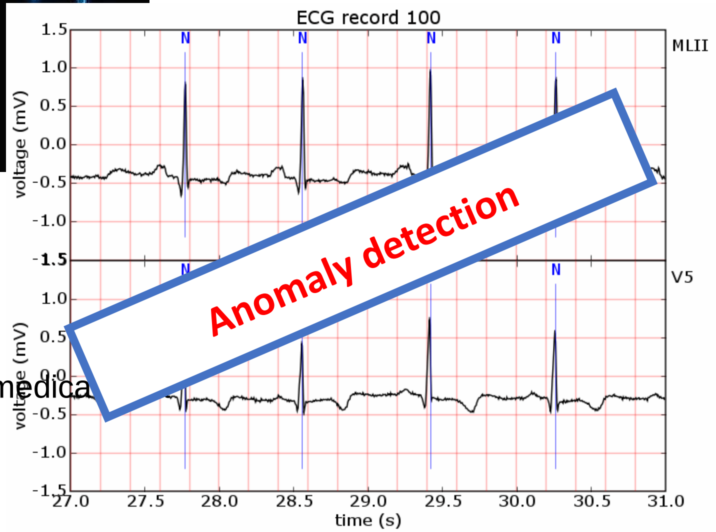
Time series analysis

Financial series



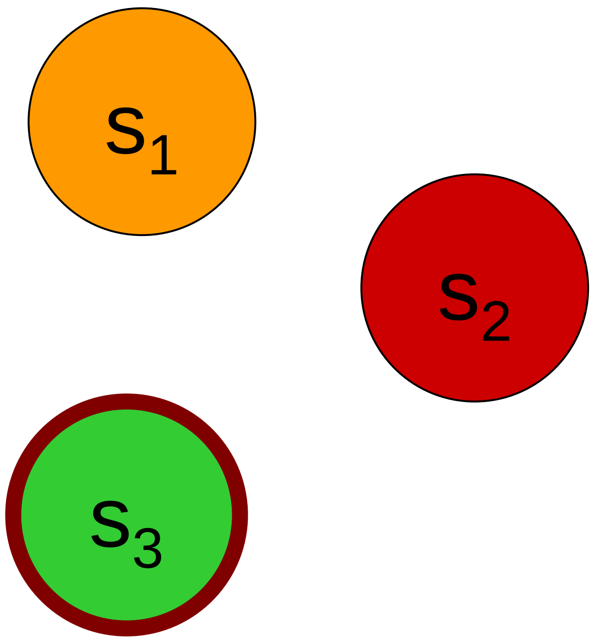
Audio signals

Gestures



Biomedical

Processo di Markov (ordine 1)



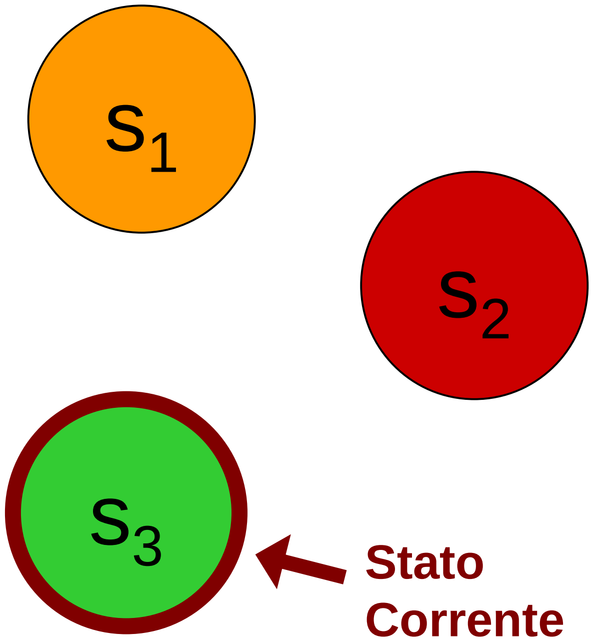
N=3
t=1

- Ha N stati , $s_1, s_2, \ldots, s_N$
- È caratterizzato da passi discreti, t=1,t=2,…
- La probabilità di partire da un determinato stato è dettata dalla distribuzione:

$$\Pi = \{\pi_i\} : \quad \pi_i = P(q_1 = s_i) \text{ con}$$

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

Processo di Markov

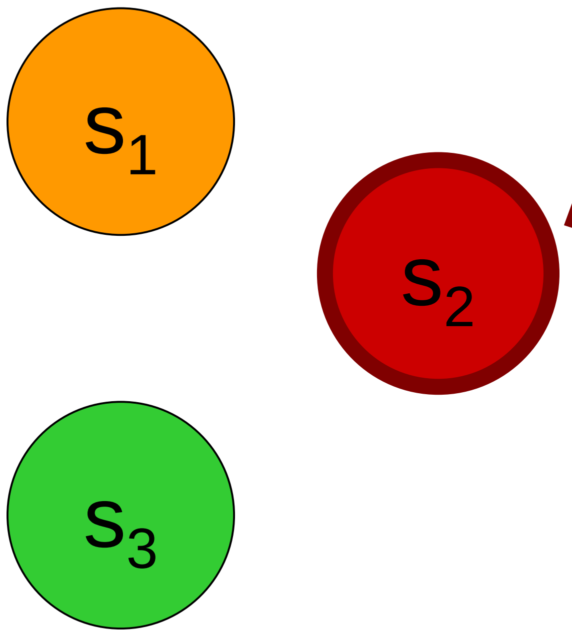


N=3
t=1

$q_1 = s_3$

- Al t-esimo istante il processo si trova esattamente in uno degli stati a disposizione, indicato dalla variabile q_t
- Nota: $q_t \in \{s_1, s_2, \ldots, s_N\}$
- Ad ogni iterazione, lo stato successivo viene scelto con una determinata probabilità

Processo di Markov



Stato Corrente

N=3 t=2

$q_2 = s_2$

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

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

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

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

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

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

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

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

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

Stato Corrente

- Tale probabilità è unicamente determinata dallo stato precedente (*markovianetà di primo ordine*):

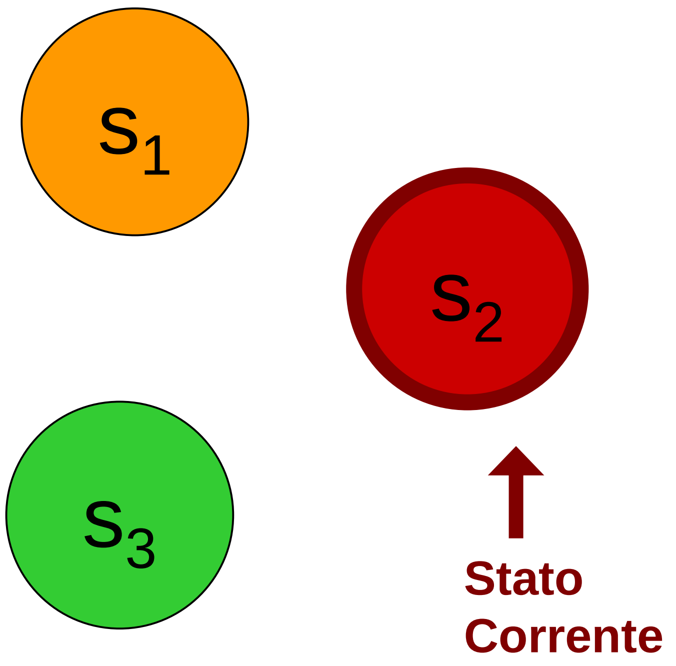
$$P(q_{t+1}= s_j|q_t= s_i,q_{t-1}=s_k,\ldots,q_1=s_l) = P(q_{t+1}= s_j|q_t= s_i)$$

Ipotesi di Markov

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

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

Processo di Markov



N=3 t=1

$q_2 = s_2$

- Definendo:

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

ottengo la matrice NxN

A di *transizione tra stati*, *invariante nel tempo*:

A=

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,1}$	$a_{3,1}$	$a_{3,3}$

Markov Models

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

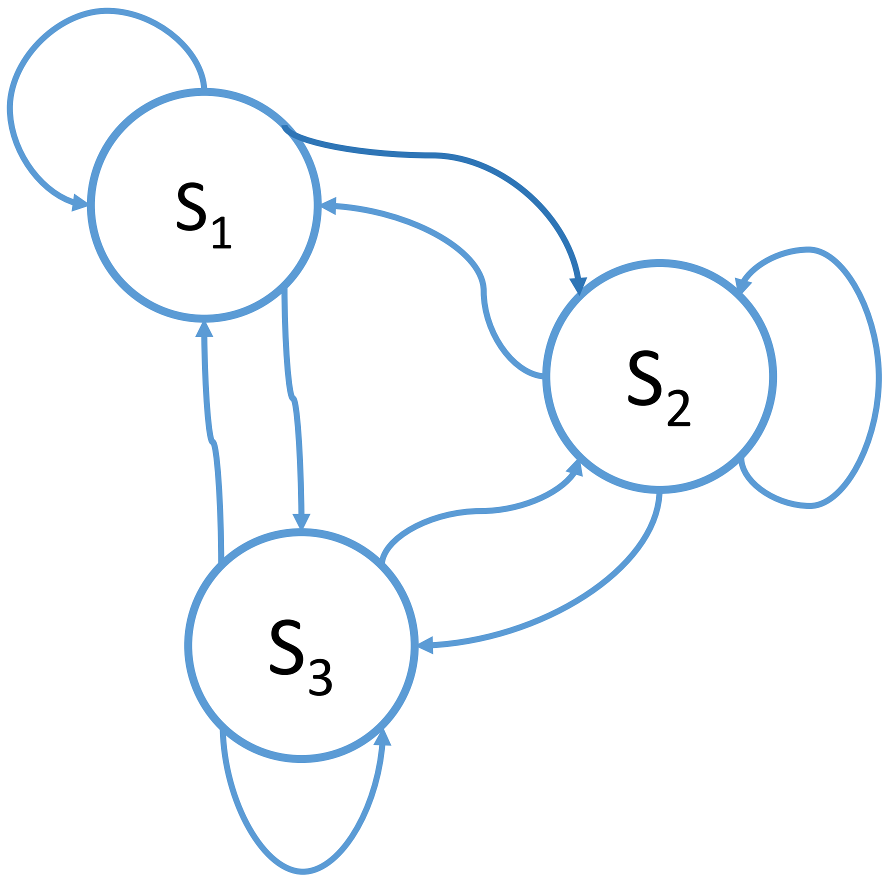
- A sequence of states $\mathcal{Q} = \{q_1, q_2, \dots, q_T\}$

- A **transition probability matrix**

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

- An **initial probability distribution** over states

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



Markov Models

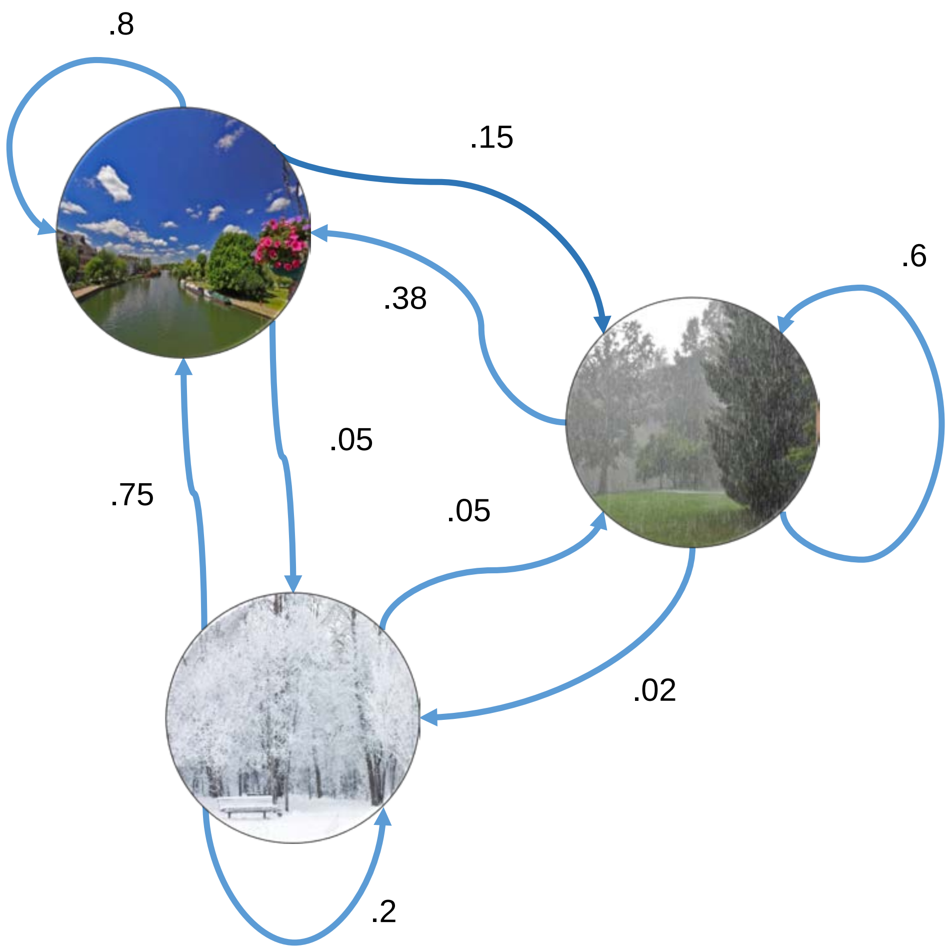
- **States:** $\{sunny, rainy, snowy\}$

- **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:

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

$$P = 0.0001512$$

Caratteristiche dei processi Markoviani

- Sono processi (discreti) caratterizzati da:
  - Markovianità del primo ordine
  - stazionarietà
  - aventi una distribuzione iniziale

- Conoscendo le caratteristiche di cui sopra, si può esibire un ***modello (probabilistico) di Markov (MM)*** come

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

Cosa serve un modello stocastico o probabilistico?

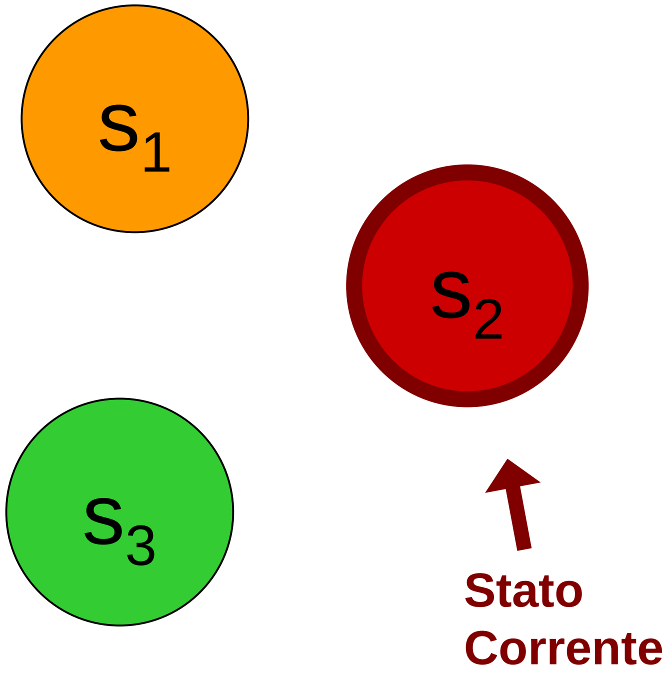
- Modella e riproduce **processi stocastici**
- Descrive tramite probabilità **le *cause* che portano da uno stato all'altro del sistema**
- In altre parole, più è probabile che dallo stato A si passi allo stato B, più è probabile che **A causi B**

Che operazioni si possono eseguire su un modello probabilistico?

- **Addestramento o training**
  - Si stimano gli elementi costituenti del modello

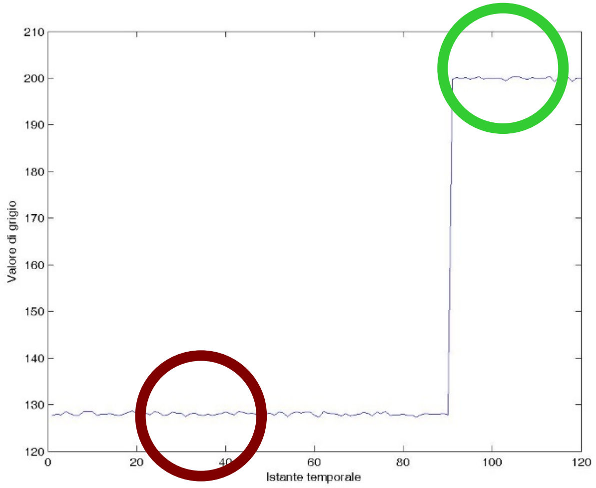
- **Inferenze di vario tipo (interrogo il modello):**
  - Probabilità di una sequenza di stati, dato il modello
  - Proprietà invarianti etc.

Cosa serve un modello di Markov?



- Modella comportamenti *stocastici Markoviani (di ordine N)* di un sistema in cui gli stati sono:
  - **Espliciti** (riesco a dar loro un nome)
  - **Osservabili** (ho delle osservazioni che univocamente identificano lo stato)

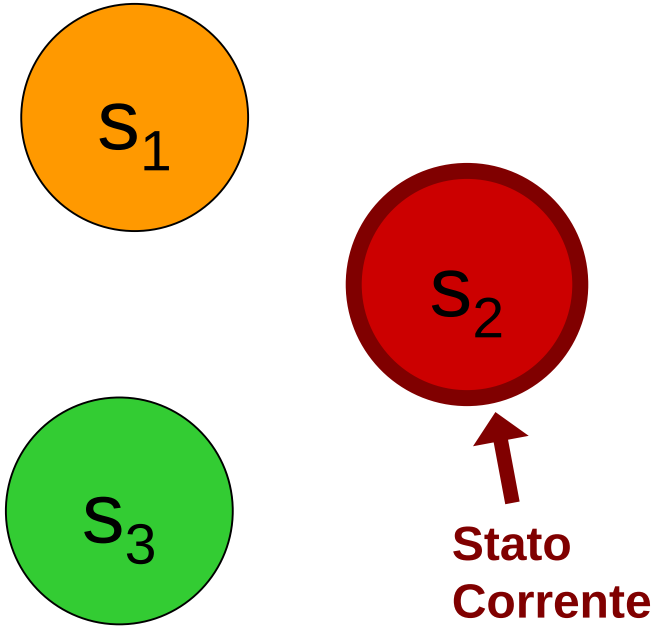
Esempio: Semaforo

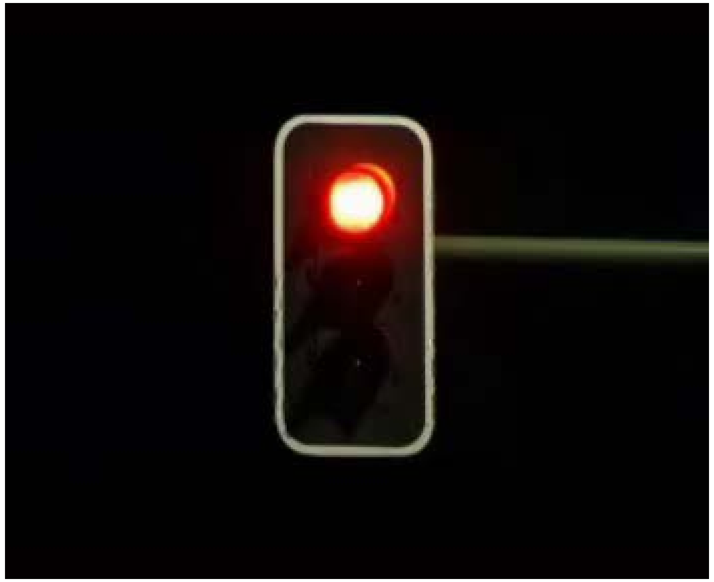


t=1 t=2 …

- È un sistema di cui gli stati sono:
  - **Espliciti** (le diverse lampade accese)
  - **Osservabili** (i colori delle lampade che osservo)

Semaforo – modello addestrato





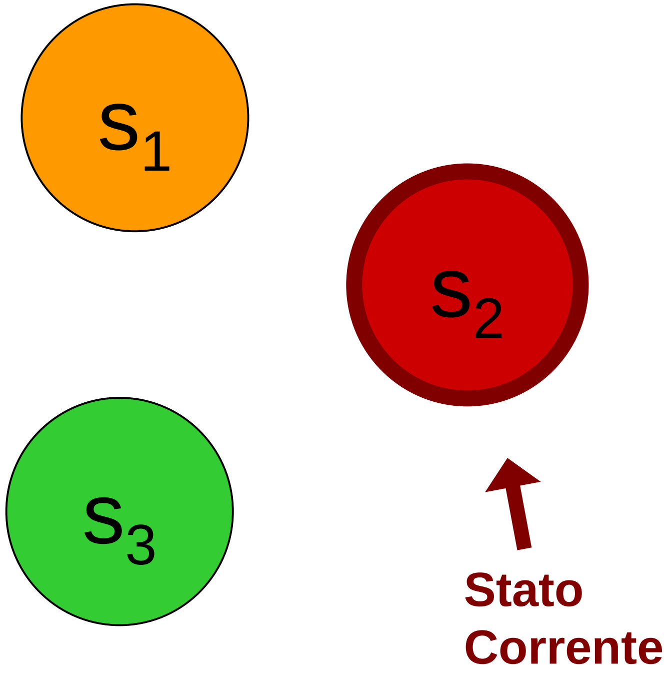
$\pi =$

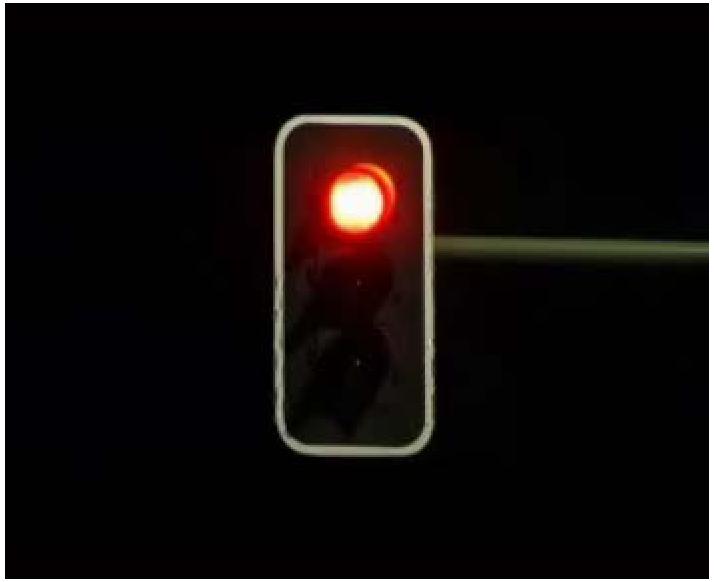
$\pi_1 = 0.33$	$\pi_2 = 0.33$	$\pi_3 = 0.33$

$\boldsymbol{A} =$

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

Semaforo – inferenze





$$O_2 =< q_2 = s_3, q_1 = s_2 >$$

Inferenza:

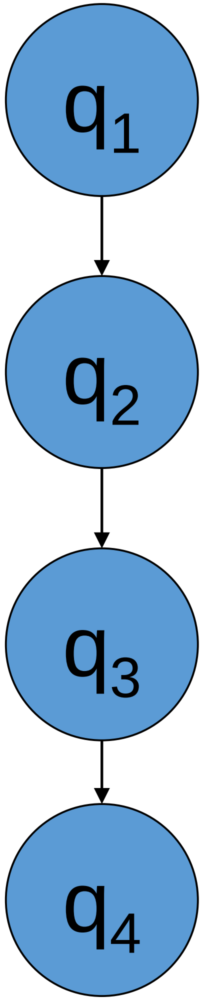
$$P(O|\lambda) = P(O) =$$

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

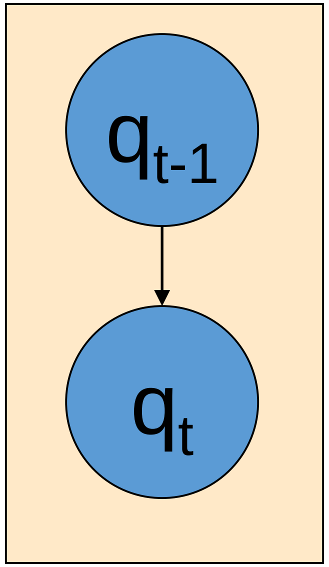
Inferenza importante

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

Modello grafico



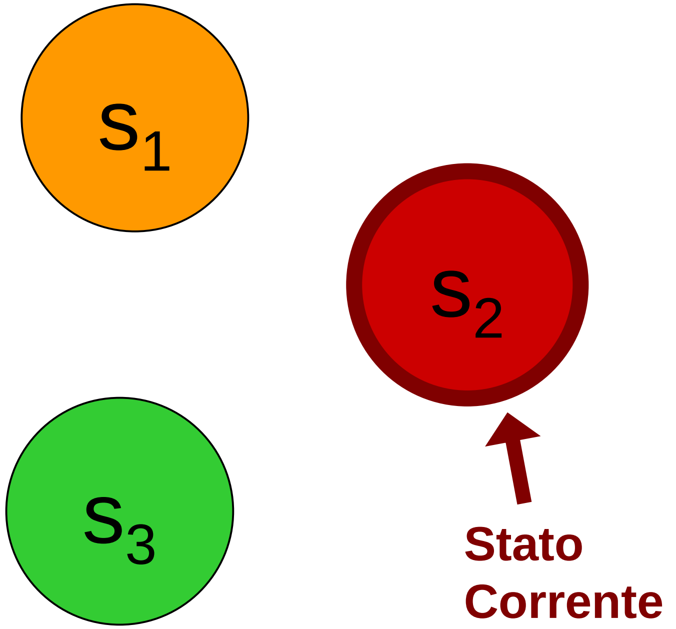
- La struttura grafica di tale probabilità congiunta si scrive in questa forma, dove

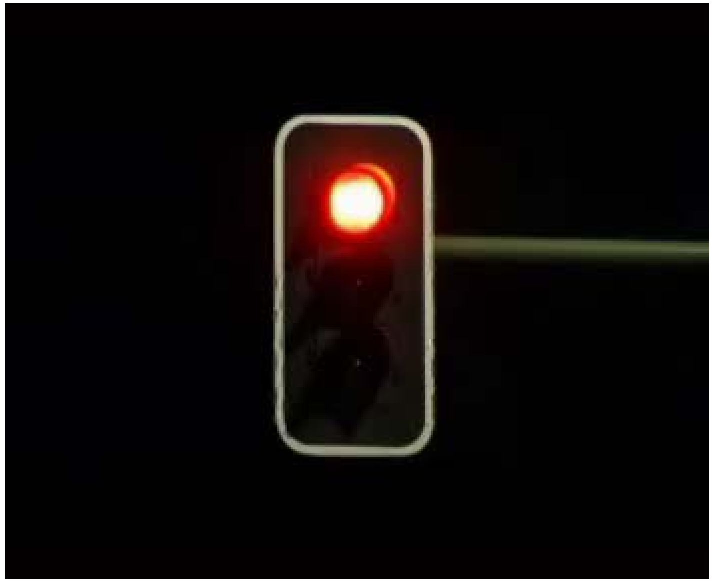


$= P(q_t \mid q_{t-1})$

$\big(= P(q_t \mid q_{t-1},\ldots,q_1)$ ***in questo caso*** $\big)$

Semaforo – inferenze, risposta





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

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

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

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

Seconda inferenza importante

- Calcolo della probabilità $P(q_T = s_j)$
- <u>STEP 1</u>:

  Valuto come calcolare P(Q) per ogni cammino di stati **Q**$=\{q_1, q_2, \ldots, q_T = s_j\}$, ossia

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

- <u>STEP 2</u>:

  Uso questo metodo per calcolare $P(q_T = s_j)$, ossia:

  o $$P(q_T = s_j) = \sum_{\mathbf{Q} \in \text{cammini di lunghezza T che finiscono in } s_j} P(\mathbf{Q})$$

  Calcolo oneroso: ESPONENZIALE in T $(O(N^T))$!

Seconda inferenza importante (2)

- Idea: per ogni stato s_j chiamo
  $p_T(j)$= prob. di trovarsi nello stato s_j al tempo T → $P(q_T = s_j)$;
  - Si può definire induttivamente:

$$\forall i \quad p_1(i) = \begin{cases} 1 & \text{se } s_i \text{ è lo stato in cui mi trovo} \\ 0 & \text{altrimenti} \end{cases}$$

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

Seconda inferenza importante (3)

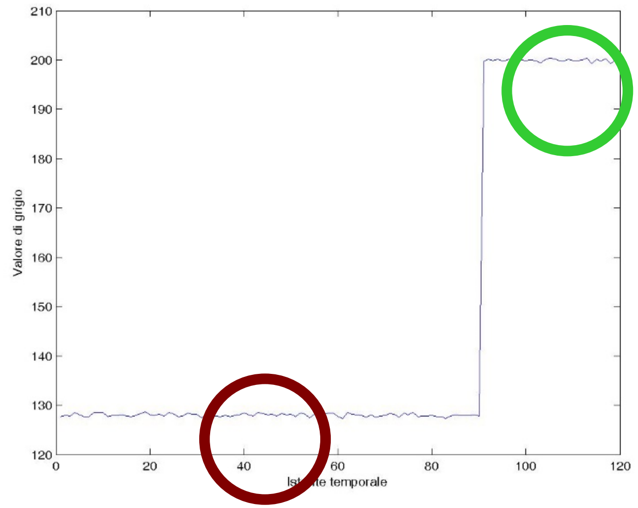
$$p_{t+1}(j) = \sum_{\mathrm{i=1}}^{\mathrm{N}} P(q_{t+1} = s_j, q_t = s_i) =$$

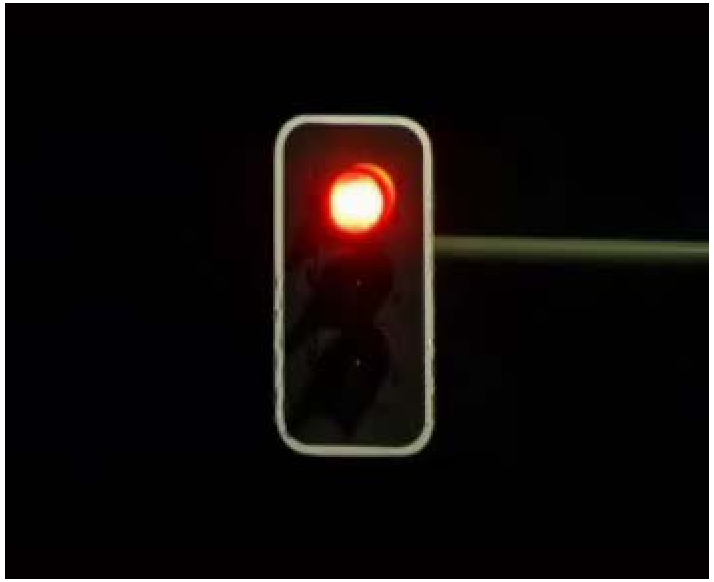
$$= \sum_{\mathrm{i=1}}^{\mathrm{N}} P(q_{t+1} = s_j \mid q_t = s_i) P(q_t = s_i) = \sum_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{a}_{ij}\, p_t(i)$$

- Uso questo metodo partendo da $P(q_T = s_j)$ e procedendo a ritroso
- Il costo della computazione in questo caso è $O(TN^2)$

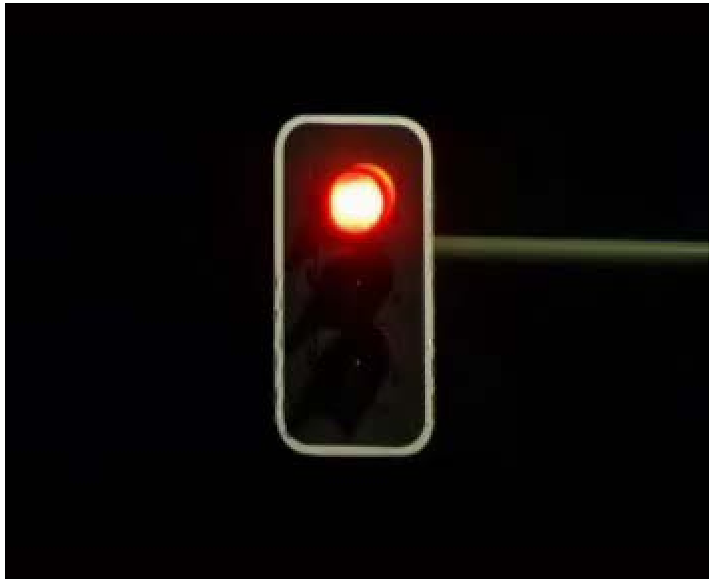
Limiti dei modelli markoviani

1. *Lo stato è sempre* ***osservabile deterministicamente***, tramite le osservazioni (non c’è rumore).

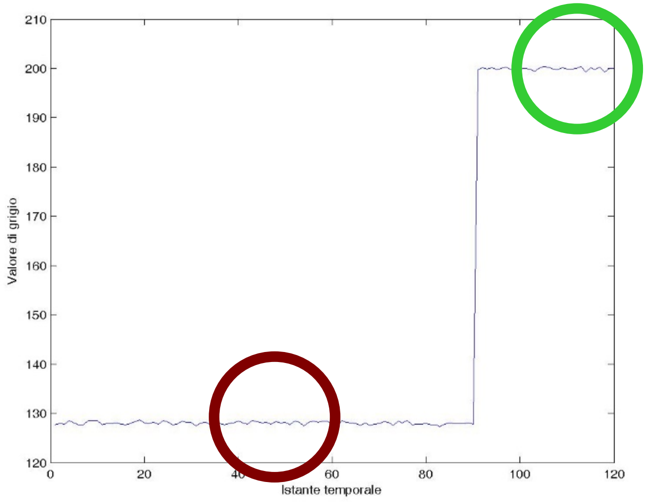




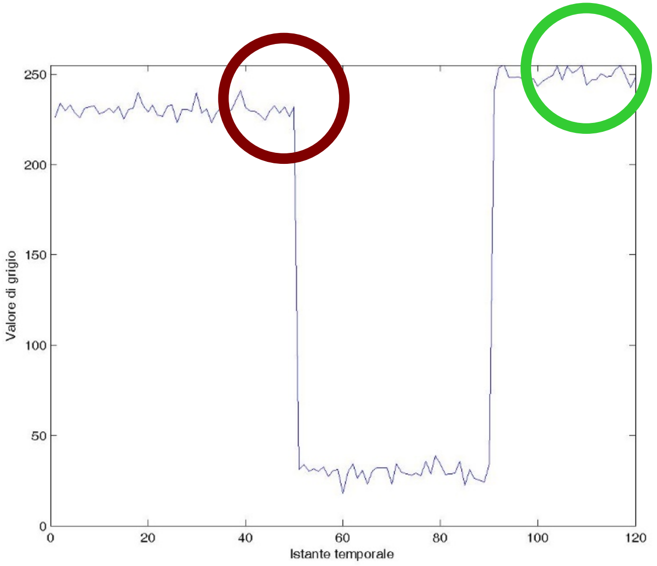
Limiti dei modelli markoviani



OK



NO

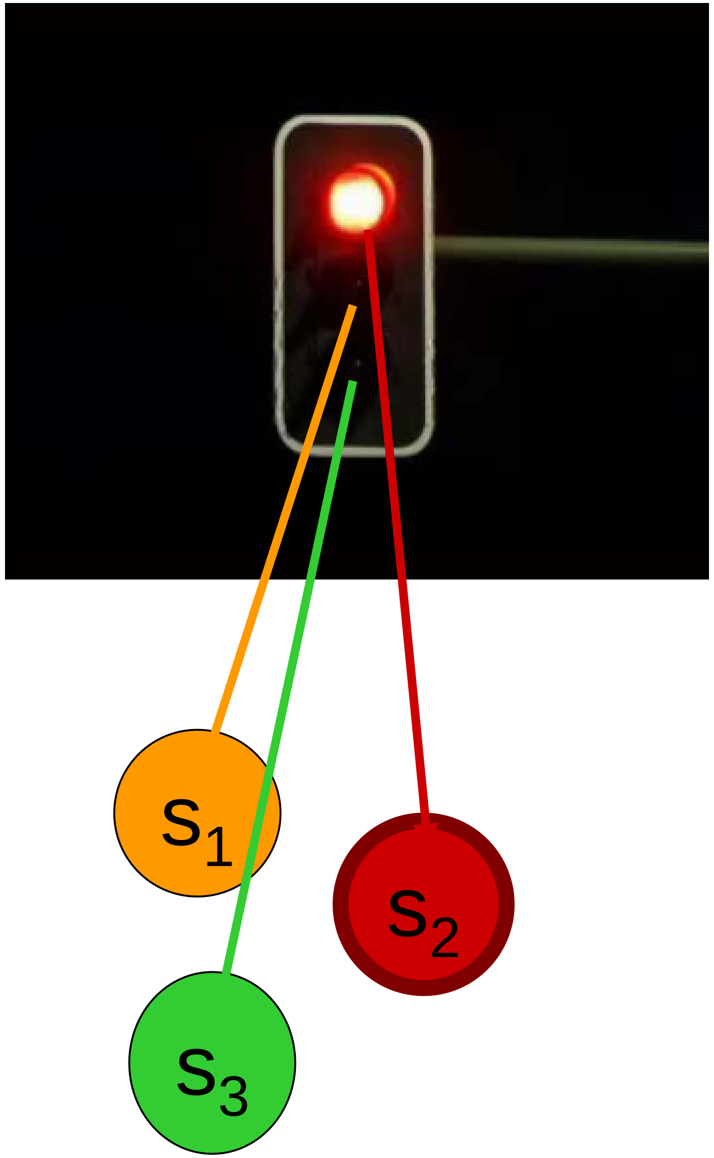


Limiti dei modelli markoviani

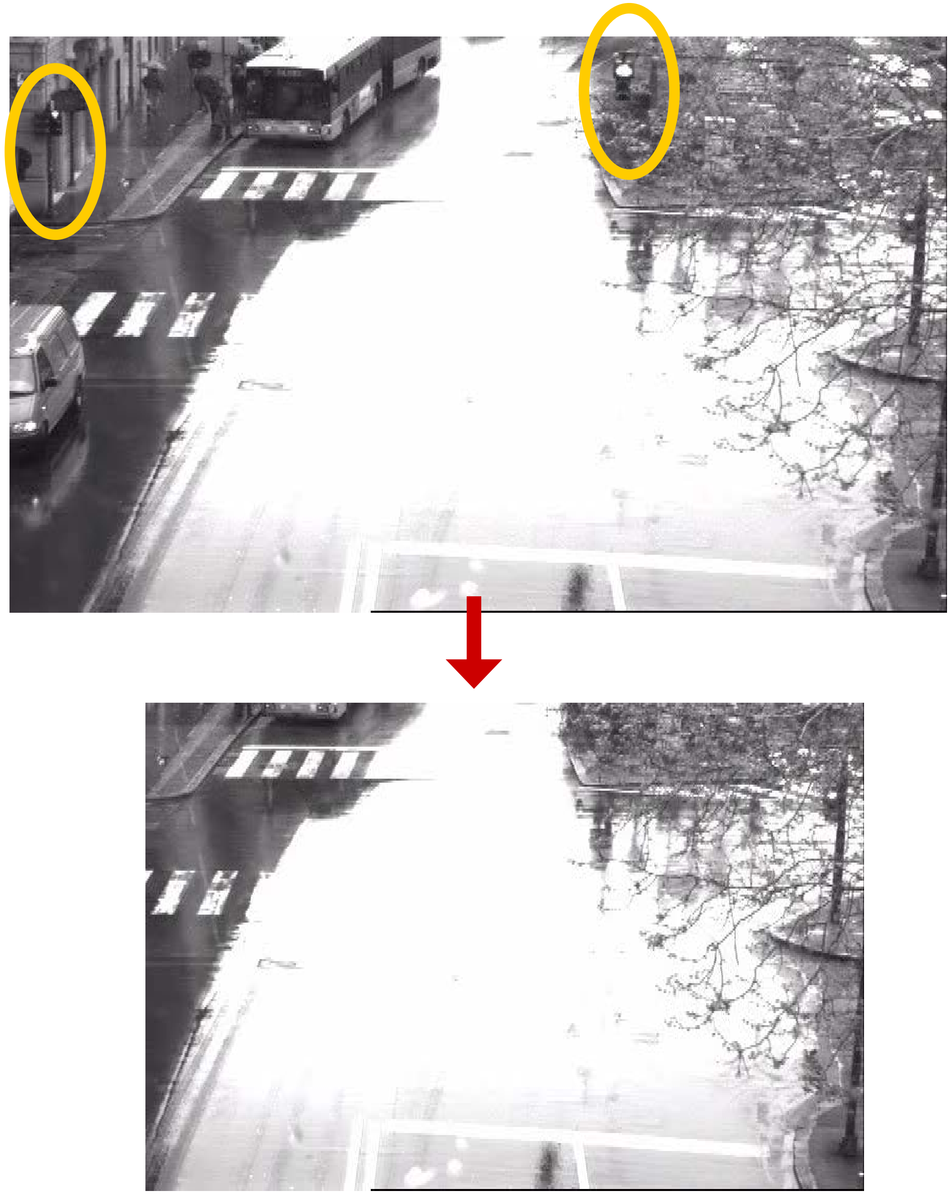
2. **(e più importante!)** Nel caso del semaforo lo stato è **esplicito**, (una particolare configurazione del semaforo), e **valutabile direttamente tramite l'osservazione**

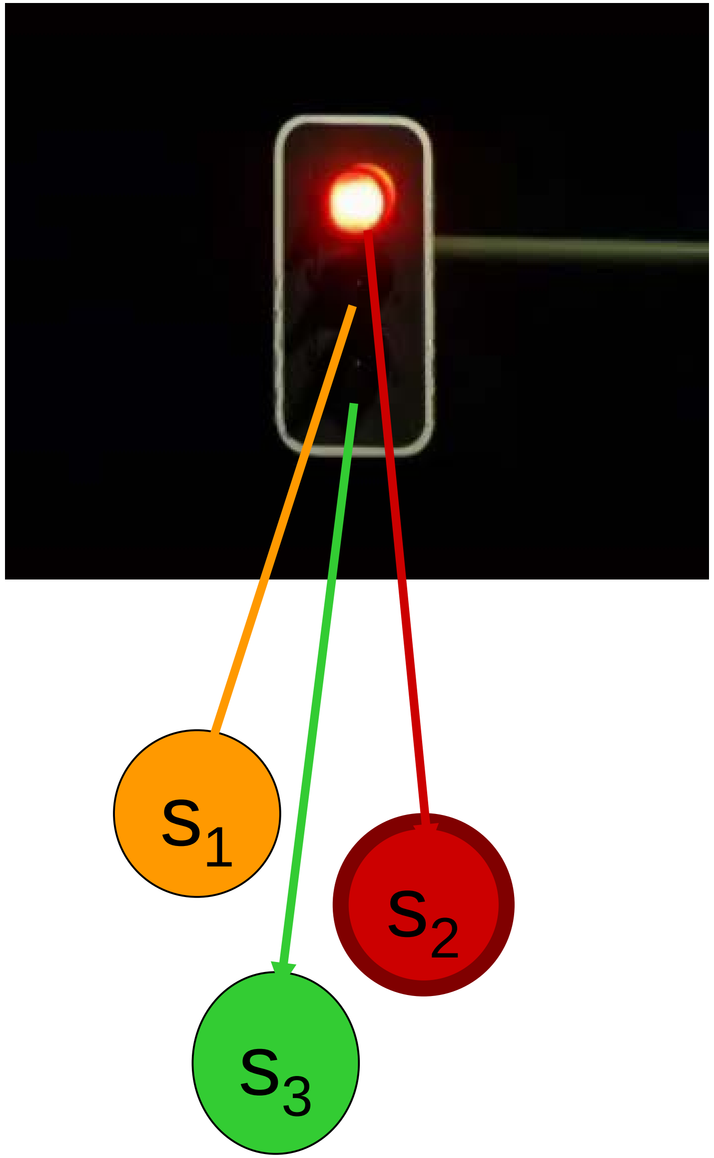
   (lo stato corrisponde al colore del semaforo)

- Non sempre accade così!

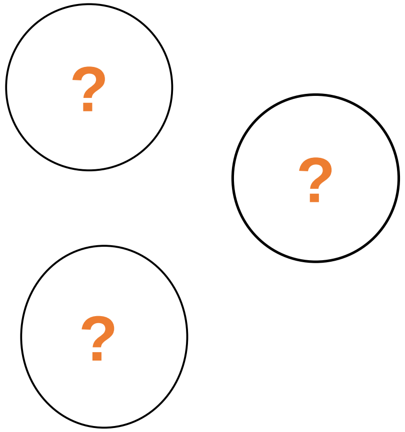


Limiti dei modelli markoviani



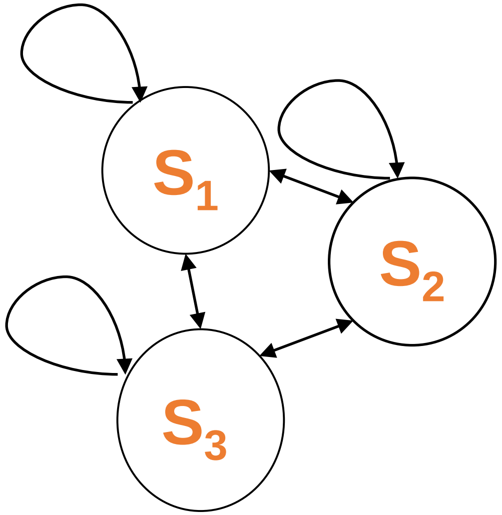


Limiti dei modelli markoviani



- *Osservo* il filmato: *osservo* che c'è un **sistema che evolve**, ma non riesco a capire quali siano gli stati regolatori.

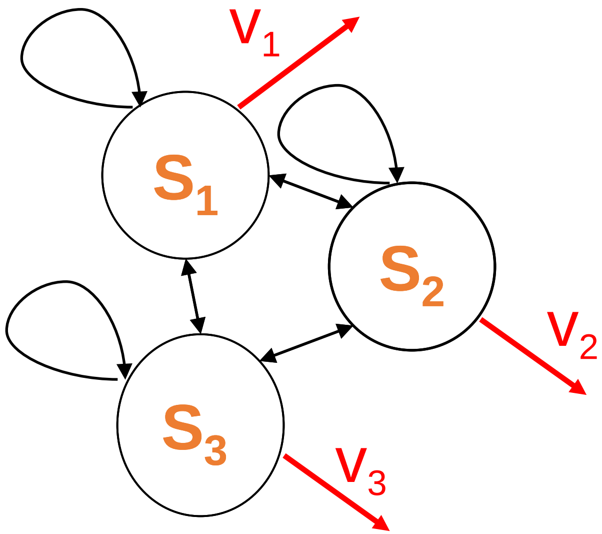
Limiti dei modelli markoviani



- *Osservo* il filmato: *osservo* che c'è un **sistema che evolve**, ma non riesco a capire quali siano gli stati regolatori.

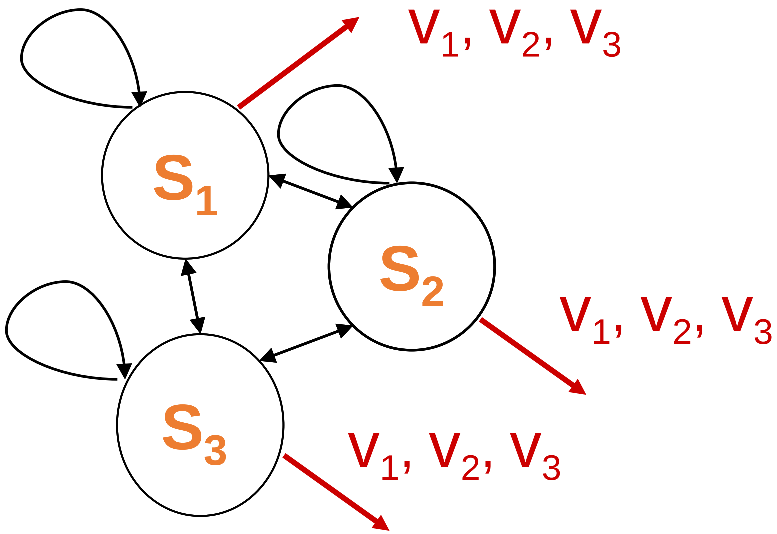
- Il sistema comunque evolve a **stati**, lo capisco *osservando* il fenomeno

Limiti dei modelli markoviani



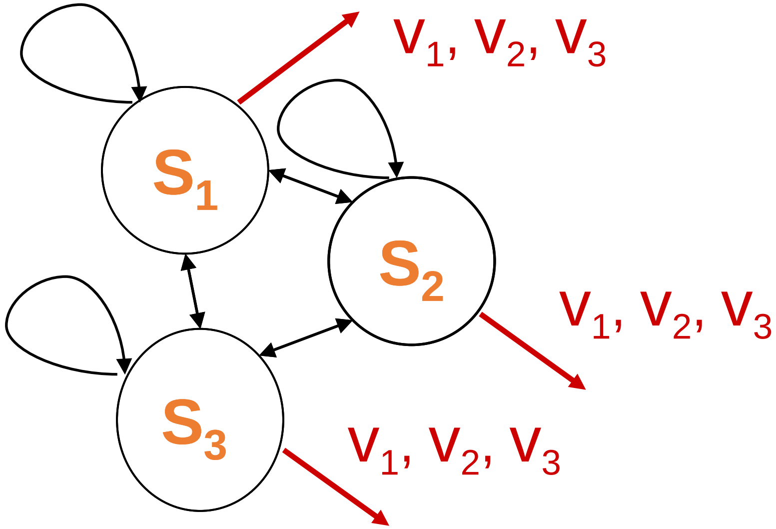
- Meglio: il sistema evolve grazie a degli **stati "nascosti"** (gli stati del semaforo, che però non vedo e di cui ignoro l'esistenza) di cui riesco ad ***osservare*** solo le probabili "conseguenze", ossia i flussi delle auto

Limiti dei modelli markoviani



- Rinuncio a dare un nome agli stati, li penso come entità nascoste e *identificabili solo tramite osservazioni* (i flussi delle auto)
- Stabilisco una **relazione tra *osservazione e stato nascosto***

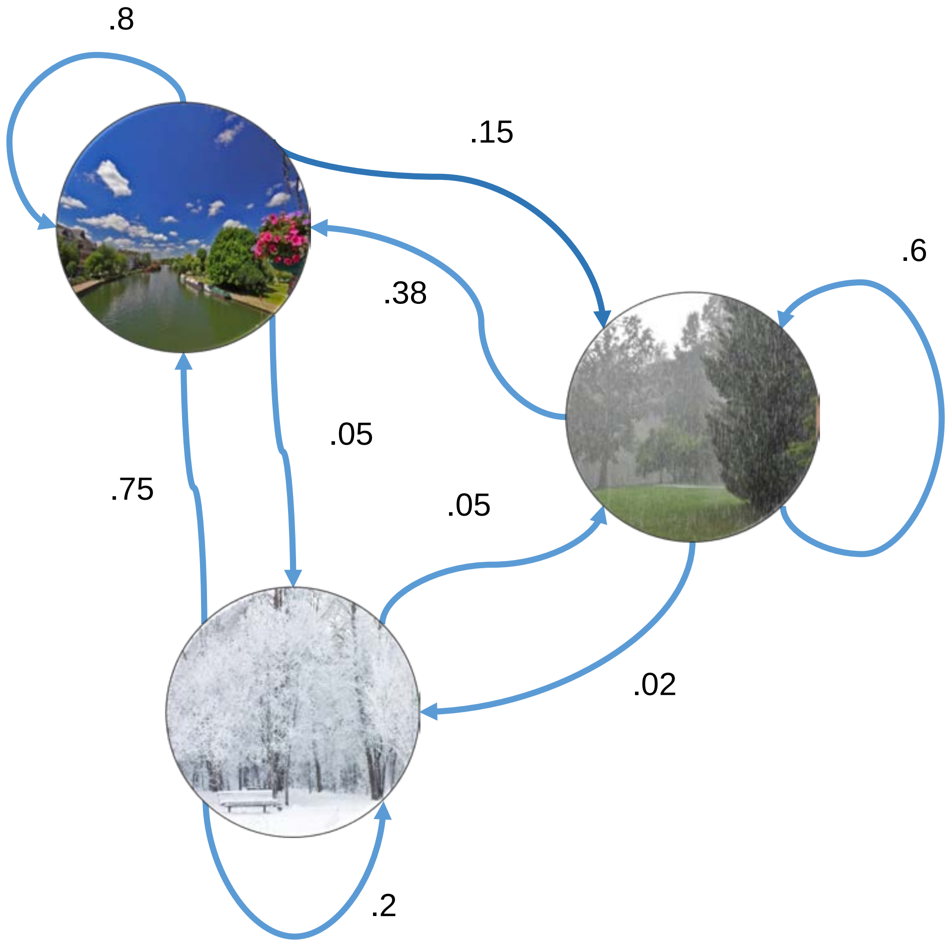
Modelli markoviani a stati nascosti (HMM)



- Gli Hidden Markov Model si inseriscono in questo contesto
- Descrivono probabilisticamente la *dinamica di un sistema* rinunciando ad identificarne direttamente le cause, o meglio, cerca di stimarle
- Gli *stati* sono identificabili solamente tramite le *osservazioni*, in maniera probabilistica.

Hidden Markov Models

States are not observable!



Hidden Markov Model (HMM)

- Classificatore statistico di sequenze, molto utilizzato negli ultimi anni in diversi contesti
- Tale modello può essere inteso come estensione del modello di Markov dal quale differisce per la **non osservabilità dei suoi stati**
- Ogni stato ha associate una funzione di probabilità che descrive la probabilità che un certo simbolo (output) sia emesso da quello stato

HMM: a formal definition

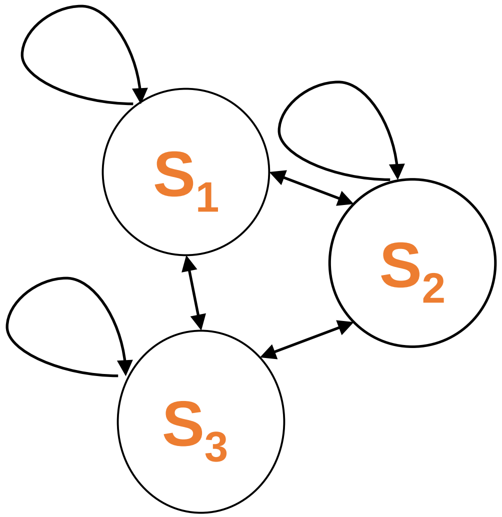
- From [Rabiner 89]:

“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: definizione formale

- Un HMM (discreto) è formato da:
  - Un insieme $S=\{s_1,s_2,\ldots,s_N\}$ di stati nascosti;
  - Una matrice di transizione $A=\{a_{ij}\}$,
    tra stati nascosti 1=<i,j=<N
  - Una distribuzione iniziale sugli stati nascosti $\pi=\{\pi_i\}$,



π =

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

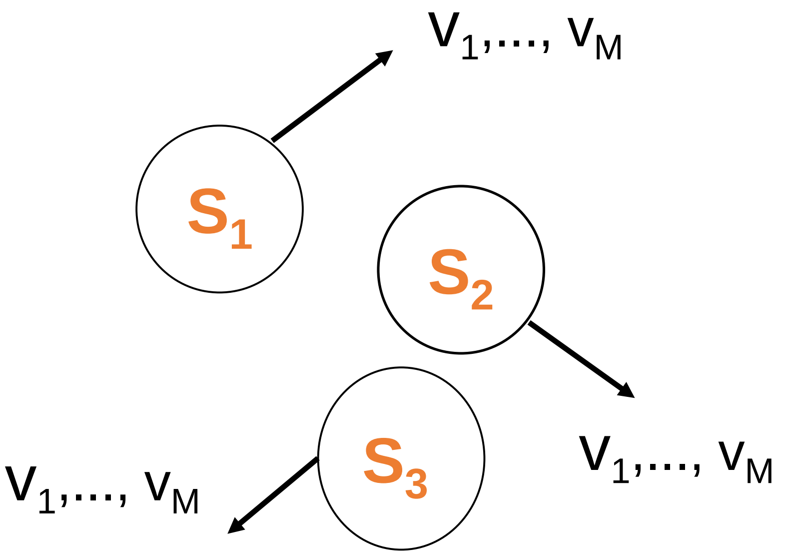
$\boldsymbol{A}$ =

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

HMM: definizione formale

- Un insieme *$V=\{v_1,v_2,\ldots,v_M\}$ di simboli d'osservazione;*

- Una distribuzione di probabilità sui simboli d'osservazione *$B=\{b_{jk}\}$, che indica la probabilità di emissione del simbolo* v_k quando lo stato del sistema è 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: definizione formale

- Denotiamo una HMM con una tripla $\lambda=(A, B, \pi)$

$\boldsymbol{\pi} =$

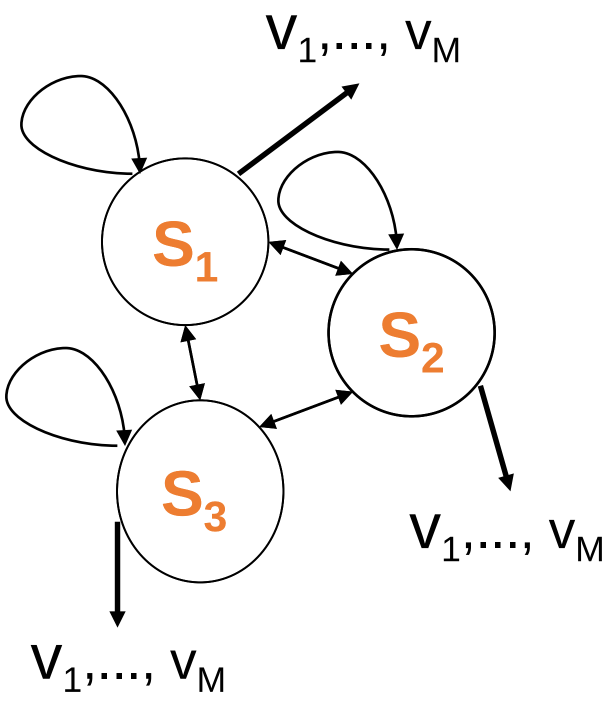
$\pi_1= 0.33$	$\pi_2= 0.33$	$\pi_3= 0.33$

$\boldsymbol{A} =$

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

$\boldsymbol{B} =$

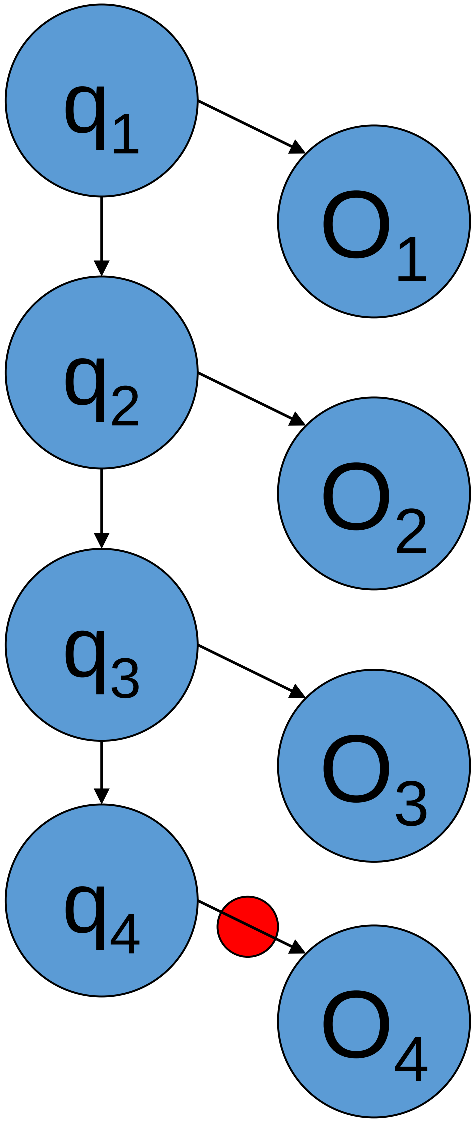
$b_{11}=0.8$	$b_{21}=0.1$	$b_{31}=0.1$
$b_{12}= 0.1$	$b_{22}= 0.8$	$b_{32}= 0.1$
⋮	⋮	⋮
$b_{1M}= 0.1$	$b_{2M}= 0.1$	$b_{3M}=0.8$



Assunzioni sull'osservazione

- Indipendenze condizionali

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

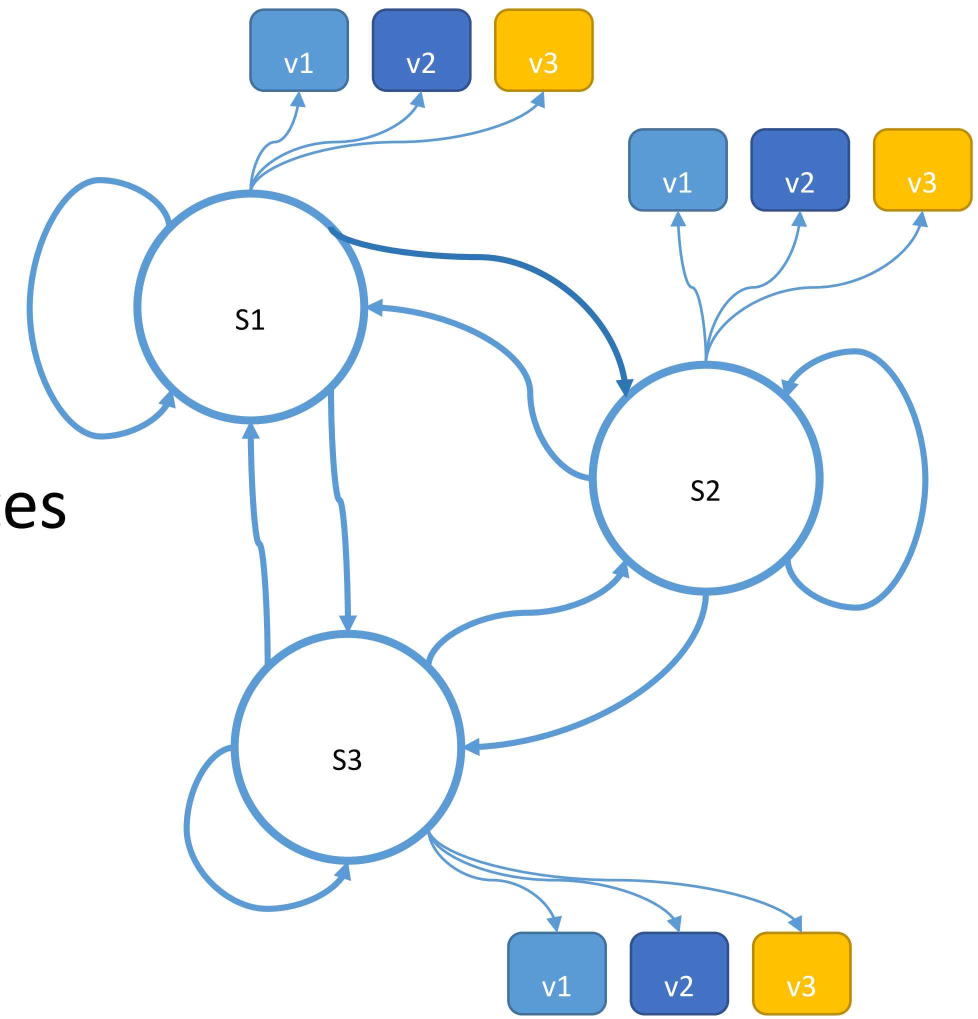


Hidden Markov Models

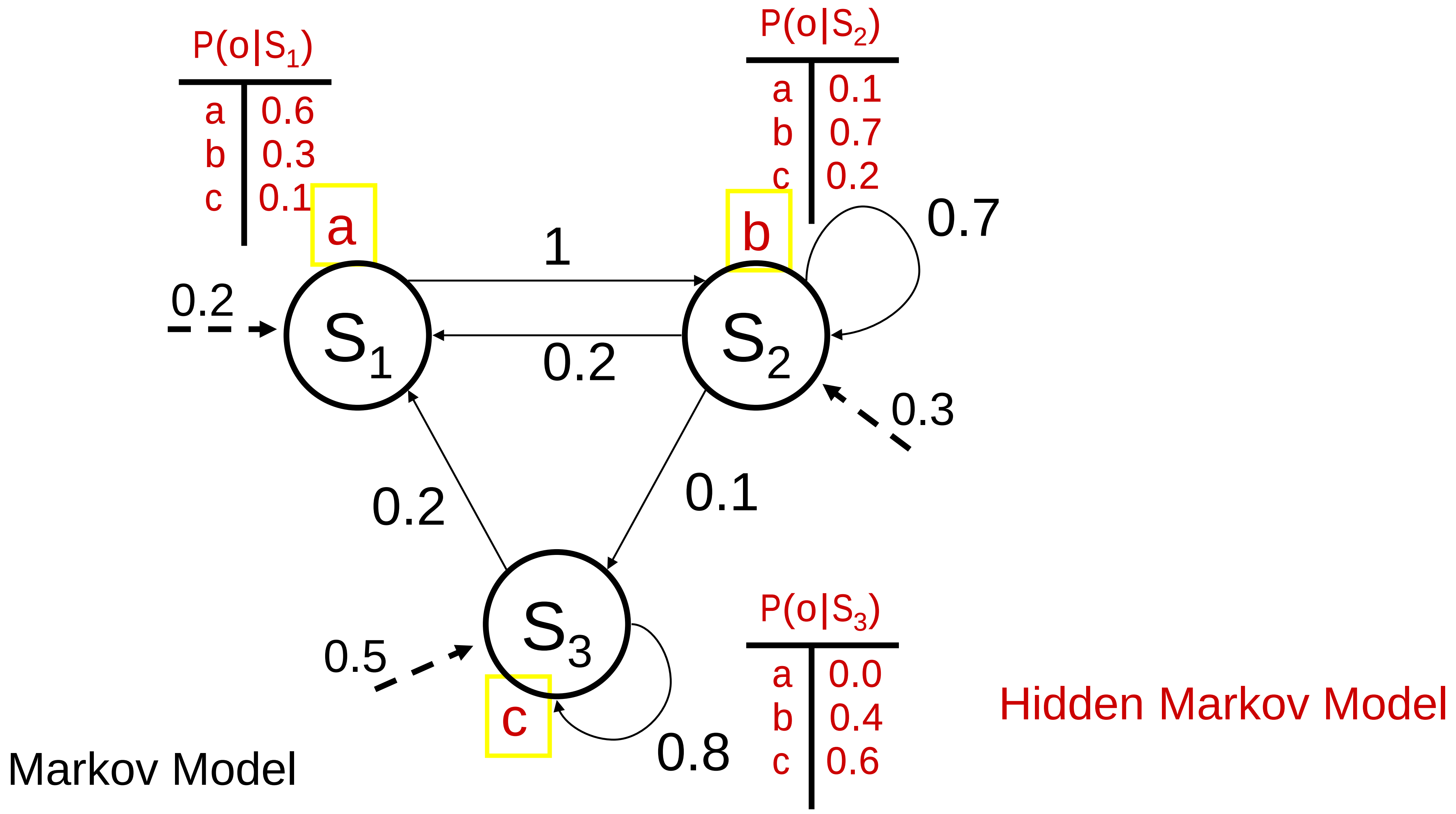
- A set of N **states** $\mathcal{S} = \{S_1, \dots, S_N\}$
- A sequence of states $\mathcal{Q} = q_1, \dots, q_T$

- 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**
$$\mathrm{B} = b_i(v_k) = P(o_t = v_k | q_t = S_i)$$

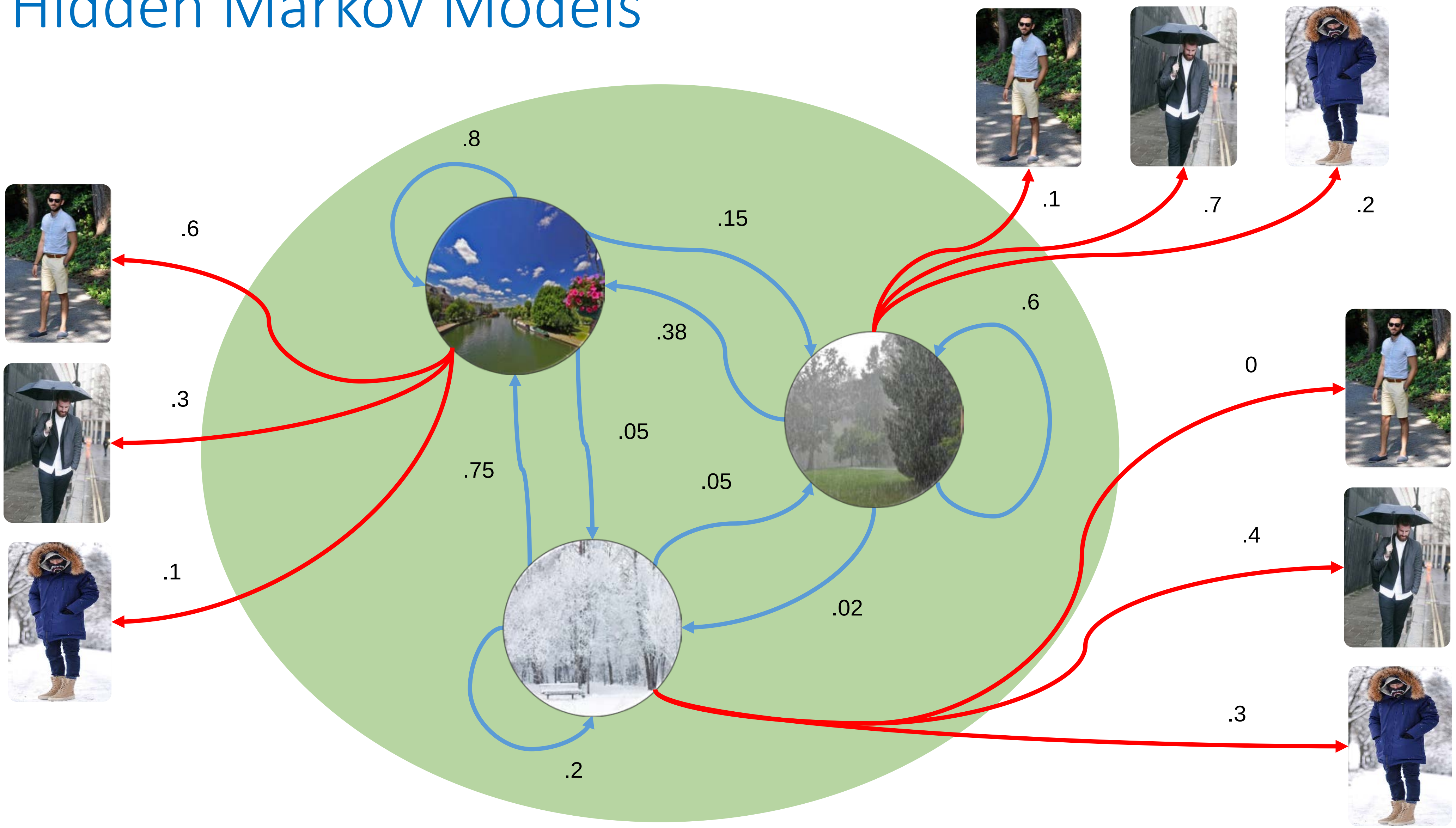
- An **observation vocabulary** $\mathcal{V} = \{v_1, \dots, v_M\}$
- A sequence of **observations** $\mathcal{O} = o_1, \dots, o_T$



Da un Markov Model ad un Hidden Markov Model



Hidden Markov Models



Problemi chiave del modello HMM

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

Dato un modello HMM λ e una stringa d'osservazione **O=(**$O_1, O_2, \ldots, O_t, \ldots, O_T$) calcolare P(**O|** λ)
→ *procedura di forward*

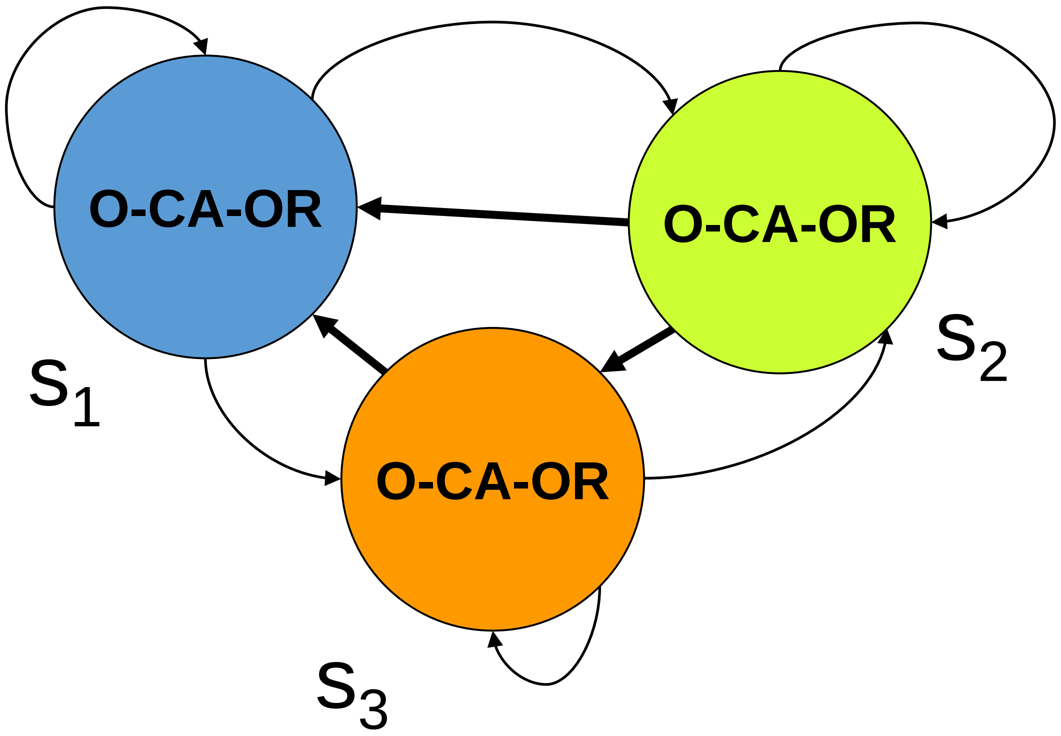
<u>Problema 2:</u> ***Decoding***

Data una stringa d'osservazione **O** e un modello HMM λ, calcolare la più probabile sequenza di stati $S_1 \ldots S_T$ che ha generato **O**
→ *procedura di Viterbi*

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

Dato un insieme di osservazioni {**O**}, determinare il miglior modello HMM $\lambda = (\pi, A, B)$, cioè il modello per cui P(**O**| λ) è massimizzata
→ *procedura di Baum Welch (forward-backword)*

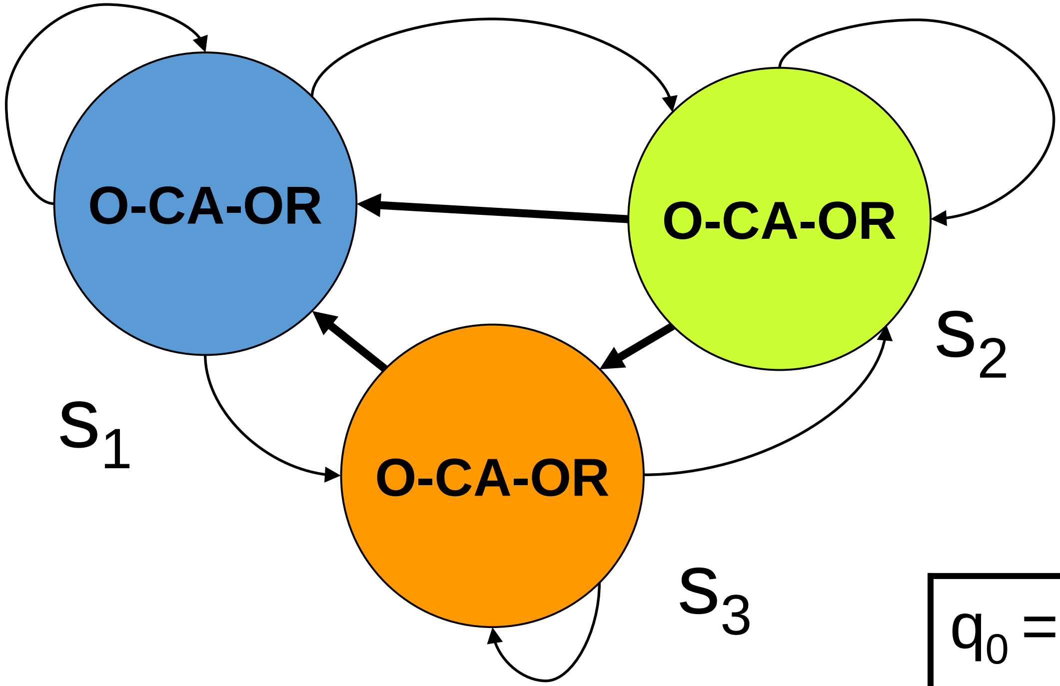
HMM – generatore di stringhe



- 3 stati: s_1 ,s_2, s_3;
- 3 simboli: 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 – generatore di stringhe



Il nostro problema è che gli stati non sono direttamente osservabili!

$q_0 =$	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

Problema 1:
Probabilità di una serie di osservazioni

- $P(\mathbf{O})=P(O_1,O_2,O_3)=P(O_1=CA,O_2=CA,O_3=O)?$
- Strategia forza bruta:

$$P(\mathbf{O}) = \sum_{\mathbf{Q} \in \text{cammini di lunghezza } 3} P(\mathbf{O},\mathbf{Q})$$

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

? ?

Problema 1:
Probabilità di una serie di osservazioni

- $P(\mathbf{O})=P(O_1,O_2\,,O_3)=P(O_1=X,O_2\,=X,O_3=Z)?$
- Strategia forza bruta:

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

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

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

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

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

<u>Ad esempio</u>, nel caso

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

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

Problema 1: Probabilità di una serie di osservazioni

- $P(\mathbf{O})=P(O_1,O_2,O_3) = P(O_1=X,O_2=X,O_3=Z)$?
- Strategia forza bruta:
  - o $P(\mathbf{O}) = \sum P(\mathbf{O},\mathbf{Q})$

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

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

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

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

Ad esempio, nel caso

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

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

Osservazioni

- Le precedenti computazioni risolvono **solo un termine della sommatoria**: per il calcolo di P(**O**) sono necessarie 27 P(Q) e 27 P(**O**|**Q**)
- Per una sequenza da 20 osservazioni necessitiamo di 3^{20} P(Q) e 3^{20} P(O|Q)
- Esiste un modo più efficace, che si basa sulla definizione di una particolare probabilità
- In generale:

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

è di complessità elevata, $O(N^T T)$, dove N è il numero degli stati, T lunghezza della sequenza

Procedura Forward

- Date le osservazioni $O_1, O_2, \ldots, O_T$ definiamo

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

  ossia:
  - *abbiamo visto le prime t osservazioni*
  - *siamo finiti in* s_i *al t-esimo stato visitato*
- Tale probabilità si può definire ricorsivamente:

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

  - Per ipotesi induttiva $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = s_i \mid \lambda)$
  - Voglio calcolare:

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

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

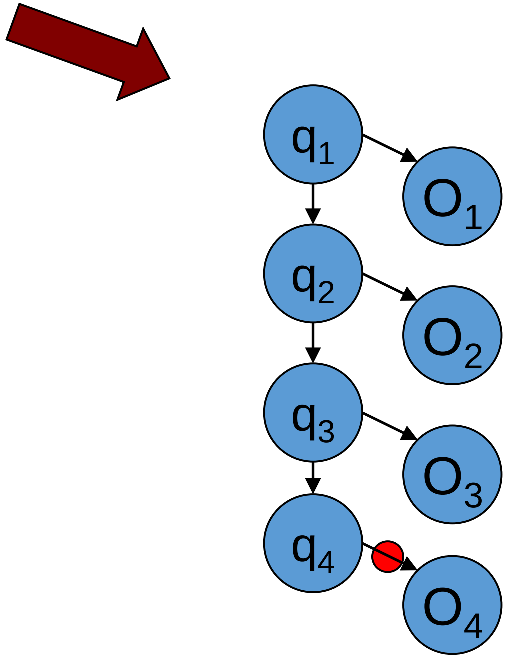
$$= \sum_{i=1}^{N} P(O_1,O_2,\ldots,O_t, q_t=s_i, \underbrace{O_{t+1}, q_{t+1}=s_j})$$

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

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

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

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



Risposta al problema 1: evaluation

- Data $O_1,O_2,\ldots, O_t, \ldots,O_T$ e conoscendo $\alpha_t(i) = P(O_1,O_2,\ldots,O_t,q_t=s_i|\lambda)$, possiamo calcolare:

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

di complessità $O(N^2T)$

- Ma anche altre quantità utili, per esempio:

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

Risposta al problema 1: evaluation

- α è chiamata variabile *forward*
- Alternativamente si può calcolare ricorsivamente introducendo un'altra variabile, cosiddetta *backward*

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

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

e quindi

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

$$= \sum_{j=1}^{N} \beta_1(j) \quad \textit{verificare}!!!$$

Problema 2: Cammino più probabile (*decoding*)

- Qual'è il cammino di stati più probabile (MPP) che ha generato $O_1, O_2, \ldots, O_T$? Ossia quanto vale

$$\underset{\mathbf{Q}}{\operatorname{argmax}}\, P(\mathbf{Q} \mid \mathrm{O}_1 \mathrm{O}_2 \ldots \mathrm{O}_T)\ ?$$

- Strategia forza bruta:

$$\underset{\mathbf{Q}}{\operatorname{argmax}} \frac{P(\mathrm{O}_1 \mathrm{O}_2 \ldots \mathrm{O}_T \mid \mathbf{Q}) P(\mathbf{Q})}{P(\mathrm{O}_1 \mathrm{O}_2 \ldots \mathrm{O}_T)}$$

$$\propto \underset{\mathbf{Q}}{\operatorname{argmax}}\, P(\mathrm{O}_1 \mathrm{O}_2 \ldots \mathrm{O}_T \mid \mathbf{Q}) P(\mathbf{Q})$$

Calcolo efficiente di MPP

- Definiamo la seguente probabilità:

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

ossia la massima probabilità dei cammini di lunghezza t-1 i quali:
  - occorrono
  - finiscono nello stato s_i all'istante t
  - producono come output $O_1, O_2, \ldots, O_t$
- Si cerca la singola miglior sequenza di stati singoli (path) massimizzando $P(Q|O, \lambda)$
- La soluzione a questo problema è una tecnica di programmazione dinamica chiamata l'algoritmo di Viterbi
  - Si cerca il più probabile stato singolo alla posizione i-esima date le osservazioni e gli stati precedenti

Algoritmo di Viterbi

1) Initialization:

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

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

Per induzione abbiamo

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

Algoritmo di Viterbi

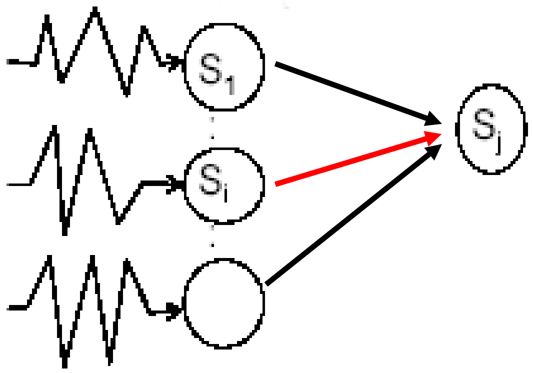
2) Recursion:

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

$$1 \le j \le N$$

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

$$1 \le j \le N.$$



ATTENZIONE:
calcolato per ogni j !!!

Algoritmo di Viterbi

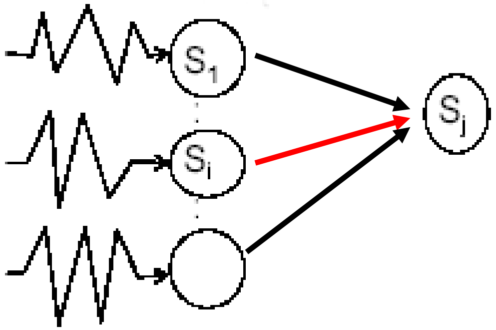
3) Termination:

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

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

4) Path (state sequence) backtracking:

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



Problema 3: Addestramento di HMM

- Si parla di processo di *addestramento* di HMM, o *fase di stima di parametri*, in cui i parametri di λ=(A,B, *π*), vengono stimati dalle *osservazioni di training*

- Di solito si usa la stima Maximum Likelihood

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \quad P(\mathrm{O_1 O_2 ... O_T} \mid \lambda)$$

- Ma si possono usare anche altre stime

$$\max_{\lambda} \quad P(\lambda \mid \mathrm{O_1 O_2 ... O_T})$$

Stima ML di HMM: procedura di ri-stima di Baum Welch

Definiamo

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

Tali quantità possono essere calcolate efficientemente (cfr. Rabiner)

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

$$\sum_{t=1}^{T-1} \xi_t(i,j) =$$ numero atteso di transizioni dallo stato *i* allo stato *j* durante il cammino

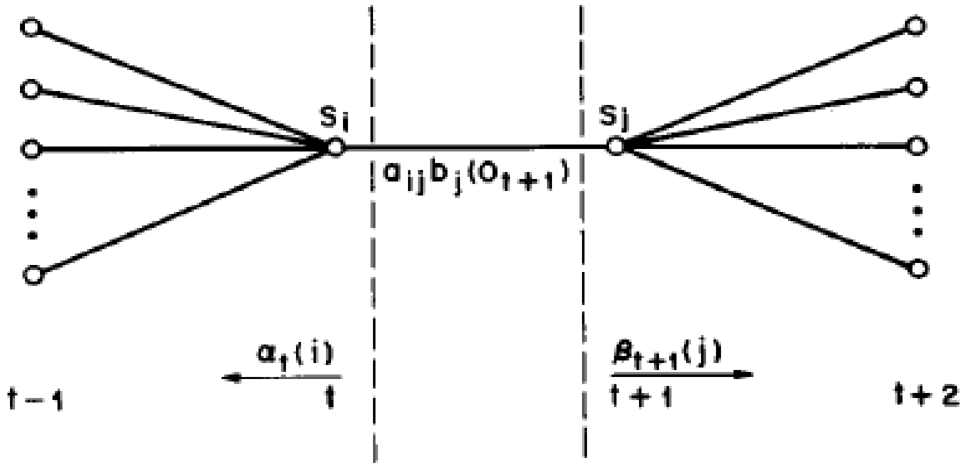
$$\sum_{t=1}^{T-1} \gamma_t(i) =$$ numero atteso di transizioni passanti dallo stato *i* durante il cammino

- Usando le variabili forward e backward, ξ è anche calcolabile come

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

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

(E step)



Stima ML di HMM:
procedura di ri-stima di Baum Welch

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

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

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

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

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

Formule di ri-stima dei parametri
(M step)

Algoritmo di Baum-Welch

- Tali quantità vengono utilizzate nel processo di stima dei parametri dell'HMM in modo iterativo

- Si utilizza una variazione dell'algoritmo di Expectation-Maximization (EM)
  - che esegue un'ottimizzazione locale
  - massimizzando la log-likelihood del modello rispetto ai dati

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

EM - BAUM WELCH (2)

- Conoscendo le quantità quali:
  - numero atteso di transizioni uscenti dallo stato i durante il cammino,
  - numero atteso di transizioni dallo stato i allo stato j durante il cammino,

  potremmo calcolare le stime correnti ML di λ, = $\overline{\lambda}$, ossia

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

- Tali considerazioni danno luogo all'algoritmo di Baum-Welch

- Algoritmo:
  1) inizializzo il modello $\bar{\lambda} = (A_0, B_0, \pi_0)$
  2) il modello corrente è $\lambda = \bar{\lambda}$
  3) uso il modello λ per calcolare la parte dx delle formule di ri-stima, ie., la statistica (E step)
  4) uso la statistica per la ri-stima dei parametri ottenendo il nuovo modello $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ (M step)
  5) vai al passo 2, finchè non si verifica la terminazione
- Baum ha dimostrato che ad ogni passo:

$$P(\mathrm{O}_1, \mathrm{O}_2, \ldots, \mathrm{O}_\mathrm{T} \mid \bar{\lambda}) > P(\mathrm{O}_1, \mathrm{O}_2, \ldots, \mathrm{O}_\mathrm{T} \mid \lambda)$$

- Condizioni di terminazione usuali:
  - dopo un numero fissato di cicli
  - convergenza del valore di likelihood

HMM training

Fundamental issue:

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

⇓

- 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

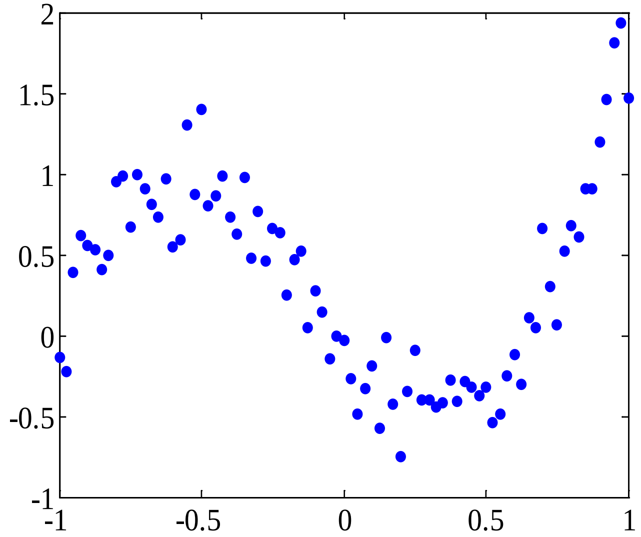
Model selection problem 1: selecting the number of states

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

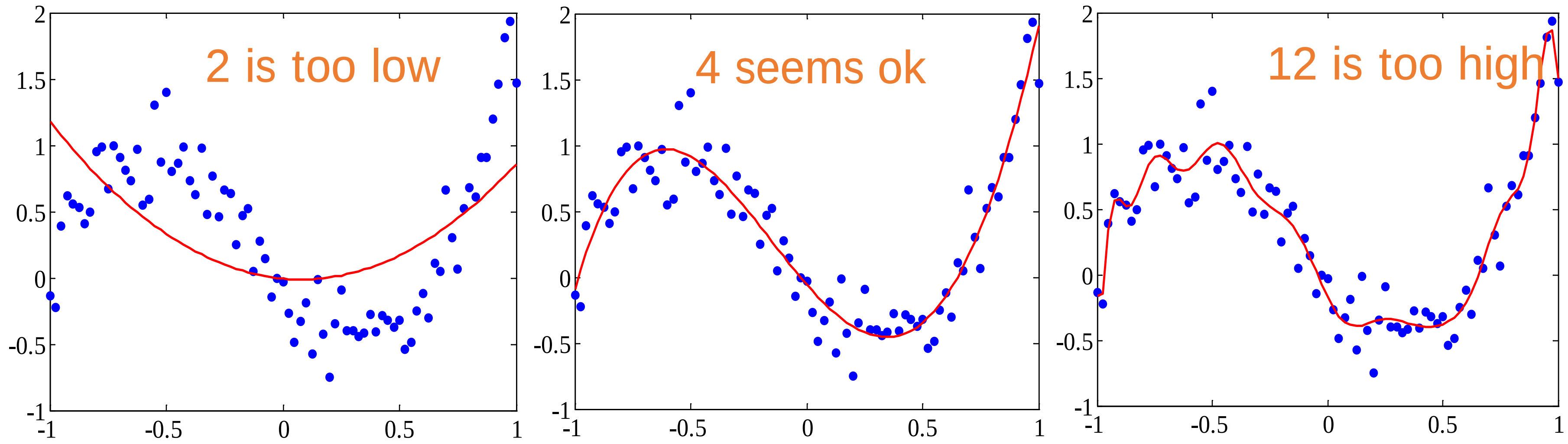
...let’s understand the concept with a toy example

What is model selection?

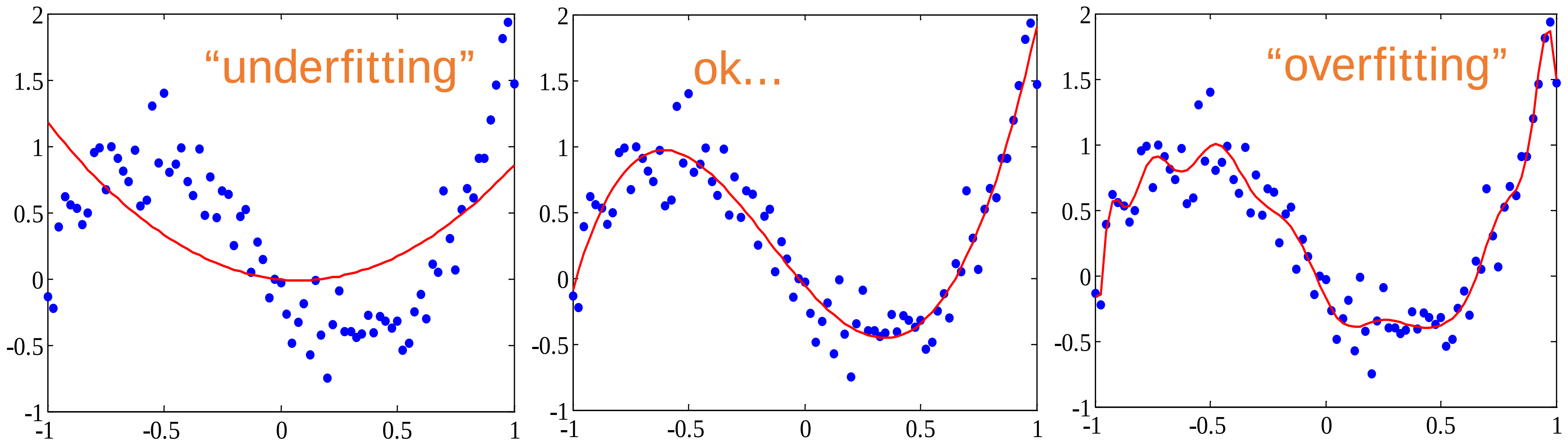
Toy example: some experimental data to which we want to fit a polynomial.



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

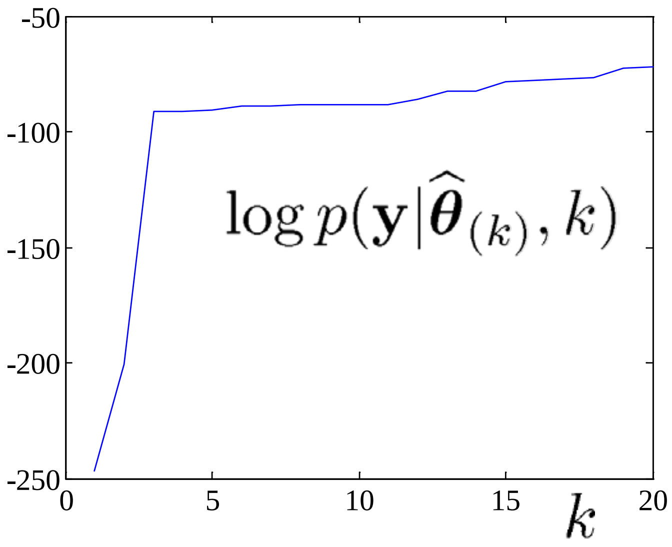
Which “optimality” criterion?

- 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

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

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

Alternative: penalized likelihood

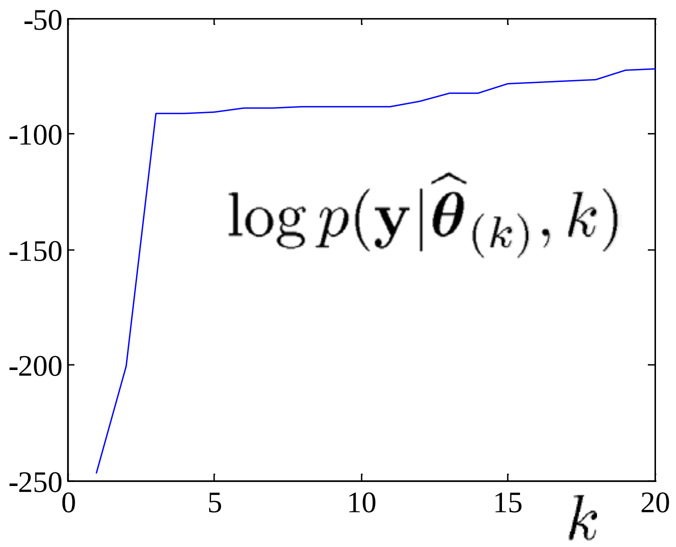
- Example: Bayesian inference criterion (BIC) [Schwartz, 1978]

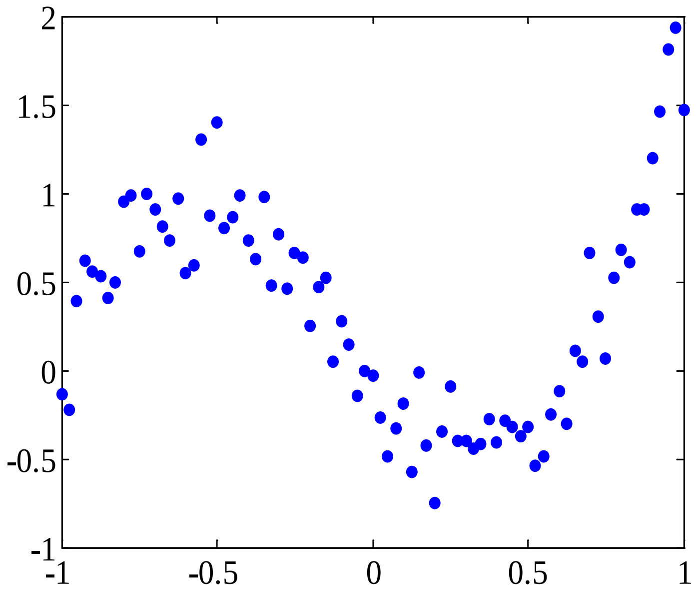
$$k_{best} = \arg\max_{k} \left\{ LL(y \mid \theta_k) - \frac{k}{2}\log(n) \right\}$$

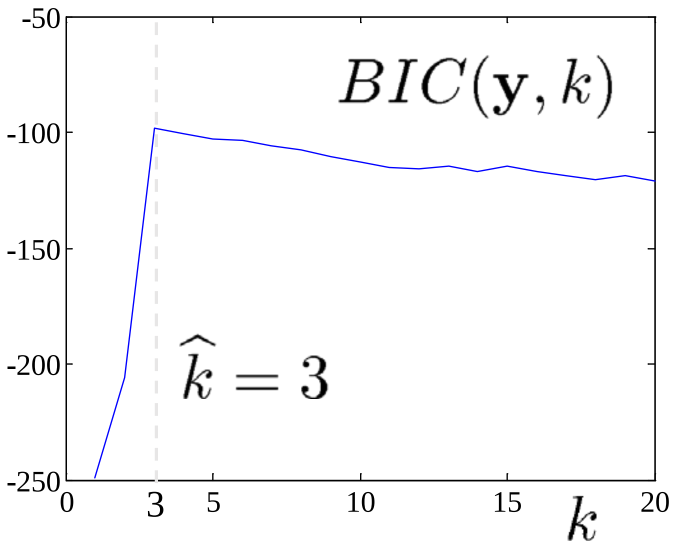
increases with k

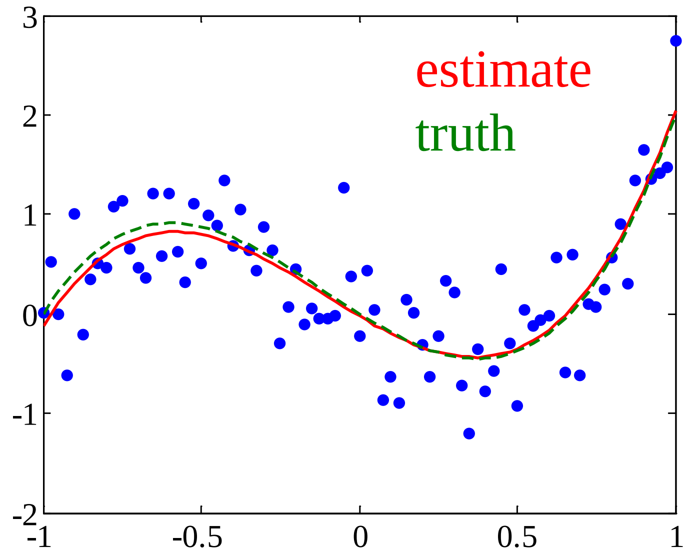
decreases with k
(penalizes larger k)

Back to the polynomial toy example





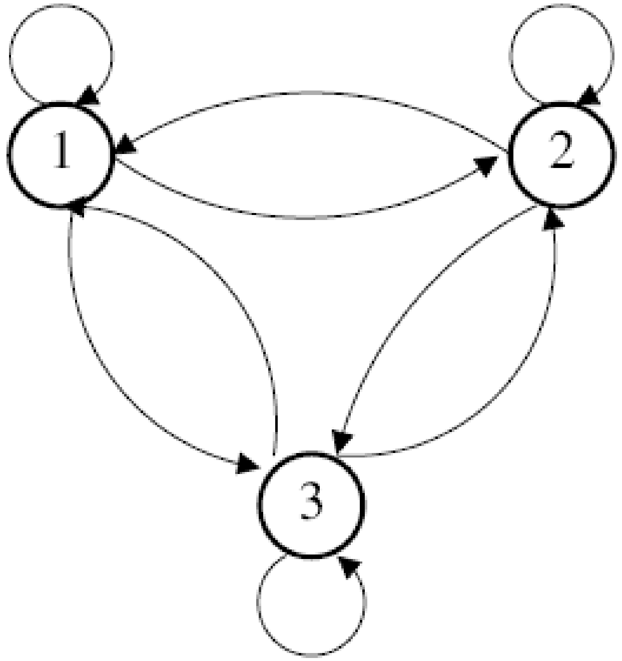




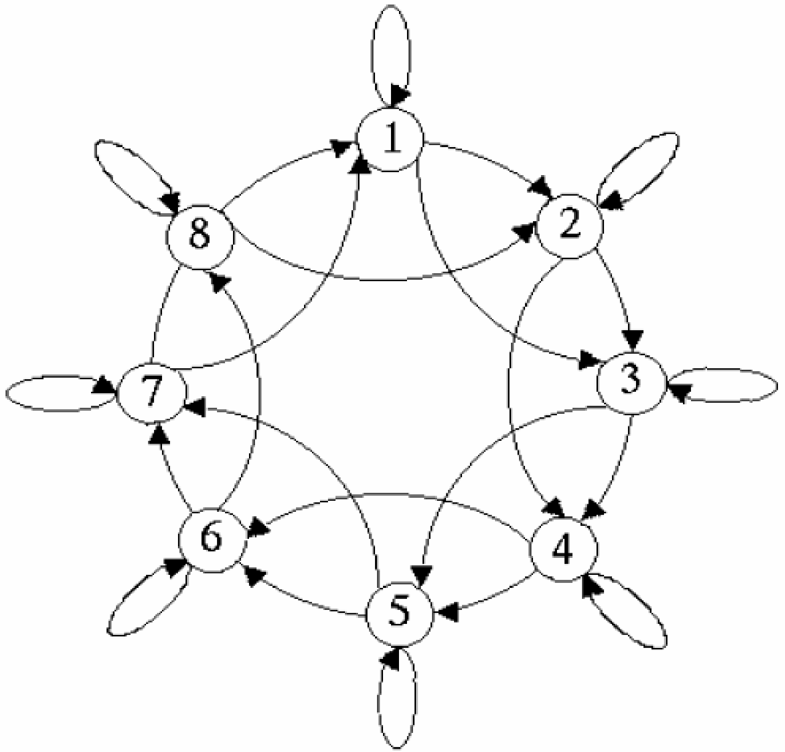
Model selection problem 2: selecting the best topology

- Problem: forcing the absence or the presence of connections

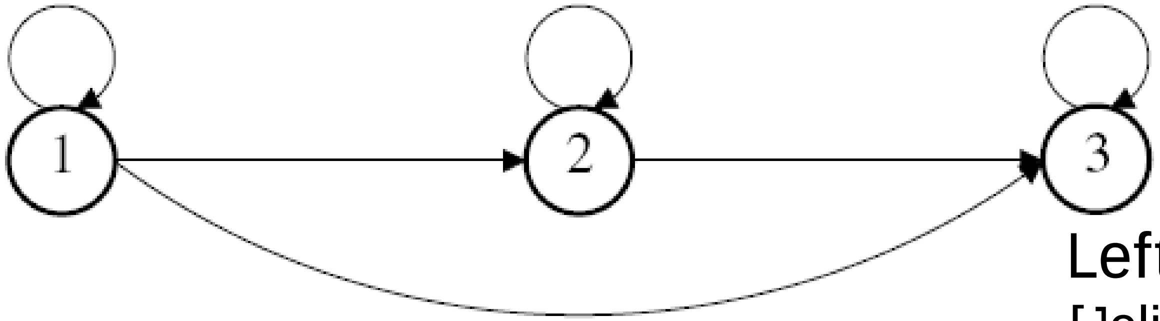
- Typical ad-hoc solutions
  - ergodic HMM (no contraints)
  - left to right HMM (for speech)
  - circular HMM (for shape recognition)



standard ergodic HMM



circular HMM [Arica,Yarman-Vural ICPR00]

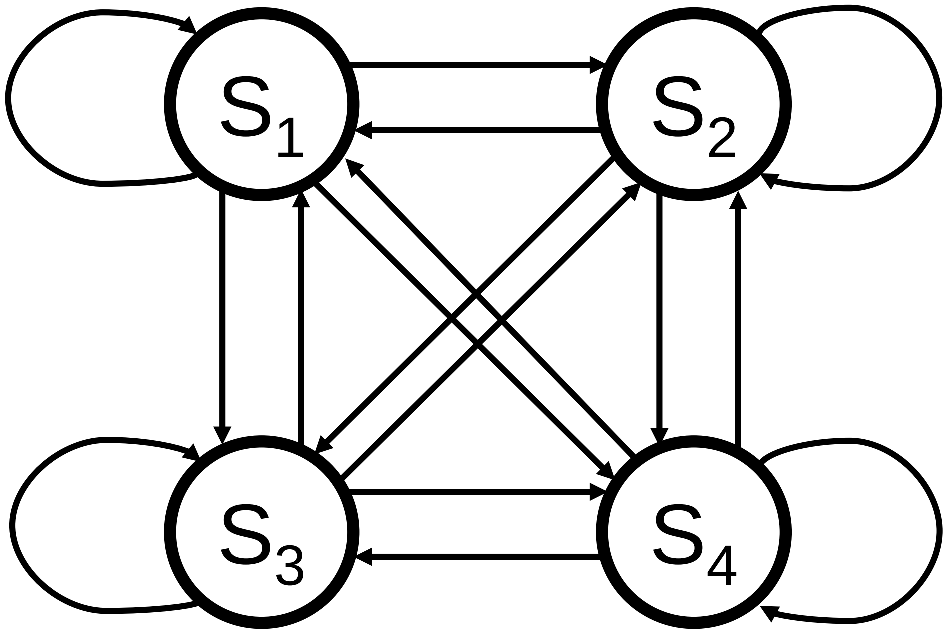


Left to right HMM
[Jelinek, Proc. IEEE 1976]

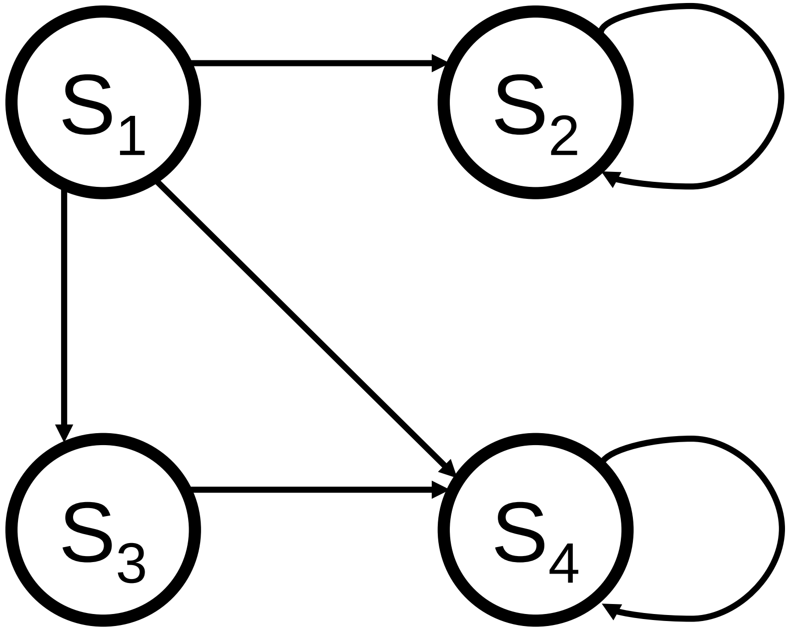
One data-driven solution

[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:

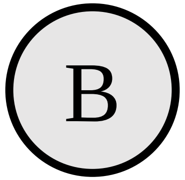
adding novel dependencies between components, in order to model different behaviours

Examples:

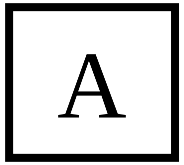
- Input/Output HMM
- Factorial HMM
- Coupled HMM
- ...

Preliminary note: the Bayesian Network formalism

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



hidden variable



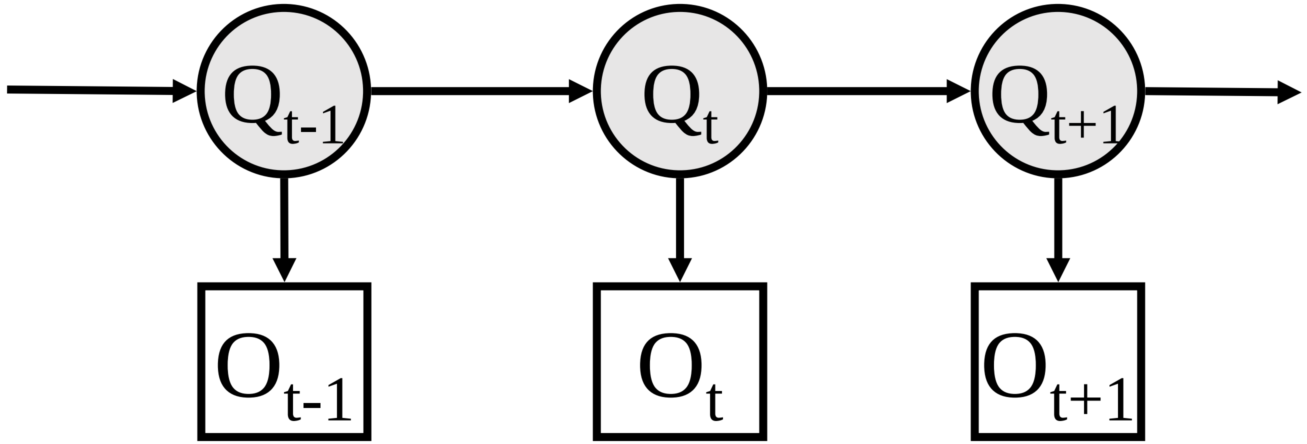
observable variable

causal dependency

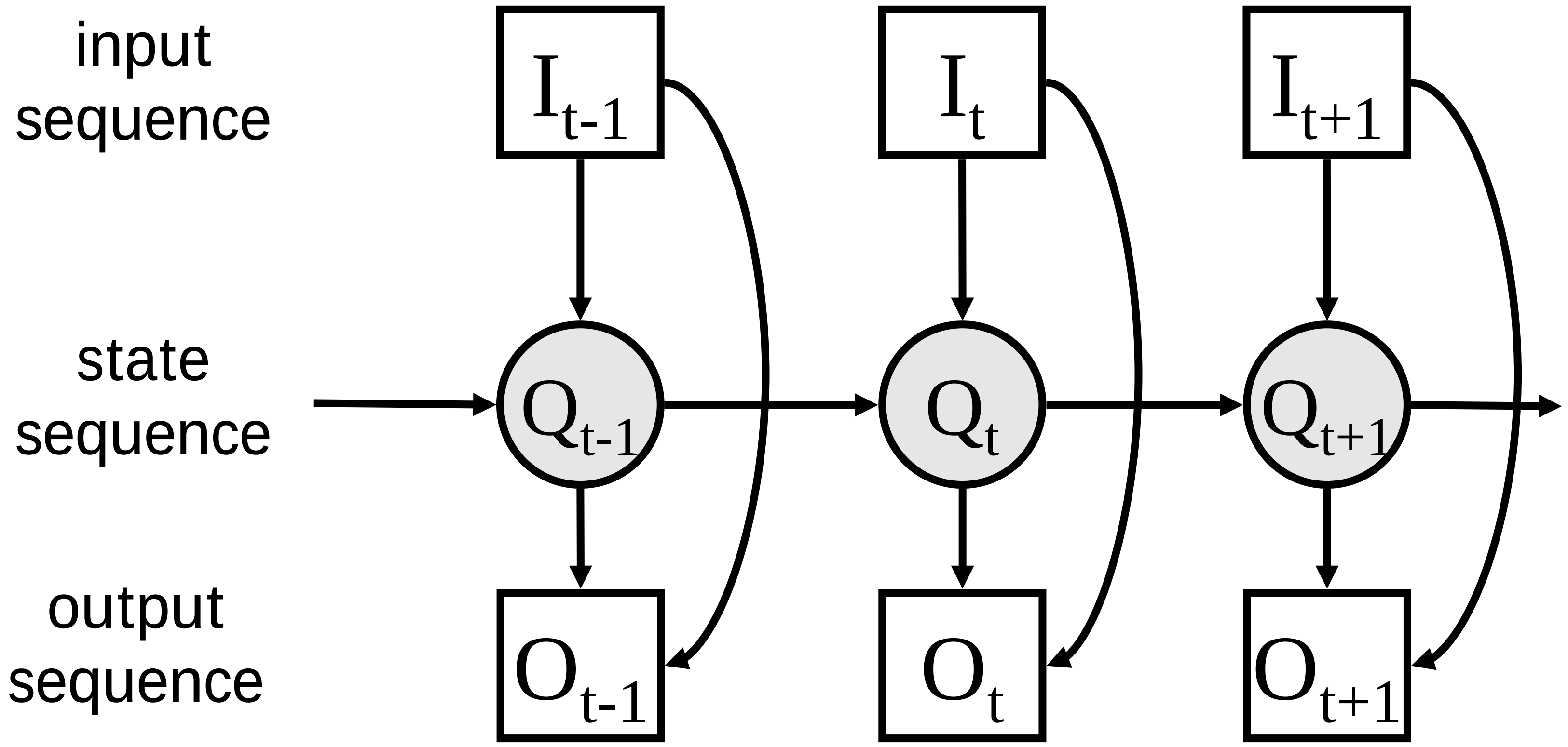
EX.: HMM

state sequence

output sequence

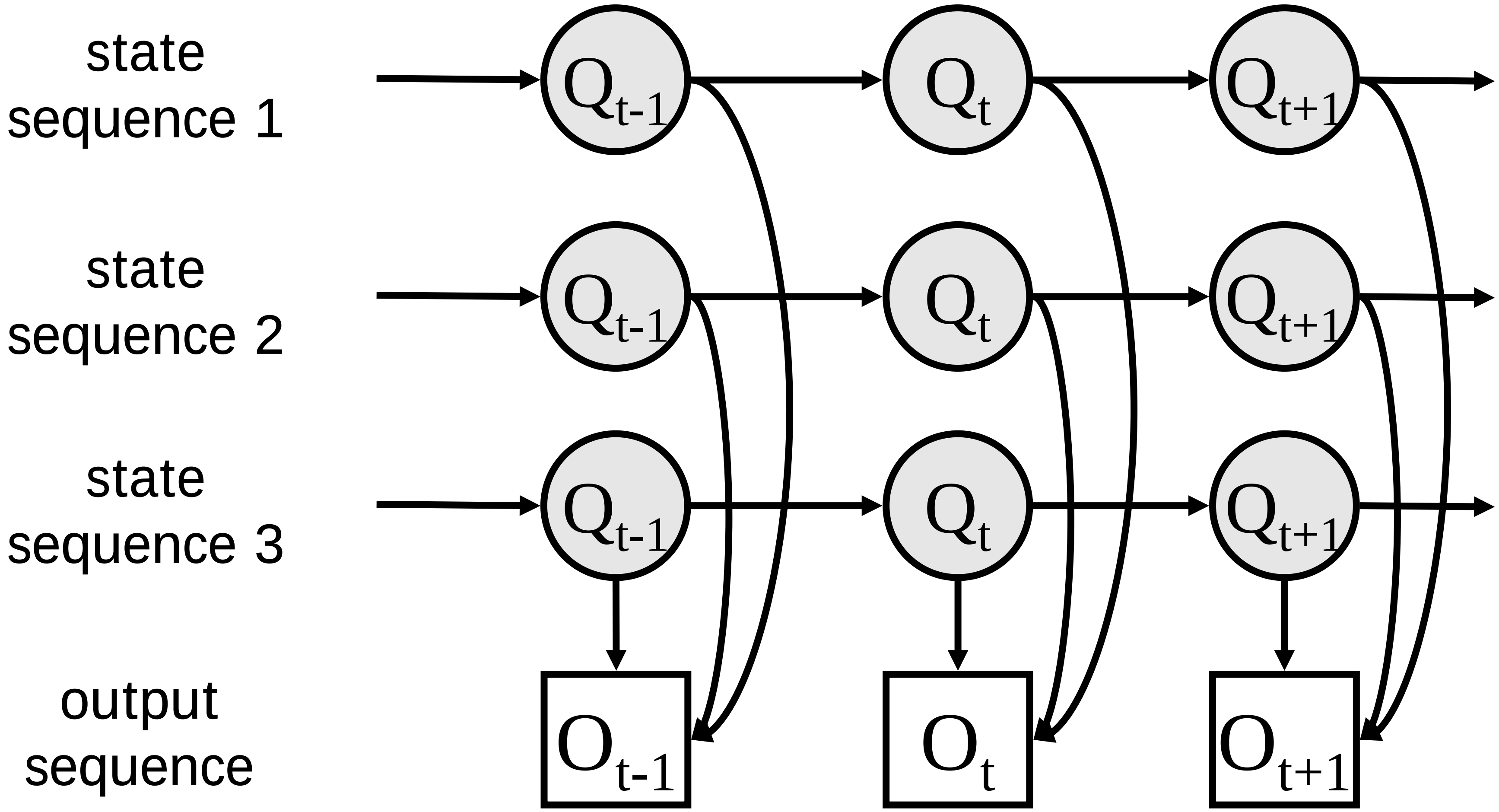


Input-Output HMM: HMM where transitions and emissions are conditional on another sequence (the input sequence)



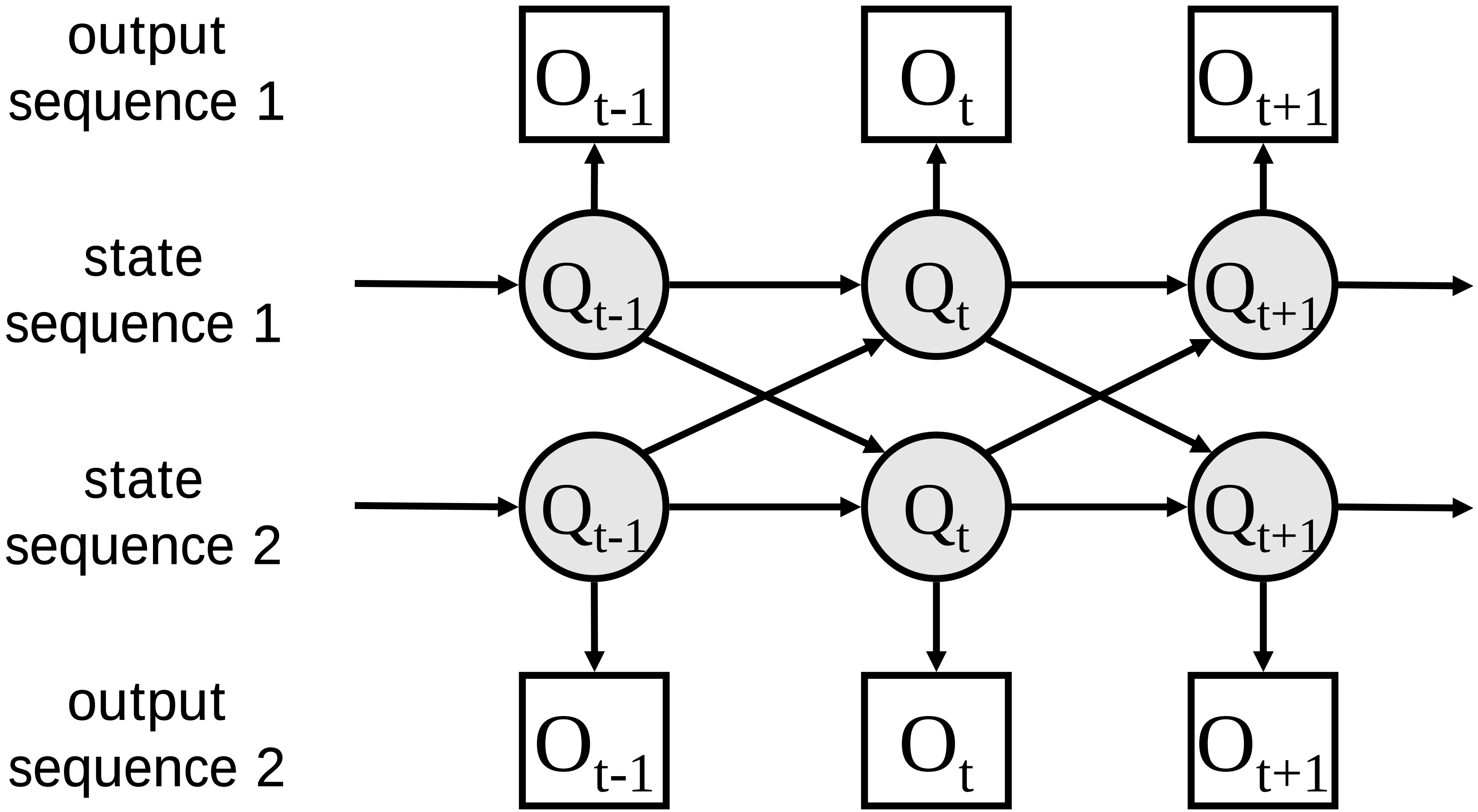
EX.: finance, the input sequence is a leading market index

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:
  - 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
  - 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
  - o Fourier Descriptor
  - o chain code
  - o curvature based techniques
  - o invariants
  - o auto-regressive coefficients
  - o Hough-based transforms
  - o 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

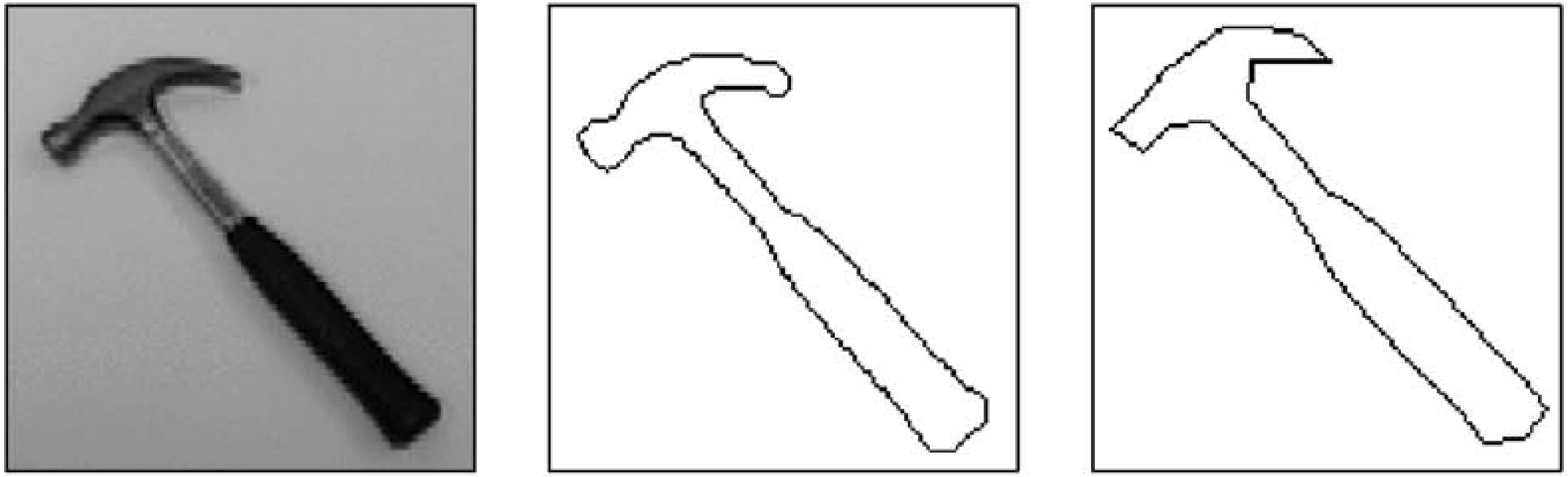
Objectives

- 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

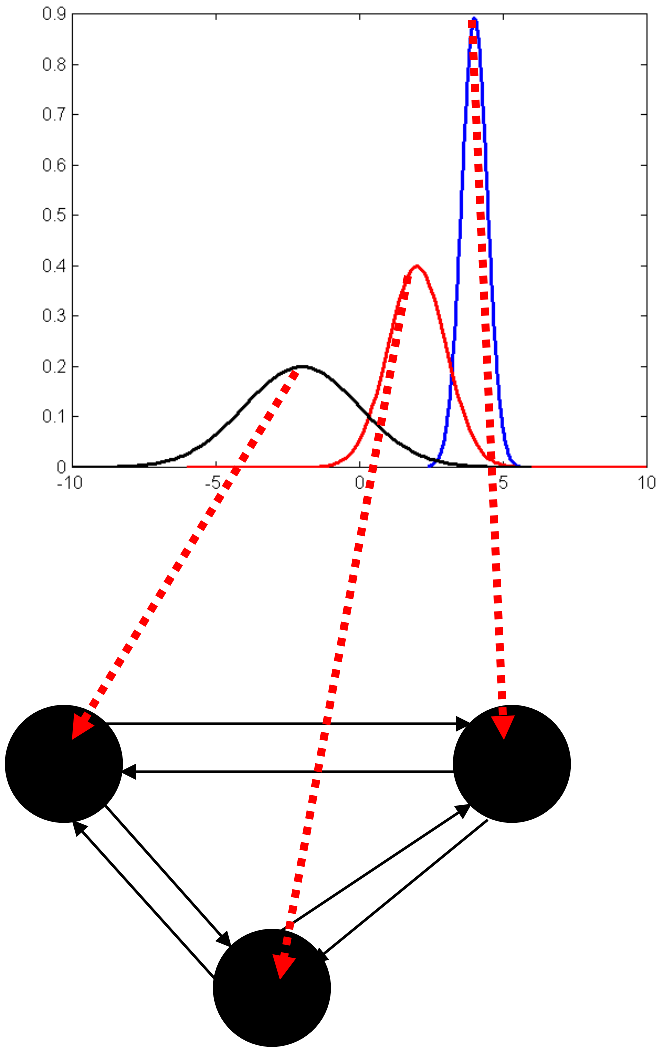
Hidden Markov Model

- 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

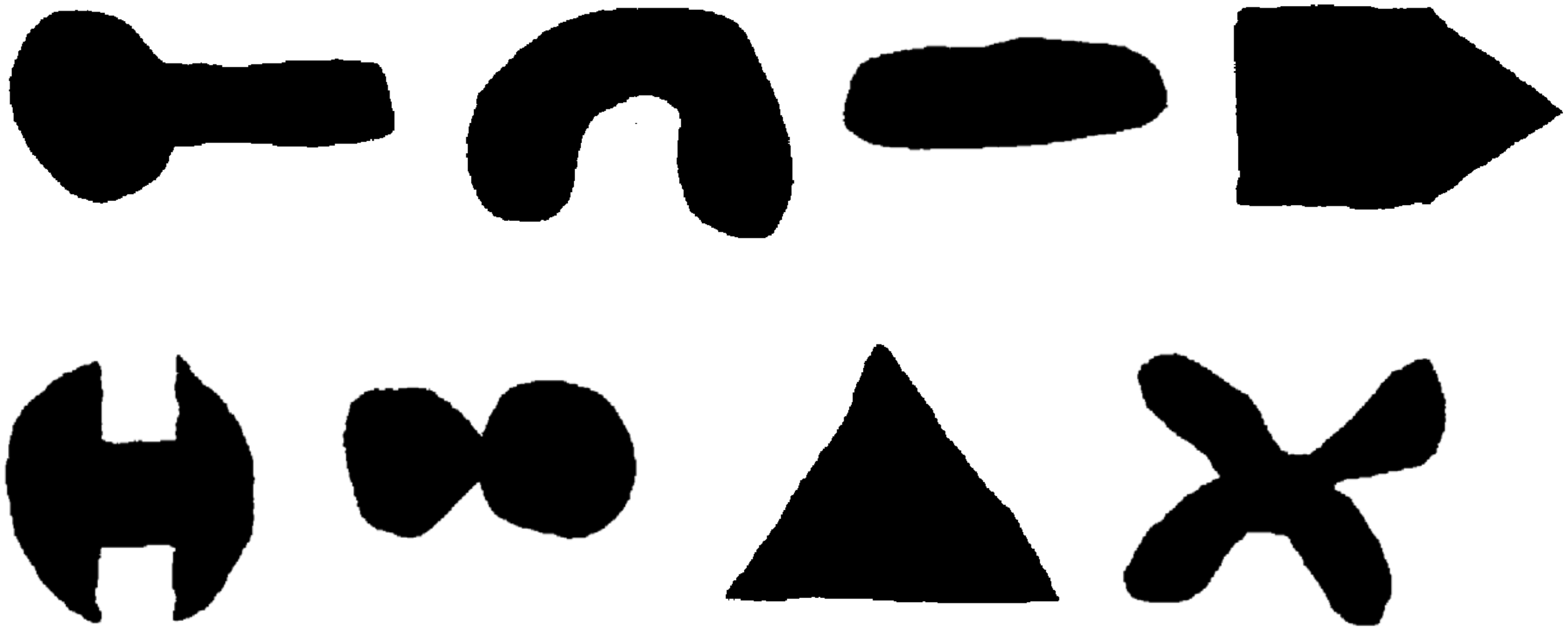
Strategy

- 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.

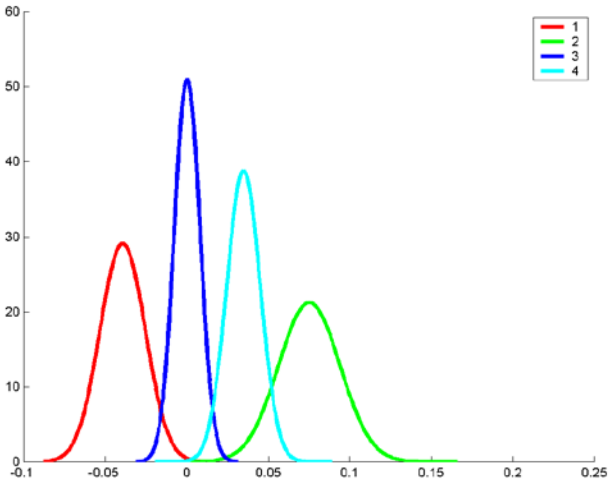
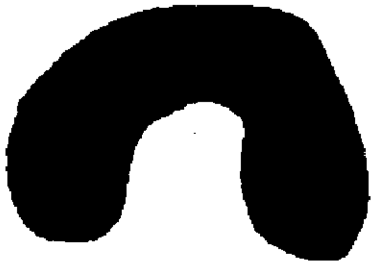
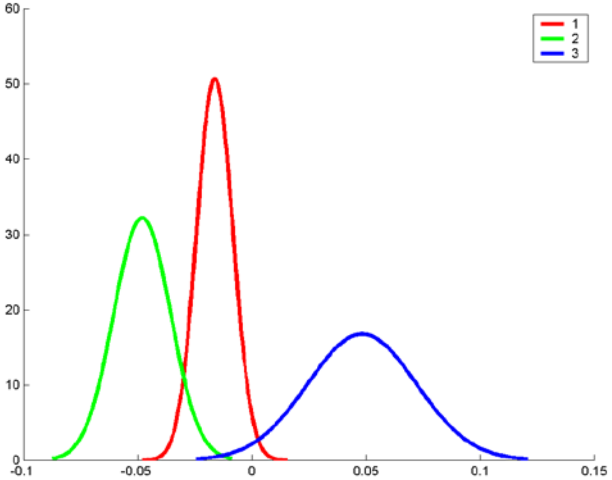
Strategy (cont.)

- 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 $P(O|\lambda_i)$.

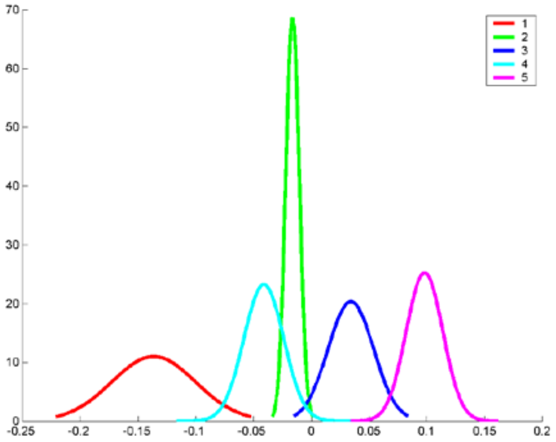
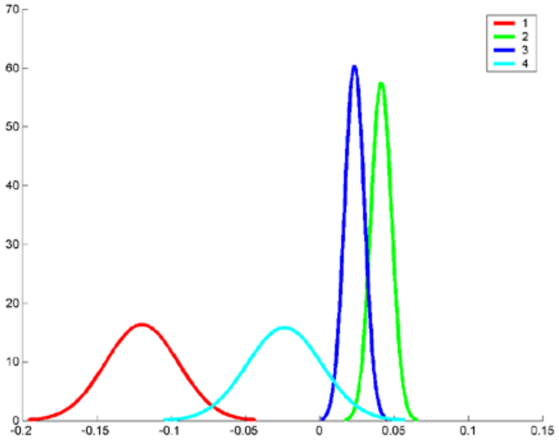
Experimental: The test set



The models

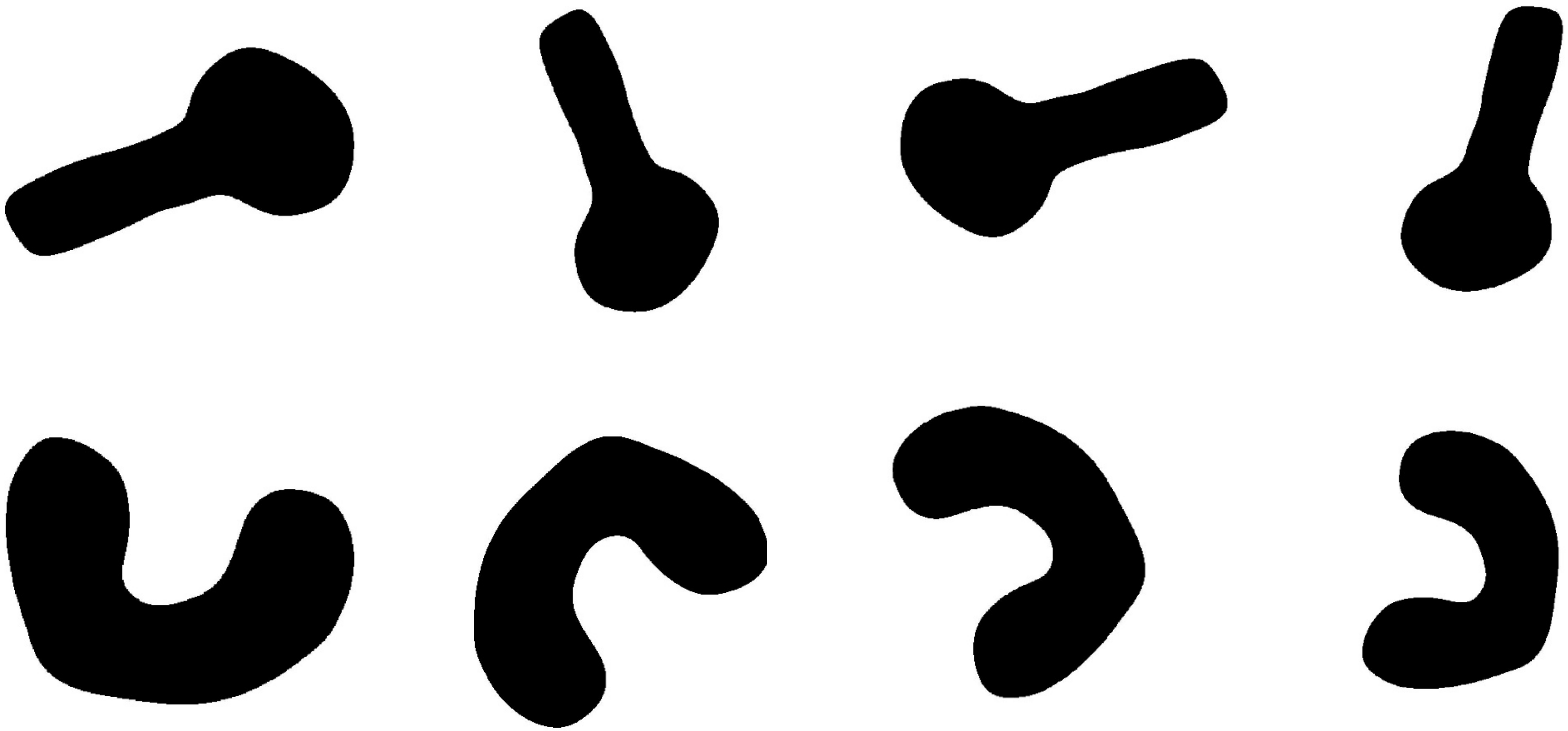
<table>
<tr><th>Shape</th><th>Emission Probability</th><th>Transition Probability</th></tr>
<tr><td></td><td>
</td><td><table>
<tr><td>0.94</td><td>0.00</td><td>0.06</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.96</td><td>0.00</td><td>0.04</td></tr>
<tr><td>0.02</td><td>0.00</td><td>0.96</td><td>0.02</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)

<table>
<tr><th>Shape</th><th>Emission Probability</th><th>Transition Probability</th></tr>
<tr><td></td><td>
</td><td>
<table>
<tr><td>0.92</td><td>0.00</td><td>0.00</td><td>0.08</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.97</td><td>0.01</td><td>0.02</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.00</td><td>0.89</td><td>0.04</td><td>0.07</td></tr>
<tr><td>0.09</td><td>0.11</td><td>0.00</td><td>0.80</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.00</td><td>0.09</td><td>0.00</td><td>0.91</td></tr>
</table>
</td></tr>
<tr><td></td><td>
</td><td>
<table>
<tr><td>0.91</td><td>0.00</td><td>0.00</td><td>0.09</td></tr>
<tr><td>0.00</td><td>0.95</td><td>0.05</td><td>0.00</td></tr>
<tr><td>0.00</td><td>0.06</td><td>0.92</td><td>0.02</td></tr>
<tr><td>0.08</td><td>0.00</td><td>0.08</td><td>0.83</td></tr>
</table>
</td></tr>
</table>

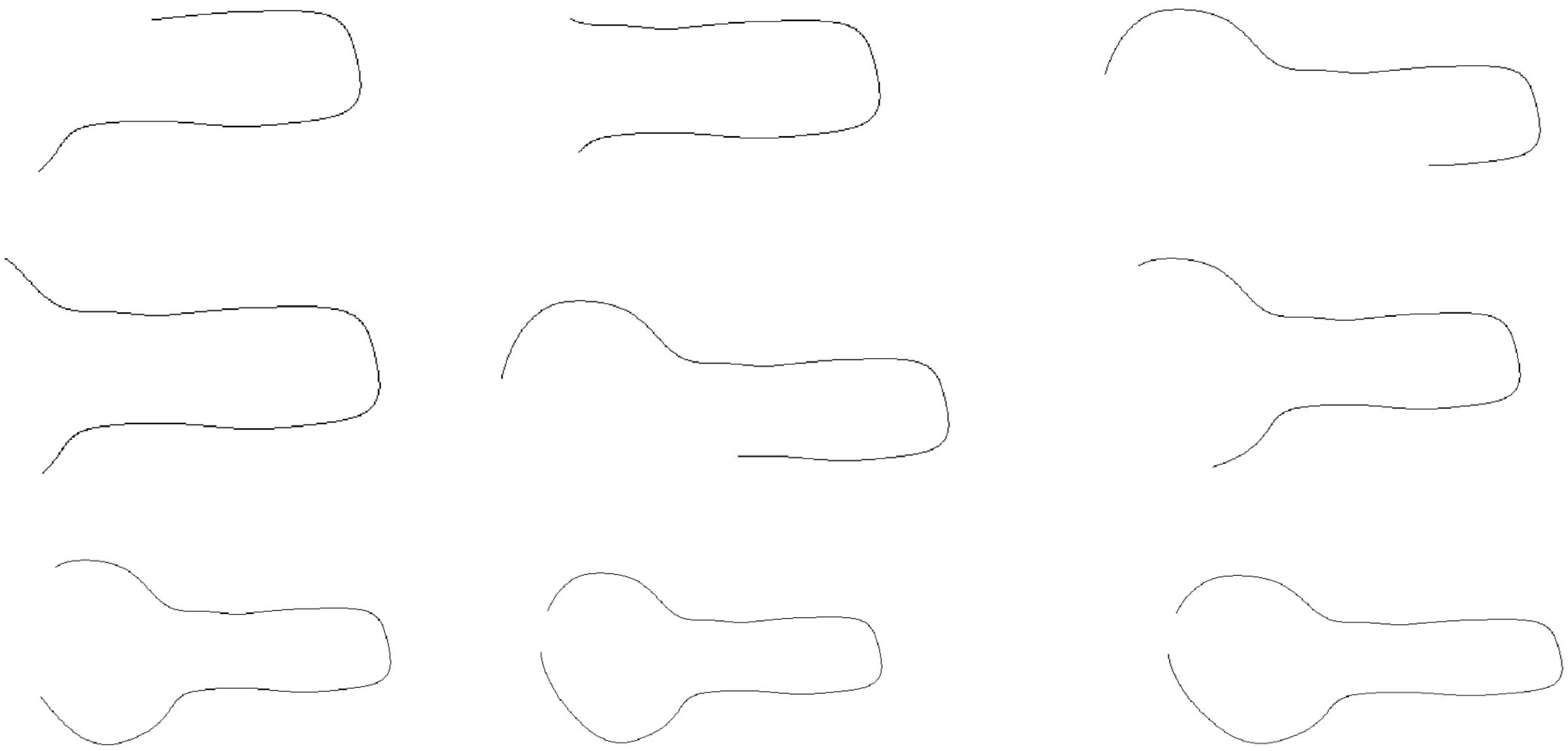
Rotations

- 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)



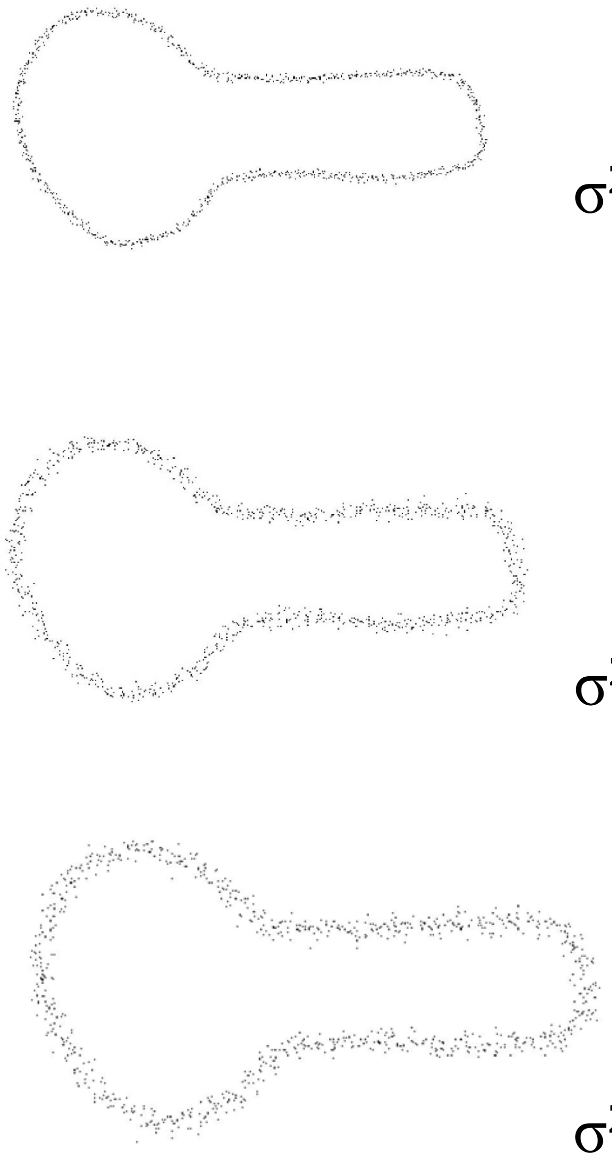
Occlusions: results

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



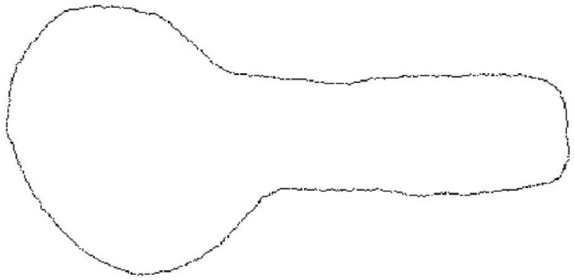
$\sigma^2=1$

$\sigma^2=4$

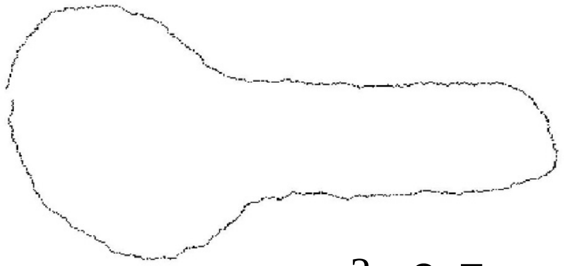
$\sigma^2=5$

Alternative Noise

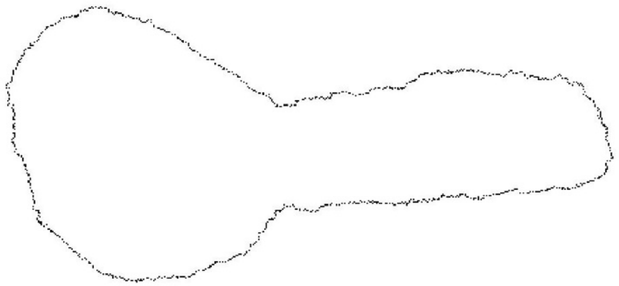
- Adding noise to the first derivative



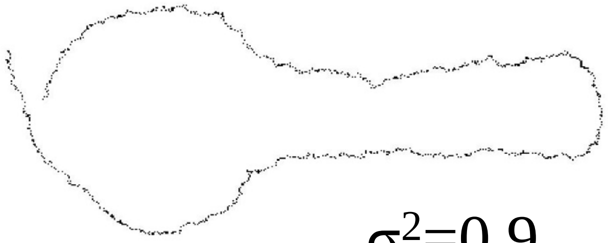
$\sigma^2=0.3$



$\sigma^2=0.5$



$\sigma^2=0.7$



$\sigma^2=0.9$

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

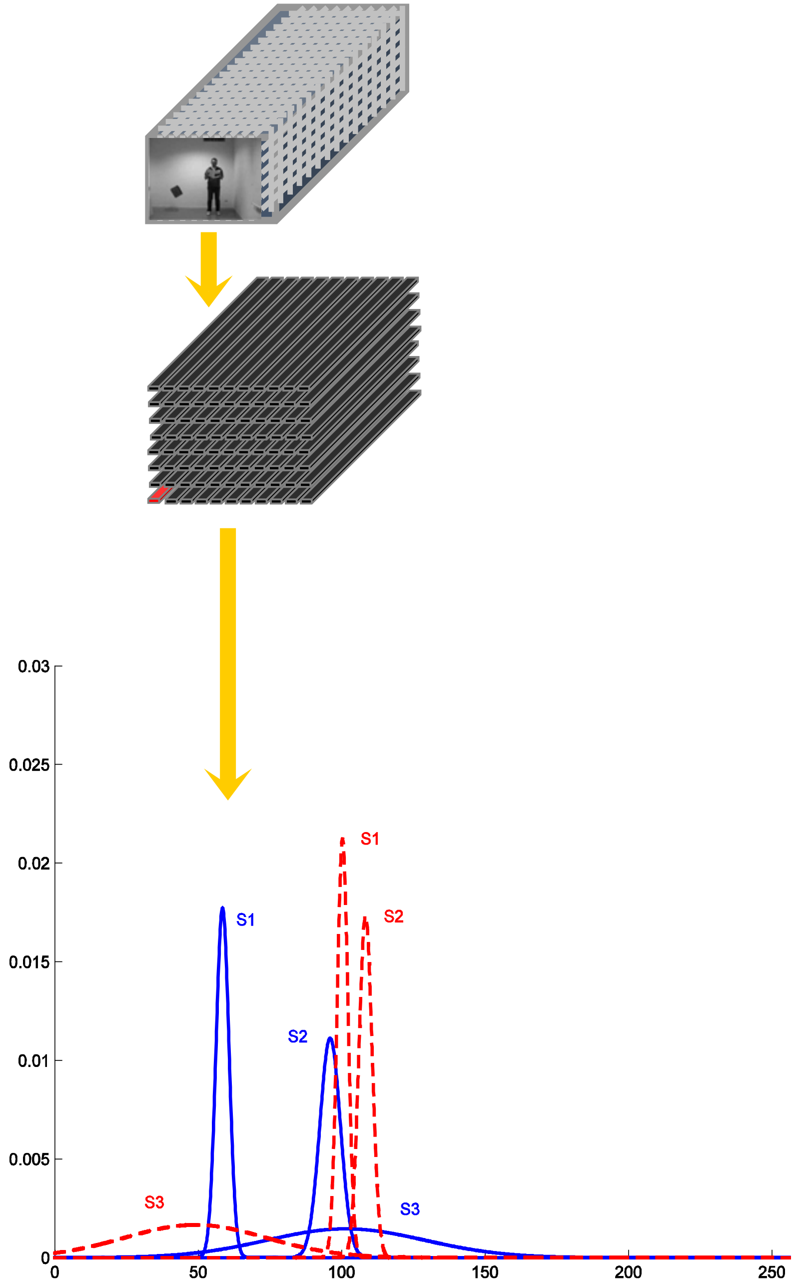
Video Analysis

Use of the HMMs: main idea

- Each pixel (signal) *v* of the sequence is modeled with an HMM λ_v=(A,B,π)
- 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