Facoltà di Scienze MM. FF. NN.

Università di Verona

A.A. 2010-11

Teoria e Tecniche del Riconoscimento

Modelli generativi: Hidden Markov Models, Observed Influence Models

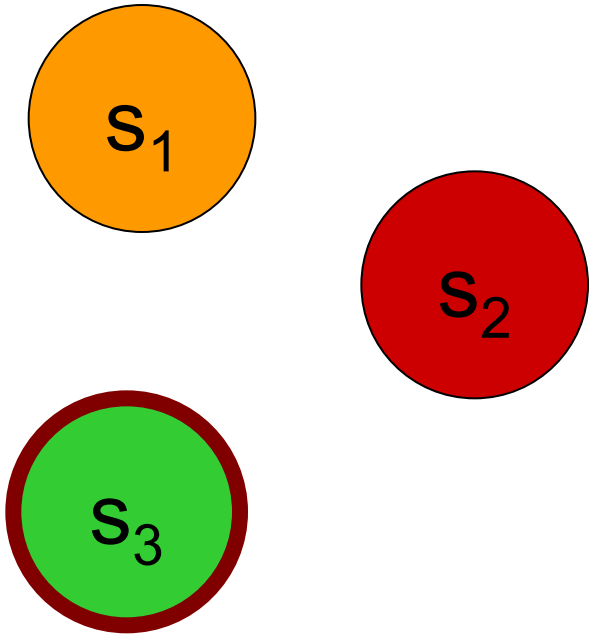
Sommario

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;

Processo di Markov (ordine 1)



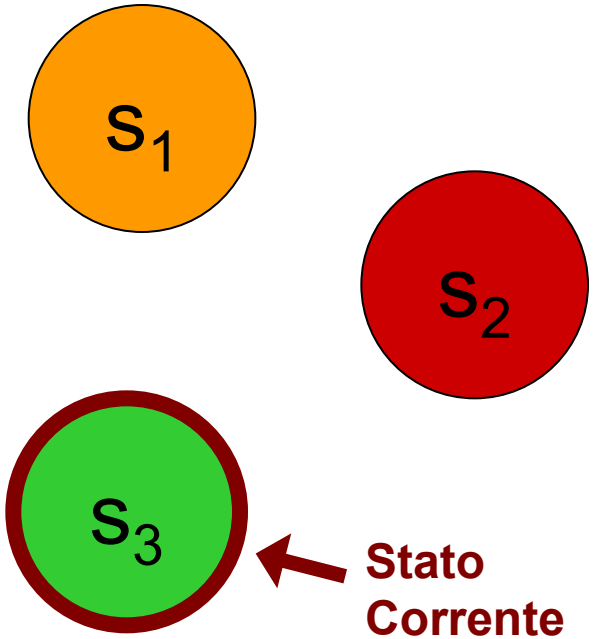
N=3
t=1

- Ha N stati , $s_1, s_2, \ldots, s_N$
- E' caratterizzato da passi discreti, t=1,t=2,…
- La probabilità di partire da un determinato stato è dettata dalla distribuzione:

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

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

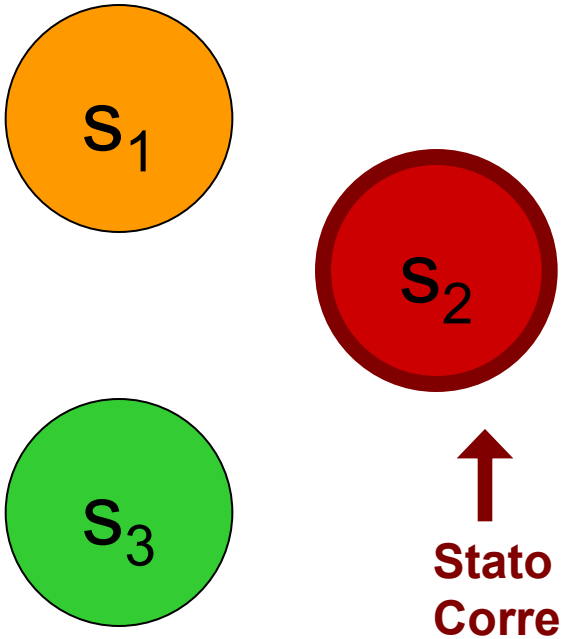
Processo di Markov



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



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

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

Stato Corrente

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

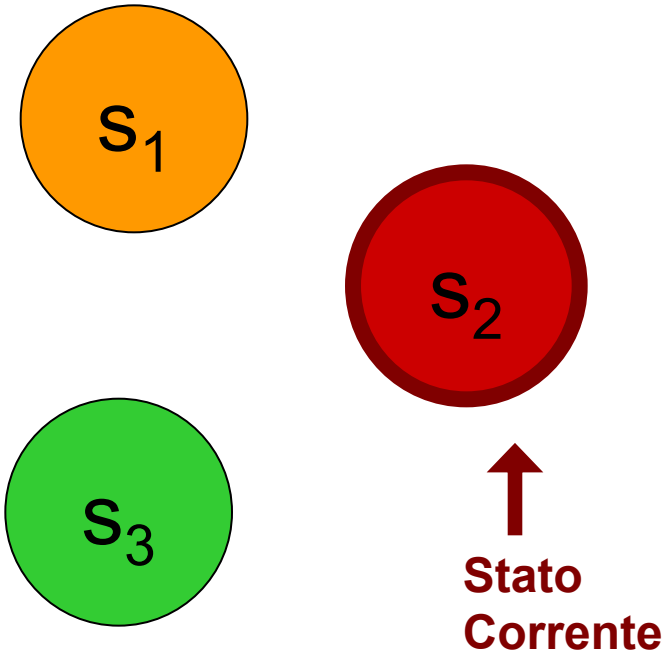
N=3 t=2

$q_2=s_2$

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

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

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*:

$A=$

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

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

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

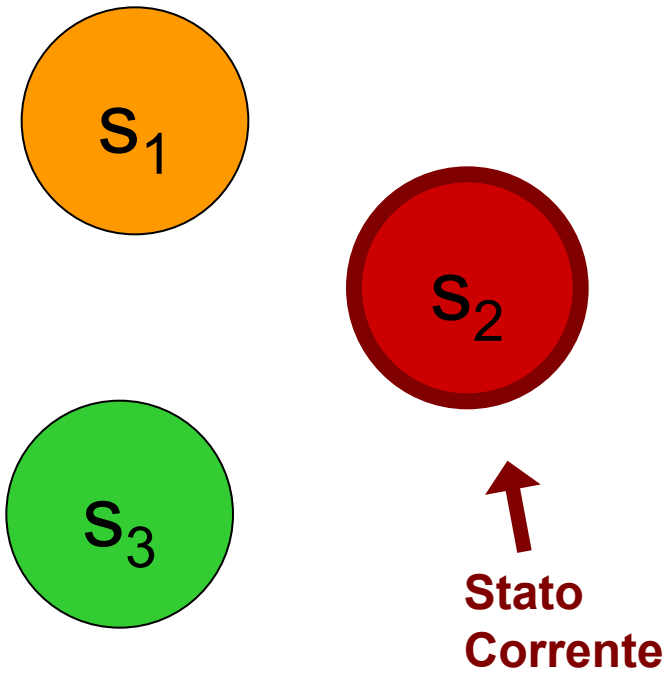
Cosa serve un modello stocastico?

- 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?

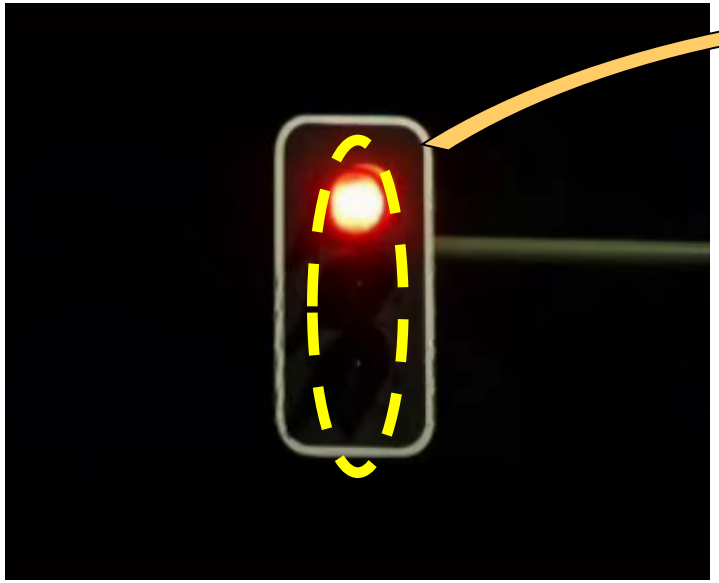
- **Addestramento o training**
  - Si costruiscono gli elementi costituenti del modello
- **Inferenze di vario tipo (interrogo il modello):**
  - Probabilità di una sequenza di stati, dato il modello
  - Proprietà invarianti etc.

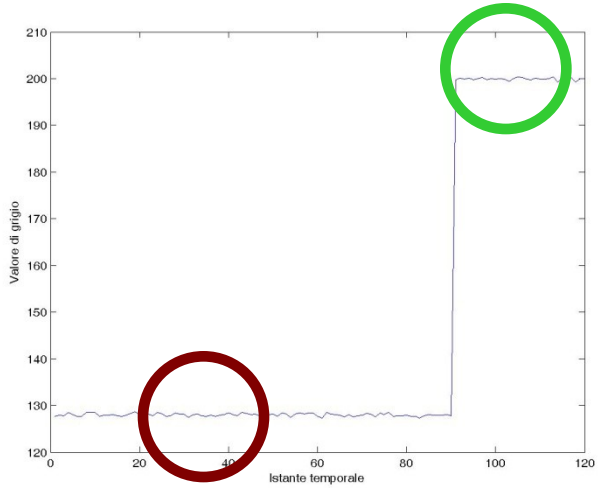
Cosa serve un modello di Markov?



- 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

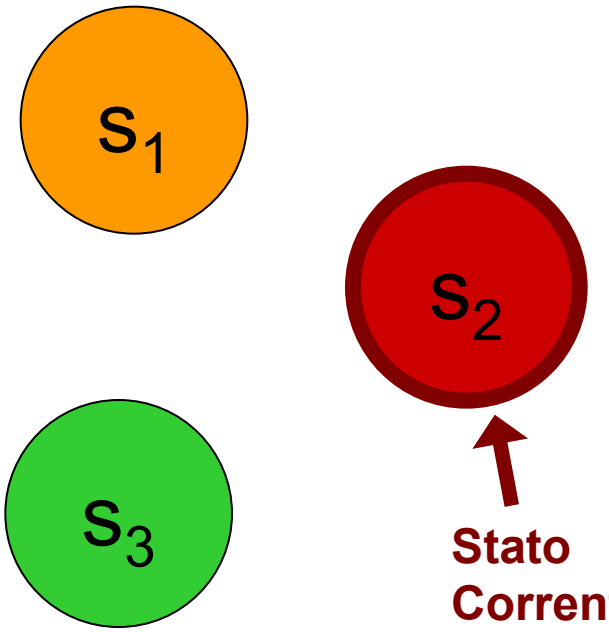


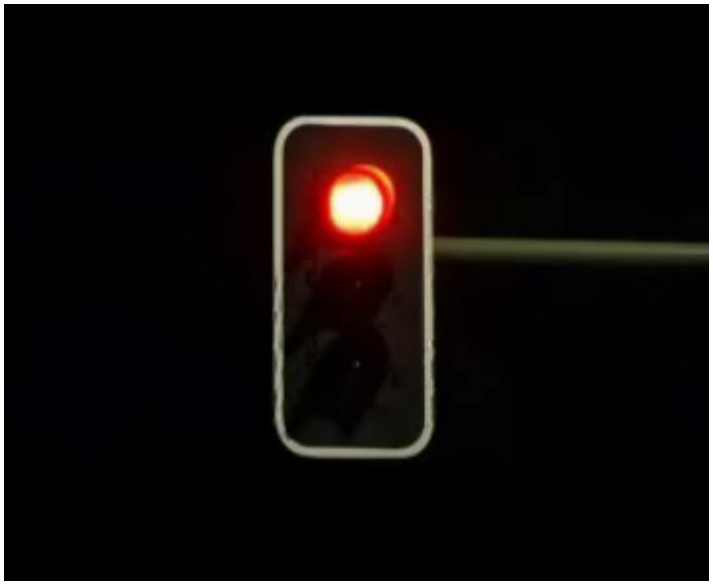


t=1 t=2 …

- E’ un sistema di cui gli stati sono:
  - **Espliciti** (le diverse lampade accese)
  - **Osservabili** (i colori delle lampade che osservo con la telecamera)

Semaforo – modello addestrato





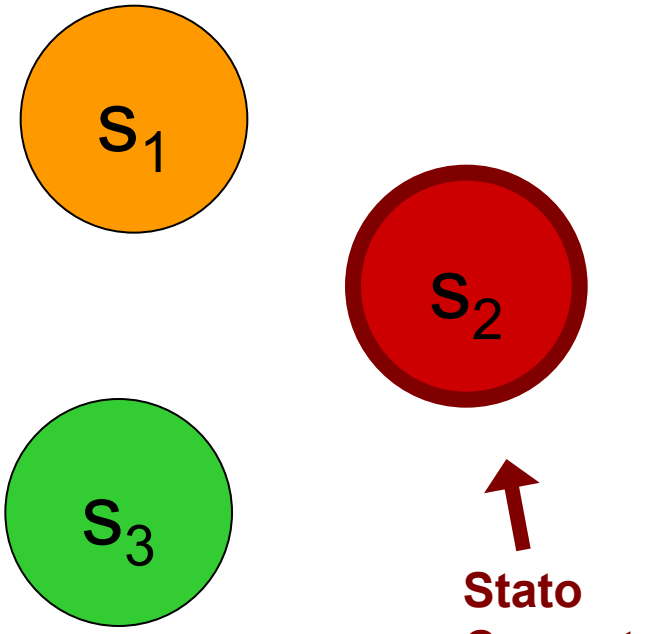
$\pi =$

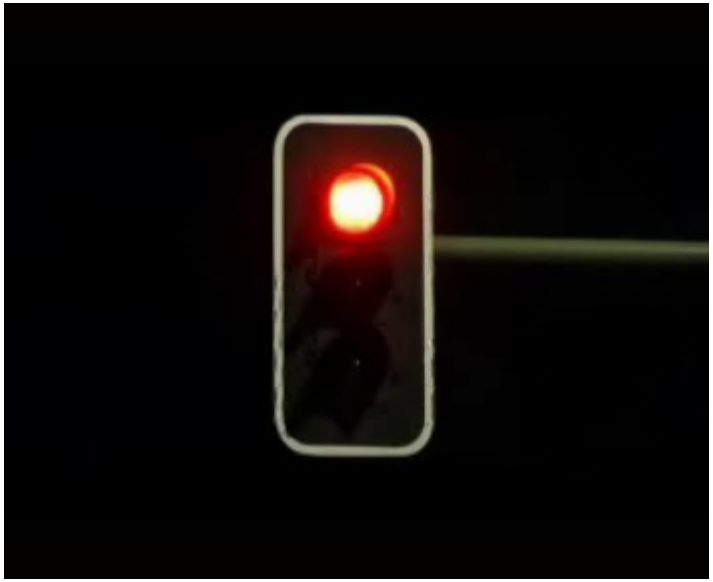
$\pi_1 = 0.33$	$\pi_2 = 0.33$	$\pi_3 = 0.33$

$A =$

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

Semaforo – inferenze





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

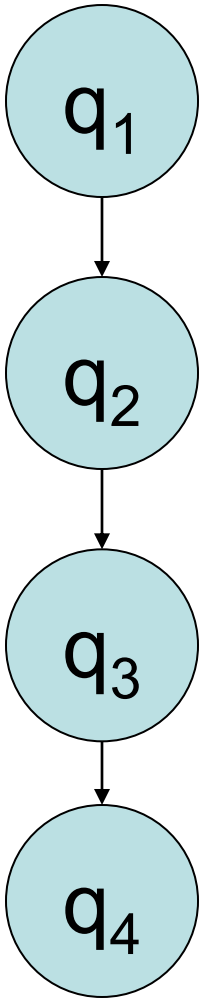
Inferenza: $P(O|\lambda) = P(O)$

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

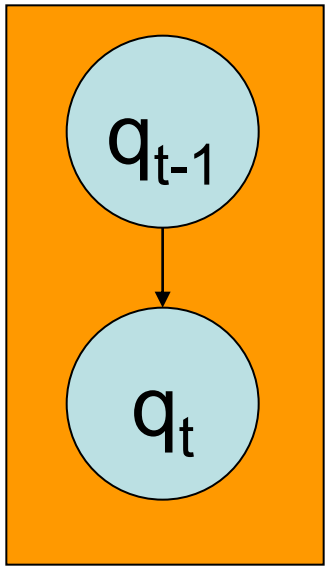
Inferenza importante

$$P(q_t,q_{t-1},\ldots,q_1) = P(q_t|q_{t-1},\ldots,q_1)\, P(q_{t-1},\ldots,q_1)$$
$$= P(q_t|q_{t-1})\, P(q_{t-1},q_{t-2},\ldots,q_1)$$
$$= P(q_t|q_{t-1})\, P(q_{t-1}|q_{t-2})\, P(q_{t-2},\ldots,q_1)$$
$$\ldots$$
$$= P(q_t|q_{t-1})\, P(q_{t-1}|q_{t-2})\ldots P(q_2|q_1)P(q_1)$$

Modello grafico



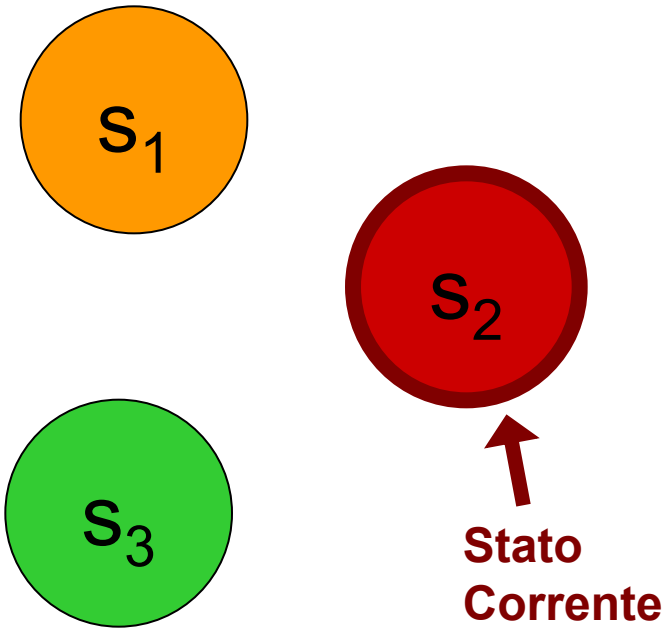
- La struttura grafica di tale probabilità congiunta si scrive in questa forma, dove

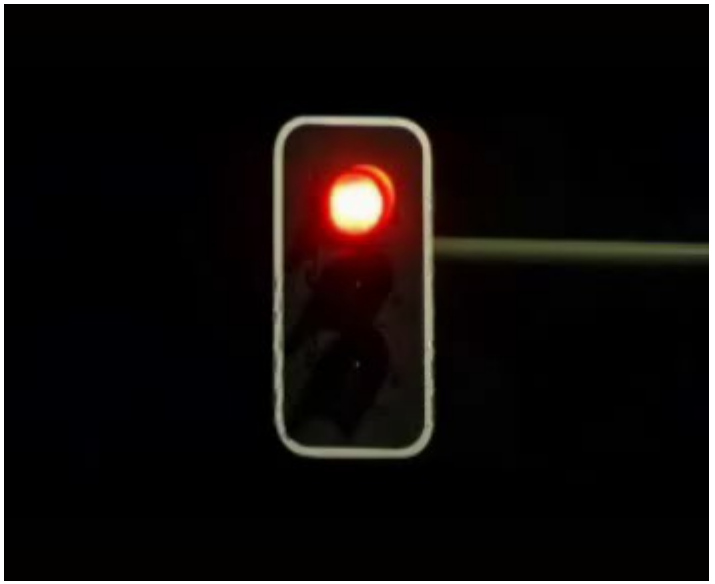


$$= P(q_t|q_{t-1})$$

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

Semaforo – inferenze, risposta





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

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

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

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

Seconda inferenza importante

- Calcolo della probabilità $P(q_T = s_j)$
- STEP 1: Valuto come calcolare P(Q) per ogni cammino di stati $\mathbf{Q}=\{q_1,q_2,\ldots,q_T=s_j\}$, ossia

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

- STEP 2: Uso questo metodo per calcolare $P(q_T = s_j)$, ossia:
  - $P(q_T = s_j) = \sum_{\mathbf{Q}\in \text{cammini di lunghezza T che finiscono in } s_j} P(\mathbf{Q})$
  - 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:

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

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

Seconda inferenza importante (3)

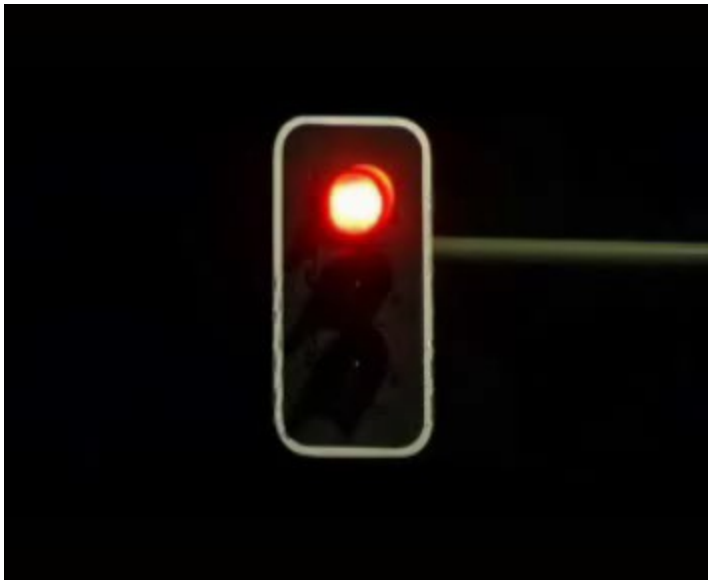
$$\sum_{i=1}^{N} P(q_{t+1} = s_j, q_t = s_i) =$$

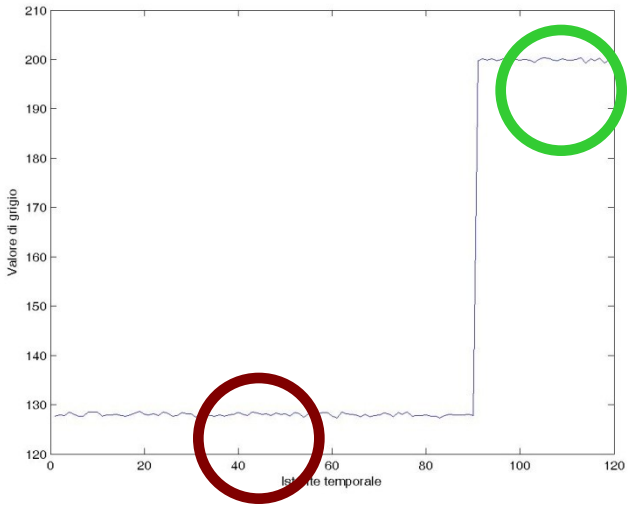
$$= \sum_{i=1}^{N} P(q_{t+1} = s_j \mid q_t = s_i) P(q_t = s_i) = \sum_{i=1}^{N} a_{ij} p_t(i)$$

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

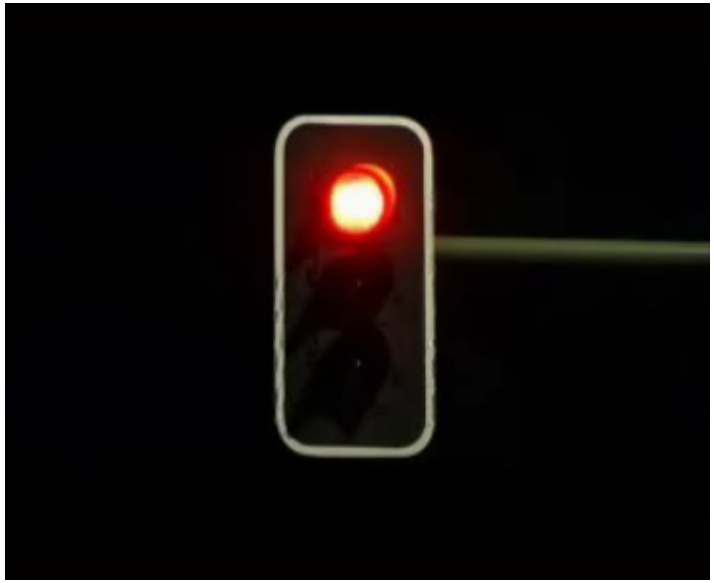
Limiti dei modelli markoviani

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

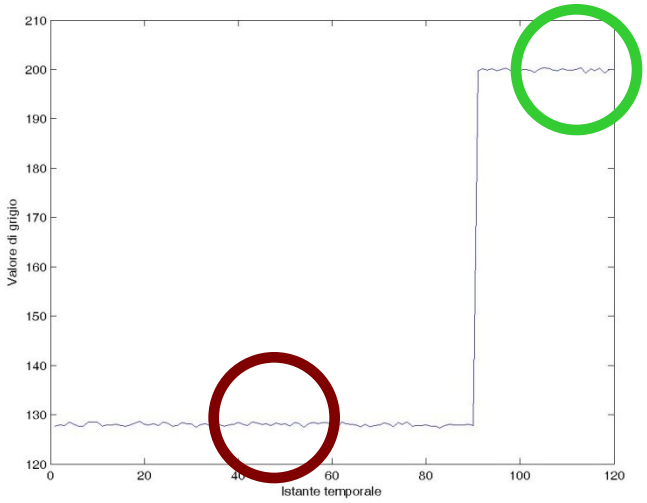




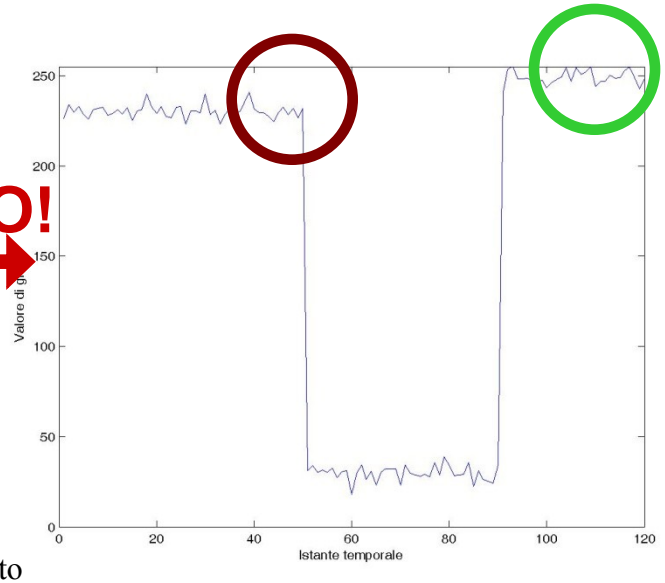
Limiti dei modelli markoviani



OK



NO!

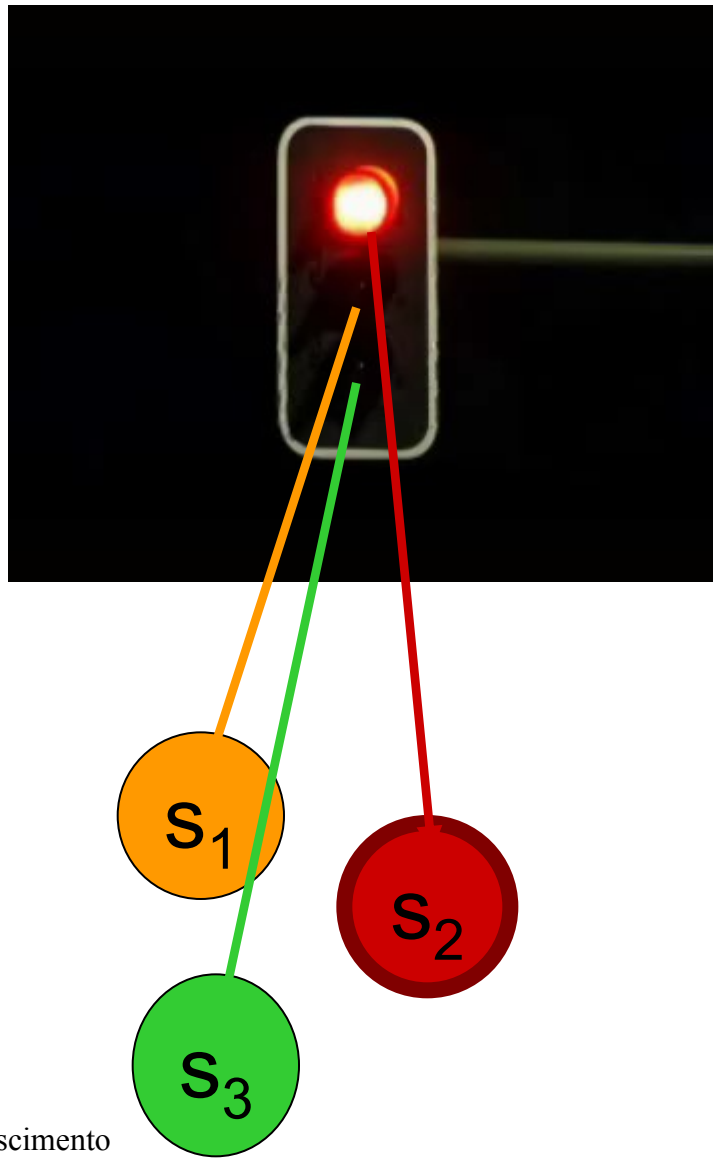


Limiti dei modelli markoviani

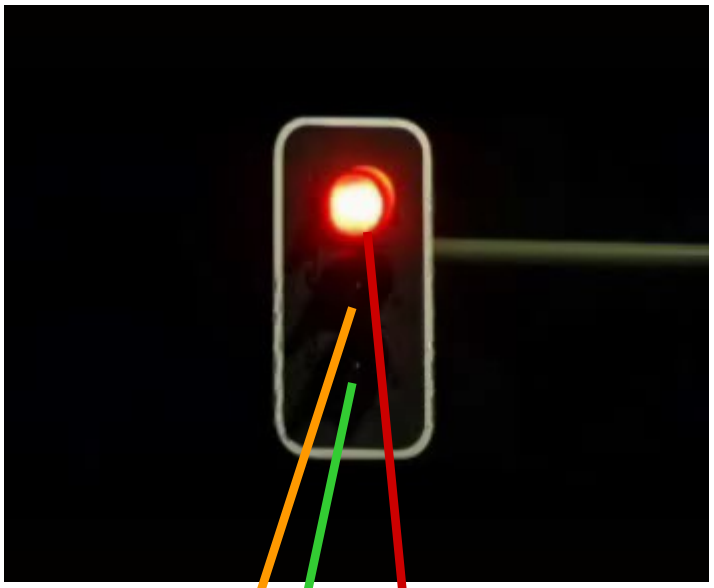
2. (Più importante!) Nel caso del semaforo lo stato è **esplicito**, (una particolare configurazione del semaforo), e **valutabile direttamente tramite l'osservazione**

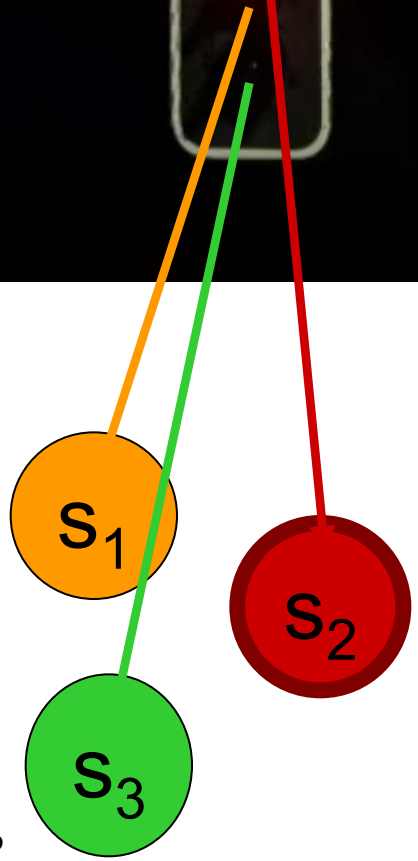
   (lo stato corrisponde al colore del semaforo)

- Non sempre accade così!



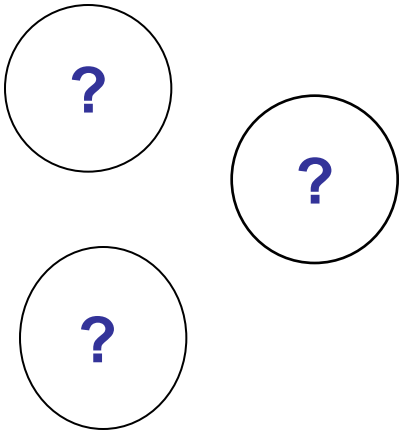
Limiti dei modelli markoviani



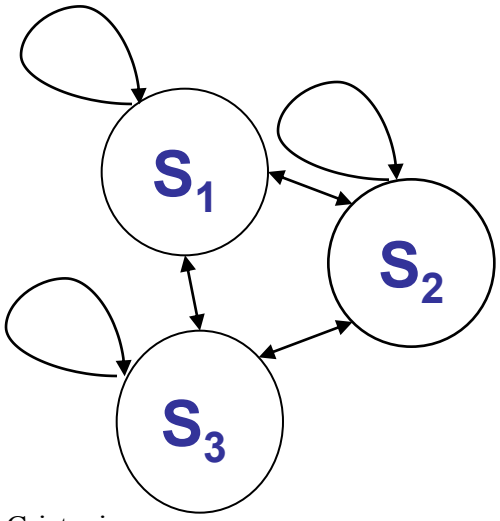


Limiti dei modelli markoviani

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

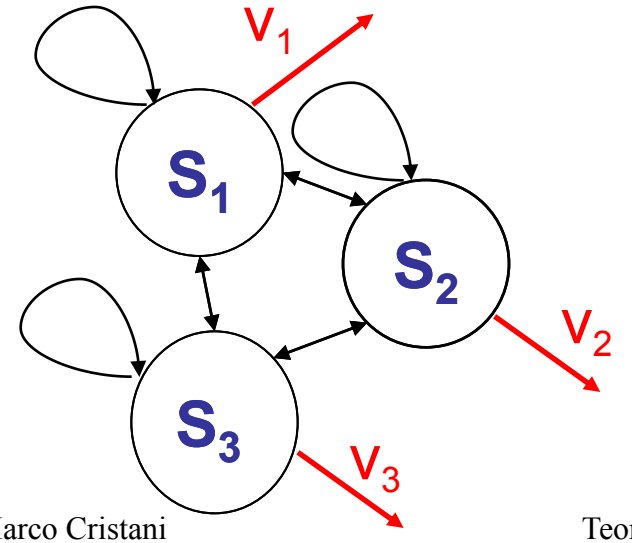


Limiti dei modelli markoviani



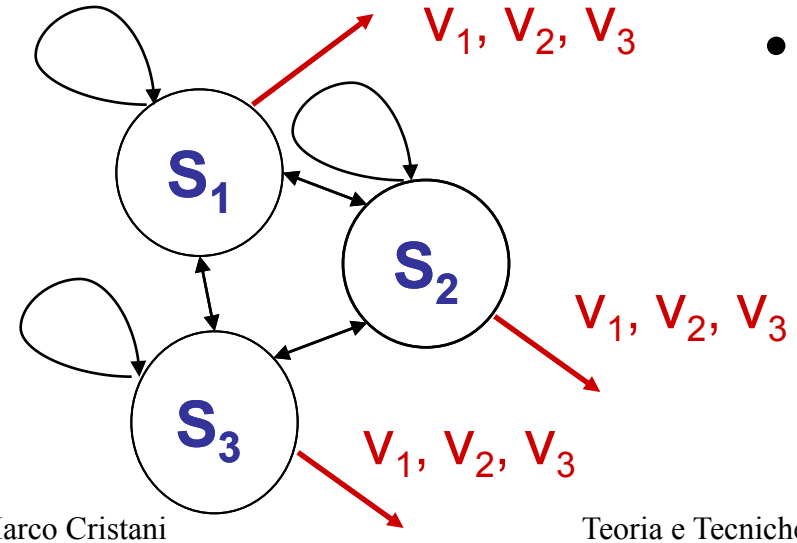
- *Osservo* il filmato: *osservo* che c'è un **sistema che evolve**, ma non riesco a capire quali siano gli stati regolatori.
- Il sistema comunque evolve a **stati**; lo capisco *osservando* il fenomeno (introduco una conoscenza "a priori")

Limiti dei modelli markoviani



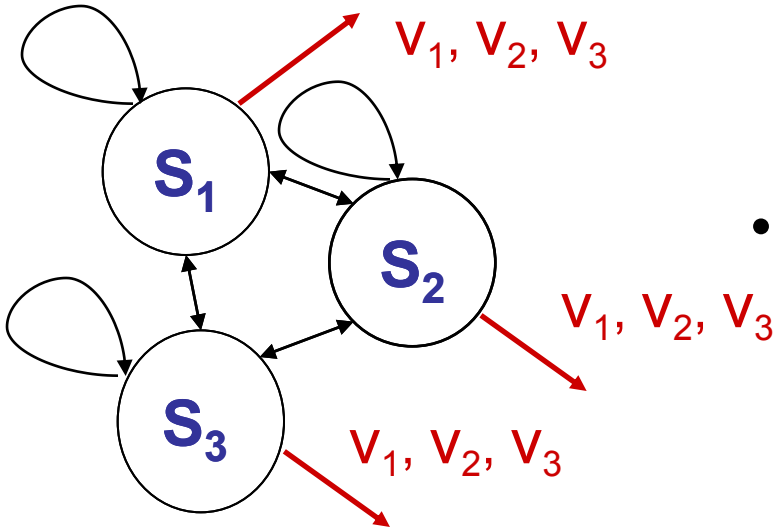
- Meglio: il sistema evolve grazie a degli **stati "nascosti"** (gli stati del semaforo, che però non vedo e di cui ignoro l'esistenza) di cui riesco ad ***osservare*** solo le probabili "conseguenze", ossia i flussi delle auto

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)



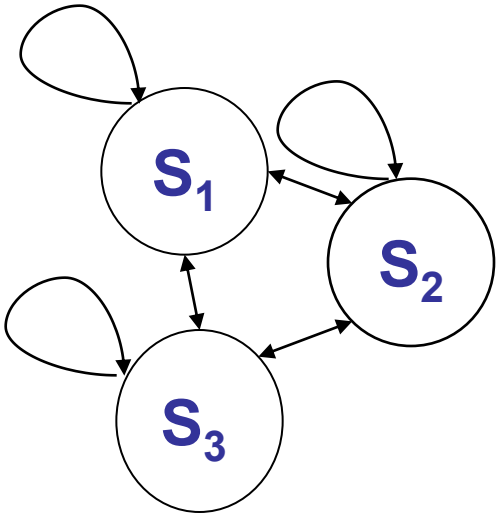
- Gli Hidden Markov Model si inseriscono in questo contesto
- Descrivono probabilisticamente la *dinamica di un sistema* rinunciando ad identificarne direttamente le cause
- Gli *stati* sono identificabili solamente tramite le *osservazioni*, in maniera probabilistica.

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
- Suppongo ora di avere un HMM addestrato, ossia in grado di descrivere un sistema stocastico come descritto sopra ...

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 $\pi=\{\pi_i\}$,



$\boldsymbol{\pi}=$

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

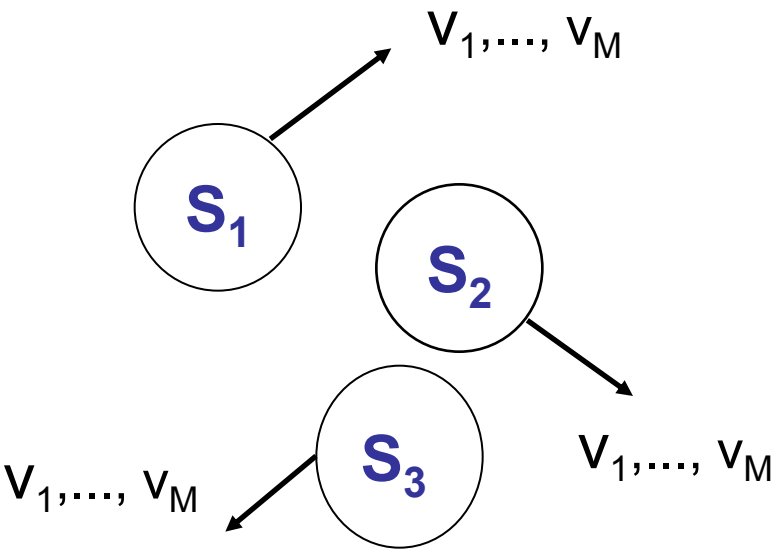
$\boldsymbol{A}=$

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

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 $B=\{b_{jk}\}=\{b_j(v_k)\}$, *che indica la probabilità di emissione del simbolo* v_k quando lo stato del sistema è s_j, $P(v_k|s_j)$



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

HMM: definizione formale

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

$\pi=$

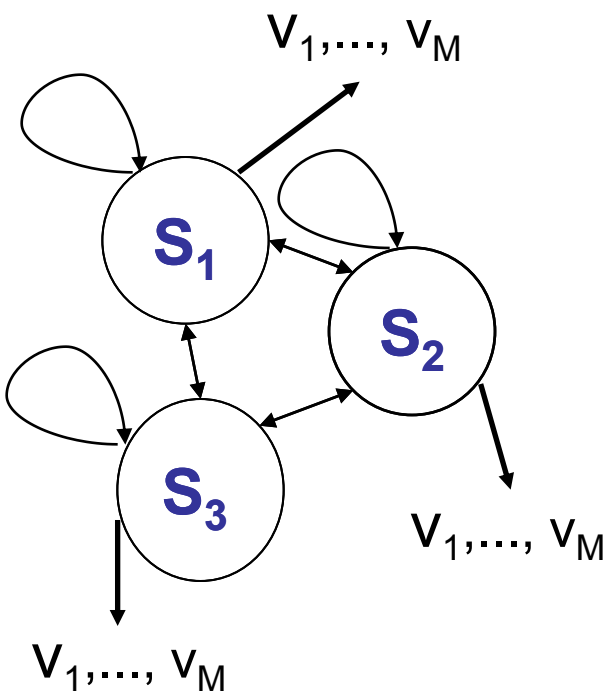
$\pi_1= 0.33$	$\pi_2= 0.33$	$\pi_3= 0.33$

$A=$

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

$B=$

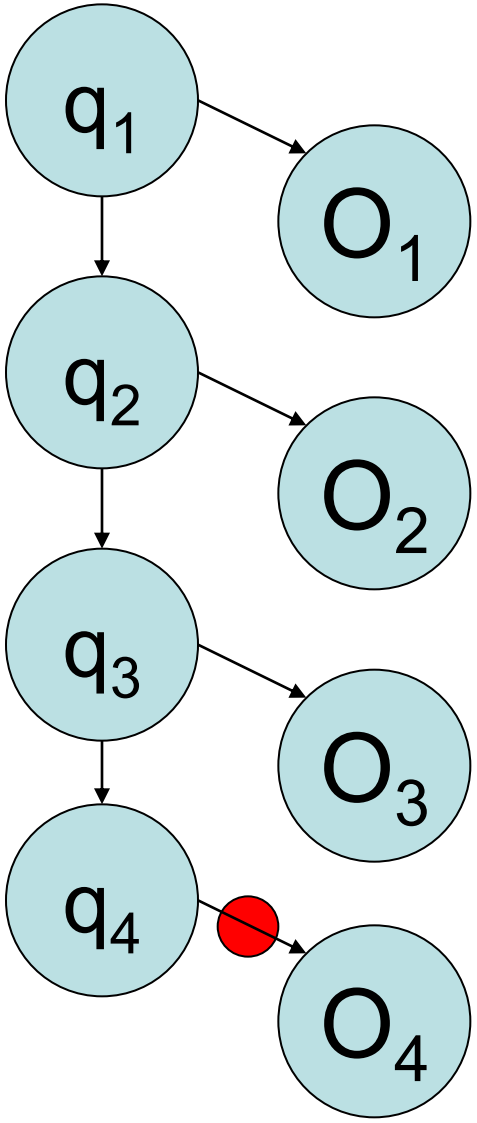
$b_{11}=0.8$	$b_{21}=0.1$	$b_{31}=0.1$
$b_{12}= 0.1$	$b_{22}= 0.8$	$b_{32}= 0.1$
$b_{1M}= 0.1$	$b_{2M}= 0.1$	$b_{3M}=0.8$



Assunzioni sull'osservazione

- Indipendenze condizionali

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



Urn & Ball – An easy example

- ***N*** large urns with ***M*** coloured balls in each
- Urns are the states and balls are the observable events
- Transition matrix for changing between urns
- Each urn has observation probabilities to determine which ball is chosen

Urn & Ball – An Example

Urn 1

Urn 2

Urn 3

P(red) = $b_1(1)$ P(red) = $b_2(1)$ P(red) = $b_3(1)$

P(blue) = $b_1(2)$ P(blue) = $b_2(2)$ P(blue) = $b_3(2)$

P(green) = $b_1(3)$ P(green) = $b_2(3)$ P(green) = $b_3(3)$

P(purple) = $b_1(4)$ P(purple) = $b_2(4)$ P(purple) = $b_3(4)$

… … …

Urn & Ball – An Example

- Initial probability to determine first urn
- At each time interval:
  - Transition probability determines the urn
  - Observation probability determines the ball
  - Ball colour added to observed event sequence and returned to urn
- Transition probability dependent on previous urn

Example Sequence Creation using Urn Ball

1. From π, 1st urn = Urn 1
2. Using $b_1(k)$, 1st ball = Red
3. From a_{1j}, 2nd urn = Urn 3 etc…

- Get observation sequence
  - Red, Blue, Purple, Yellow, Blue, Blue
- From state sequence
  - Urn1, Urn 3, Urn 3, Urn 1, Urn, 2, Urn 1

Problemi chiave del modello HMM

Evaluation

1. Data una stringa d'osservazione **O=(**O_1,O_2,…,O_t, …,O_T) calcolare P(**O|** λ) → *procedura di forward*

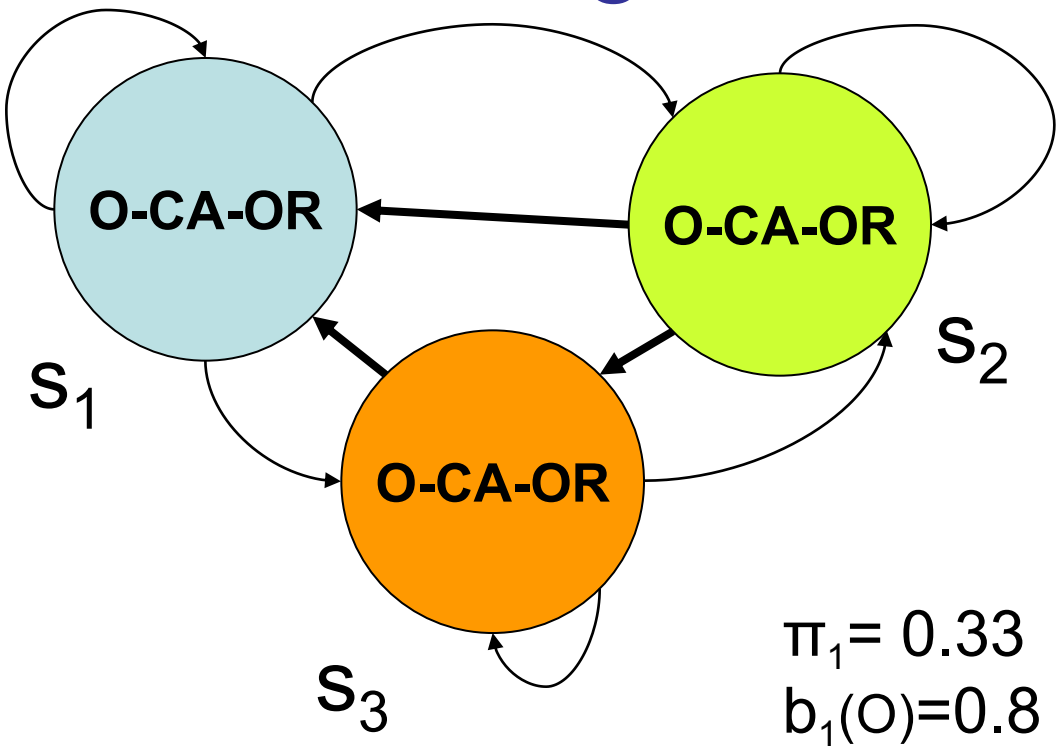
Decoding

2. Data una stringa d'osservazione **O** e un modello HMM λ, calcolare la più probabile sequenza di stati s_1…s_T che ha generato **O** → *procedura di Viterbi*

Training

3. Dato un insieme di osservazioni {**O**}, determinare il miglior modello HMM λ, cioè il modello per cui P(O| λ) è massimizzata → *procedura di Baum Welch (forward-backword)*

HMM – generatore di stringhe



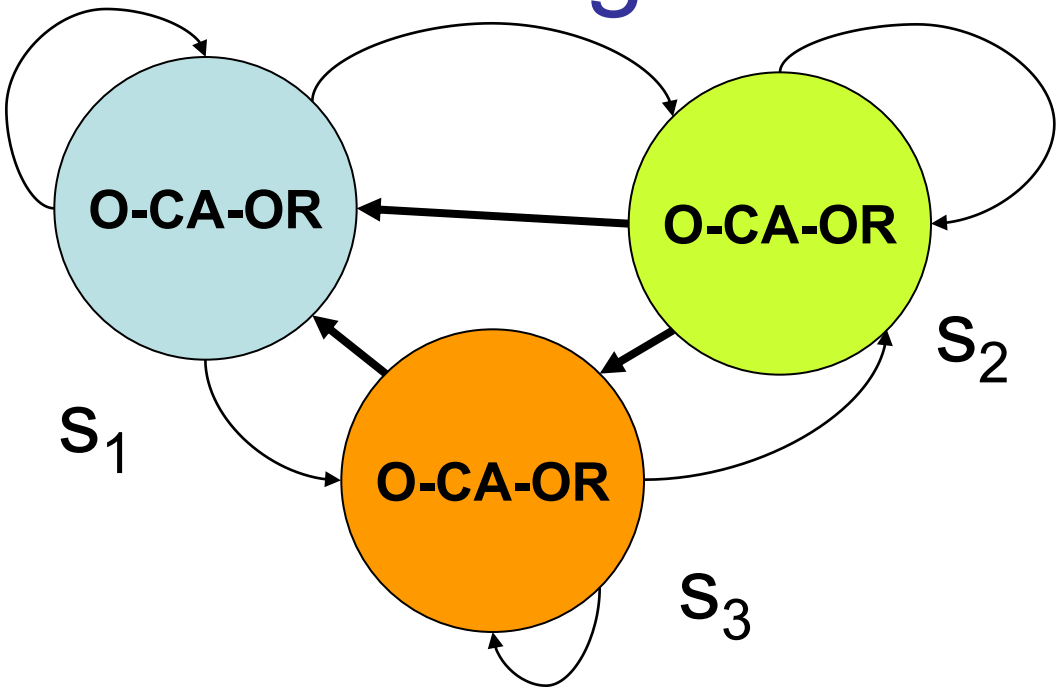
- 3 stati: s_1 ,s_2, s_3;
- 3 simboli: O,CA,OR

$\pi_1 = 0.33$
$b_1(O)=0.8$
$b_1(OR)= 0.1$
$b_1(CA)= 0.1$
$a_{11}= 0$
$a_{21}= 1/3$
$a_{31}= 1/2$

$\pi_2 = 0.33$
$b_2(O)=0.1$
$b_2(OR)= 0.0$
$b_2(CA)= 0.9$
$a_{12}= 1$
$a_{22}= 2/3$
$a_{32}= 1/2$

$\pi_3 = 0.33$
$b_3(O)=0.1$
$b_3(OR)= 0.8$
$b_3(CA)= 0.1$
$a_{13}= 0$
$a_{23}= 0$
$a_{33}= 0$

HMM – generatore di stringhe



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

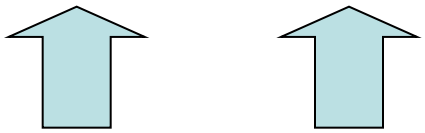
Il nostro problema è che gli stati non sono direttamente osservabili!

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

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

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

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

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

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

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

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

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

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

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:

$$P(O|\lambda) = \sum_{\text{All sequences } Q_1,\ldots,Q_T} \pi_{Q_1} b_{Q_1}(O_1) a_{Q_1Q_2} b_{Q_2}(O_2) a_{Q_2Q_3} \ldots$$

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

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

ossia:

  - *Abbiamo visto le prime t osservazioni*
  - *Siamo finiti in s_i, come 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 $\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = s_i | \lambda)$
- Voglio calcolare:

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

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

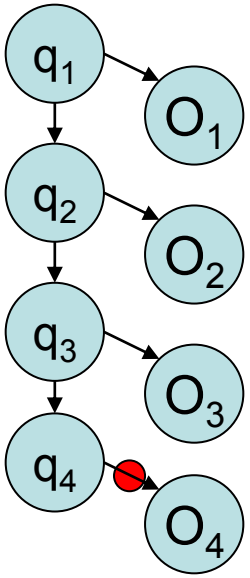
$$= \quad (O_1,O_2,\ldots,O_t, \quad O_{t+1}, q_{t+1}=s_j)$$

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

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

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

$$= \sum_{i=1}^{N} \quad _{ij}\ \alpha_t(i)]\ b_j(O_{t+1})$$



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:

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

di complessità $O(N^2T)$

- Ma anche altre quantità utili, per esempio:

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

Risposta al problema 1: evaluation

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

$$\beta_t(j) = P(O_{t+1} \ldots O_T \mid q_t = s_j, \lambda) =$$

$$= \sum_{i=1}^{N} \quad _{t+1}(i)\, a_{ij}\, b_j(O_{t+1})$$

e quindi

$$P(O \mid \lambda) = \sum_{j=} \quad (j)\beta_t(j) \quad \forall$$

$$= \sum_{j=} \quad (j) \quad \textit{verificare!!}$$

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

$$\underset{\mathbf{Q}}{\operatorname{argmax}}\ P(\mathbf{Q} \mid O_1 O_2 \ldots O_T)\ ?$$

- Strategia forza bruta:

$$\underset{\mathbf{Q}}{\operatorname{argmax}}\ \frac{P(O_1 O_2 \ldots O_T \mid \mathbf{Q}) P(\mathbf{Q})}{P(O_1 O_2 \ldots O_T)}$$

$$\propto \underset{\mathbf{Q}}{\operatorname{argmax}}\ P(O_1 O_2 \ldots O_T \mid \mathbf{Q}) P(\mathbf{Q})$$

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
  - 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:

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

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

Per induzione abbiamo

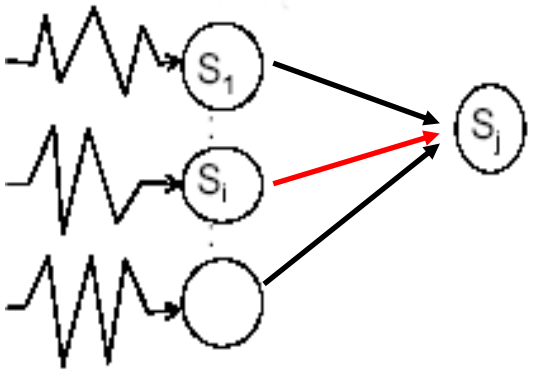
$$\delta_{t+1}(j) = [\max_i \delta_t(i) a_{ij}] \cdot b_j(O_{t+1}).$$

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

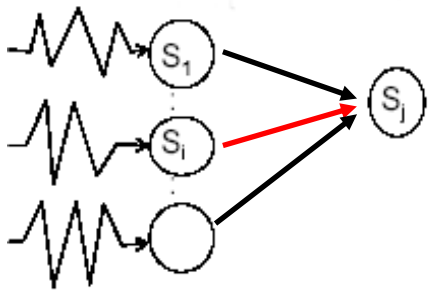
3) Termination:

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

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

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 $\lambda=(A,B,\pi)$, vengono stimati dalle *osservazioni di training*
- Di solito si usa la stima Maximum Likelihood

$$\lambda^* = \arg\max_{\lambda} \quad P(O_1 O_2 ... O_T \mid \lambda)$$

- Ma si possono usare anche altre stime

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

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

Definiamo

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

Tali quantità possono essere calcolate efficientemente (cfr. Rabiner)

$$\sum_{j=}^{N} \quad (i,j) = \quad (i)$$

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

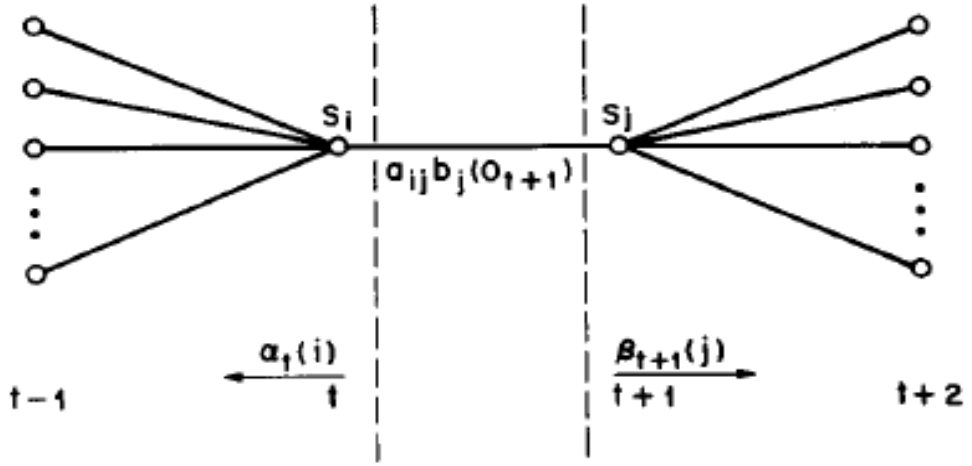
$$\sum_{t=1}^{T-1} \gamma \quad (i) = \text{numero atteso di transizioni passanti dallo stato i durante il cammino}$$

- 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, = λ , ossia

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

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

- Algoritmo:
  1) inizializzo il modello $\lambda = A_0, B_0, \pi_0)$
  2) il modello corrente è $\lambda = \lambda$
  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 $\lambda = \overline{A}, \overline{B}, \pi$ (M step)
  5) vai al passo 2, finchè non si verifica la terminazione
- Baum ha dimostrato che ad ogni passo:

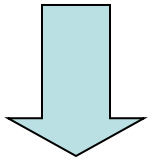
$$P(\mathrm{O}_1, \mathrm{O}_2, ..., \mathrm{O}_\mathrm{T} \mid \lambda > P(\mathrm{O}_1, \mathrm{O}_2, ..., \mathrm{O}_\mathrm{T} \mid \lambda)$$

- 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

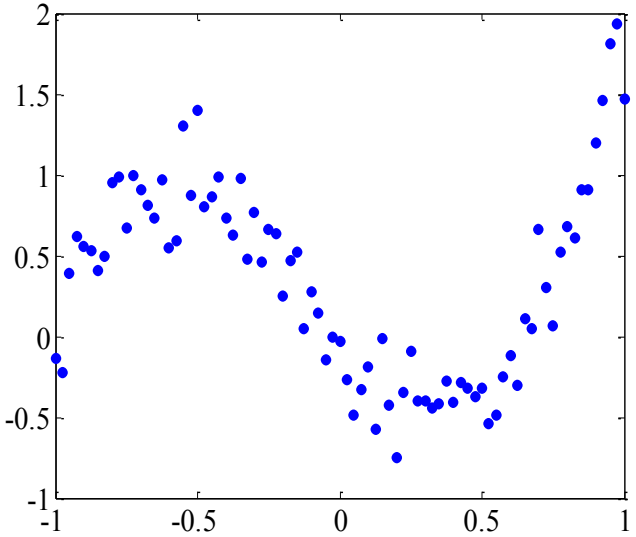
Model selection problem 1: selecting the number of states

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

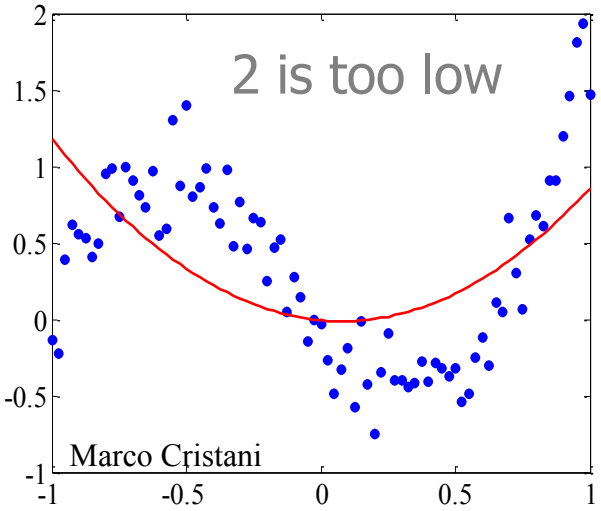
...let's understand the concept with a toy example

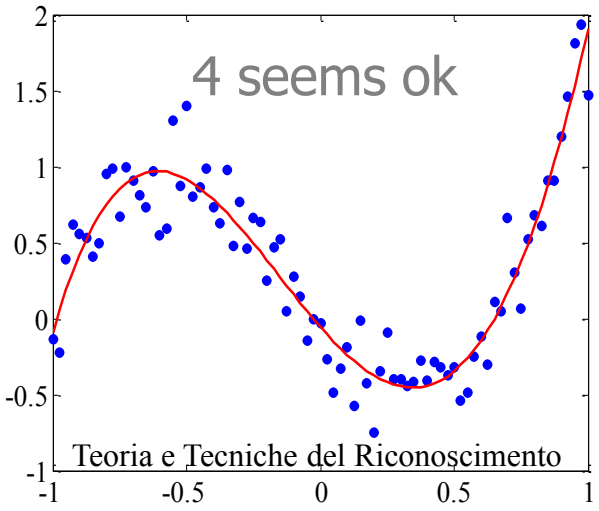
What is model selection?

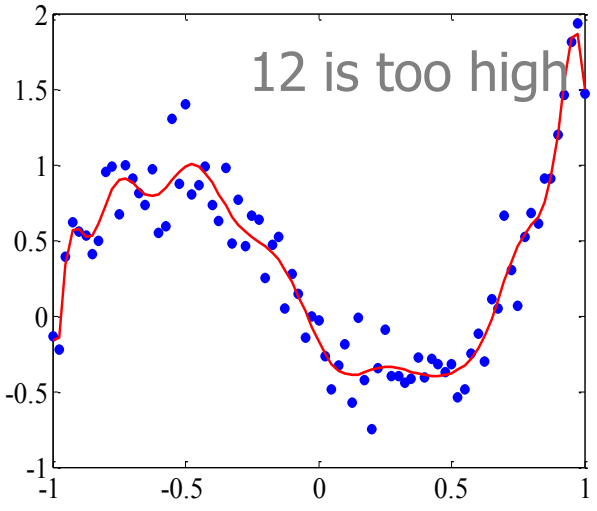
Toy example: some experimental data to which we want to fit a polynomial.



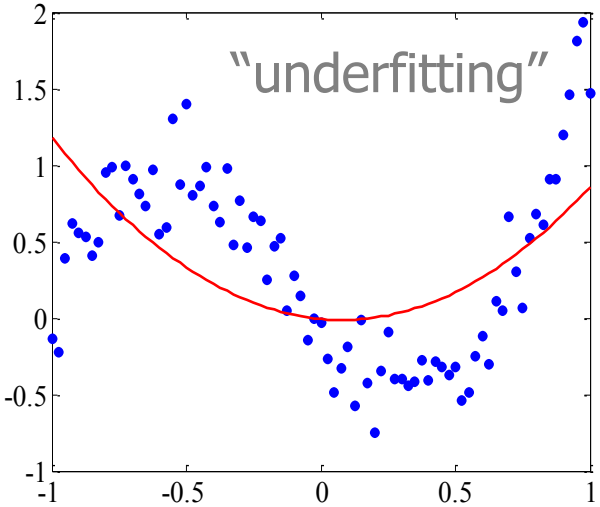
The model selection question is: which order?

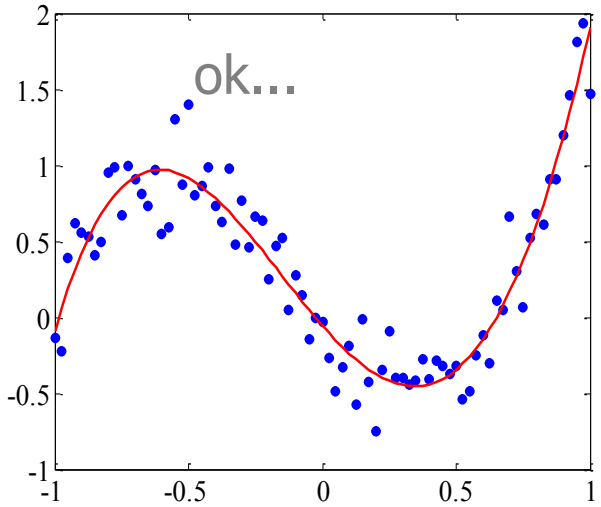


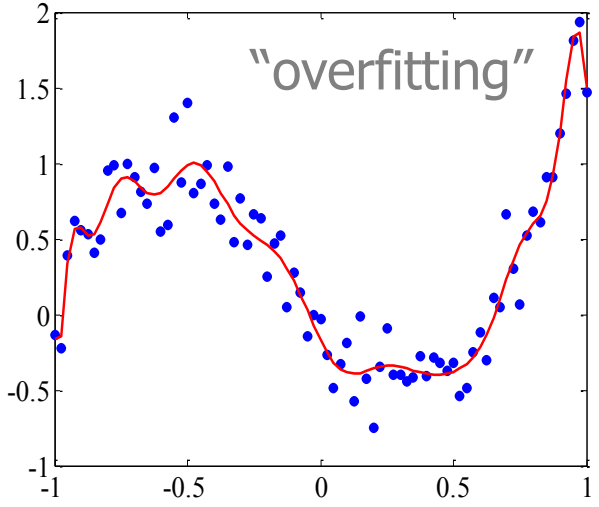




What is model selection?







Model selection goal:

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

Model selection: solutions

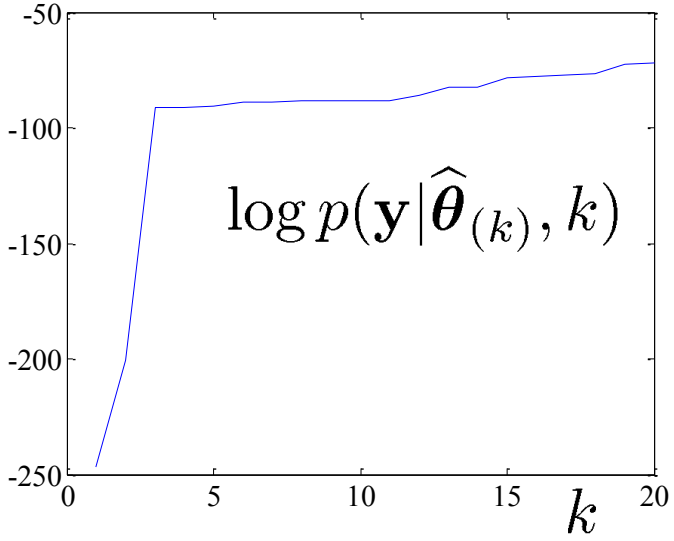
- Typical solution (usable for many probabilistic models)
  - train several models with different orders k
  - choose the one maximizing an “optimality” criterion

  Which “optimality” criterion?

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

Maximizing Log Likelihood

- Problem: Log Likelihood is not 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

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

complexity penalty

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

Alternative: penalized likelihood

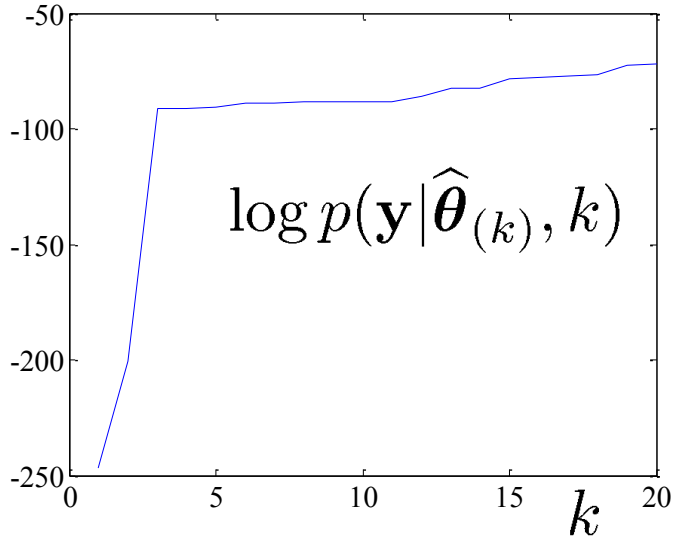
- Example: Bayesian inference criterion (BIC) [Schwartz, 1978]

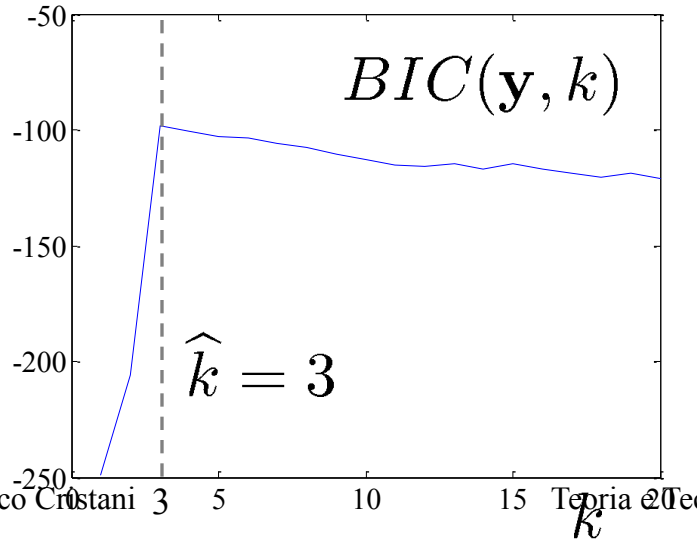
$$k_{best} = \arg\max_{k} \left\{ LL(y \mid \theta_k) - \frac{k}{2} \log(n) \right\}$$

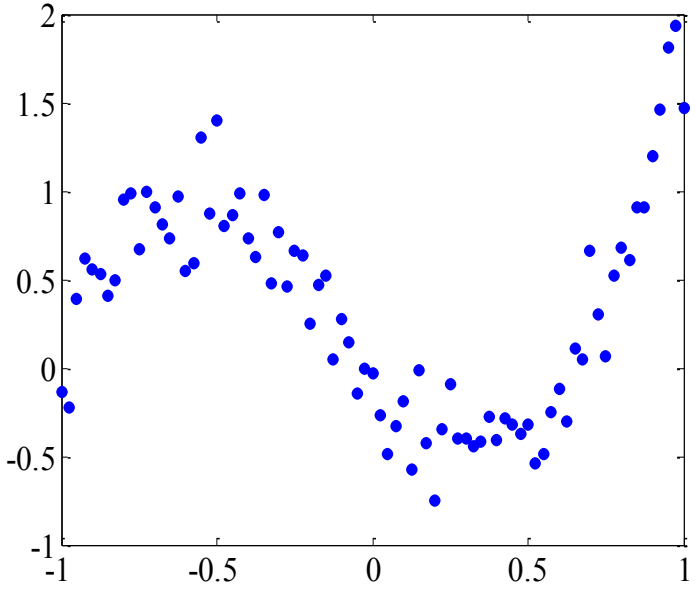
increases with k

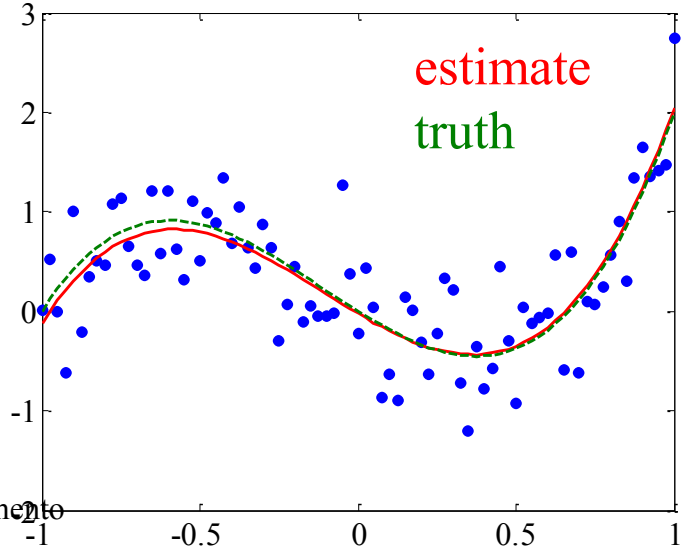
decreases with k
(penalizes larger k)

Back to the polynomial toy example









Some more HMM-oriented solutions

- Application driven model selection: states have some physical meaning

  [Hannaford,Lee IJRR 91]

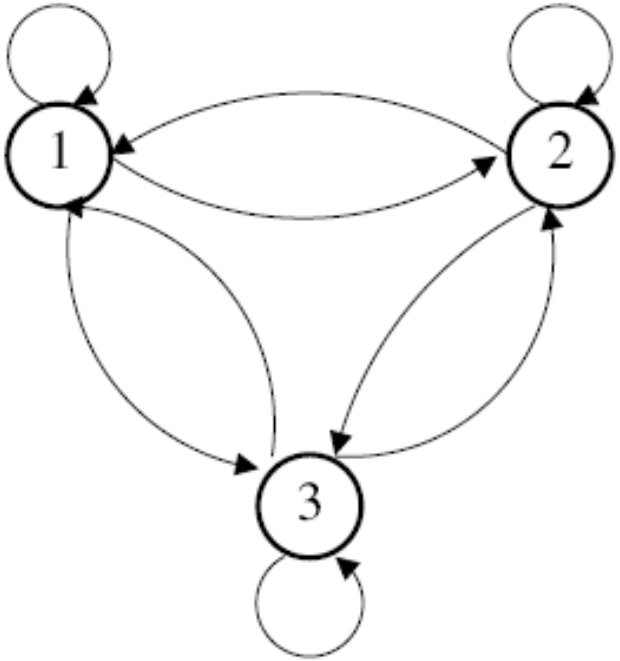
- Split and merge approaches: starting from an inappropriate but simple model, the correct model is determined by successively applying a splitting (or merging) operation

  [Ikeda 93] [Singer,Ostendorf ICASSP 96]
  [Takami, Sagayama ICASSP 92]

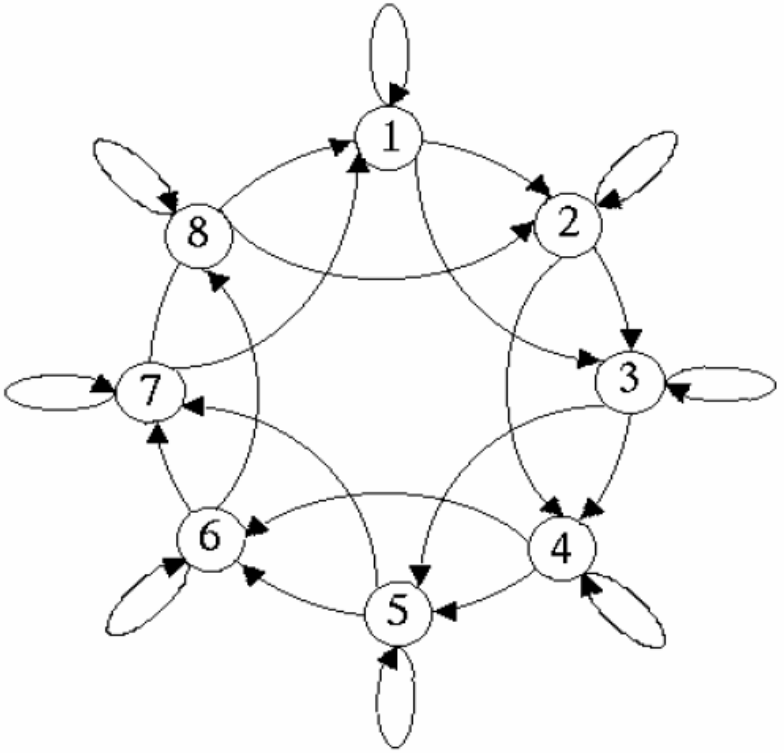
Model selection problem 2: selecting the best topology

- Problem: forcing the absence or the presence of connections

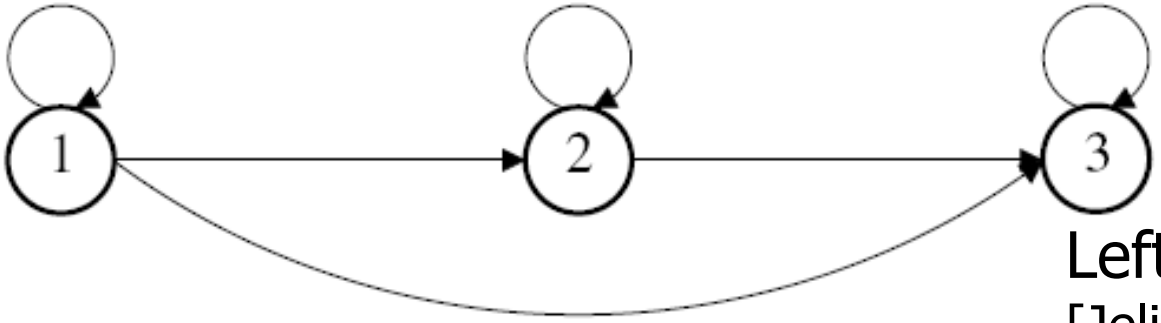
- Typical ad-hoc solutions
  - ergodic HMM (no contraints)
  - left to right HMM (for speech)
  - circular HMM (for shape recognition)



standard ergodic HMM



circular HMM [Arica,Yarman-Vural ICPR00]

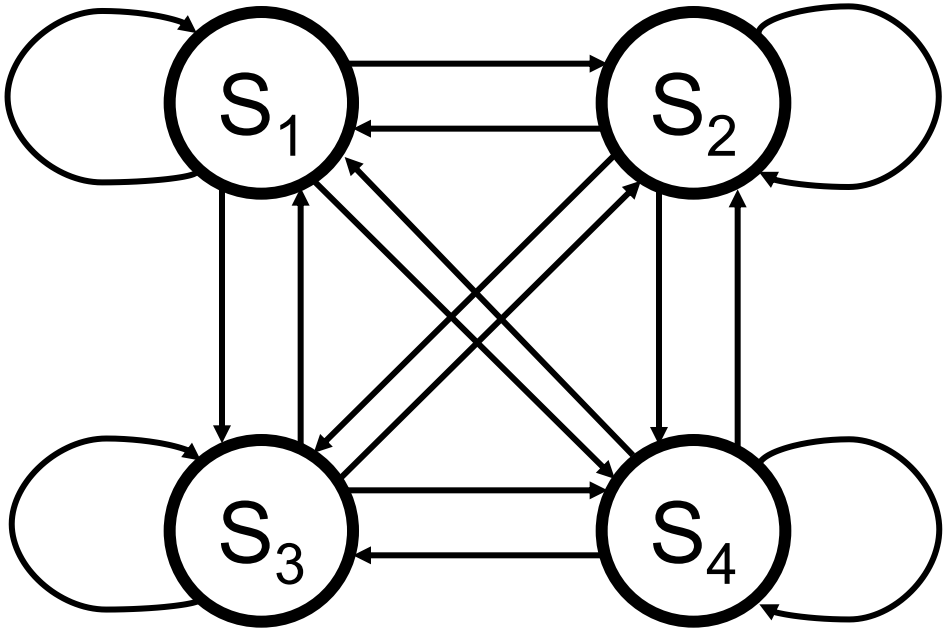


Left to right HMM
[Jelinek, Proc. IEEE 1976]

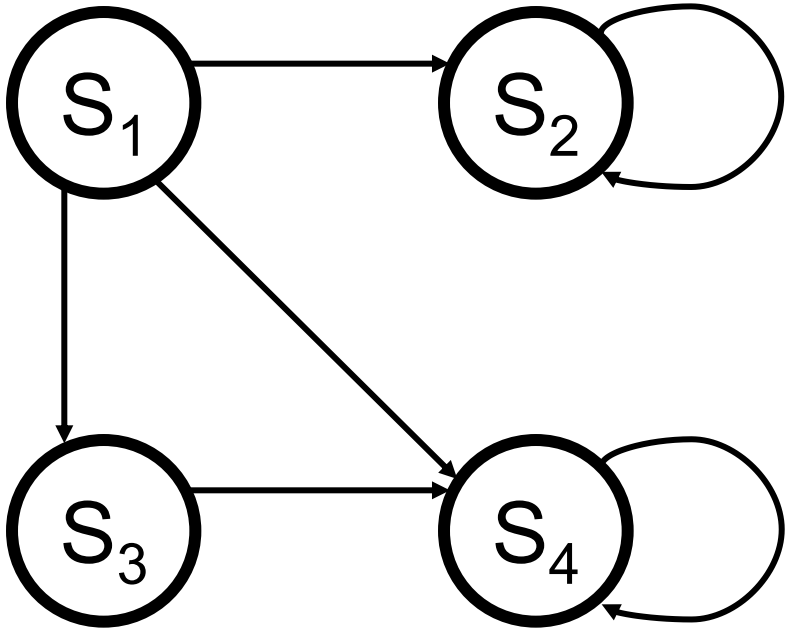
One data-driven solution

[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)

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:

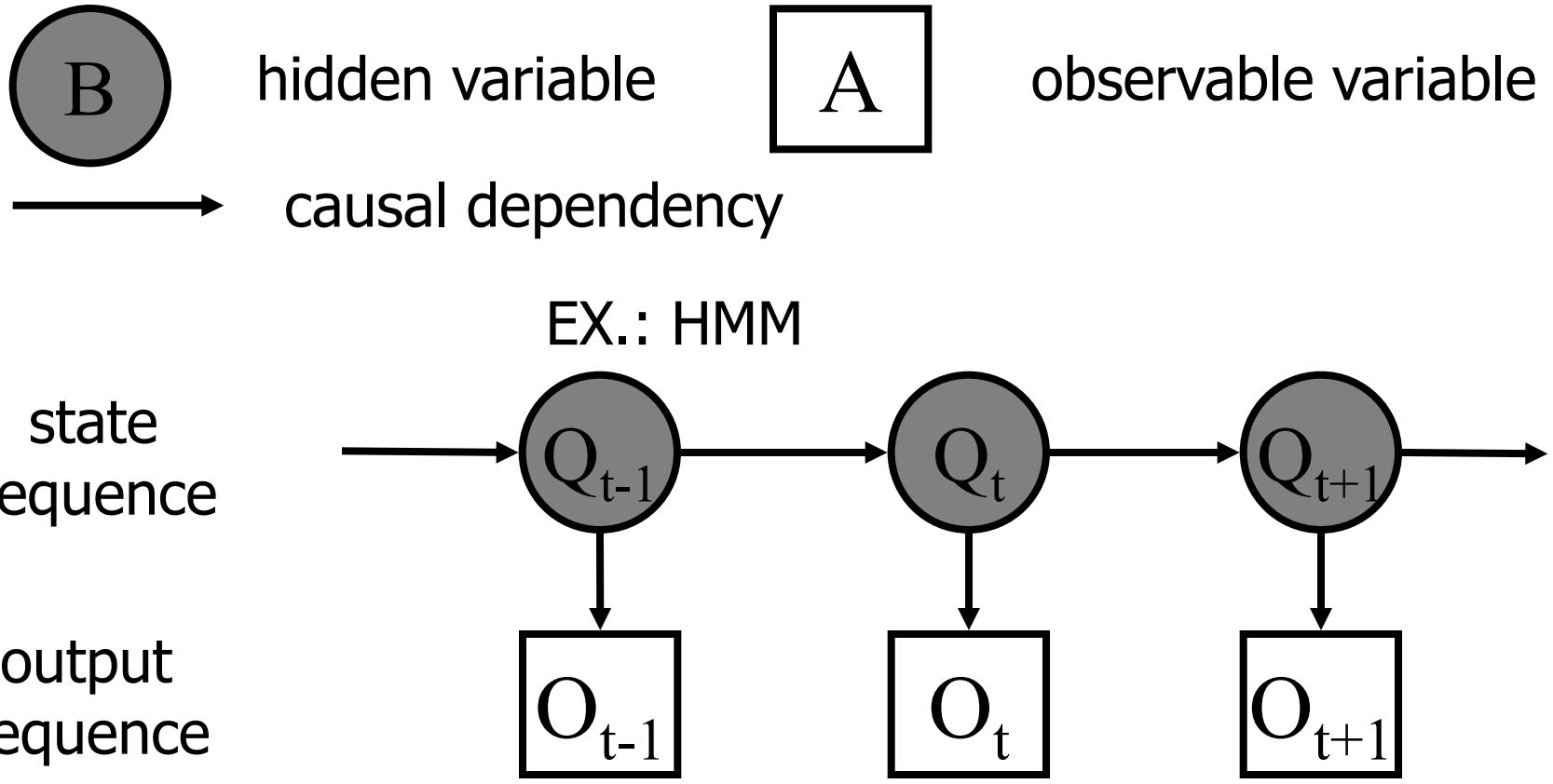
adding novel dependencies between components, in order to model different behaviours

Examples:

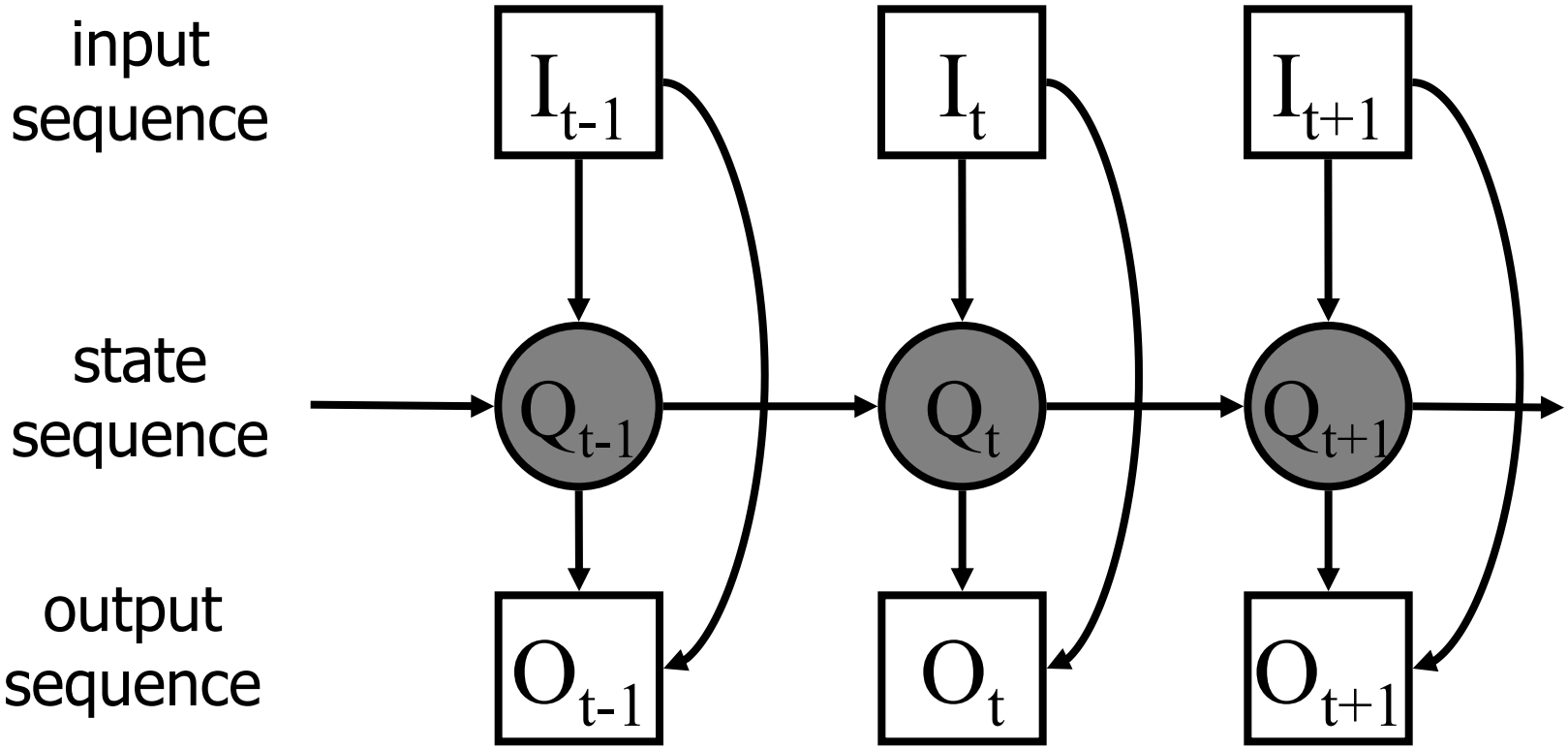
- Input/Output HMM
- Factorial HMM
- Coupled HMM
- ...

Preliminary note: the Bayesian Network formalism

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

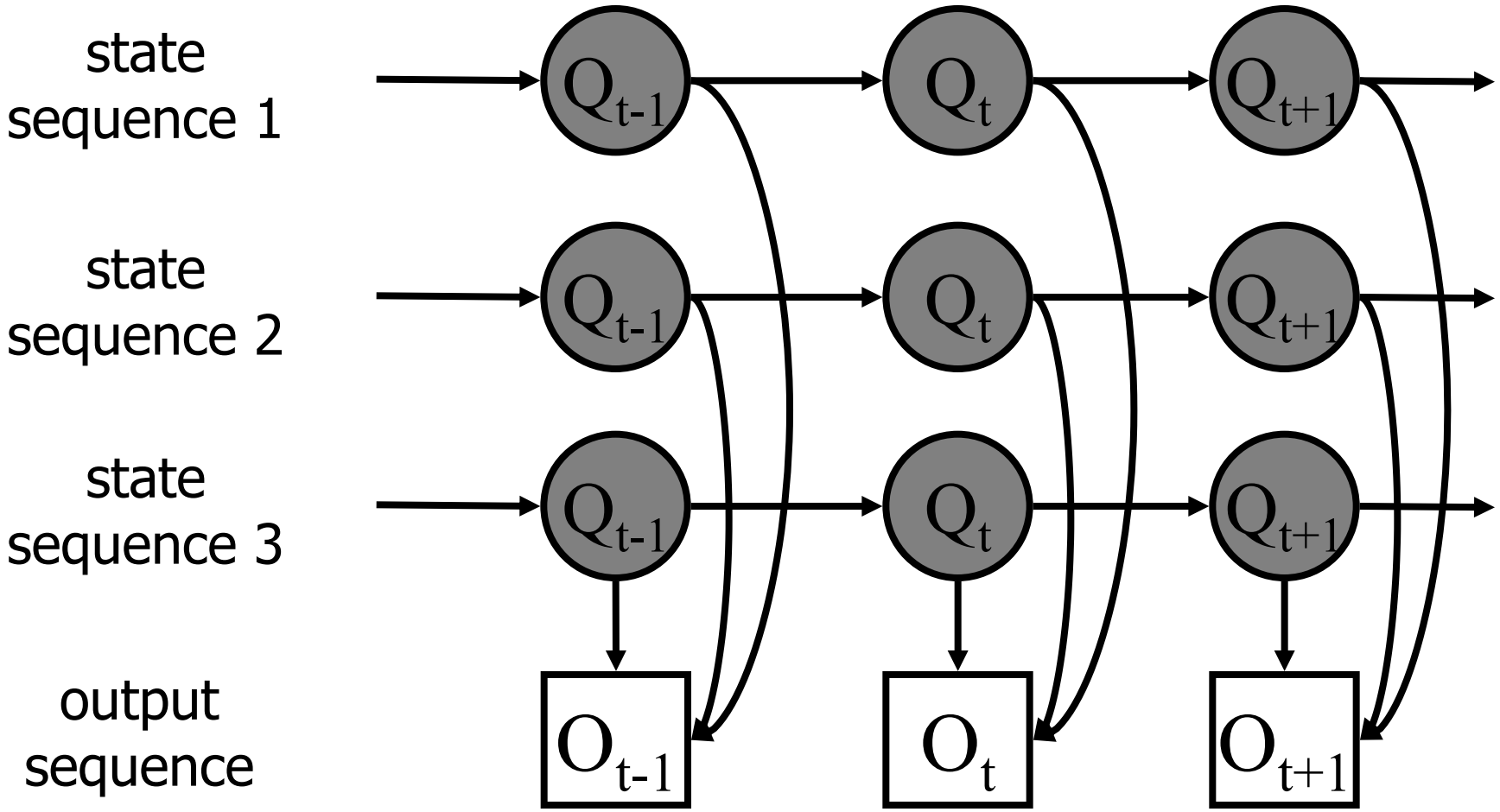


Input-Output HMM: HMM where transitions and emissions are conditional on another sequence (the input sequence)



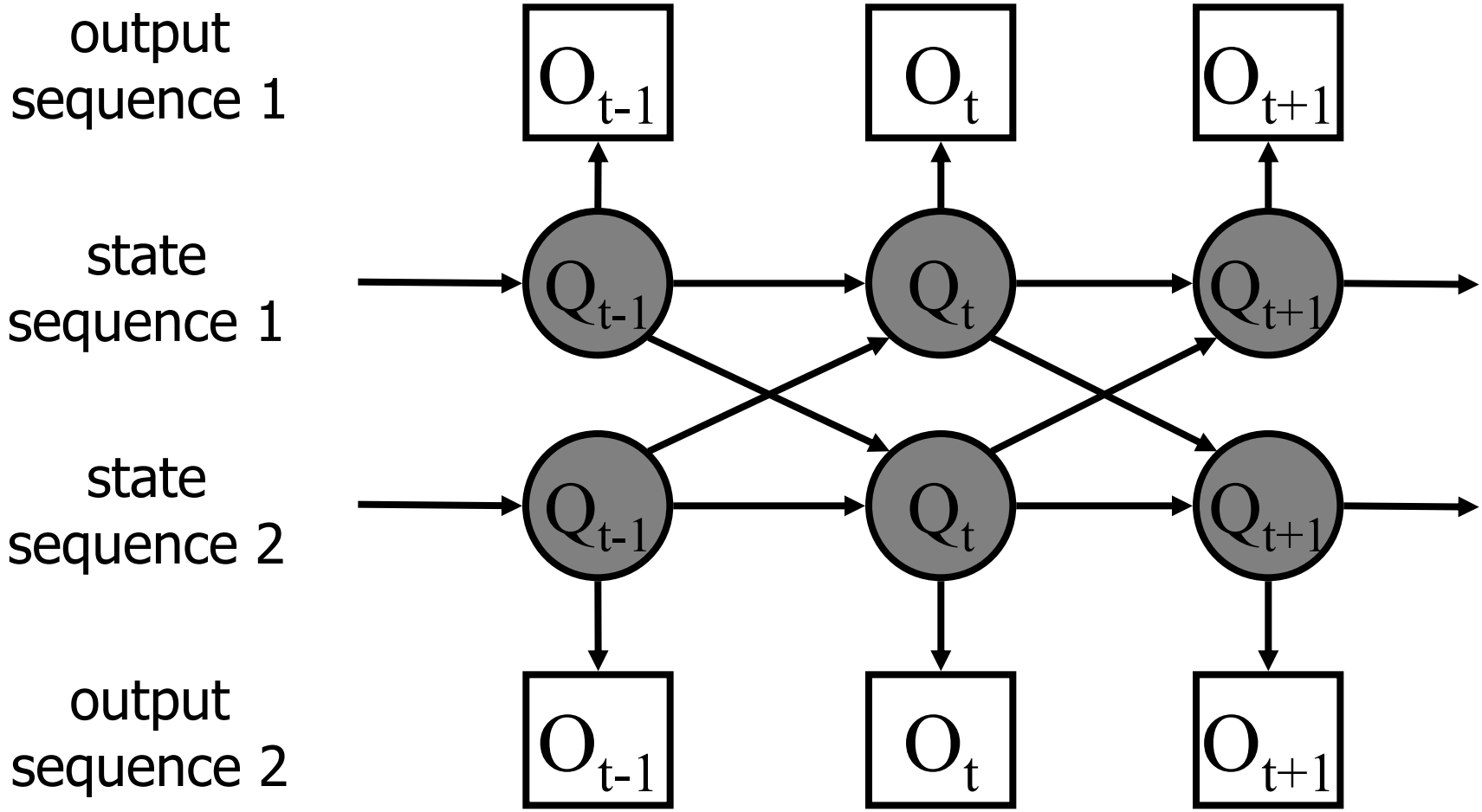
EX.: finance, the input sequence is a leading market index

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]

Extending standard models (2)

Examples:

- 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

Some open issues/research trends

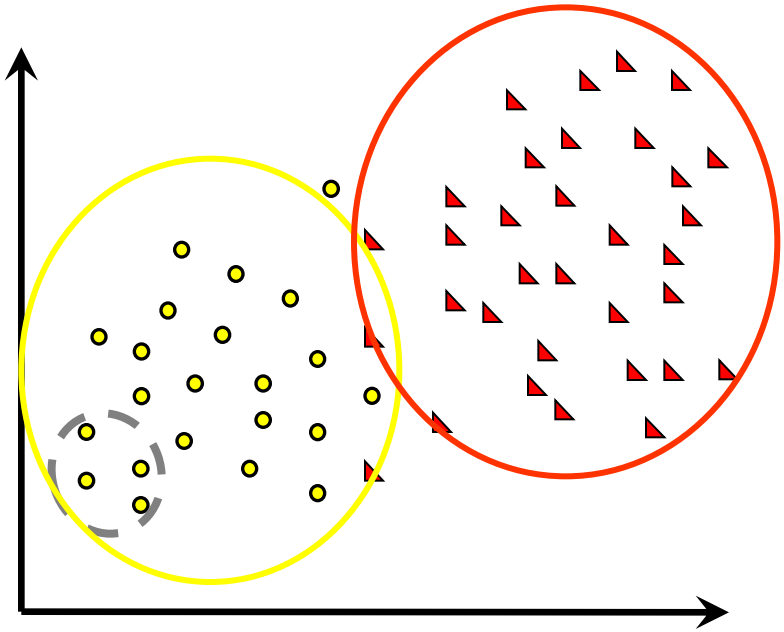
1. Model selection
   - how many states?
   - which topology?

2. Extending standard models
   - modifying dependencies or components

3. Injecting discriminative skills into HMM

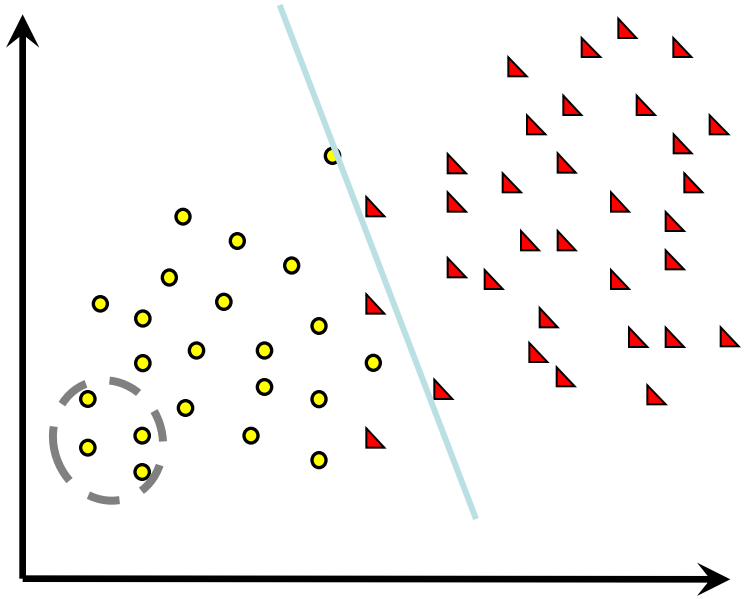
The general problem: generative vs discriminative modelling

Generative: one model for each class/group (e.g. HMM)



generative are better in describing classes

Discriminative: just model how to separate classes (e.g. SVM)



discriminative are better in solving the problem

Injecting discriminative information into HMM

- HMM are generative models, could be improved injecting discriminative information (information from other classes)
- Two ways:
  - inject discriminative information in the training phase
  - inject discriminative information in the classification phase

Discriminative training

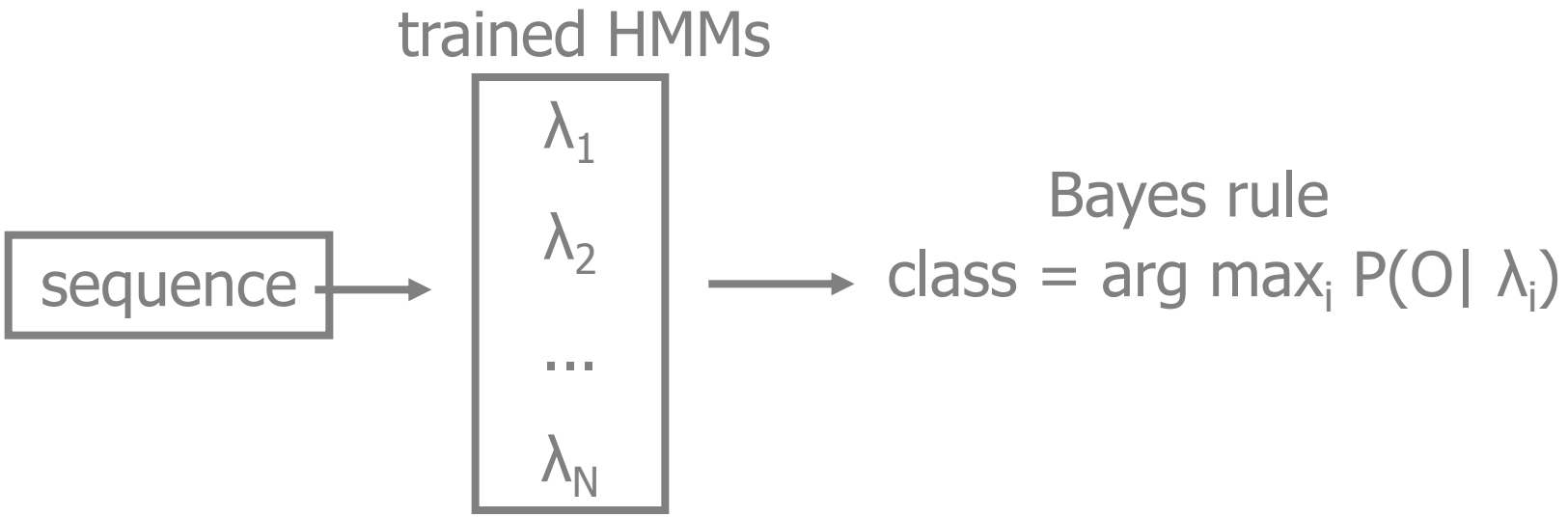
- Standard HMM training is blind (no information from other classes is used)
- IDEA: training HMMs with discriminative criteria, i.e. considering also other classes' information
- Two popular examples:
  - maximum mutual information (MMI)

  [Bahl, Brown, de Souza, Mercer, ICASSP 00]
    - maximize likelihood for the objects in the class while minimizing the likelihood for the other obejcts
  - minimum Bayes risk (MBR)

  [Kaiser, Horvat, Kacic, ICSLP 00]

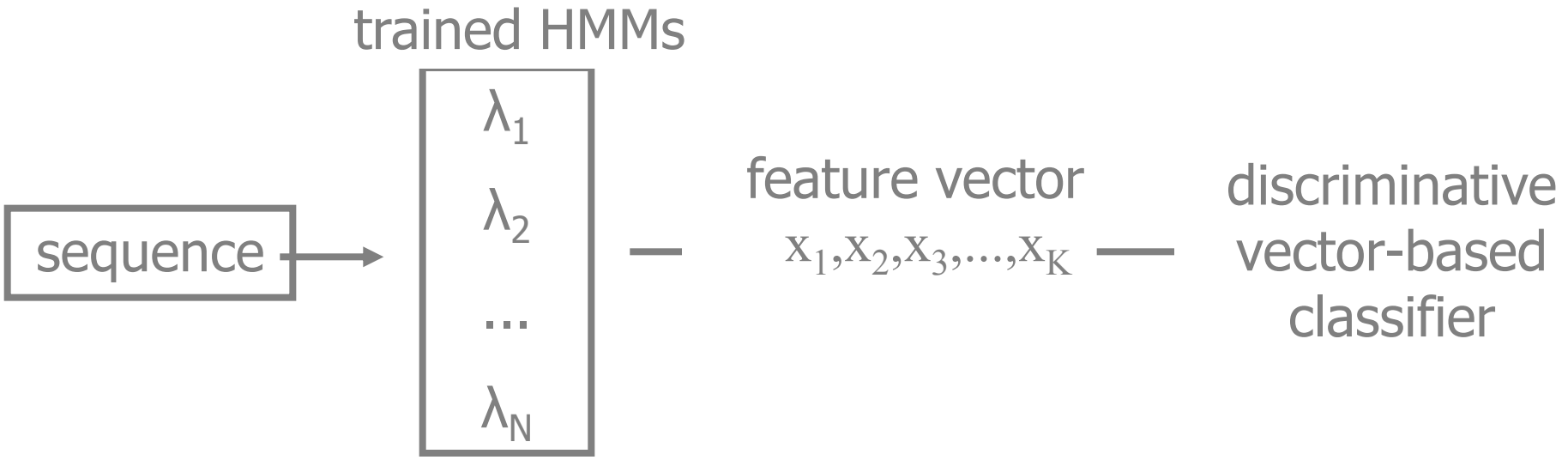
Discriminative classification

Standard HMM classification: train one HMM per class and apply the Bayes rule



Discriminative classification

Idea of discriminative classification: using trained HMMs to derive a feature space, where a discriminative classifiers is trained



Discriminative classification

- Kind of features:
  - the gradient of the Log Likelihood (or other related quantities):
    - this is the well known Fisher Kernel:

    [Jaakkola, Haussler, NIPS 99]

  - the log likelihood itself (or other quantities directly computable from the posterior probability)
    - using “score spaces”
    - using the “dissimilarity-based representation” paradigm

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

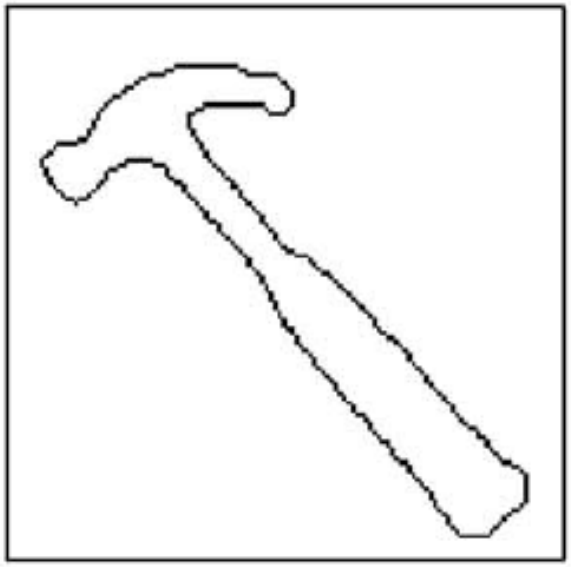
Motivations

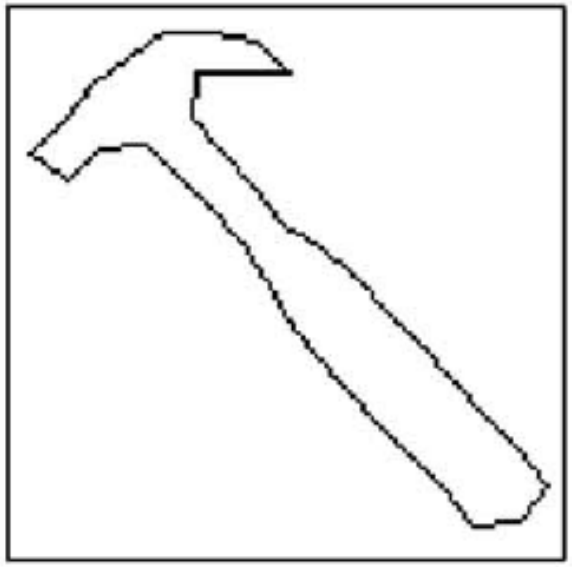
- The use of HMM for shape analysis is very poorly addressed
- Previous works:
  - He Kundu (PAMI - 91) using AR coefficients
  - Fred Marques Jorge 1997 (ICIP 97) using chain code
  - Arica Yarman Vural (ICPR 2000) using circular HMM
- Very low entity occlusion
- Closed contours
- Noise sensitivity 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





- Contour is smoothed by a gaussian filter before computing the curvature

Curvature representation

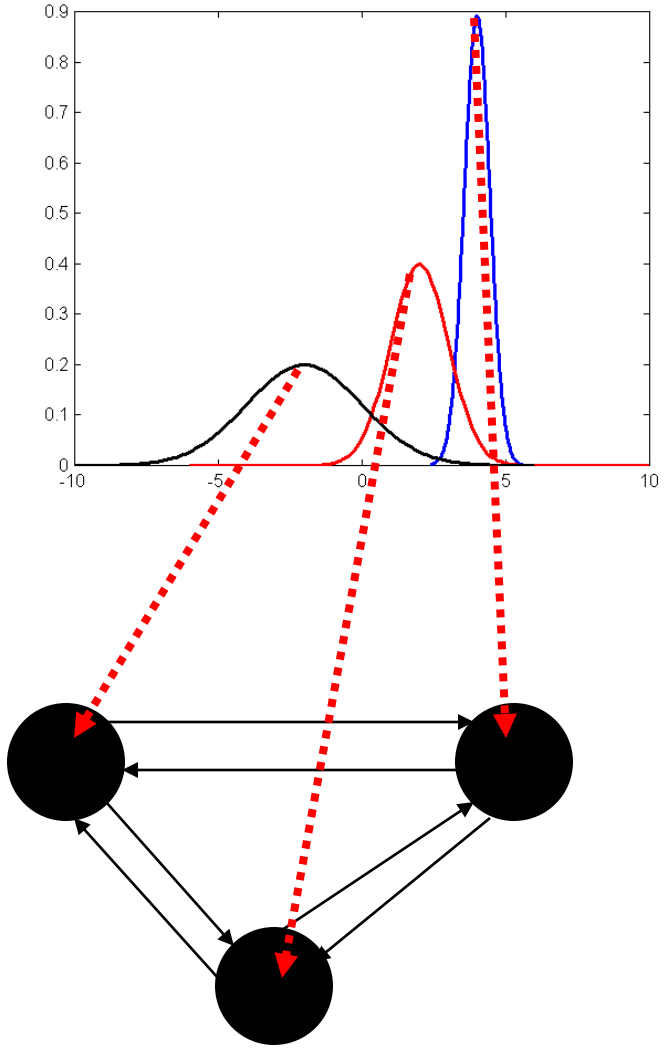
- Advantages
  - invariant to object translation
  - rotation of object is equal to phase translation of the curvature signal;
  - can be calculated for open contours
- Disadvantages
  - noise sensitivity

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
  - Using BIC on the gaussian mixture model clustering in order to choose the optimal number of states.
  - Advantage: only one HMM training session

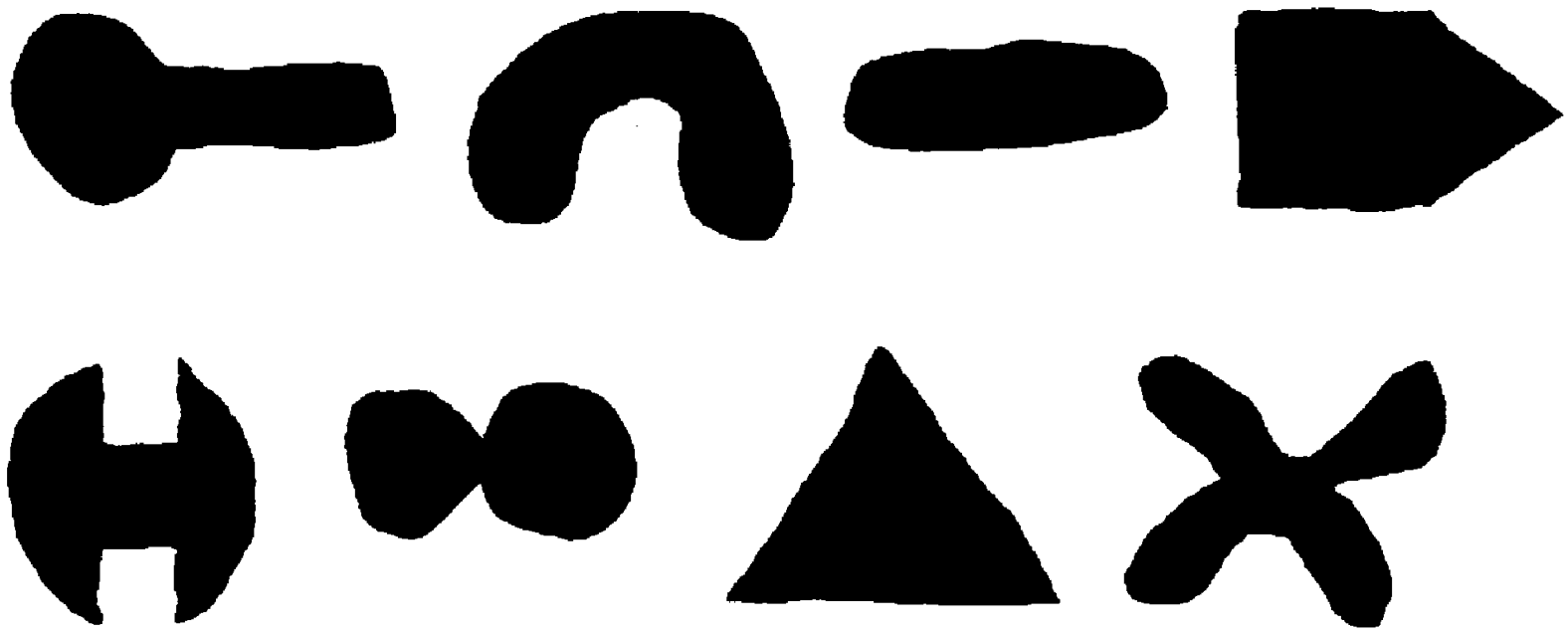
Strategy

- 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.

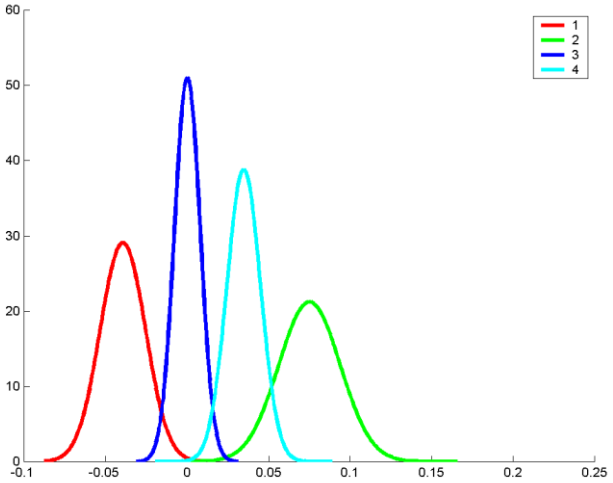
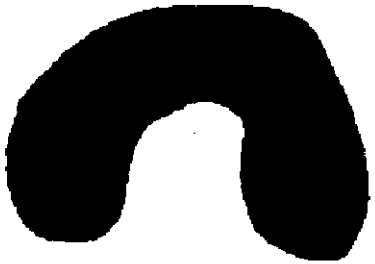
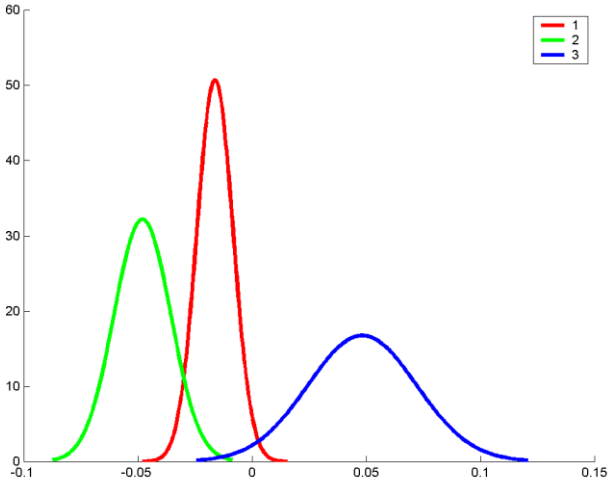
Strategy (cont.)

- 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 $P(O|\lambda_i)$.

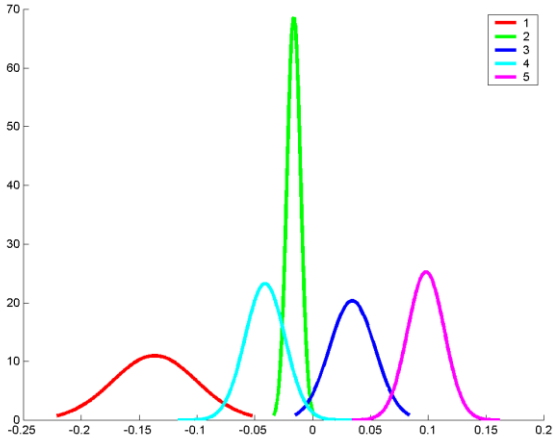
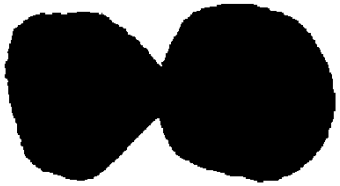
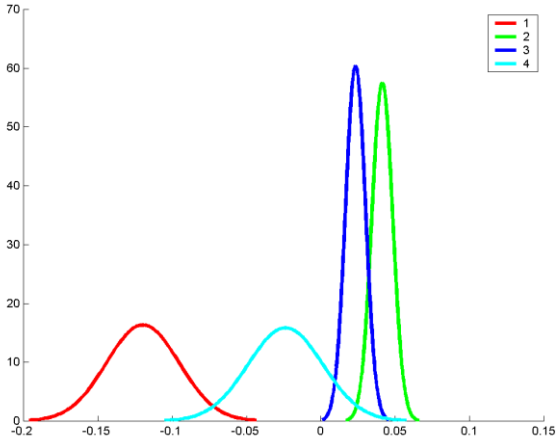
Experimental: The test set



The models

Shape	Emission Probability	Transition Probability
		0.94 0.00 0.06 0.00 0.00 0.96 0.00 0.04 0.02 0.00 0.96 0.02 0.00 0.02 0.02 0.96
		0.98 0.01 0.01 0.03 0.97 0.00 0.02 0.00 0.98

The models (2)

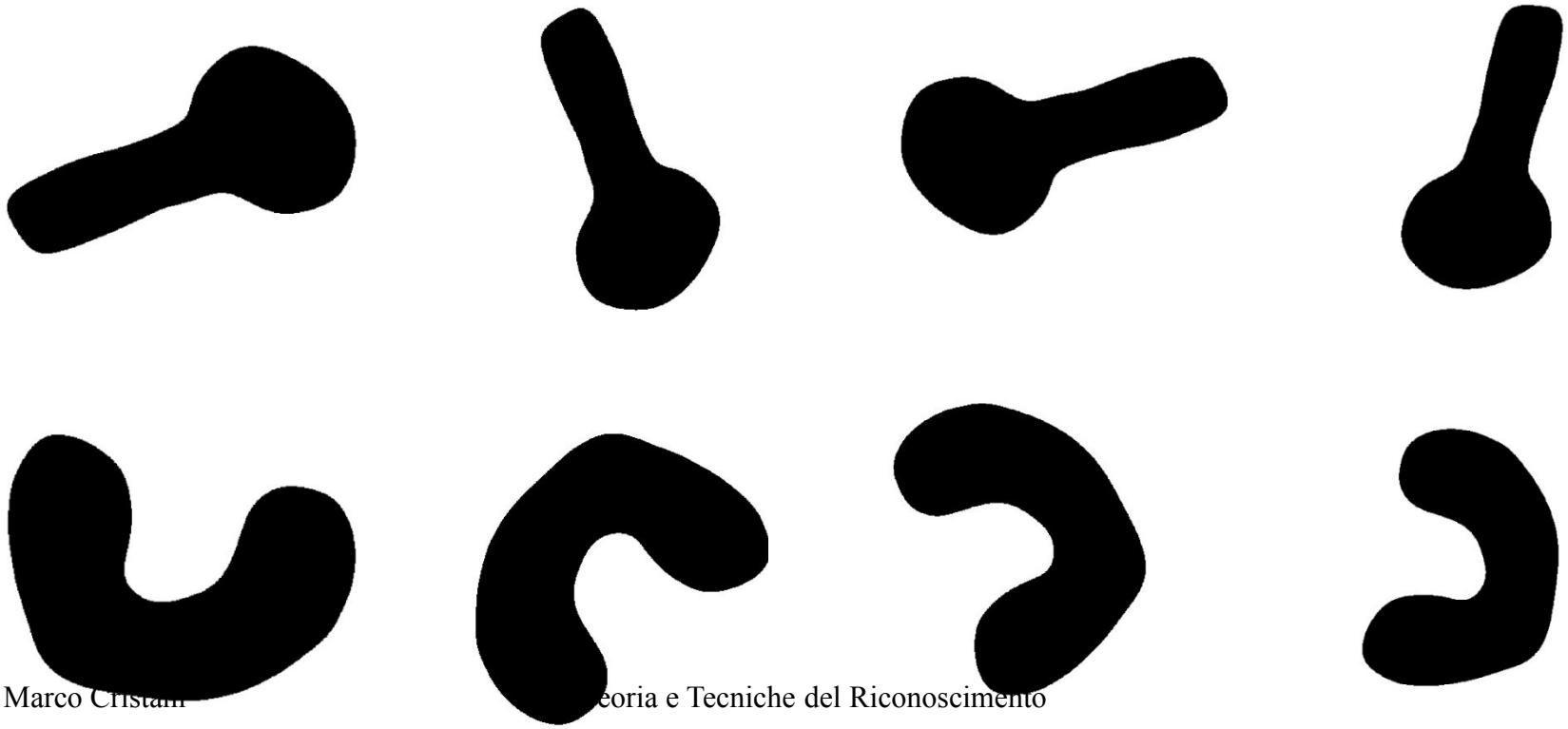
Shape	Emission Probability	Transition Probability
		
		

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

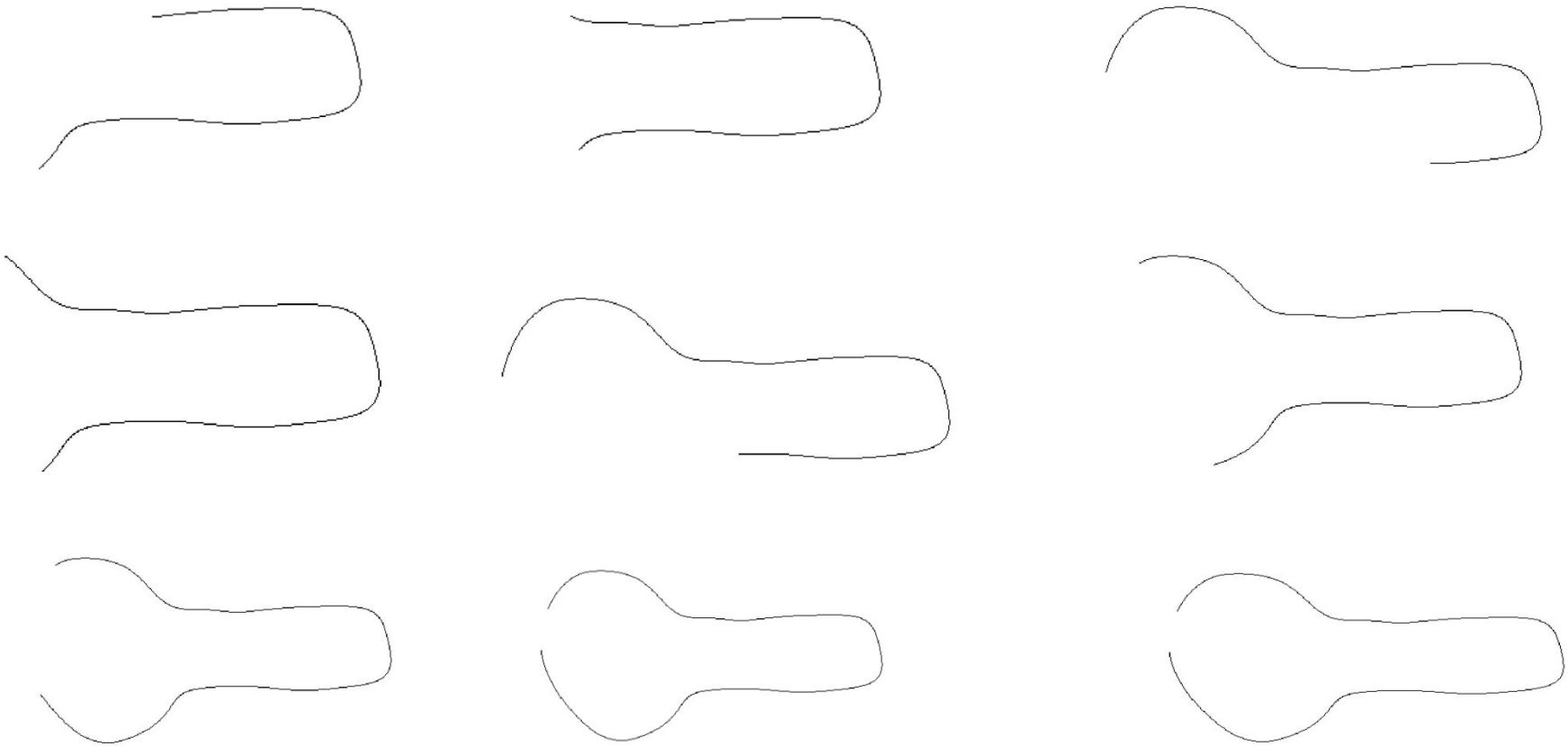
Rotation

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



Occlusion

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



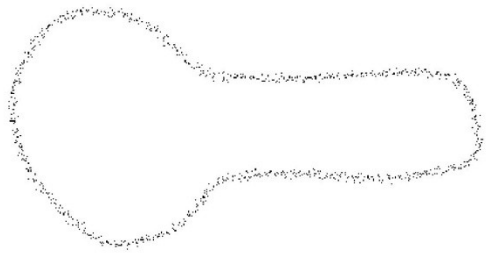
Occlusion: results

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



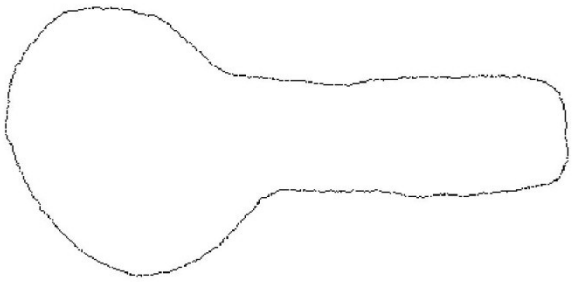
$\sigma^2=1$

$\sigma^2=4$

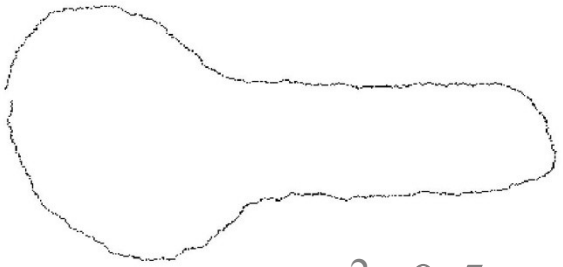
$\sigma^2=5$

Alternative Noise

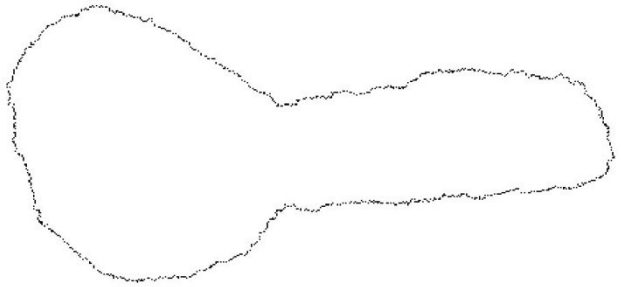
- Adding noise to the first derivative



$\sigma^2=0.3$



$\sigma^2=0.5$



$\sigma^2=0.7$

$\sigma^2=0.9$

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%

Occlusion and Rotation: 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%

Occlusion, Rotation 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 Projection

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 Projection: 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

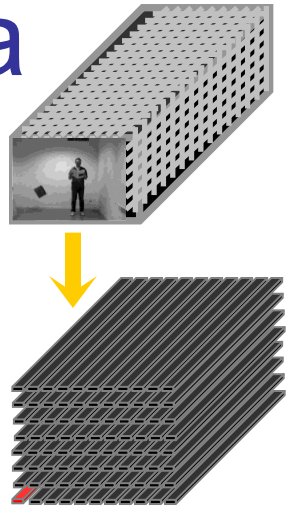
Conclusions

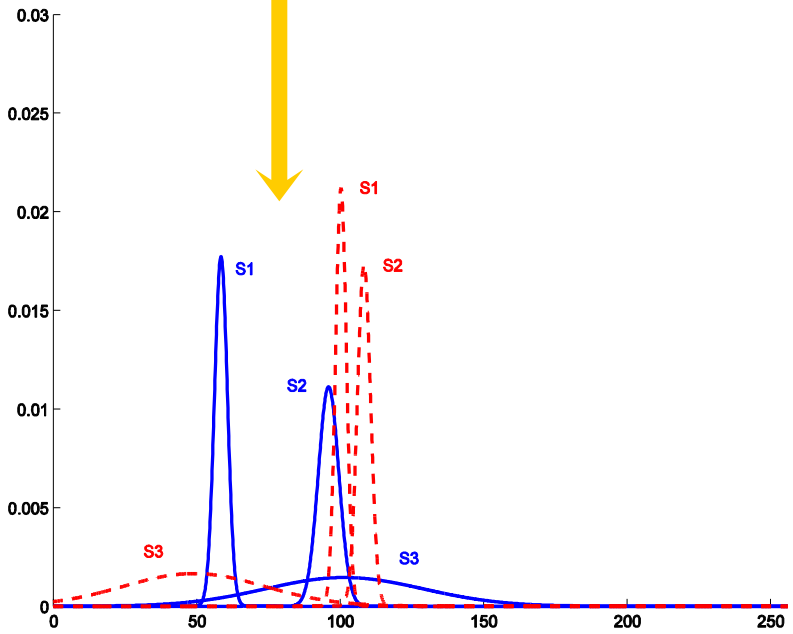
- System is able to recognize object that could be translated, rotated and occluded, also in presence of noise.
- Translation invariance: due to Curvature
- Rotation invariance: due to Curvature and HMM
- Occlusion invariance: due to HMM
- Robustness to noise: due to HMM

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 = [redacted] represents gray level ranges assumed by the v-th pixel signal, and
- The larger the [redacted], 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 [redacted]

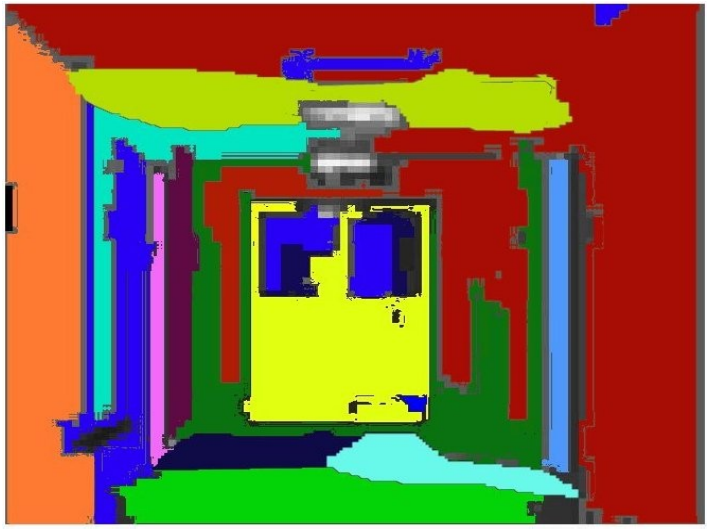
- We use a similar distance, *more robust*, which weights more the states in which the model stands more time

Results (real)

Corridoio.avi

Image based segmentation

HMM based segmentation



For more (and other) info ...

- M. Bicego, M. Cristani, V. Murino, "Unsupervised Scene Analysis: A Hidden Markov Model Approach", *Computer Vision and Image Understanding*, Vol. 102/1, pp. 22-41, April 2006.

- M. Cristani, M. Bicego, V. Murino, “Multi-level Background Initialization using Hidden Markov Models”, First ACM SIGMM Int’l Wks on Video Surveillance, Berkeley, CA, USA, pp. 11-20, Nov. 2003.

- M. Cristani, M. Bicego, V. Murino, “Integrated Region- and Pixel-based Approach to Background Modelling”, IEEE Wks. on Motion and Video Computing, Orlando, FL, pp. 3-8, Dec. 2002.

- A.Perina, M. Cristani, V. Murino et al. “Unsupervised haplotype reconstruction and LD blocks discovery in a hidden Markov framework”, CIBB 2007.