

# *Hidden Markov Model (HMM)*

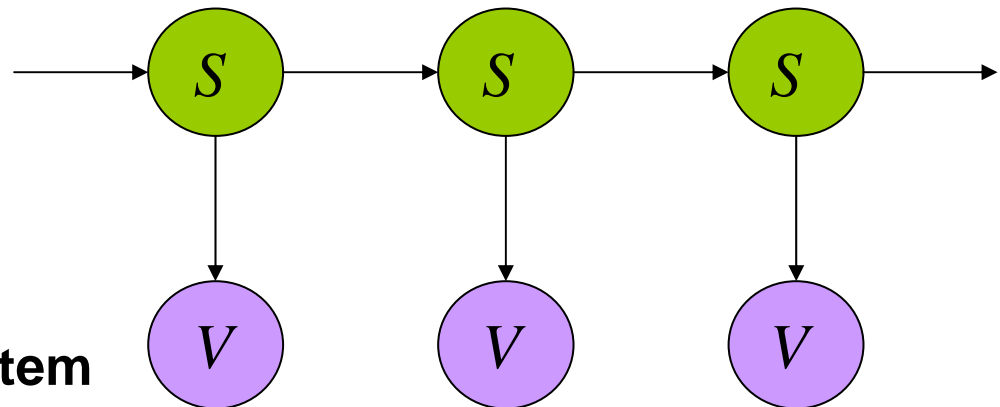


# Why HMM?

Hidden Markov Models (HMMs) are used to model the **temporal evolution of a complex problem** of which one can have only a **partial description**.

One may be provided solely with the current state of the system or with several of the previous states of the system.

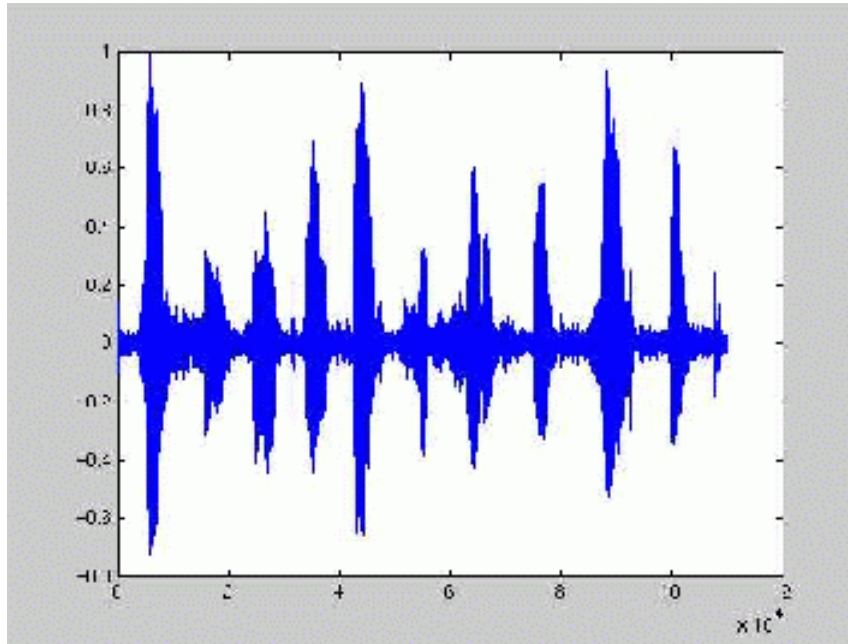
**Hidden States**



**Observations**

**Partial state of the system**

# Applications of HMM



Speech Processing: How to segment continuous speech signal into sets of distinct words?

HMM are used to model the temporal evolution of speech pattern and to recognize either complete words or parts of the words, such as phonemes.

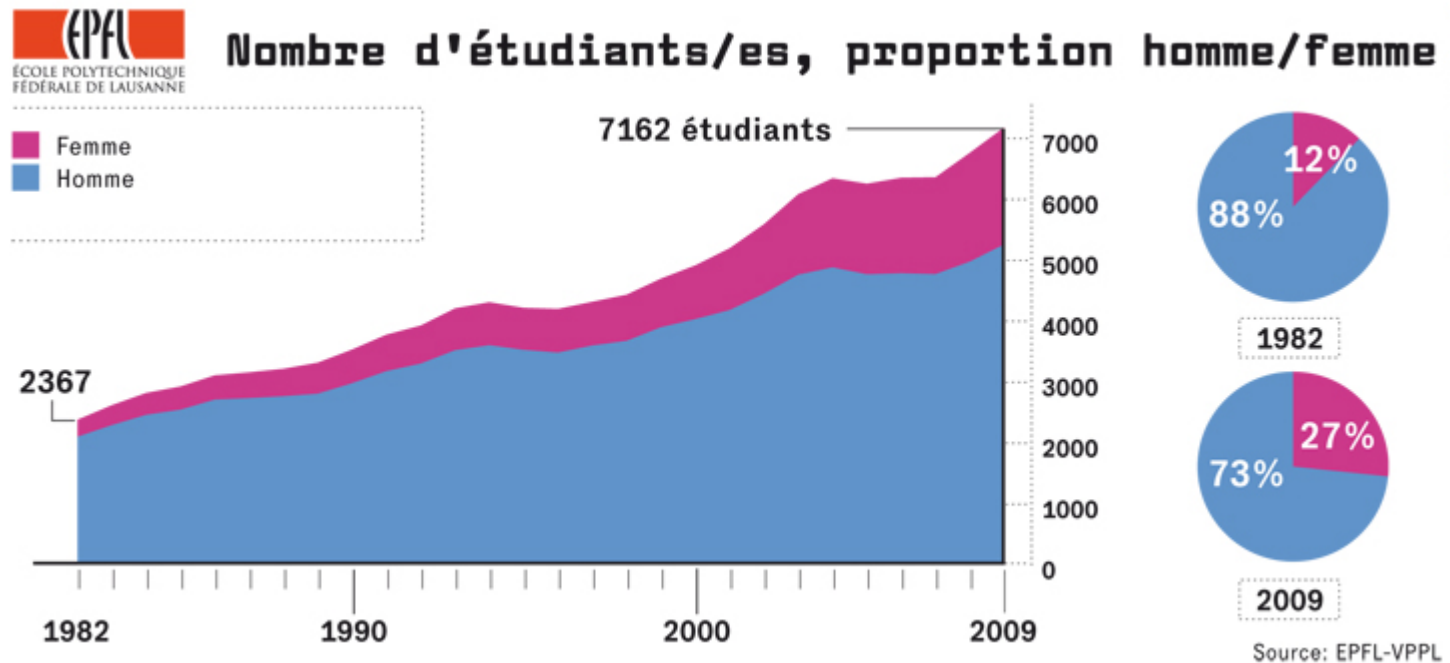
# Applications of HMM



Gesture Recognition: How to segment continuous motion of joints into subsets of motions (so-called motion primitives)?

HMM are used to model the temporal evolution of joint motion and to recognize either complete motion for classification of gestures or to decompose this into subsets of motions.

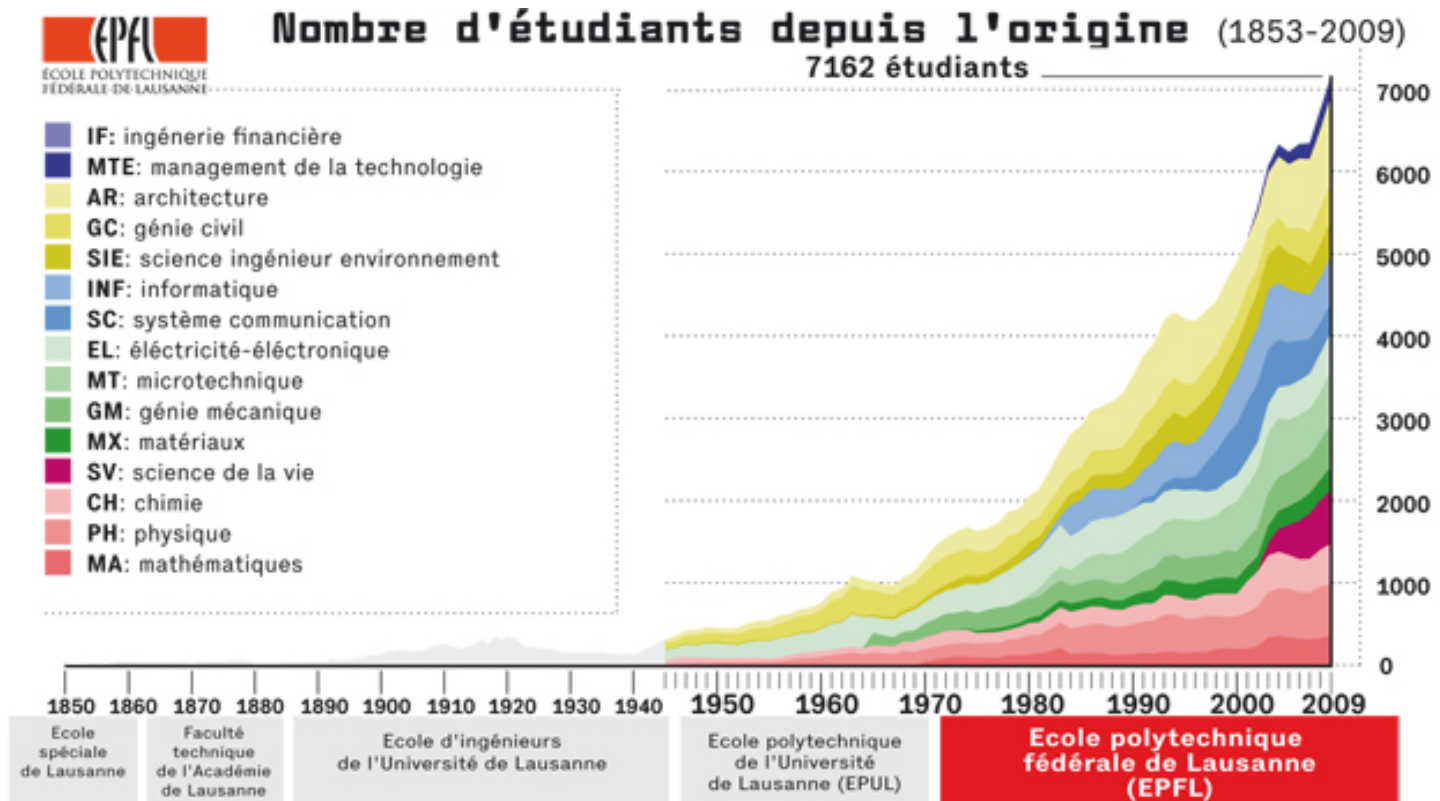
# Applications of HMM



- The temporal evolution of any process is due to many (hidden) variables which we will never be able to fully observe.
- With enough data, one may be able to build a good estimate of the process at hand and be able to use this to predict future states.

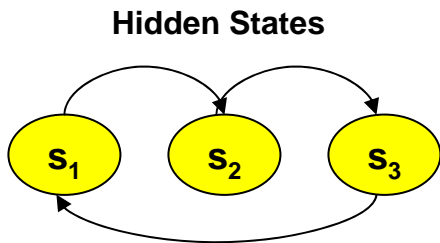
# Applications of HMM

- The distribution can be multi-variate and prediction can apply to the complete distribution.
- Predict the number of diplomas that will be awarded in the next ten years across the two EPF based on a field-distribution.



# Why HMM?

Number of pastries sold at the Coupole each day over a 7-day period



**3-state Model**

# Why HMM?

Number of pastries sold at the Coupole each day over a 7-day period



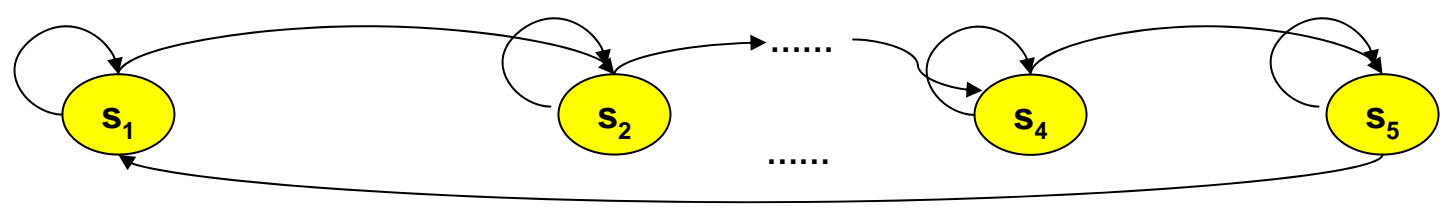
Hidden States



15-state Model

# Why HMM?

Number of pastries sold at the Coupole each day over a 7-day period



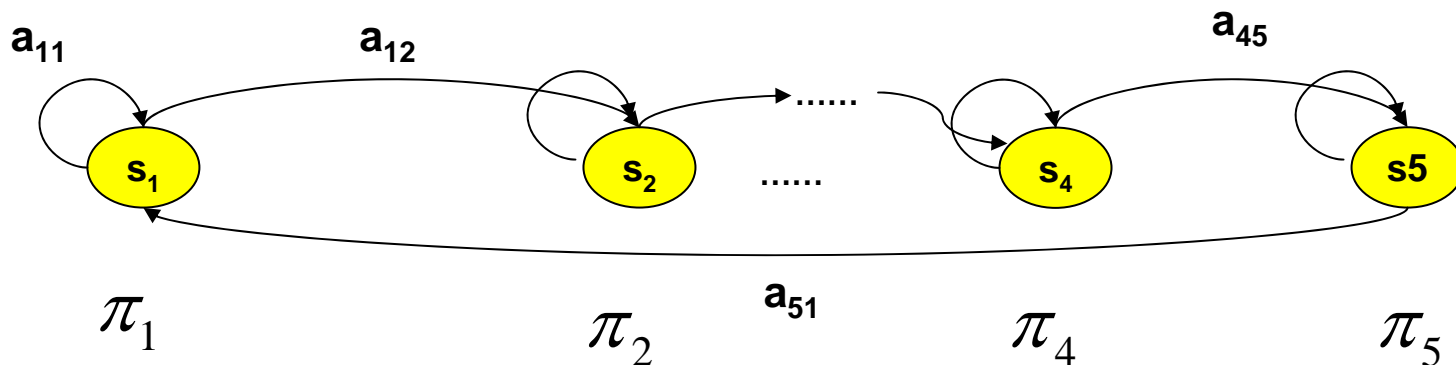
**5-state Model**

# HMM Parameters

For a fixed set of states, one must determine the probabilities of transiting from one state to the other and the probability to start the complete process in one particular state.

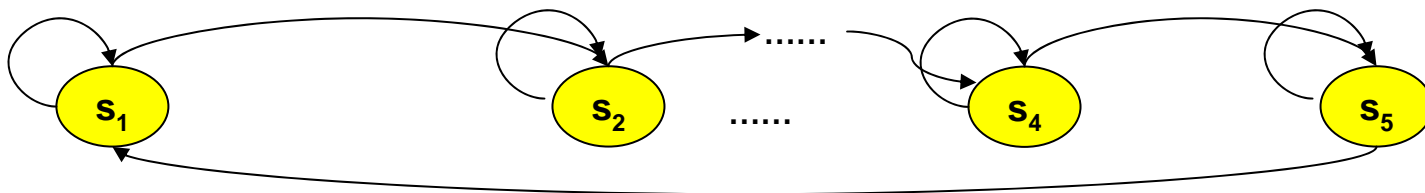
Transition probability matrix  $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{15} \\ \dots & & & \\ a_{51} & a_{52} & \dots & a_{55} \end{bmatrix}$

Initial state probabilities  $\pi = [\pi_1, \dots, \pi_5]$



# HMM parameters

The observations  $o_t \in \mathbb{R}^3 \quad \forall t$

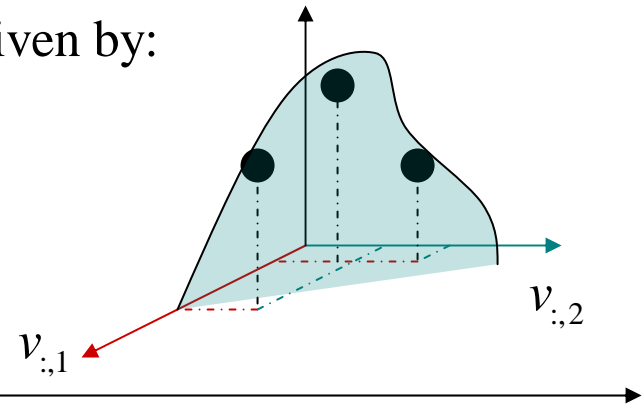
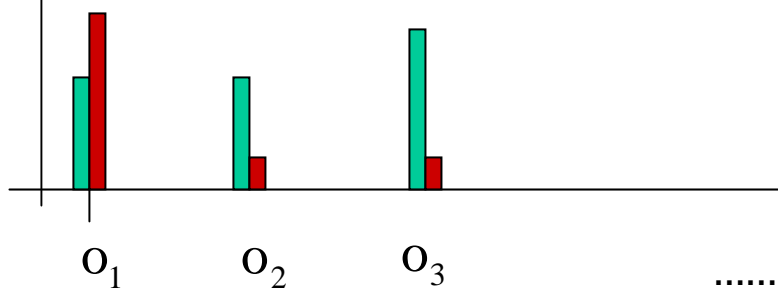


# HMM parameters

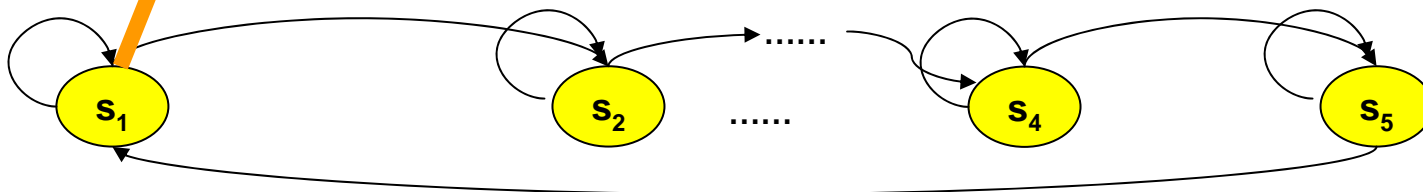
$b_1$  : probability distribution of the observables for state  $s_1$

The prob. that observable  $o_t$  takes value  $v_k$  is given by:

$$p(o_t = v_k | s_1) = p(o_{t,1} = v_{k,1}, o_{t,2} = v_{k,2}, o_{t,3} = v_{k,3} | s_1)$$



$b_1$ : Emission probability at state 1



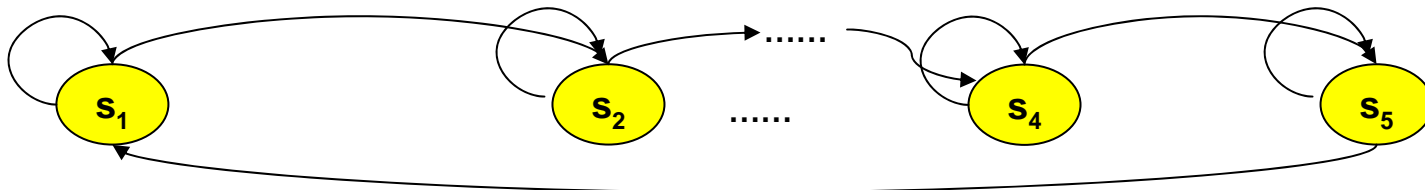
# HMM parameters

Need to learn the following parameters

Transition probability matrix  $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{15} \\ \dots & & & \\ a_{51} & a_{52} & \dots & a_{55} \end{bmatrix}$

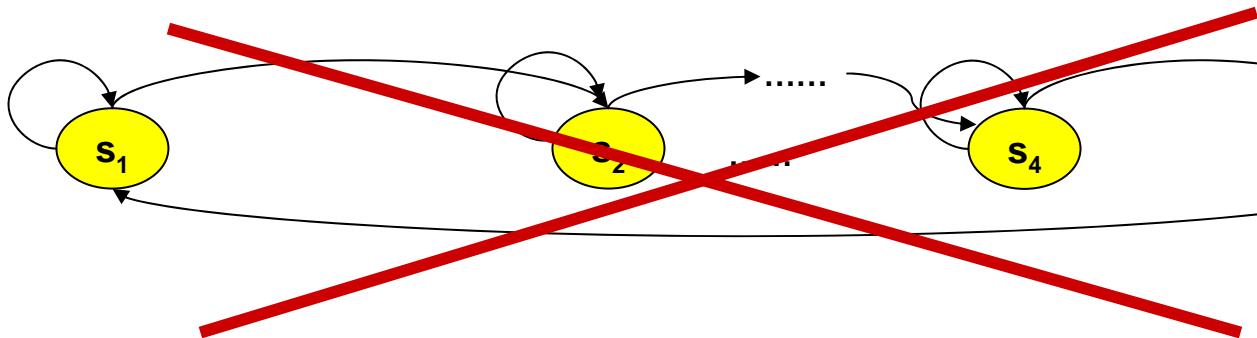
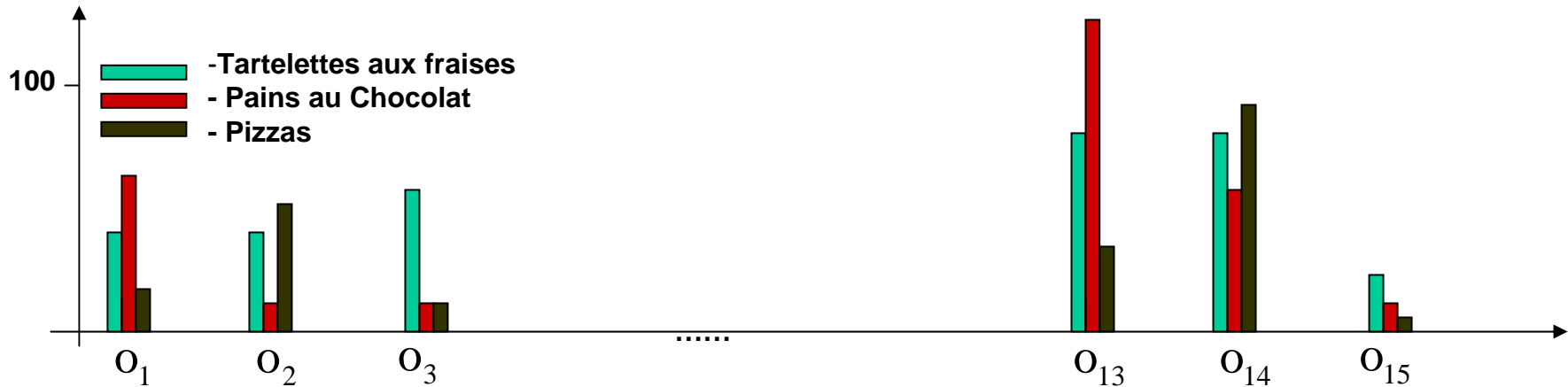
Initial state probabilities  $\pi = [\pi_1, \dots, \pi_5]$

Emission state probabilities  $b_1, \dots, b_5$  ← Could be a multivariate complex pdf modeled e.g by GMM



# HMM parameters

Number of states is given (iterative methods exist to also estimate these)  
Automatic assignment of observations to states!

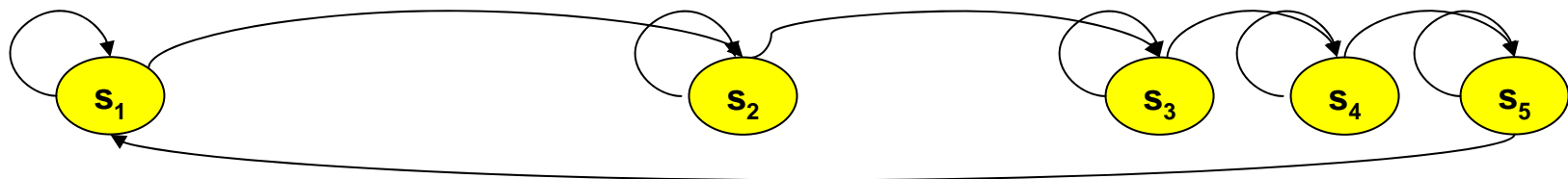


States do not necessarily correspond to our intuition

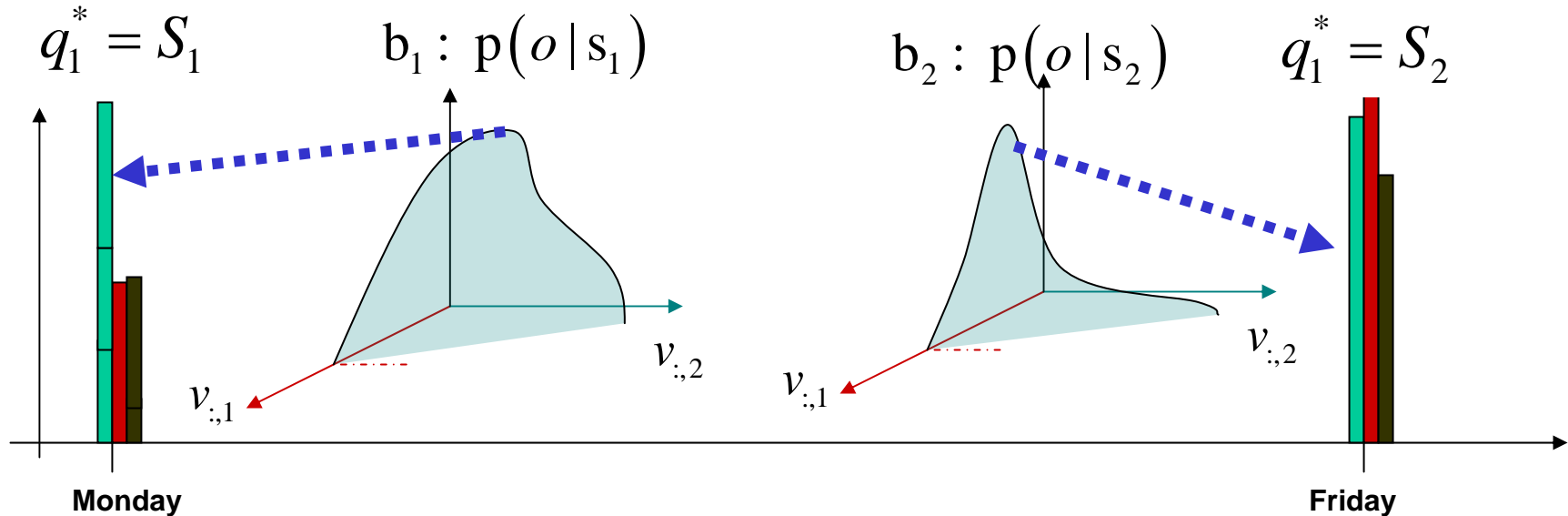
# HMM parameters

Determine the most likely state sequence  $Q = \{q_1, \dots, q_T\}$

Each observation is associated the most likely state.



# HMM decoding



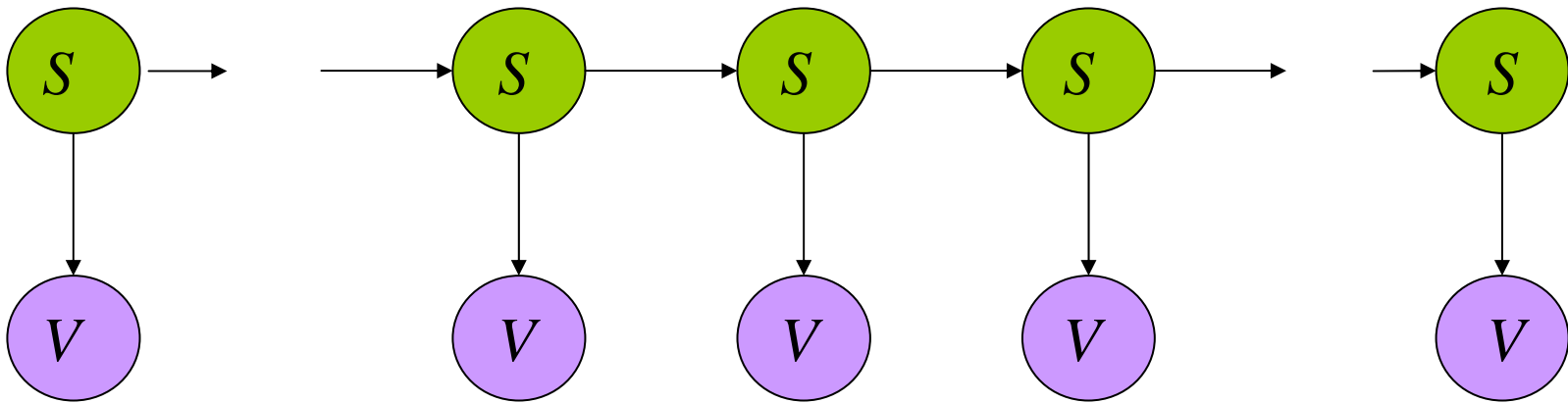
Once the HMM is constructed, it can be used to predict how many pastries of each type will be sold each day (can be used by a secondary process to determine the optimal number of pastries to produce each day)

→ Need to determine the most likely sequence of states  $Q^* = \{q_1^*, \dots, q_5^*\}$  and use this to determine the expected numbers of pastries in each category for each state.

# Why HMM?

- To encode efficiently a varying signal (e.g. **changing through time**)
  - ⇒ Compact representation
  - ⇒ Statistical model
- To **recognize** new signals
- To **predict** the outcome of a signal
- To **generate** new signals that are generalized over several examples

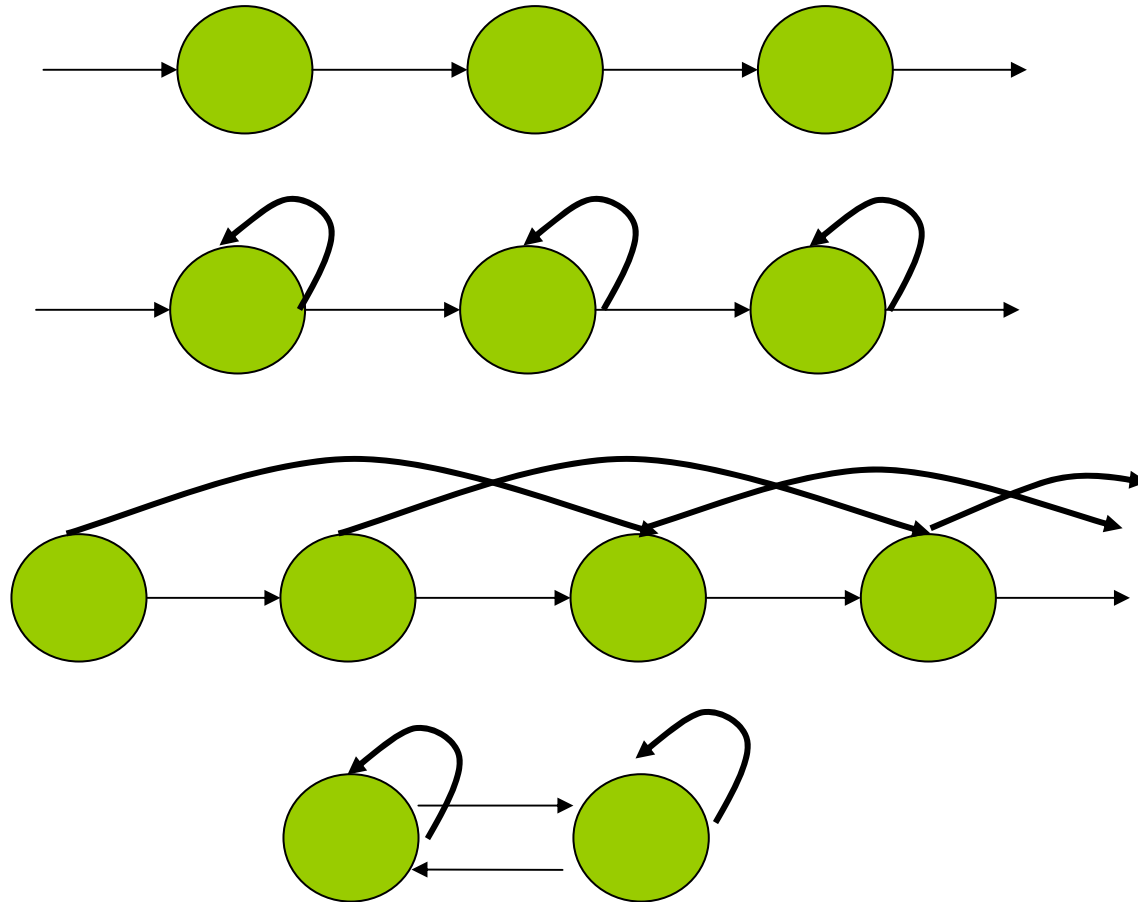
# HMM Formalism



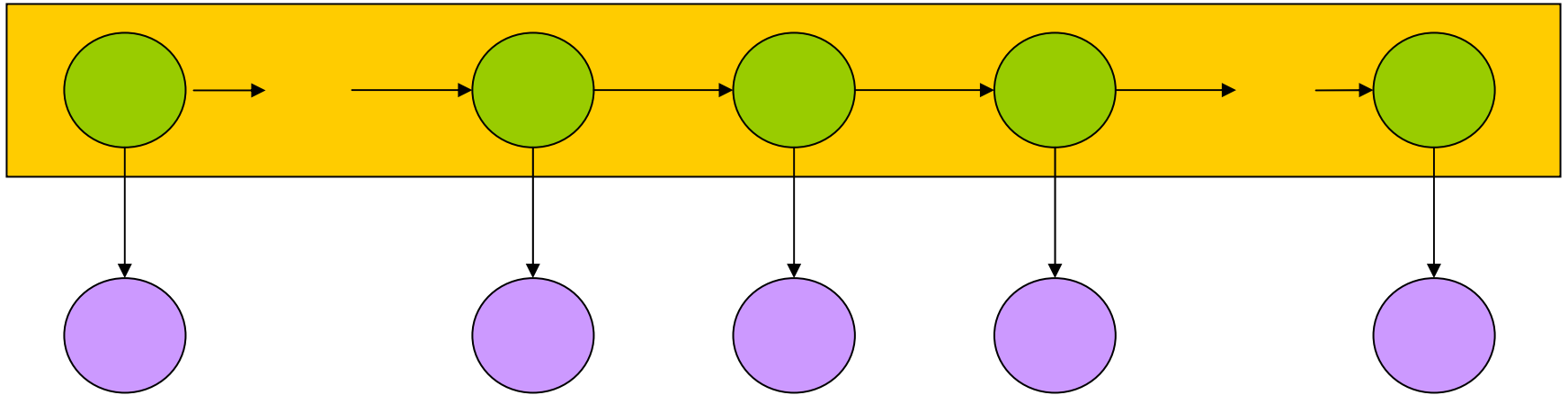
- Graphical Model
- Arrows indicate transition probabilities between states
- Green circles are the ***hidden states***
- Purple circles are the ***observations***

# HMM Formalism

- Different topologies

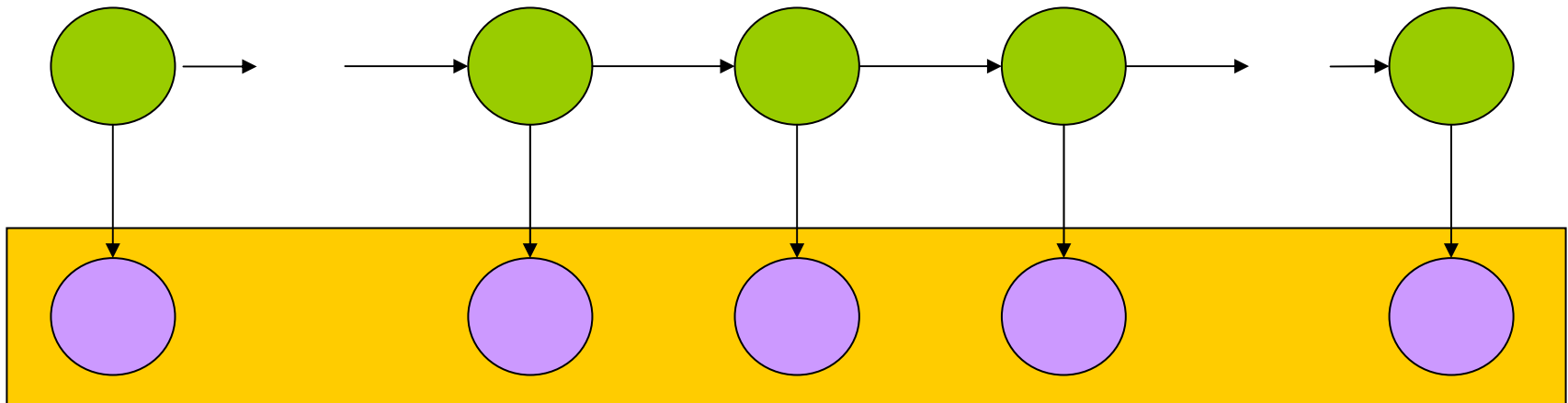


# HMM Formalism



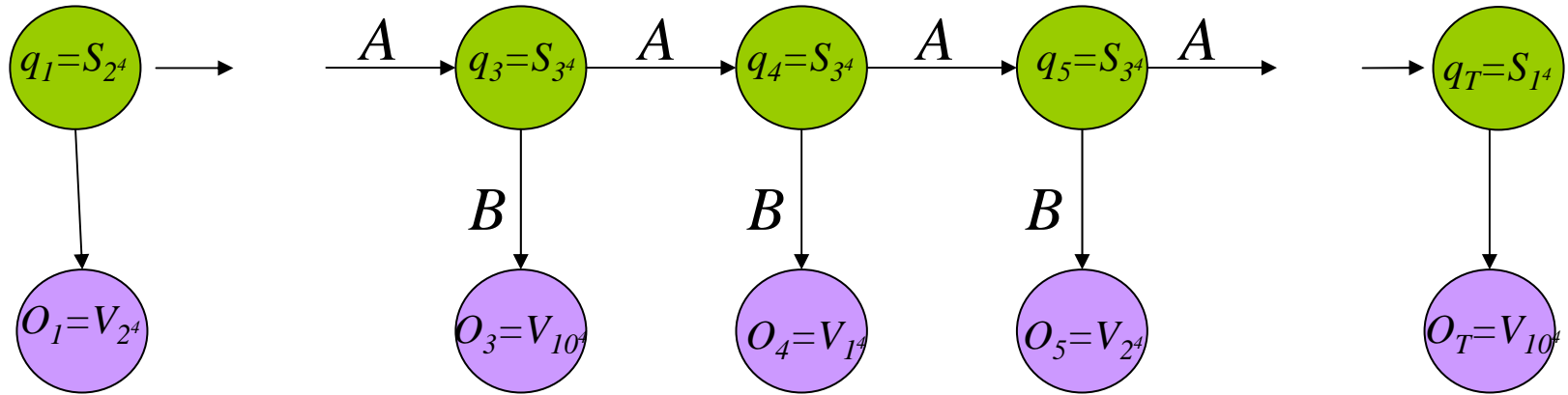
- Green circles are the *hidden states*
- $S : \{s_1 \dots s_N\}$  is the set of hidden states
- Dependent only on the previous state (1<sup>st</sup> order Markov assumption)
  - ⇒ “The future is independent of the past given the present.”

# HMM Formalism



- Purple circles are the ***observations***
- $V : \{v_1 \dots v_M\}$  is the set of possible observations (*discrete case*); generalization to continuous case possible.
- $O : \{o_1 \dots o_T\}$  is the set of actual observations. Their probability distributions depend only on their associated hidden states.

# HMM Formalism



- $\Pi = \{\pi_i\}$  are the **initial state probabilities**

$$\pi_i = P(q_1=s_i)$$

- $A = \{a_{ij}\}$  are the **transition probabilities**

$$a_{ij} = P(q_{t+1}=s_j \mid q_t=s_i), \quad i,j=1,\dots,N, \quad t=1,\dots,T-1$$

- $B = \{b_{ik}\}$  are the **emission probabilities**

$$b_{ik} = P(o_t=v_k \mid q_t=s_i) \quad i=1,\dots,N, k=1,\dots,M, \quad t=1,\dots,T$$

# The Three Basic HMM Problems

- Problem 1 (**Estimation**): Given the observation sequence  $O = \{o_1, \dots, o_T\}$  and an HMM model  $\lambda = (A, B, \pi)$ , how do we compute  $P(O | \lambda)$ , the probability of  $O$  given the model?
- Problem 2 (**Decoding**): Given the observation sequence  $O = \{o_1, \dots, o_T\}$  and an HMM model  $\lambda = (A, B, \pi)$ , how do we find the state sequence  $Q = \{q_1, \dots, q_T\}$  that best explains the observations?
- Problem 3 (**Maximization**): How do we adjust the model parameters  $\lambda = (A, B, \pi)$  to maximize  $P(O | \lambda)$ ?

# Problem 1: Estimation

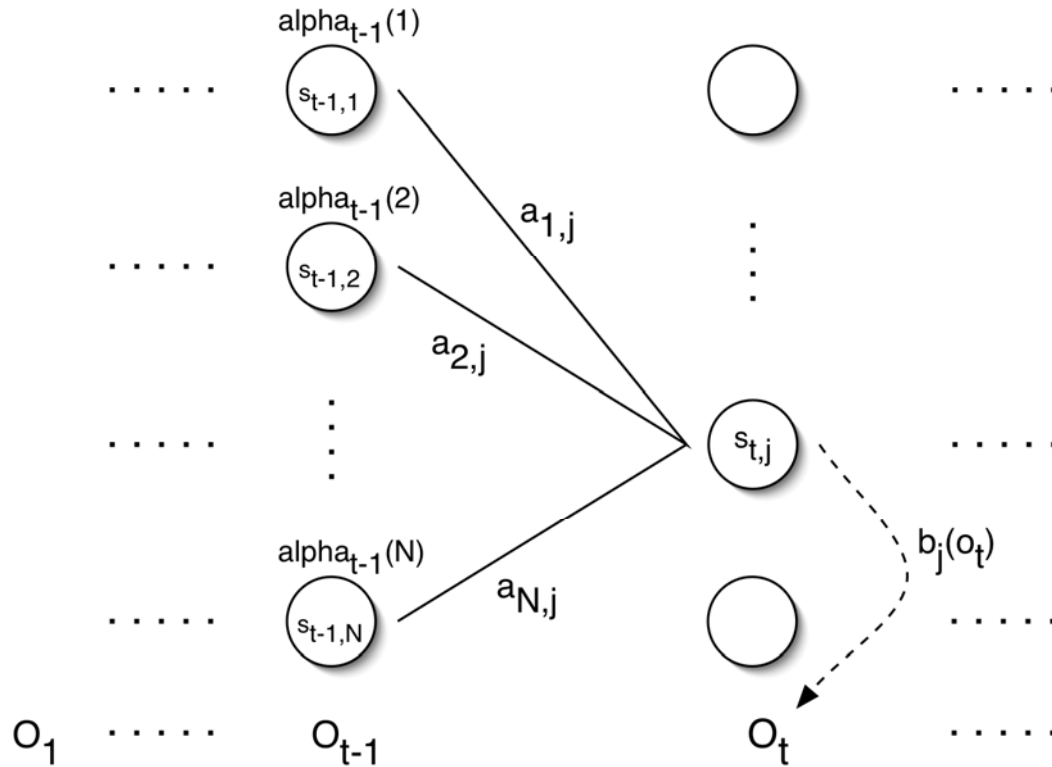
- $P(O | \lambda)$  could be computed by considering all possible state sequences in the HMM:

$$P(O | \lambda) = \sum_q P(O | q, \lambda) P(q | \lambda)$$

- However, naïve computation is very expensive. Given  $T$  observations and  $N$  states, there are  $N^T$  possible state sequences ( $(2T-1) N^T$  multiplications and  $N^T-1$  additions).  
⇒ Even small HMMs, e.g.  $T=10$  and  $N=10$ , contain 10 billion different paths
- Solution to this and problem 2 is to use dynamic programming (**Forward-Backward computation**)

# Forward Procedure

$$\alpha_{t-1}(i) = P(o_1 \dots o_{t-1}, q_{t-1} = s_i \mid \lambda)$$



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

# Forward Procedure

- **Initialization:**  $\alpha_1(i) = \pi_i b_i(o_1) \quad 1 \leq i \leq N$

- **Induction:**

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

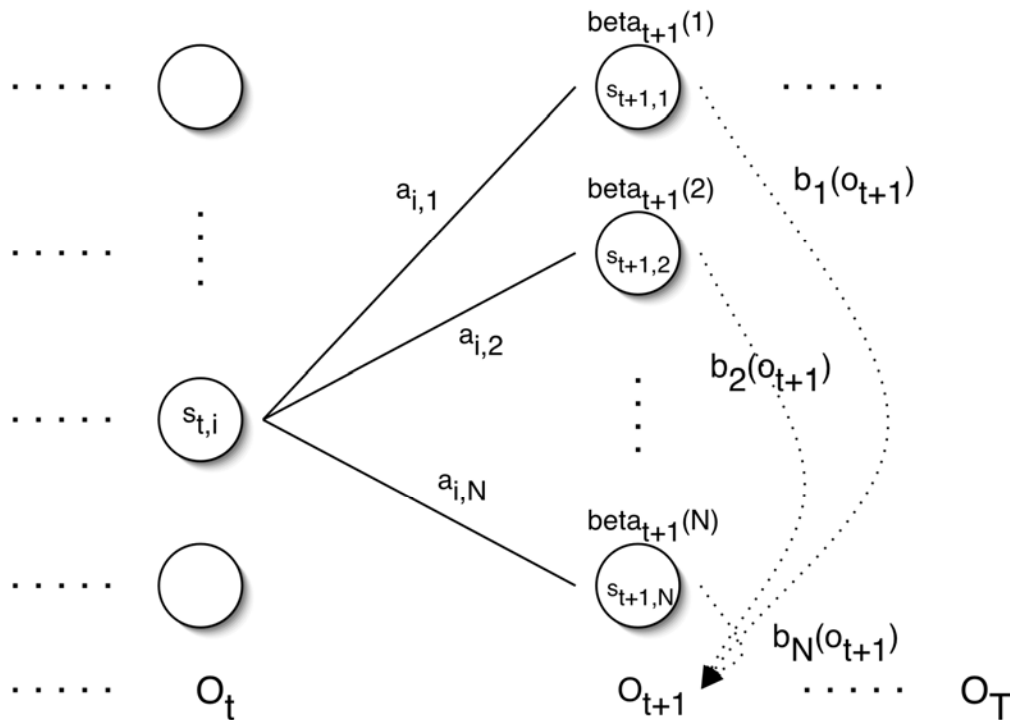
- **Termination:**  $P(O | \lambda) = \sum_{i=1}^N \alpha_T(i)$

# Forward Procedure Complexity

- In the naïve approach to solving problem 1 it would take on the order of  $2T*N^T$  computations
- The forward algorithm takes on the order of  $N^2T$  computations.

# Backward Procedure

$$\beta_t(i) = P(o_{t+1} \dots o_T \mid q_t = s_i, \lambda)$$



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

# Backward Procedure

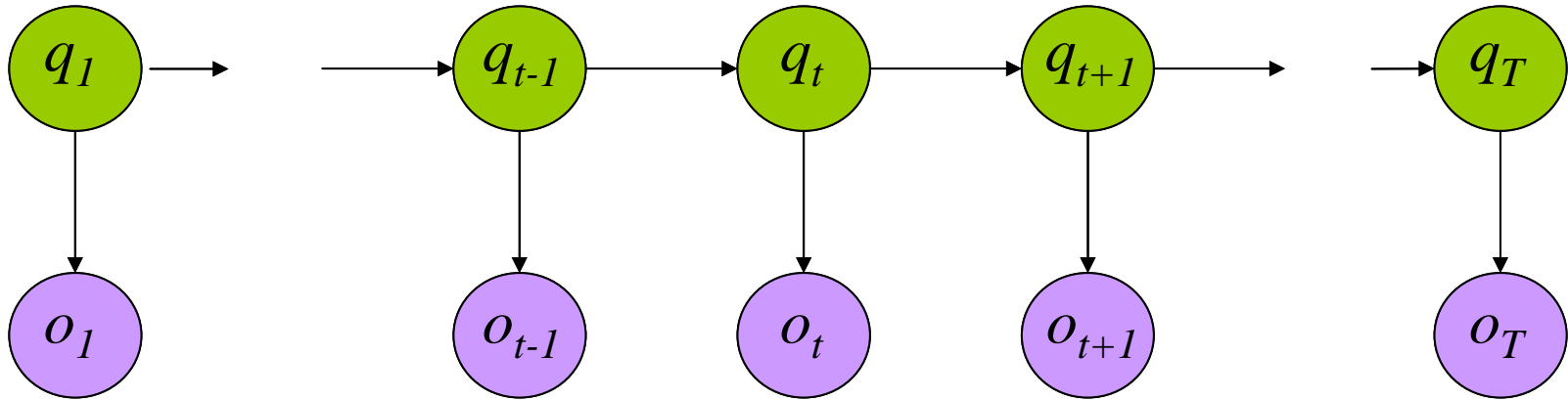
- **Initialization:**  $\beta_T(i) = 1, \quad 1 \leq i \leq N$

- **Induction:**

$$\beta_t(i) = \left[ \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right] \quad t = T-1 \dots 1, 1 \leq i \leq N$$

- **Termination:**  $P(O | \lambda) = \sum_{i=1}^N \pi_i \beta_1(i)$

# Problem 1: Estimation



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

**Forward Procedure**

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

**Backward Procedure**

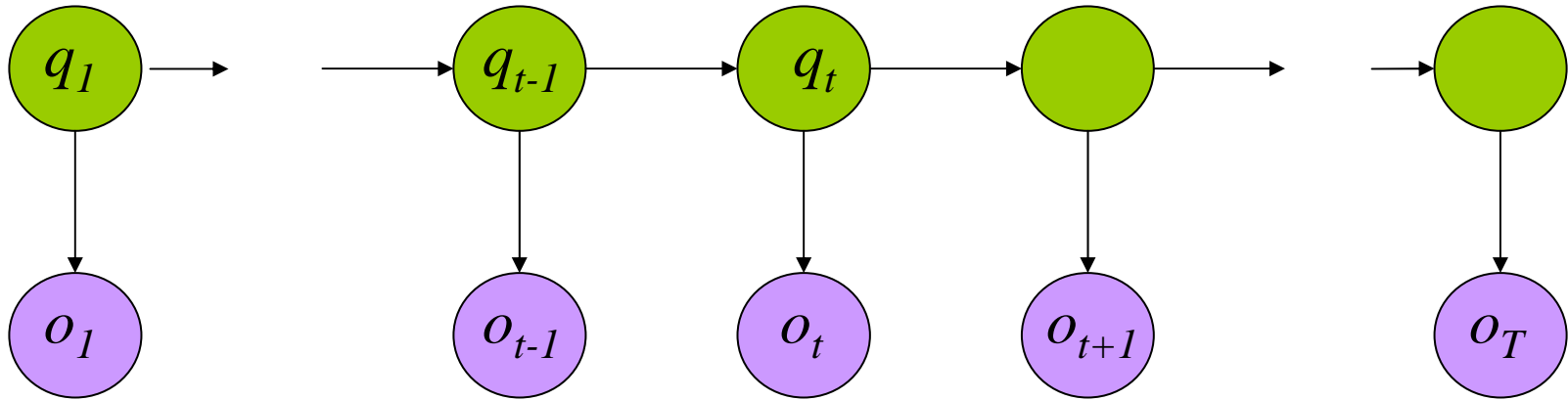
$$P(O | \lambda) = \sum_{i=1}^N \alpha_i(t) \beta_i(t) \quad \forall t=1, \dots, T$$

**Combination**

# Problem 2: Decoding

- For Problem 1 (Estimation), we have computed the probability of an observed sequence by considering different possible paths.
- For Problem 2, we want to find **the path with the highest probability**
  - ⇒ We want to find the state sequence  $Q=q_1\dots q_T$ , such that  $Q = \underset{Q'}{\operatorname{argmax}} P(Q' | O, \lambda)$

# Problem 2: Decoding



$$\gamma_t(j) = \frac{\alpha_j(t) \beta_j(t)}{\sum_{i=1}^N \alpha_i(t) \beta_i(t)}$$

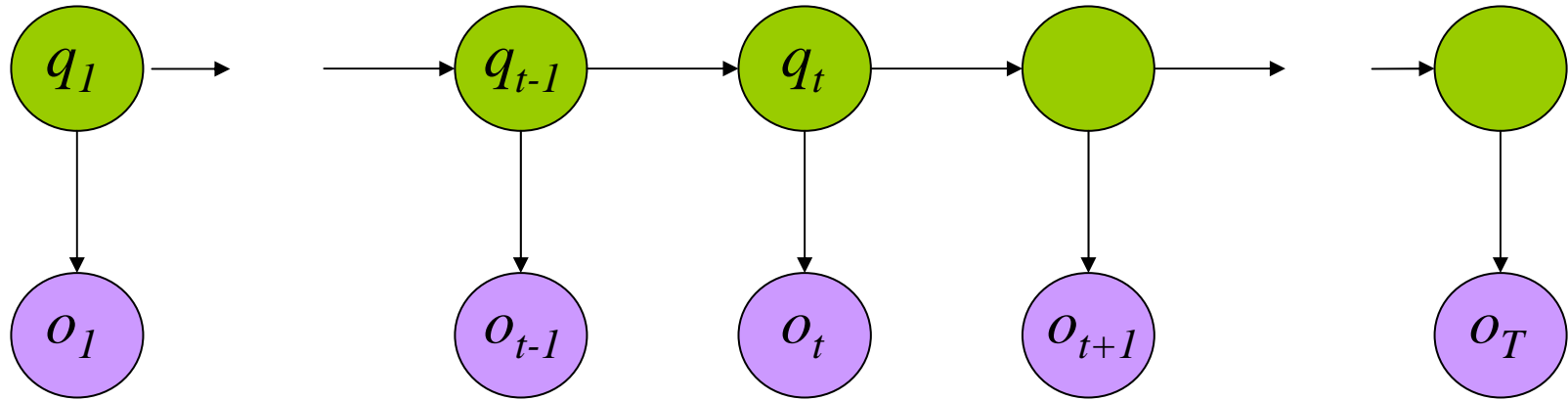
Find the optimum state at each time step individually.

## Problem 2: Decoding

- We can then find the most likely sequence  $Q$  by computing the most likely state  $q$  at each time step  $t$ , using:

$$q_t = \arg \max_{1 \leq i \leq N} (\gamma_t(i))$$

# Problem 2: Decoding - Viterbi Algorithm



$$\delta_j(t) = \max_{q_1 \dots q_{t-1}} p(q_1 \dots q_{t-1}, q_t = j, o_1 \dots o_t)$$

The state sequence maximizing the probability of a path which accounts for the first  $t$  observations and ends in state  $j$ .

## Problem 2: Decoding - **Viterbi** Algorithm

- **Initialization:**  $\delta_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N$
- **Induction:**  $\delta_t(j) = \left[ \max_{1 \leq i \leq N} \delta_{t-1}(i) a_{ij} \right] b_j(o_t)$
- **Termination:**  $q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$

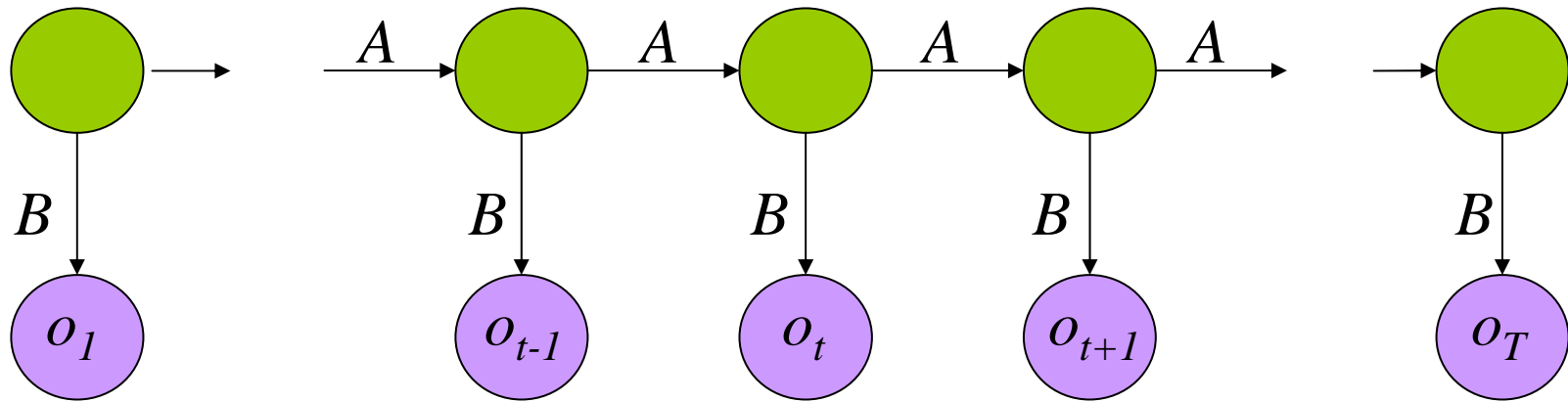
# Problem 3: Maximization

- Up to now we have assumed that we knew the underlying model  $\lambda = (A, B, \pi)$ .
- We want **to adjust the parameters with respect to the training data**, i.e., we are looking for a model such that  $\lambda' = \underset{\lambda}{\operatorname{argmax}} P(O | \lambda)$ .
- Unfortunately, there is no known way to analytically find a global maximum, but it is possible to find a local maximum through iterations.

# Problem 3: Maximization

- Use of the **Baum-Welch** algorithm, which is similar to **Expectation-Maximization (EM)**
- Given an initial model  $\lambda$ , find a model  $\lambda'$ , such that  $P(O | \lambda') \geq P(O | \lambda)$
- Using an initial set of parameters, the algorithm **iteratively re-estimates the parameters** and improves the probability that the given observations have been generated by the model

# Problem 3: Maximization



- **E-step:** Given an observation sequence and a model, find the state sequence that is most likely to have produced that sequence.
- **M-step:** Given a model and observation sequence, update the model parameters to better fit the observations.

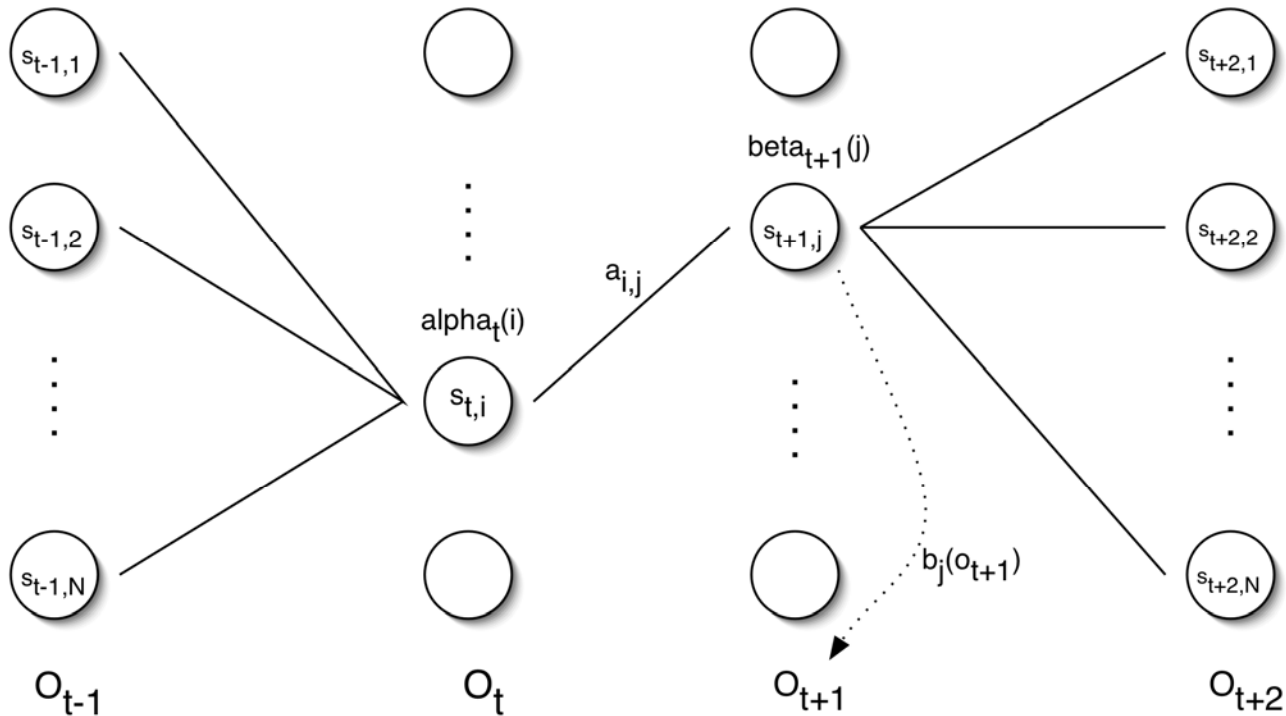
# Problem 3: Maximization

- Three parameters need to be re-estimated:
  - Initial state distribution:  $\pi_i$
  - Transition probabilities:  $a_{ij}$
  - Emission probabilities:  $b_i$
- What is the probability of being in state  $s_i$  at time  $t$  and going to state  $s_j$ , given the current model and observation sequence?

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

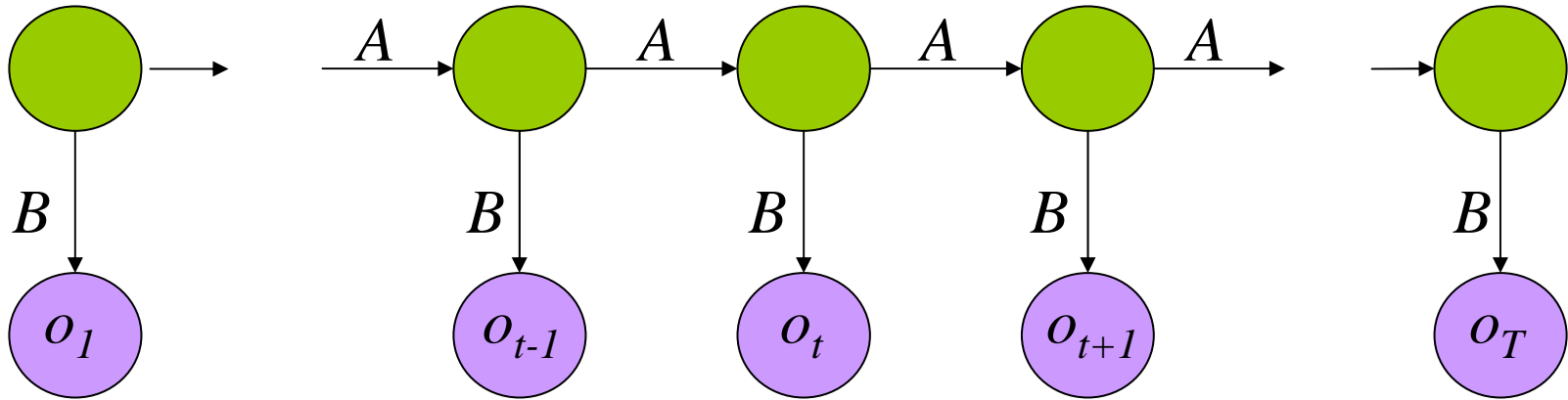
# Re-estimating Transition Probabilities

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



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

# Problem 3: Maximization



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

Probability of traversing an arc

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

Probability of being in state  $i$

# Problem 3: Maximization

- Update of the **initial states probabilities**

$\hat{\pi}_i$  = expected number of times in state  $s_i$  at time 1

$$\hat{\pi}_i = \gamma_1(i)$$

# Problem 3: Maximization

- Update of the **transition probabilities**

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i}$$

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

# Problem 3: Maximization

- Update of the **emission probabilities**

$$\hat{b}_i(v_k) = \frac{\text{expected number of times in state } s_i \text{ and observe symbol } v_k}{\text{expected number of times in state } s_i}$$

$$\hat{b}_i(v_k) = \frac{\sum_{\{t:o_t=v_k\}} \gamma_t(i)}{\sum_{t=1}^T \gamma_t(i)}$$

Can be replaced by an estimate of continuous density, such as GMM

# HMM extensions

HMM is a parametric technique (Fixed number of states, fixed topology)

→ Heuristics to determining the optimal number of states

$X$  : dataset;  $N$  : number of datapoints;  $K$  : number of free parameters

- Akaike Information Criterion:  $AIC = -2 \ln L + 2K$

- Bayesian Information Criterion:  $BIC = -2 \ln L + K \ln(N)$

$L$ : maximum likelihood of the model given  $K$  parameters

Choosing AIC versus BIC depends on the application:

→ Is the purpose of the analysis to make predictions, or to decide which model best represents reality?

AIC may have better predictive ability than BIC, but BIC finds a computationally more efficient solution.

# HMM extensions

HMM is a parametric technique (Fixed number of states, fixed topology)

→ Heuristics to determining the optimal number of states

$X$  : dataset;  $N$  : number of datapoints;  $K$  : number of free parameters

- Deviation Information Criterion:

$$\text{DIC} = E\{D(K)\} - D(E\{K\}) \quad \text{with} \quad D(K) = -2 \ln p(X | K)$$

Models with smaller DIC should be preferred to models with larger DIC.

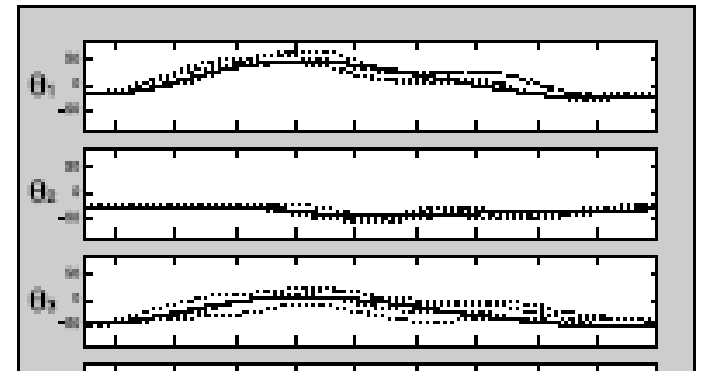
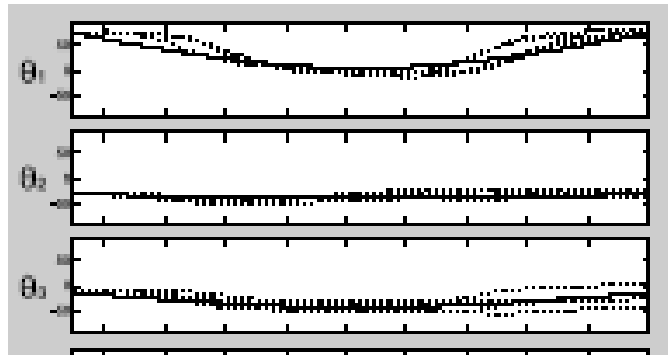
Models are penalized by the value of  $D(E\{K\})$  which favors a good fit, and (in common with AIC and BIC) by the effective number of parameters.

# HMM: Applications



# HMM: Applications

## Recognizing Motions: Which Representation?



PCA and ICA techniques are used to project data onto an uncorrelated or independent representation, to reduce the dimensionality of the dataset and to proceed to a classification across the principal components

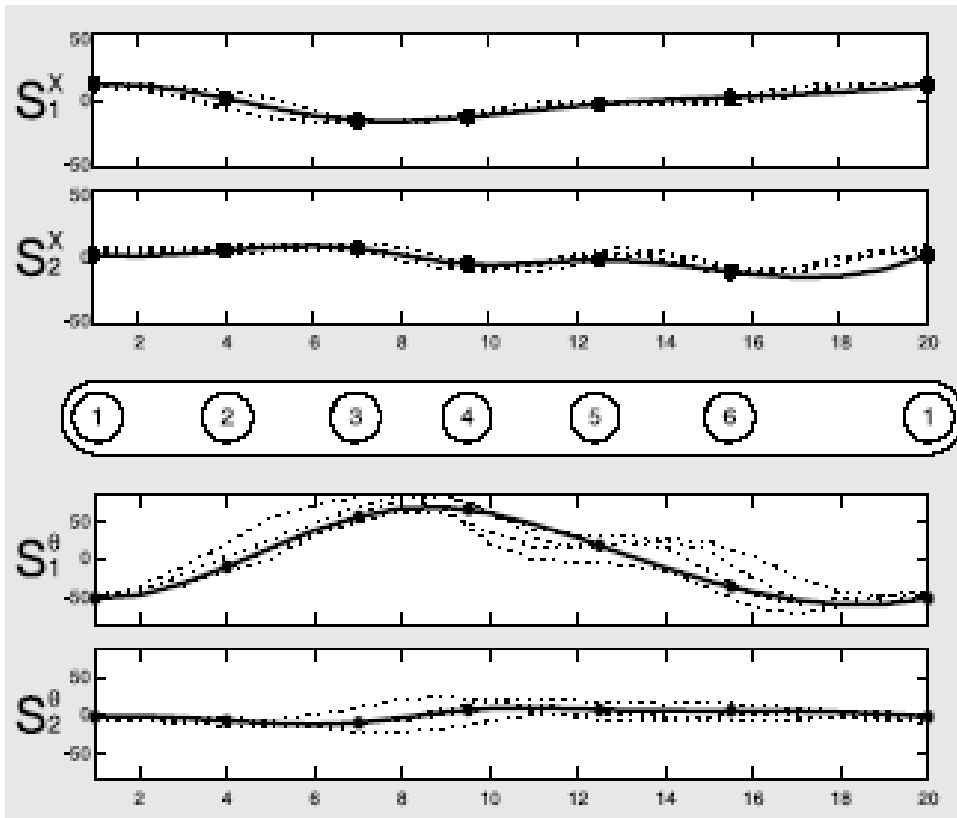
# HMM: Applications

## Encoding and Recognizing Gestures



1<sup>st</sup> and 2<sup>nd</sup> PC components  
In hand path

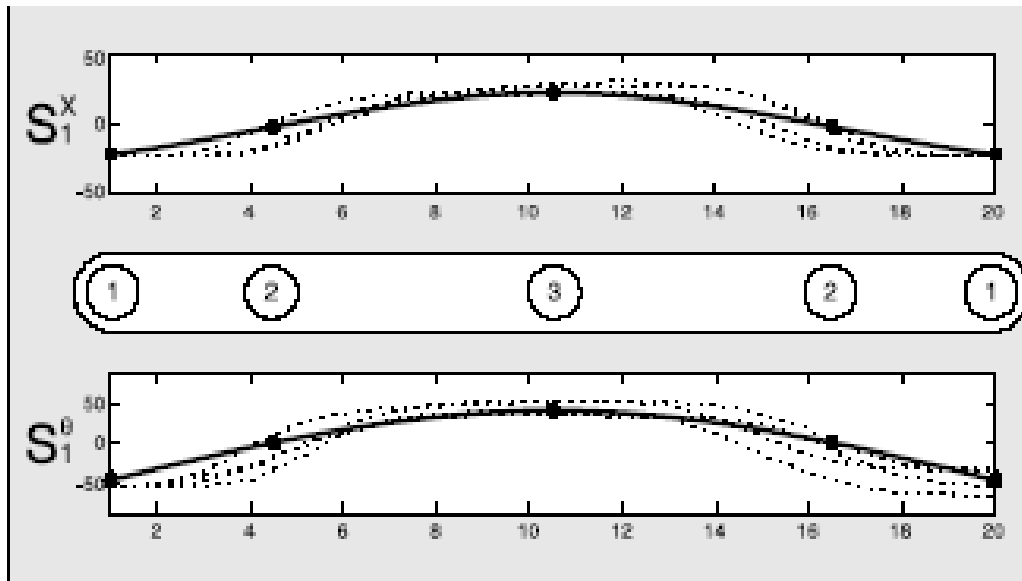
1<sup>st</sup> and 2<sup>nd</sup> PC component  
In joint trajectories



Dimensionality Reduction: 4  $\rightarrow$  2

# HMM: Applications

## Encoding and Recognizing Gestures



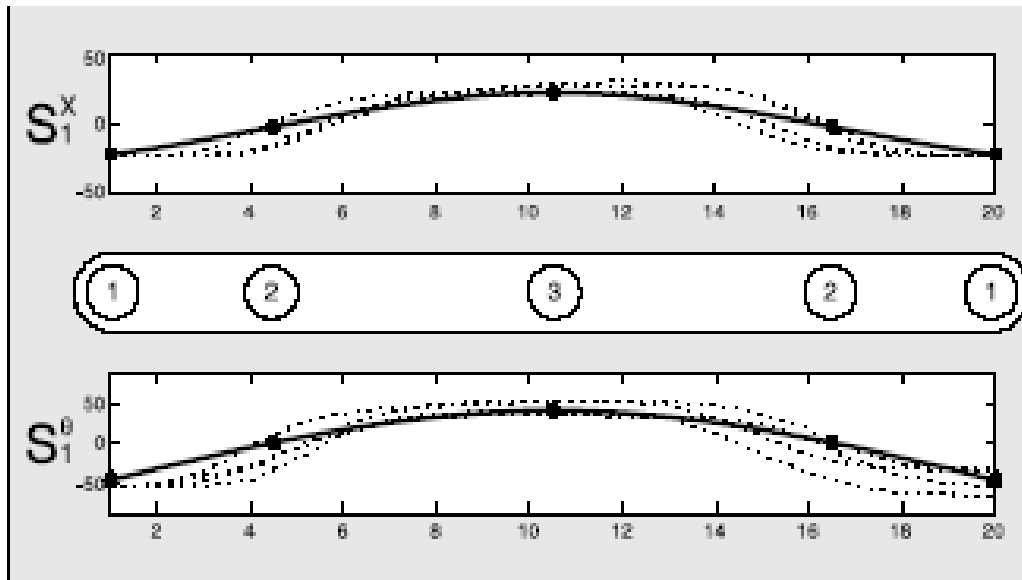
1<sup>st</sup> PC component  
In hand path

1<sup>st</sup> PC component  
In joint trajectories

Dimensionality Reduction: 4  $\rightarrow$  1

# HMM: Applications

## Encoding and Recognizing Gestures



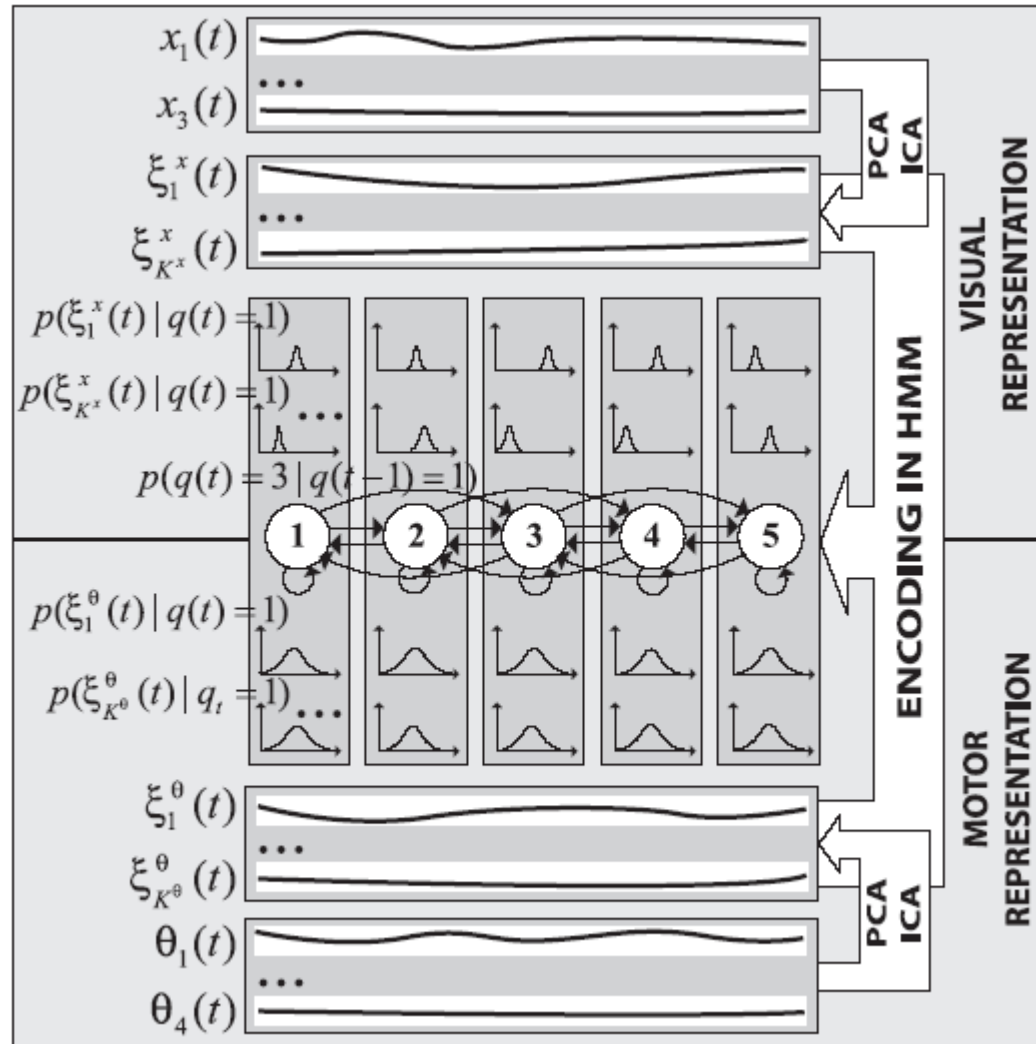
1<sup>st</sup> PC component  
In hand path

1<sup>st</sup> PC component  
In joint trajectories

Trajectories are then segmented and encoded as time series in either Hidden Markov Models or time-delay neural networks

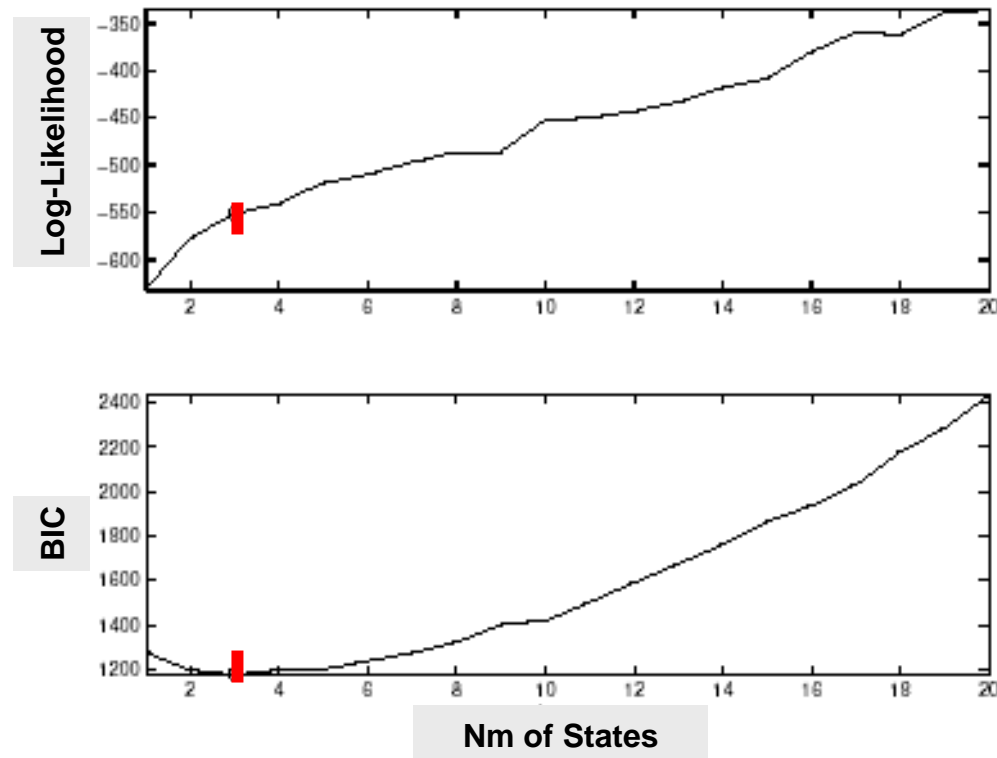
# HMM: Applications

## From Recognizing to Reproducing Gestures



# HMM: Applications

## From Recognizing to Reproducing Gestures

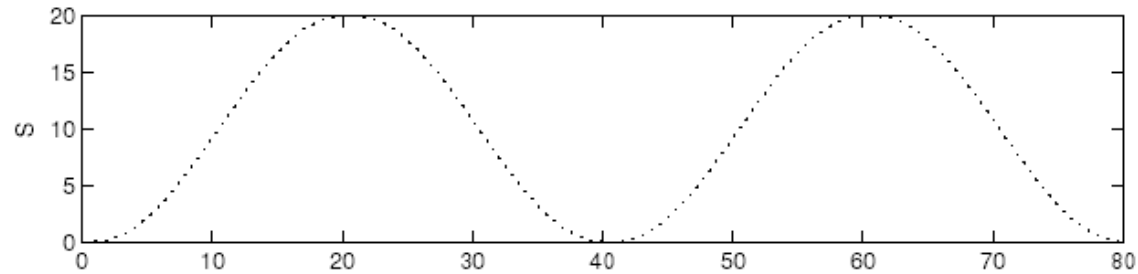


BIC Criterion: To determine the optimal number of states in the HMM to encode the data. Tradeoff between maximizing the log-likelihood of the model and the number of parameters required to encode the data (e.g. 3 states in the “waving gesture”).

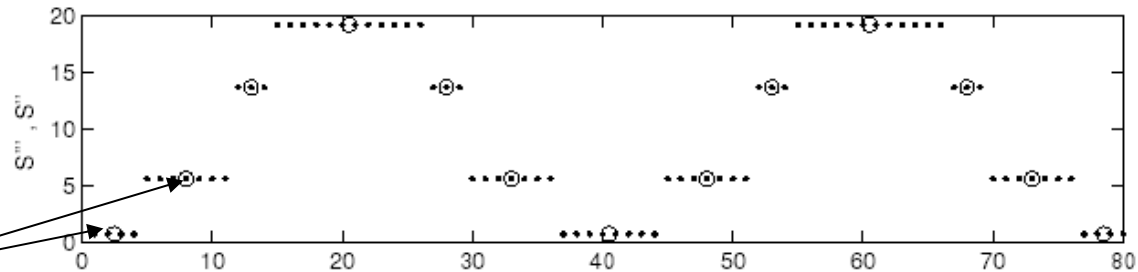
# HMM: Applications

## From Recognizing to Reproducing Gestures

Original Signal

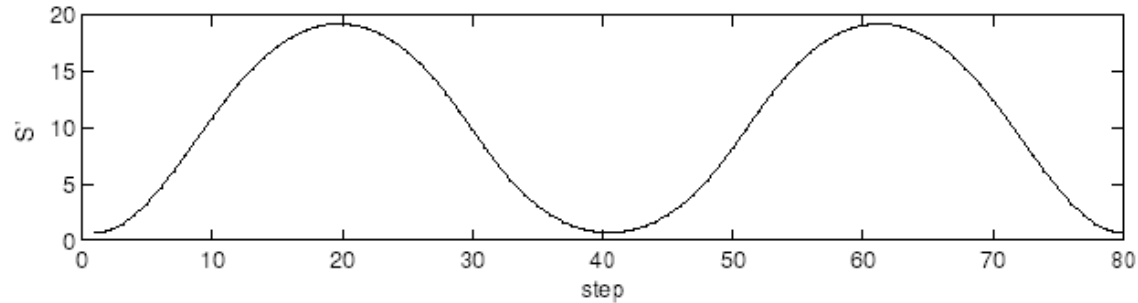


Sequence of states retrieved by Viterbi.



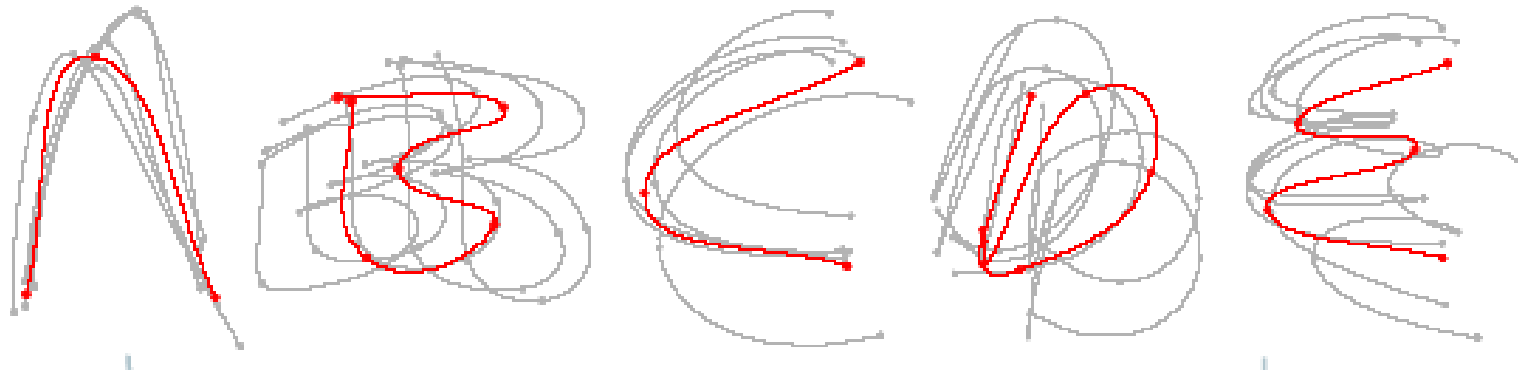
Keypoints

Reconstructed Signal



# HMM: Applications

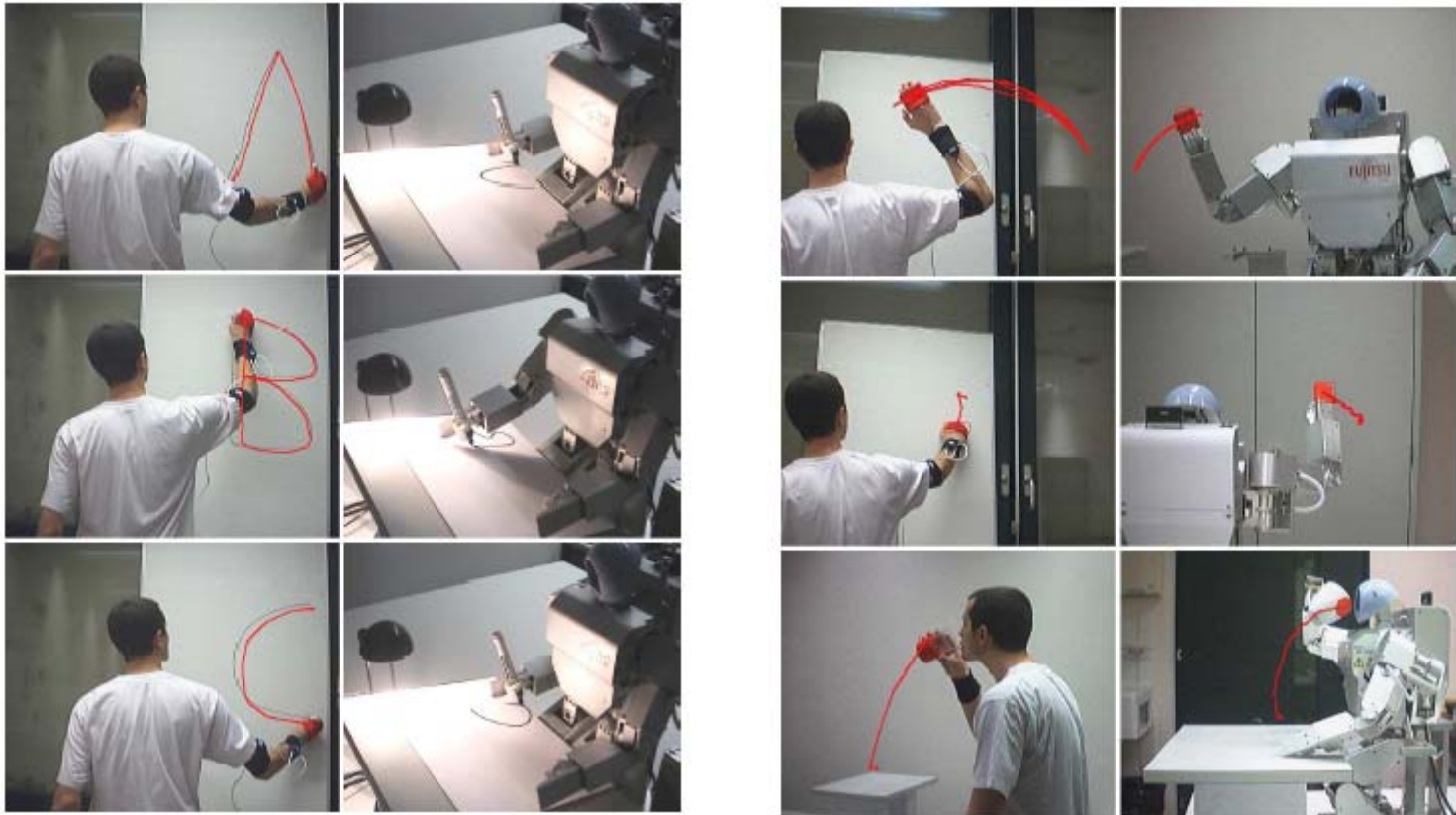
From Recognizing to Reproducing Gestures



Recognizing and Reproducing Drawings of Letters of the Alphabet

# HMM: Applications

## From Recognizing to Reproducing Gestures



The method allows to disambiguate very robustly  
across various gestures

# HMM summary and extensions

## Pros:

- Was used for a variety of applications (speech processing, gesture recognition, robotics, vision, etc)
- Well documented, lots of source code
- Nice extension to, e.g., hierarchical HMM, using GMM for modeling the density of the observations

## Cons & extensions:

- Choosing the topology of the HMM may be difficult → methods for estimating it automatically
  - Not incremental
- Particle filters address partly these two drawbacks

# HMM reference

- Rabiner, L.R. (1989) **“A tutorial on hidden Markov models and selected applications in speech recognition”**, Proceedings of the IEEE, 77:2