



# Hidden Markov Model

---

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA' DI PISA

DAVIDE.BACCIU@UNIFI.IT

# Lecture Plan (Part I and II)

---

- A probabilistic model for sequences: **Hidden Markov Models (HMMs)**
- Exact inference on a **chain** with **observed and unobserved** variables
  - **Sum-product** message passing example
  - **Max-product** message passing example
- Using inference to **learn**: the Expectation-Maximization algorithm for HMMs
- Graphical models with **varying structure**: Dynamic Bayesian Networks





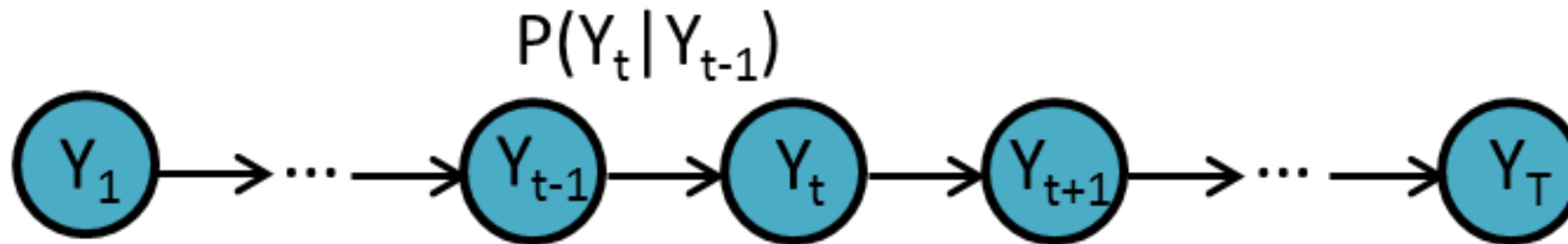
# Part I

---

Introduction to HMM and forward-backward recursion

# Sequences

---

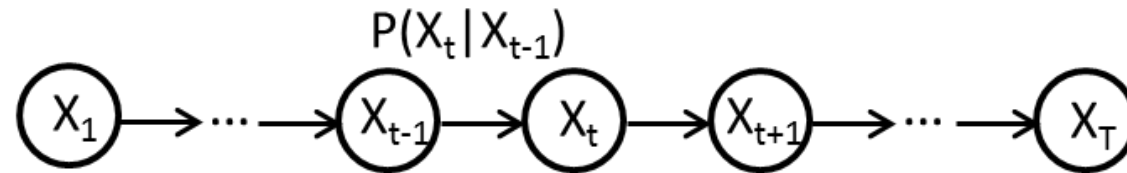


- A sequence  $\mathbf{y}$  is a collection of observations  $y_t$  where  $t$  represent the **position of the element** according to a (complete) order (e.g. **time**)
- Reference population is a set of i.i.d sequences  $\mathbf{y}^1, \dots, \mathbf{y}^N$
- Different sequences  $\mathbf{y}^1, \dots, \mathbf{y}^N$  generally have **different lengths**  $T^1, \dots, T^N$

# Markov Chain

## First-Order Markov Chain

Directed graphical model for sequences s.t. element  $X_t$  only depends on its predecessor in the sequence



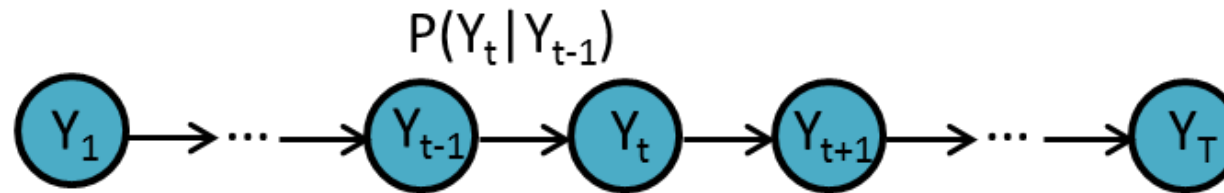
- Joint probability factorizes as

$$P(\mathbf{X}) = P(X_1, \dots, X_T) = P(X_1) \prod_{t=2}^T P(X_t | X_{t-1})$$

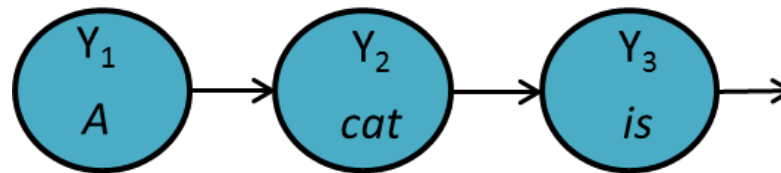
- $P(X_t | X_{t-1})$  is the transition distribution;  $P(X_1)$  is the prior distribution
- General form: an  $L$ -th order Markov chain is such that  $X_t$  depends on  $L$  predecessors

# Observed Markov Chains

Can we use a Markov chain to model the relationship between observed elements in a sequence?



Of course yes, but..

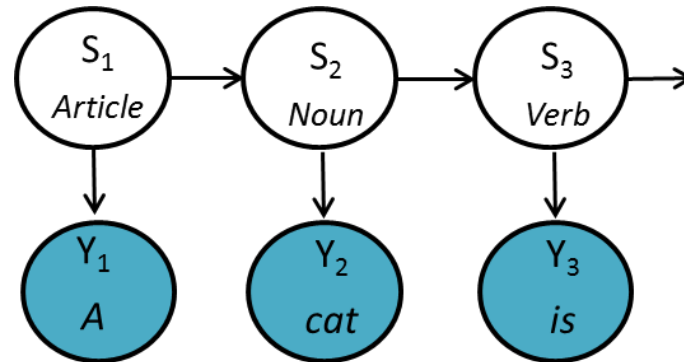


Does it make sense to represent  $P(is|cat)$ ?



# Hidden Markov Model (HMM) (I)

Stochastic process where **transition dynamics** is **disentangled from observations** generated by the process



State transition is an **unobserved** (hidden/latent) process characterized by the **hidden state variables**  $S_t$

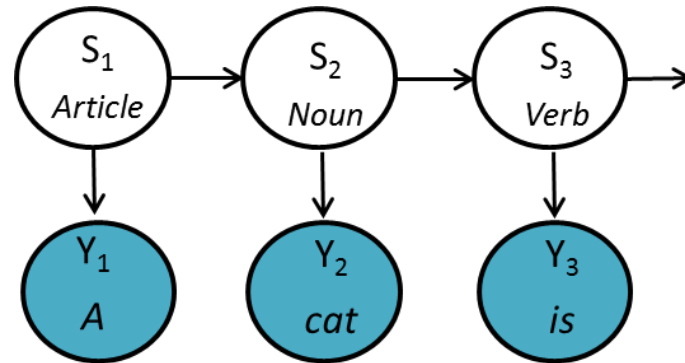
- $S_t$  are often **discrete** with value in  $\{1, \dots, C\}$
- Multinomial **state transition** and prior probability (**stationarity assumption**)

$$A_{ij} = P(S_t = i | S_{t-1} = j) \text{ and } \pi_i = P(S_1 = i)$$



# Hidden Markov Model (HMM) (II)

Stochastic process where **transition dynamics** is **disentangled from observations** generated by the process



Observations are generated by the **emission distribution**

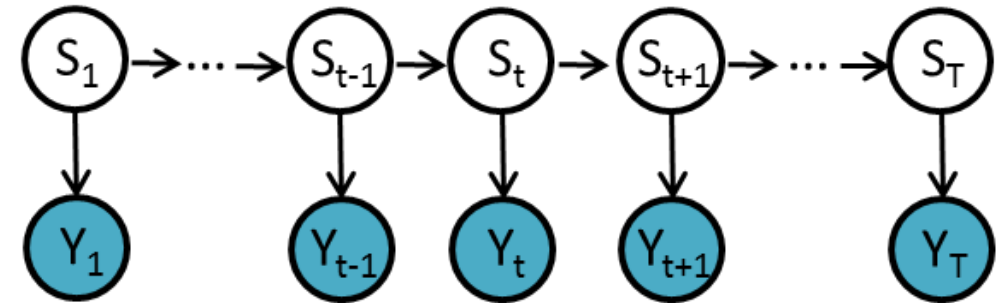
$$b_i(y_t) = P(Y_t = y_t | S_t = i)$$



# HMM Joint Probability Factorization

Discrete-state HMMs are parameterized by  $\theta = (\pi, A, B)$  and the finite number of hidden states  $C$

- State transition and prior distribution  $A$  and  $\pi$
- Emission distribution  $B$  (or its parameters)

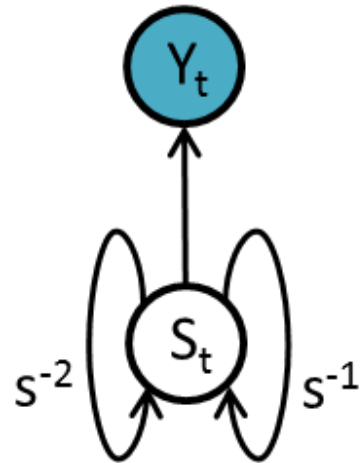
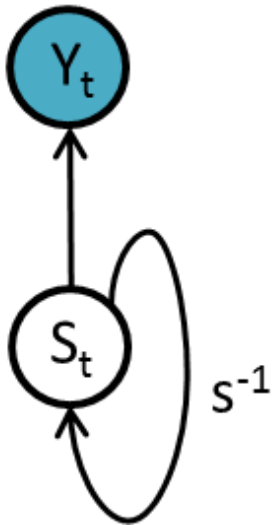


$$P(Y = \mathbf{y}) = \sum_{\mathbf{s}} P(Y = \mathbf{y}, \mathbf{S} = \mathbf{s})$$
$$= \sum_{S_1, \dots, S_T} \left\{ P(S_1 = s_1) P(Y_1 = y_1 | S_1 = s_1) \prod_{t=2}^T P(S_t = s_t | S_{t-1} = s_{t-1}) P(Y_t = y_t | S_t = s_t) \right\}$$



# HMMs as a Recursive Model

A graphical framework describing **how contextual information is recursively encoded** by both probabilistic and neural models

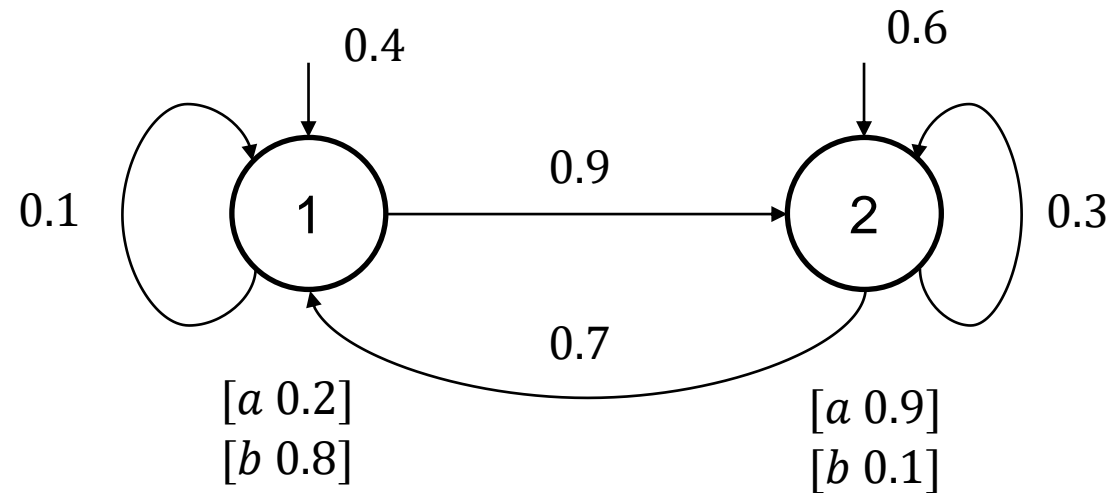


- Indicates that the hidden state  $S_t$  at time  $t$  is dependent on **context information** from
  - The previous timestep  $s^{-1}$
  - Two timesteps earlier  $s^{-2}$
  - ...
- When applying the recursive model to a sequence (**unfolding**), it generates the corresponding **directed graphical model**



# HMMs as Automata

---



Can be generalized to transducers

# 3 Notable Inference Problems

## Definition (Smoothing)

Given a model  $\theta$  and an observed sequence  $\mathbf{y}$ , determine the **distribution of the hidden state at time  $t$**   $P(S_t | \mathbf{Y} = \mathbf{y}, \theta)$

## Definition (Learning)

Given a dataset of  $N$  observed sequences  $\mathcal{D} = \{\mathbf{y}^1, \dots, \mathbf{y}^N\}$  and the number of hidden states  $C$ , **find the parameters  $\pi, A$  and  $B$**  that maximize the probability of model  $\theta = \{\pi, A, B\}$  having generated the sequences in  $\mathcal{D}$

## Definition (Optimal State Assignment)

Given a model  $\theta$  and an observed sequence  $\mathbf{y}$ , find an **optimal state assignment**  $\mathbf{s} = s_1^*, \dots, s_T^*$  for the hidden Markov chain



# Forward-Backward Algorithm

**Smoothing** - How do we determine the posterior  $P(S_t = i | \mathbf{y})$ ?

Exploit factorization

$$\begin{aligned} P(S_t = i | \mathbf{y}) &\propto P(S_t = i, \mathbf{y}) = P(S_t = i, \mathbf{Y}_{1:t}, \mathbf{Y}_{t+1:T}) \\ &= P(S_t = i, \mathbf{Y}_{1:t}) P(\mathbf{Y}_{t+1:T} | S_t = i) = \alpha_t(i) \beta_t(i) \end{aligned}$$

$\alpha$ -term computed as part of **forward recursion** ( $\alpha_1(i) = b_i(y_1) \pi_i$ )

$$\alpha_t(i) = P(S_t = i, \mathbf{Y}_{1:t}) = b_i(y_t) \sum_{j=1}^c A_{ij} \alpha_{t-1}(j)$$

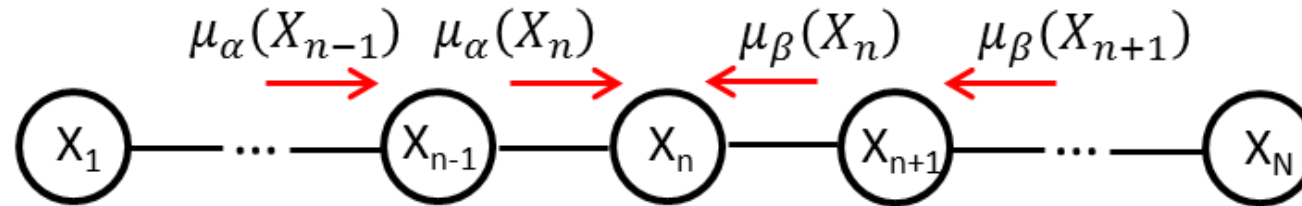
$\beta$ -term computed as part of **backward recursion** ( $\beta_T(i) = 1, \forall i$ )

$$\beta_t(j) = P(\mathbf{Y}_{t+1:T} | S_t = j) = \sum_{i=1}^c b_i(y_{t+1}) \beta_{t+1}(i) A_{ij}$$



# Sum-Product Message Passing

The **Forward-Backward algorithm** is an example of a **sum-product message passing** algorithm



A general approach to *efficiently* perform exact inference in graphical models

- $\alpha_t \equiv \mu_\alpha(X_n) \rightarrow$  forward message

$$\underbrace{\mu_\alpha(X_n)}_{\alpha_t(i)} = \sum_{\substack{X_{n-1} \\ \sum_{j=1}^C}} \underbrace{\psi(X_{n-1}, X_n)}_{b_i(y_t)A_{ij}} \underbrace{\mu_\alpha(X_{n-1})}_{\alpha_{t-1}(j)}$$



# Sum-Product Message Passing

---

The **Forward-Backward algorithm** is an example of a **sum-product message passing** algorithm

A general approach to *efficiently* perform exact inference in graphical models

- $\alpha_t \equiv \mu_\alpha(X_n) \rightarrow$  forward message
- $\beta_t \equiv \mu_\beta(X_n) \rightarrow$  backward message

$$\underbrace{\mu_\beta(X_n)}_{\beta_t(j)} = \sum_{\substack{X_{n+1} \\ \sum_{i=1}^C}} \underbrace{\psi(X_n, X_{n+1})}_{b_i(y_{t+1})A_{ij}} \underbrace{\mu_\beta(X_{n+1})}_{\beta_{t+1}(i)}$$





# Part II

---

Parameter estimation and Viterbi Algorithm



# Learning in HMM

Learning HMM parameters  $\theta = (\pi, A, B)$  by **maximum likelihood**

$$\mathcal{L}(\theta) = \log \prod_{n=1}^N P(\mathbf{Y}^n | \theta) = \log \prod_{n=1}^N \left\{ \sum_{s_1^n, \dots, s_{T_n}^n} P(s_1^n) P(Y_1^n | s_1^n) \prod_{t=2}^{T_n} P(s_t^n | s_{t-1}^n) P(Y_t^n | s_t^n) \right\}$$

- How can we deal with the unobserved random variables  $S_t^n$  and the nasty summation in the log ?
- Expectation-Maximization algorithm
  - Maximization of the **complete likelihood**  $\mathcal{L}_c(\theta)$
  - Completed with **indicator variables**

$$z_{ti}^n = \begin{cases} 1 & \text{if } n\text{-th chain is in state } i \text{ at time } t \\ 0 & \text{otherwise} \end{cases}$$



# Complete HMM Likelihood

Introduce indicator variables in  $L(\theta)$  together with model parameters  $\theta = (\pi, A, B)$

$$\begin{aligned}\mathcal{L}_c(\theta) &= \log P(\mathcal{X}, \mathcal{Z}|\theta) = \log \prod_{n=1}^N \left\{ \prod_{i=1}^C [P(S_1 = 1)P(Y_1^n|S_1 = i)]^{z_{1i}^n} \right. \\ &\quad \left. \prod_{t=2}^{T_n} \prod_{i,j=1}^C P(S_t = i|S_{t-1} = j)^{z_{ti}^n z_{(t-1)j}^n} P(Y_t^n|S_t = i)^{z_{ti}^n} \right\} \\ &= \sum_{n=1}^N \left\{ \sum_{i=1}^C z_{1i}^n \log \pi_i + \sum_{t=2}^{T_n} \sum_{i,j=1}^C z_{ti}^n z_{(t-1)j}^n \log A_{ij} + \sum_{t=1}^{T_n} \sum_{i=1}^C z_{ti}^n \log b_i(y_t^n) \right\}\end{aligned}$$



# Expectation-Maximization

---

A 2-step iterative algorithm for the maximization of **complete likelihood**  $\mathcal{L}_c(\theta)$  w.r.t. model parameters  $\theta$

**E-Step:** Given the current estimate of the model parameters  $\theta^{(t)}$ , compute

$$Q^{(k+1)}(\theta|\theta^{(k)}) = E_{Z|X,\theta^{(t)}} [\log P(X, Z|\theta)]$$

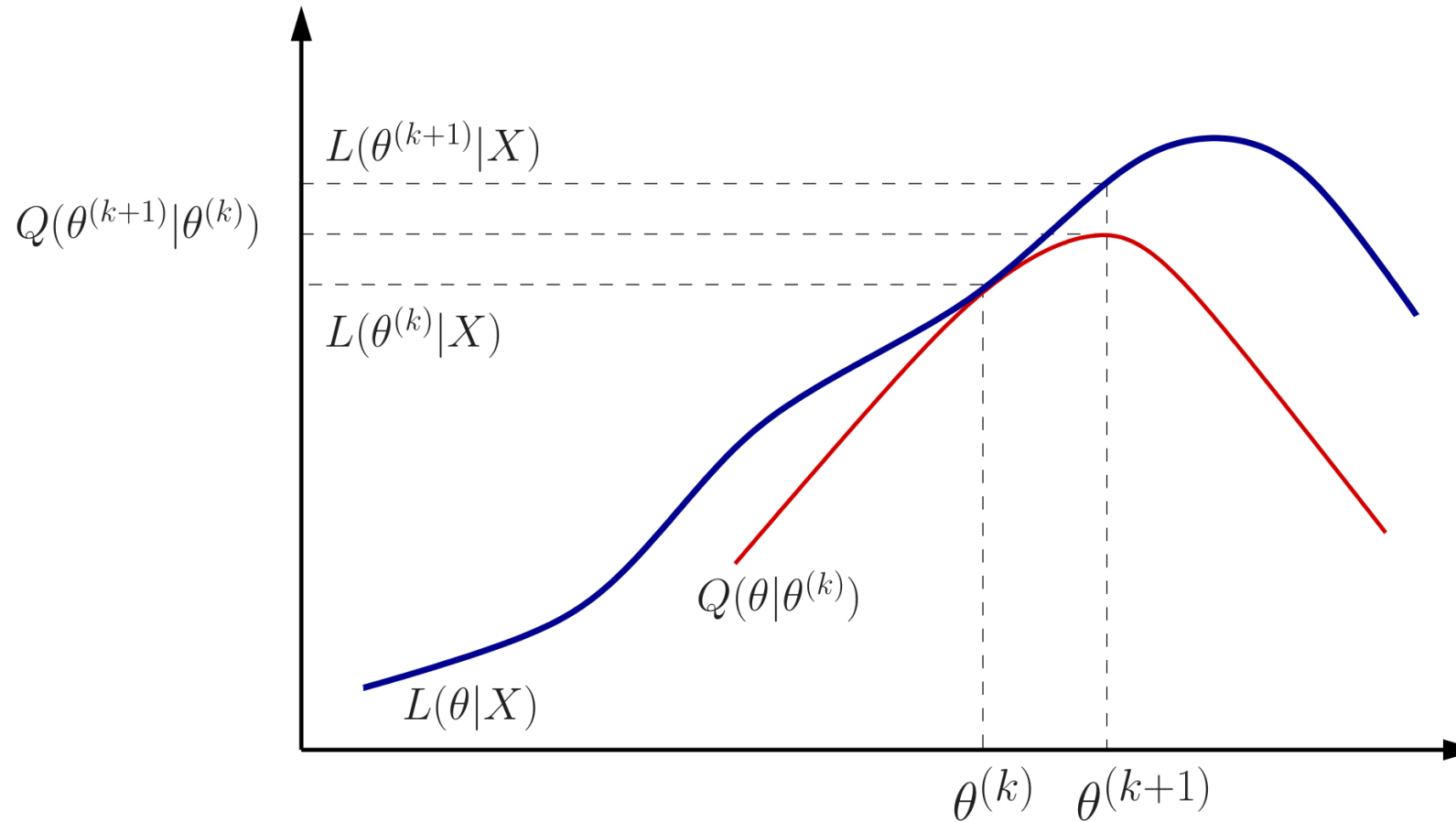
**M-Step:** Find the new estimate of the model parameters

$$\theta^{(k+1)} = \arg \max_{\theta} Q^{(k+1)}(\theta|\theta^{(k)})$$

Iterate 2 steps until  $|\mathcal{L}_c(\theta)^{k+1} - \mathcal{L}_c(\theta)^k| < \epsilon$  (or stop if maximum number of iterations is reached)



# EM Graphically



# E-Step (I)

---

Compute the expectation of the complete log-likelihood **w.r.t indicator variables**  $z_{ti}^n$  assuming (estimated) **parameters**  $\theta^k = (\pi^k, A^k, B^k)$  fixed at iteration  $k$  (i.e. **constants**)

$$Q^{(k+1)}(\theta|\theta^{(k)}) = E_{Z|\mathcal{X},\theta^{(k)}} [\log P(\mathcal{X}, \mathcal{Z}|\theta)]$$

Expectation w.r.t a (discrete) random variable  $z$  is

$$E_Z[Z] = \sum_z z \cdot P(Z = z)$$

To compute the conditional expectation  $Q^{(t+1)}(\theta|\theta^{(t)})$  for the complete HMM log-likelihood we need to estimate

$$E_{Z|\mathbf{Y},\theta^{(k)}} [z_{ti}] = P(S_t = i|\mathbf{y})$$

$$E_{Z|\mathbf{Y},\theta^{(k)}} [z_{ti}z_{(t-1)j}] = P(S_t = i, S_{t-1} = j|\mathbf{Y})$$



# E-Step (II)

---

We know how to compute the posteriors by the **forward-backward algorithm!**

$$\gamma_t(i) = P(S_t = i | \mathbf{Y}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^C \alpha_t(j)\beta_t(j)}$$

$$\gamma_{t,t-1}(i,j) = P(S_t = i, S_{t-1} = j | \mathbf{Y}) = \frac{\alpha_{t-1}(j)A_{ij}b_i(y_t)\beta_t(i)}{\sum_{m,l=1}^C \alpha_{t-1}(m)A_{lm}b_l(y_t)\beta_t(l)}$$



# M-Step (I)

---

Solve the **optimization problem**

$$\theta^{(k+1)} = \arg \max_{\theta} Q^{(k+1)}(\theta | \theta^{(k)})$$

using the information computed at the E-Step (the posteriors).

**How?**

As usual

$$\frac{\partial Q^{(k+1)}(\theta | \theta^{(k)})}{\partial \theta}$$

where  $\theta = (\pi, A, B)$  are now variables.

## Attention

Parameters can be distributions  $\Rightarrow$  need to preserve sum-to-one constraints  
(Lagrange Multipliers)



# M-Step (II)

---

State distributions

$$A_{ij} = \frac{\sum_{n=1}^N \sum_{t=2}^{T^n} \gamma_{t,t-1}^n(i, j)}{\sum_{n=1}^N \sum_{t=2}^{T^n} \gamma_{t-1}^n(j)} \quad \text{and} \quad \pi_i = \frac{\sum_{n=1}^N \gamma_1^n(i)}{N}$$

Emission distribution (multinomial)

$$B_{ki} = \frac{\sum_{n=1}^N \sum_{t=1}^{T_n} \gamma_t^n(i) \delta(y_t = h)}{\sum_{n=1}^N \sum_{t=1}^{T_n} \gamma_t^n(i)}$$

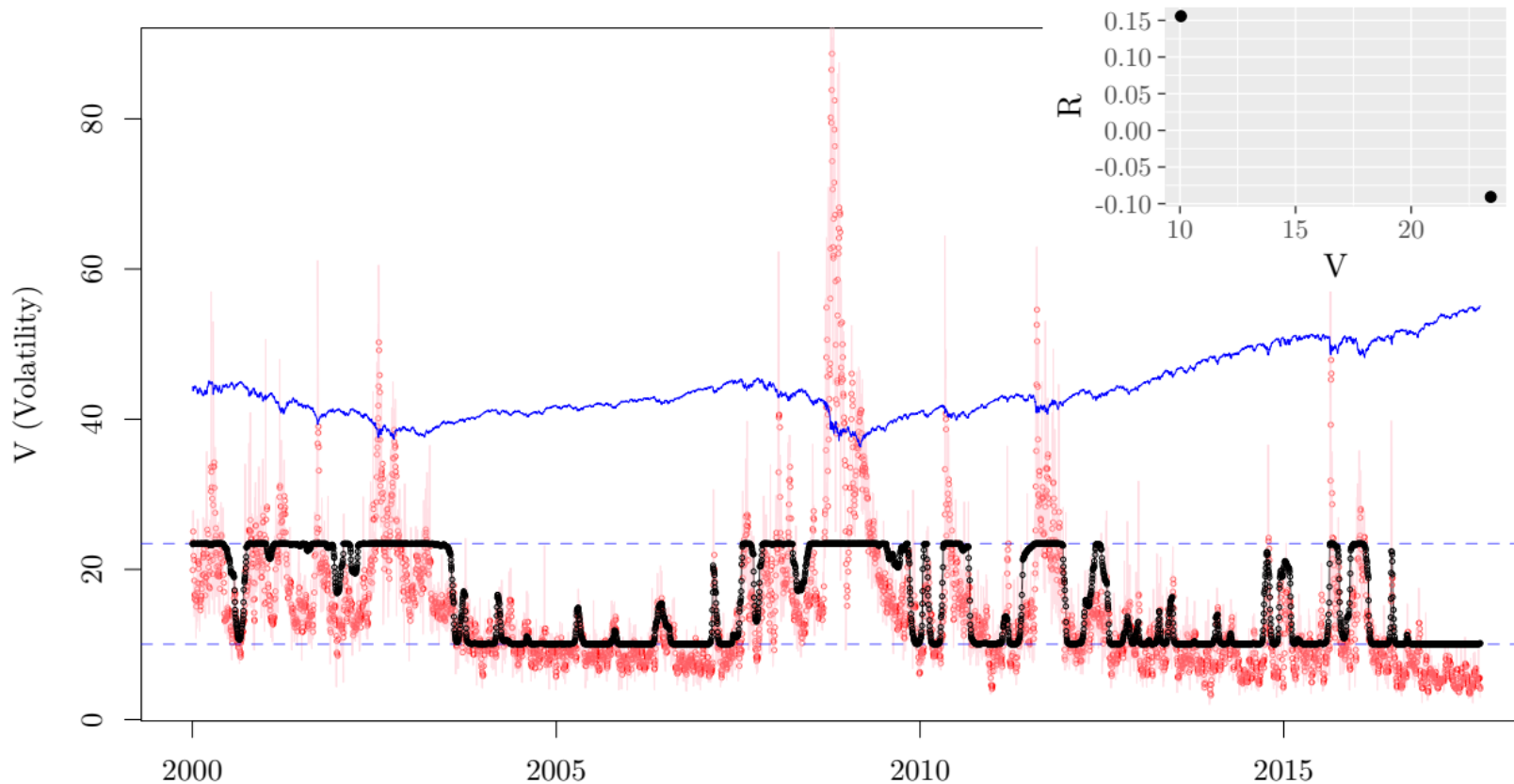
where  $\delta(\cdot)$  is the indicator function for emission symbols  $h$





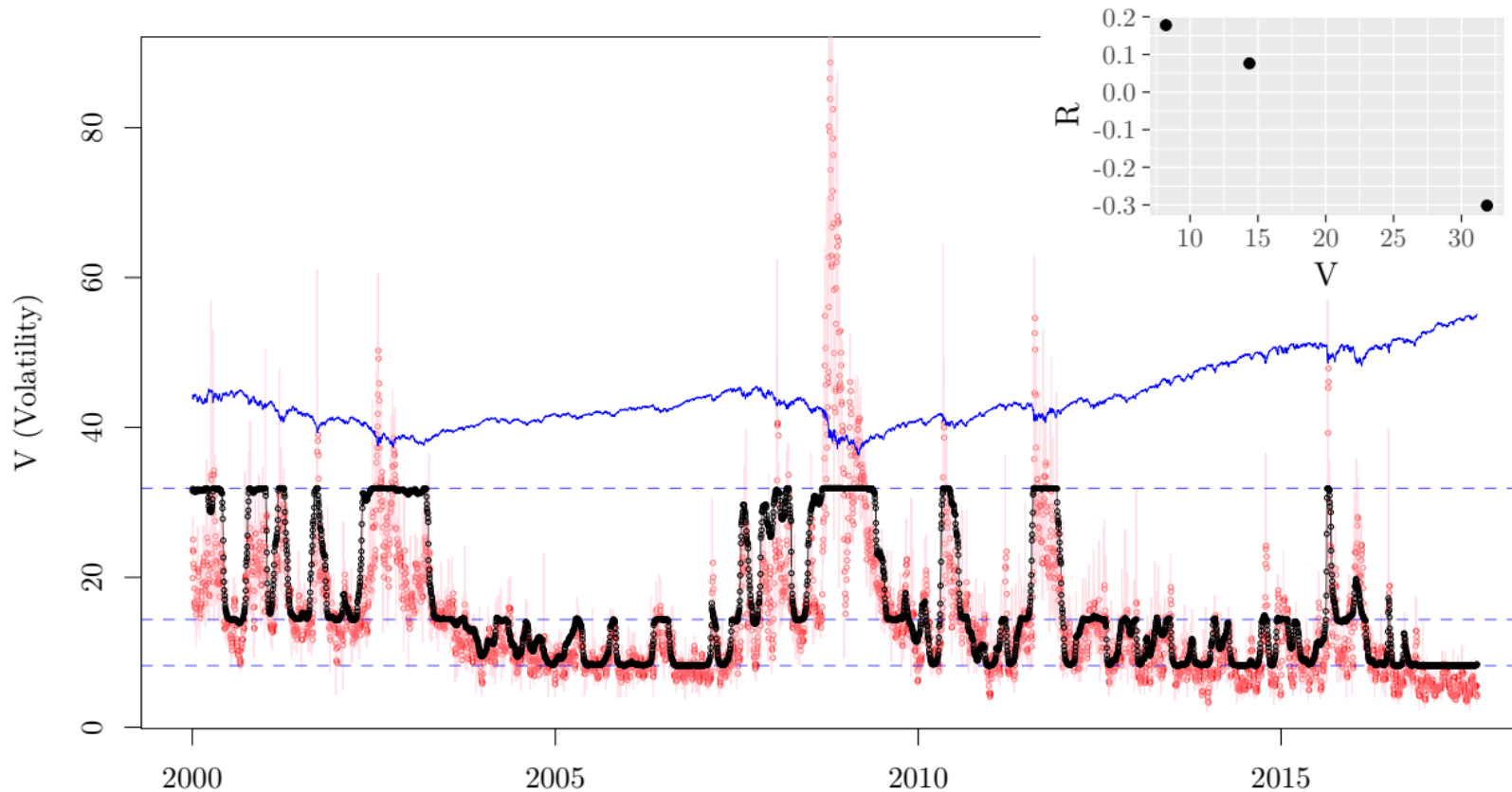
# HMM in PR - Regime Detection

2-State HMM (SPX2.r) vs Realized Vol (SPX2.rv)



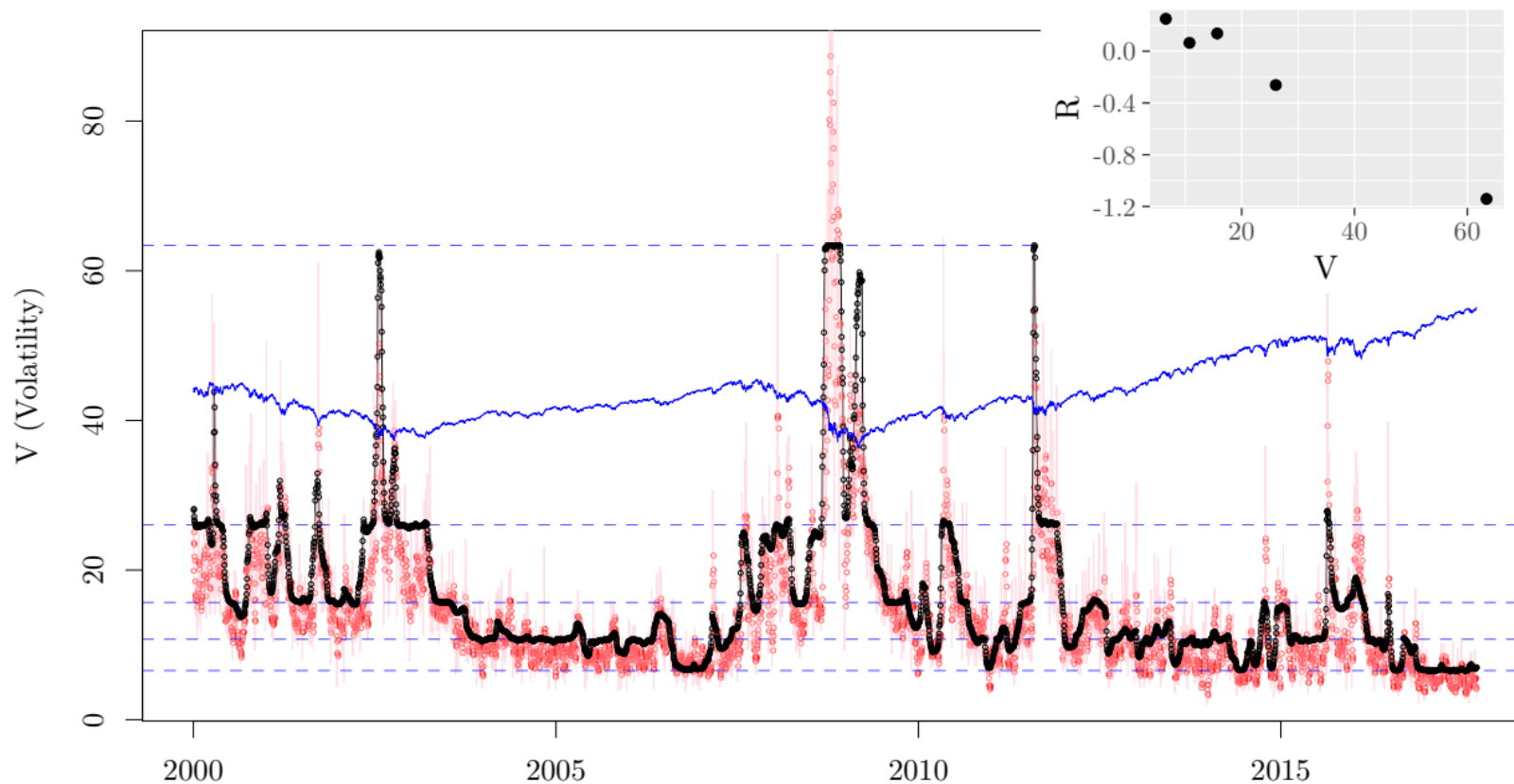
# HMM in PR - Regime Detection

3-State HMM (SPX2.r) vs Realized Vol (SPX2.rv)



# HMM in PR - Regime Detection

5-State HMM (SPX2.r) vs Realized Vol (SPX2.rv)



# Decoding Problem

---

- Find the **optimal hidden state assignment**  $\mathbf{s} = s_1^*, \dots, s_T^*$  for an observed sequence  $\mathbf{y}$  given a trained HMM

- No unique interpretation of the problem

- Identify the **single hidden states**  $s_t$  that maximize the posterior

$$s_t^* = \arg \max_{i=1, \dots, C} P(S_t = i | Y)$$

- Find the most likely **joint hidden state assignment**

$$\mathbf{s}^* = \arg \max_{\mathbf{s}} P(\mathbf{Y}, \mathbf{S} = \mathbf{s})$$

- The last problem is addressed by the **Viterbi algorithm**



# Viterbi Algorithm

An efficient **dynamic programming** algorithm based on a **backward-forward recursion**

An example of a **max-product message passing algorithm**

Recursive backward term

$$\epsilon(s_{t-1}) = \max_{s_t} P(Y_t | S_t = s_t) P(S_t = s_t | S_{t-1} = s_{t-1}) \epsilon(s_t),$$

Root optimal state

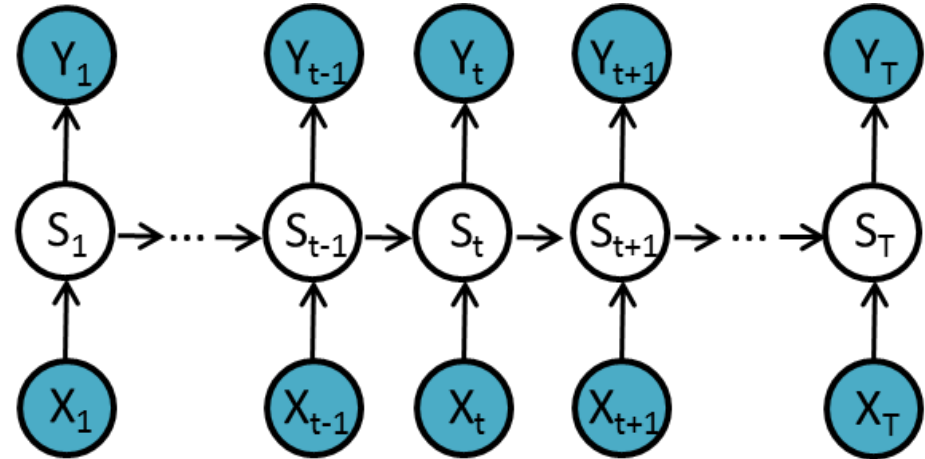
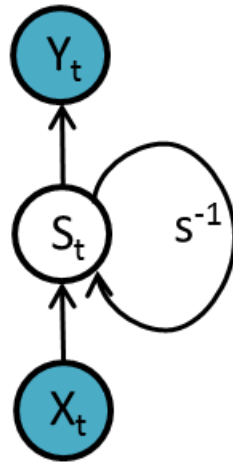
$$s_1^* = \arg \max_s P(Y_t | S_1 = s) P(S_1 = s) \epsilon(s).$$

Recursive forward optimal state

$$s_t^* = \arg \max_s P(Y_t | S_t = s) P(S_t = s | S_{t-1} = s_{t-1}^*) \epsilon(s).$$



# Input-Output Hidden Markov Models

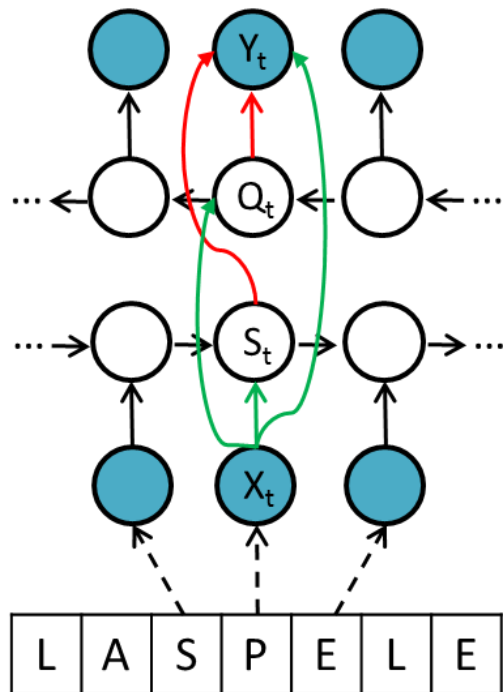


- Translate an input sequence into an output sequence (**transduction**)
- State transition and emissions depend on input observations (**input-driven**)
- Recursive model highlights analogy with **recurrent neural networks**



# Bidirectional Input-driven Models

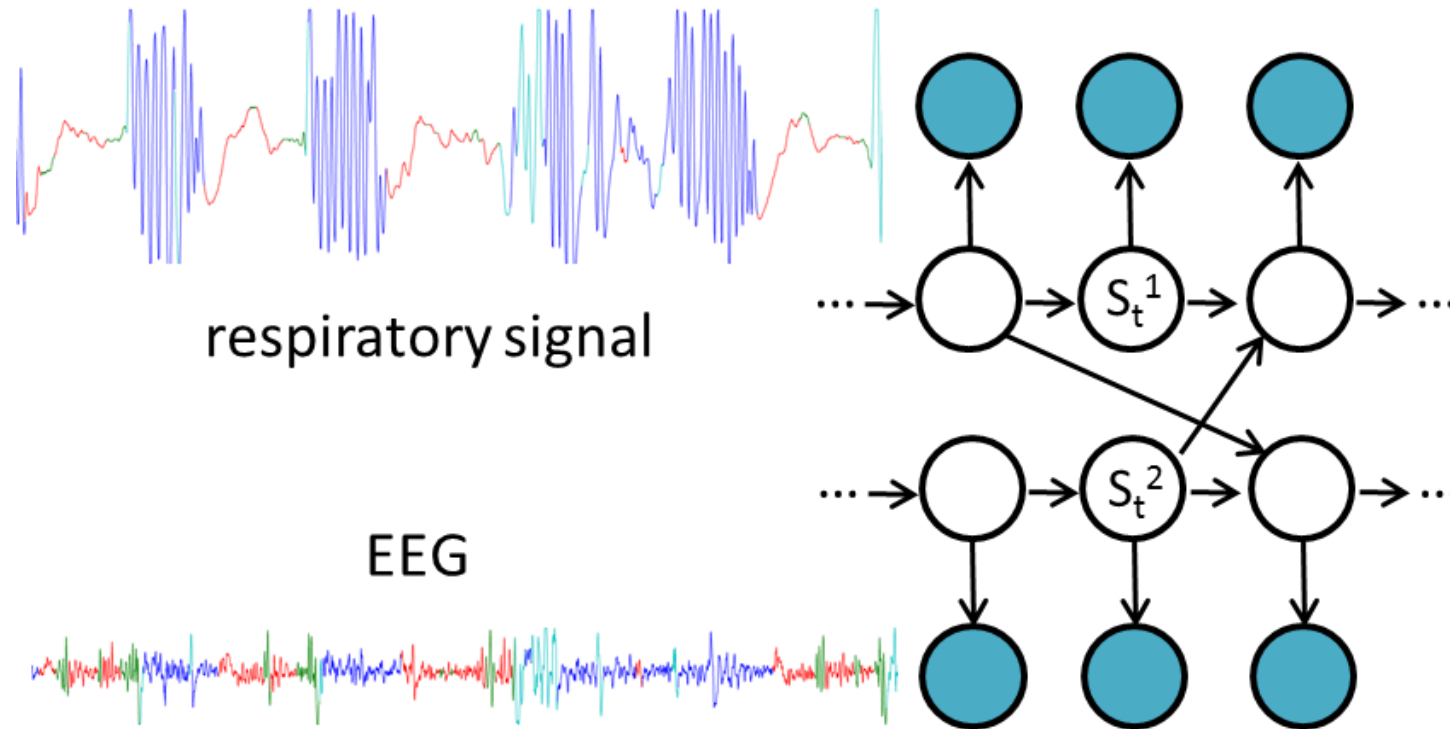
Remove **causality assumption** that current observation does not depend on the future and **homogeneity assumption** that an state transition is not dependent on the position in the sequence



- Structure and function of a region of DNA and protein sequences may depend on upstream and downstream information
- Hidden state transition distribution changes with the amino-acid sequence being fed in input

# Coupled HMM

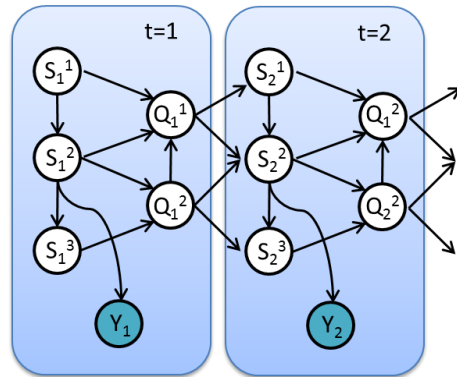
Describing **interacting processes** whose observations follow different dynamics while the underlying generative processes are interlaced



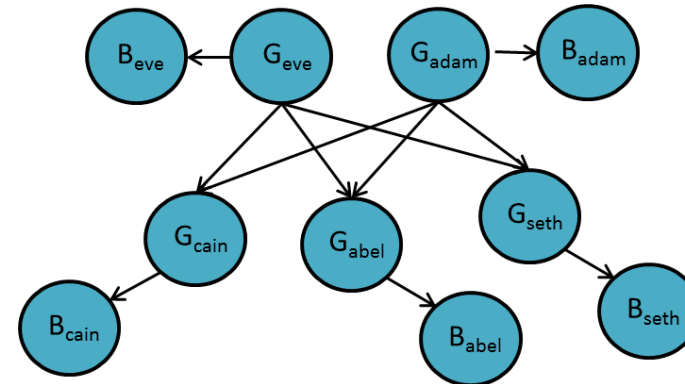


# Dynamic Bayesian Networks

HMMs are a specific case of a class of directed models that represent **dynamic processes** and data with changing **connectivity template**



Hierarchical HMM



Structure changing information

## Dynamic Bayesian Networks (DBN)

Graphical models whose structure changes to represent evolution across time and/or between different samples



# HMM in Matlab

An official implementation by Mathworks available as a set of **inference and learning functions**

Estimate distributions (based on initial guess)

```
%Initial distribution guess
tg = rand (N,N); %Random init
tg = tg ./ repmat (sum( tg , 2 ) , [ 1 N ] ); %Normalize
... %Similarly for eg
[ test , eest ] = hmmtrain ( seq , tg , eg );
```

Estimate posterior states

```
pstates = hmmdecode( seq , test , eest )
```

Estimate Viterbi states

```
vstates = hmmviterbi ( seq , test , eest )
```

Sample a sequence from the model

```
[ seq , states ] = hmmgenerate ( len , test , eest )
```



# HMM in Python

---

- **hmmlearn** - The official scikit-like implementation of HMM
  - 3 classes depending on emission type: `MultinomialHMM`, `GaussianHMM`, and `GMMHMM`

---

```
from hmmlearn . hmm import GaussianHMM
...
# Create an HMM and fit it to data X
model = GaussianHMM ( n_components = 4 , covariance_type = "diag" , n_iter = 1000 ) . fit ( X )
# Decode the optimal sequence of internal hidden state ( Viterbi )
hidden_states = model . predict ( X )
# Generate new samples ( visible, hidden )
X1 , Z1 = model . sample ( 500 )
```

---

- **hmms 0.1** - A scalable implementation for both **discrete and continuous-time** HMMs



# Take Home Messages

---

- Hidden Markov Models
  - Hidden states used to realize an **unobserved generative process for sequential data**
  - A **mixture model** where selection of the next component is regulated by the transition distribution
  - Hidden states **summarize (cluster) information** on subsequences in the data
- Inference in HMMS
  - **Forward-backward** - Hidden state posterior estimation
  - **Expectation-Maximization** - HMM parameter learning
  - **Viterbi** - Most likely hidden state sequence
- Dynamic Bayesian Networks
  - A graphical **model whose structure changes** to reflect information with variable size and connectivity patterns
  - Suitable for modeling **structured data** (sequences, tree, ...)



# Next Lecture/Lectures

---

## Markov Random Fields

- Learning in undirected graphical models
- Introduction to message passing algorithms
- Conditional random fields
- Pattern recognition applications

