

# Hidden Markov Models

Chris Williams

School of Informatics, University of Edinburgh

November 2010

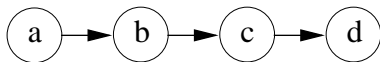
# Overview

- Definitions
- Inference Problems
- Recursion formulae
- Viterbi alignment
- Training a HMM
- Reading: Bishop §13.1, 13.2 (but not 13.2.3, 13.2.4, 13.2.5), Rabiner paper

Dynamical models used in many areas for modelling sequences, including

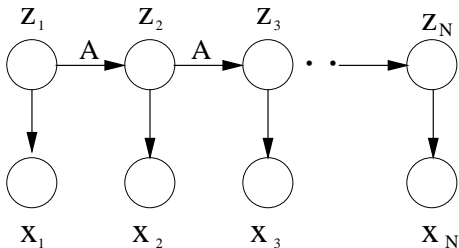
- Speech recognition
- Molecular biology sequences
- Linguistic sequences (e.g. part-of-speech tagging)
- Multi-electrode spike-train analysis
- Tracking objects through time

## Markov Chain



$$p(a, b, c, d) = p(a)p(b|a)p(c|b)p(d|c)$$

## Hidden Markov Model



A HMM is defined by

- $K$  the number of states
- $A$  the state transition matrix
- $p(\mathbf{x}_n|\mathbf{z}_n)$  the output probability distribution (independent of  $n$ )
- $\pi$  the initial transition probabilities
- $\mathbf{z}_n$  is a multinomial variable with components  $z_{ni}$ , if  $\mathbf{z}_n$  is in state  $i$  then  $z_{ni} = 1$  and  $z_{nj} = 0$  for  $j \neq i$

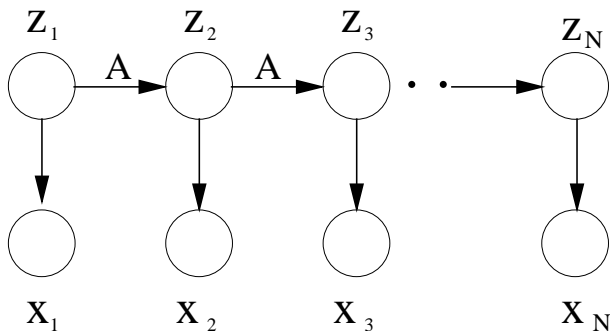
- Let  $\pi_{\mathbf{z}_1} = \prod_{i=1}^K [\pi_i]^{z_{1i}}$  and

$$a_{\mathbf{z}_n \mathbf{z}_{n+1}} = \prod_{i,j} [a_{ij}]^{z_{ni} z_{n+1,j}}$$

- Let  $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_N$  and  $\mathbf{Z} = \mathbf{z}_1, \dots, \mathbf{z}_N$
- For any state sequence  $\mathbf{Z}$

$$p(\mathbf{X}, \mathbf{Z}) = \pi_{\mathbf{z}_1} p(\mathbf{x}_1 | \mathbf{z}_1) a_{\mathbf{z}_1 \mathbf{z}_2} p(\mathbf{x}_2 | \mathbf{z}_2) \dots a_{\mathbf{z}_{N-1} \mathbf{z}_N} p(\mathbf{x}_N | \mathbf{z}_N)$$

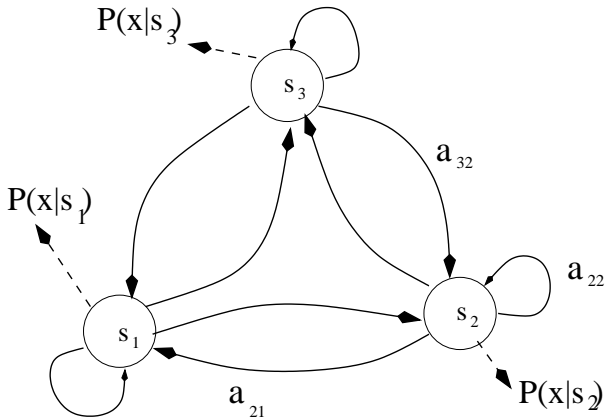
## Independence relationships



- Conditioning on  $\mathbf{z}_n$  renders  $\mathbf{z}_{n-1}$  and  $\mathbf{z}_{n+1}$  independent, i.e.  $I(\mathbf{z}_{n-1}, \mathbf{z}_{n+1} | \mathbf{z}_n)$
- $I(\mathbf{z}_s, \mathbf{z}_u | \mathbf{z}_n)$  for all  $s < n, u > n$

- $I(\mathbf{x}_s, \mathbf{x}_u | \mathbf{z}_n)$  for all  $s \leq n, u > n$
- the future is independent of the past given the present
- Note that conditioning on  $\mathbf{x}_n$  does not yield any conditional independences
- HMM as a dynamical mixture model; choice of state is not independent at each time frame, but depends on the past

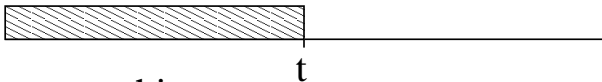
- HMM as a finite state automaton



# Inference Problems

- $p(\mathbf{z}_1 \dots \mathbf{z}_N | \mathbf{X})$  inferring hidden state given  $\mathbf{X}$
- $p(\mathbf{z}_n | \mathbf{X})$  marginal of above
- $p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_n)$  filtering
- $p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_s)$   $n > s$ , prediction
- $p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_u)$   $n < u$ , smoothing
- $p(\mathbf{x}_1, \dots, \mathbf{x}_N)$  likelihood calculation
- Find sequence  $\mathbf{z}_1^* \dots \mathbf{z}_N^*$  that maximizes  $p(\mathbf{Z} | \mathbf{X})$  [Viterbi alignment]

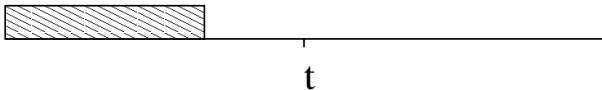
filtering



smoothing

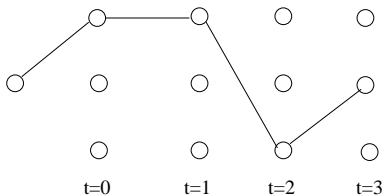


prediction



denotes the extent of data available

$$p(\mathbf{x}_1, \dots, \mathbf{x}_N) = \sum_{\mathbf{z}_1} \sum_{\mathbf{z}_2} \dots \sum_{\mathbf{z}_N} \pi_{\mathbf{z}_1} \prod_{n=1}^{N-1} a_{\mathbf{z}_n \mathbf{z}_{n+1}} \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n)$$



- Naive approach  $O(K^N)$
- Efficient  $O(K^2N)$  algorithm is available by pushing sums through products

## Computing $p(\mathbf{z}_n|\mathbf{X})$

$$\begin{aligned} p(\mathbf{z}_n|\mathbf{X}) &= \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1} \dots \mathbf{x}_N|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1} \dots \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} \\ &\equiv \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})} \end{aligned}$$

The alphas and betas can be calculated recursively.

$$\sum_{\mathbf{z}_n} p(\mathbf{z}_n|\mathbf{X}) = 1 = \frac{\sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

implies

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)$$

- Define  $p(\mathbf{z}_n|\mathbf{X}) = \gamma(\mathbf{z}_n)$

# Recursion formulae

- Alpha

$$\alpha(\mathbf{z}_{n+1}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n) a_{\mathbf{z}_n \mathbf{z}_{n+1}} p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1})$$

Initialization

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{x}_1 | \mathbf{z}_1) p(\mathbf{z}_1) = p(\mathbf{x}_1 | \mathbf{z}_1) \pi_{\mathbf{z}_1}$$

- Beta

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) a_{\mathbf{z}_n \mathbf{z}_{n+1}} p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1})$$

Initialization:  $\beta(\mathbf{z}_N)$  is the vector of ones as

$$\sum_i \alpha(z_{Ni}) \beta(z_{Ni}) = \sum_i \alpha(z_{Ni}) = \sum_i p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_N = i) = p(\mathbf{X})$$

Each step is  $O(K^2)$

# Viterbi alignment

Find the best state sequence  $\mathbf{z}_1^* \mathbf{z}_2^* \dots \mathbf{z}_N^*$  that maximizes  $p(\mathbf{Z}|\mathbf{X})$

Define

$$\delta_n(i) = \max_{\mathbf{z}_1, \dots, \mathbf{z}_{n-1}} p(\mathbf{z}_1, \dots, \mathbf{z}_n = i, \mathbf{x}_1 \dots \mathbf{x}_n)$$

i.e.  $\delta_n(i)$  is the best score along a single path up to time  $n$  which account for the first  $n$  observations and ends in state  $S_i$ .

- There is a recursive formula for the  $\delta$ s similar to the  $\alpha$ -recursion, except that a max rather than sum operation is used
- For further details see, for example, L. Rabiner, Proc. IEEE 77(2) 1989 pp 257-285

## Calculating $p(\mathbf{z}_n, \mathbf{z}_{n+1} | \mathbf{X})$

$$\begin{aligned}\xi(\mathbf{z}_n, \mathbf{z}_{n+1}) &\equiv p(\mathbf{z}_n, \mathbf{z}_{n+1} | \mathbf{X}) \\ &= \frac{p(\mathbf{z}_n, \mathbf{z}_{n+1}, \mathbf{X})}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{X} | \mathbf{z}_n, \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= p(\mathbf{x}_0, \dots, \mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) \times \\ &\quad \frac{p(\mathbf{x}_{n+2} \dots \mathbf{x}_N | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_n) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) \beta(\mathbf{z}_{n+1}) a_{\mathbf{z}_n \mathbf{z}_{n+1}}}{p(\mathbf{X})}\end{aligned}$$

## Training a HMM

- Use the EM algorithm to estimate  $\pi$ ,  $A$  and  $\eta$ , the parameters of  $p(\mathbf{x}_n|\mathbf{z}_n)$ . Let  $\theta = (\pi, A, \eta)$
- If we knew the “true” state sequence, parameter estimation would be easy. The trick is to use the probability distribution over state paths to weight these estimates

$$\hat{\pi}_i \leftarrow \gamma(z_{1i})$$

$$\hat{a}_{ij} \leftarrow \frac{\sum_{n=1}^{N-1} \xi(z_{ni}, z_{n+1,j})}{\sum_{n=1}^{N-1} \gamma(z_{ni})}$$

If the output is a multinomial distribution with  $p(x_{nj} = 1 | z_{ni} = 1) = \eta_{ij}$  and  $\sum_k x_{nk} = 1$

$$\hat{\eta}_{ij} \leftarrow \frac{\sum_{n=1}^N \gamma(z_{ni}) x_{nj}}{\sum_{t=1}^N \gamma(z_{ni})}$$

For HMMs these are known as the Baum-Welch equations

# Example: Harmonizing Chorales in the Style of J S Bach

- Moray Allan and Chris Williams (NIPS 2004)  
<http://www.tardis.ed.ac.uk/~moray/harmony/>,  
online demo at  
<http://www.anc.inf.ed.ac.uk/demos/hmmbach/>
- Visible states are the melody (quarter notes)
- Hidden states are the harmony (which chord)
- Trained using labelled melody/harmony data (no need for EM)
- Task: find Viterbi alignment for harmony given melody (or sample from  $p(\text{harmony}|\text{melody})$ .)
- Actually uses HMMs for three subtasks: harmonic skeleton, chord skeleton, ornamentation