Chris Bishop's PRML
Chapter 13: Sequential data

Moray Allan & Tingting Jiang

26 June 2008

Chapter outline

- (Hidden) Markov models
- Linear Dynamical Systems

The chapter section by section

13.1

- Markov models

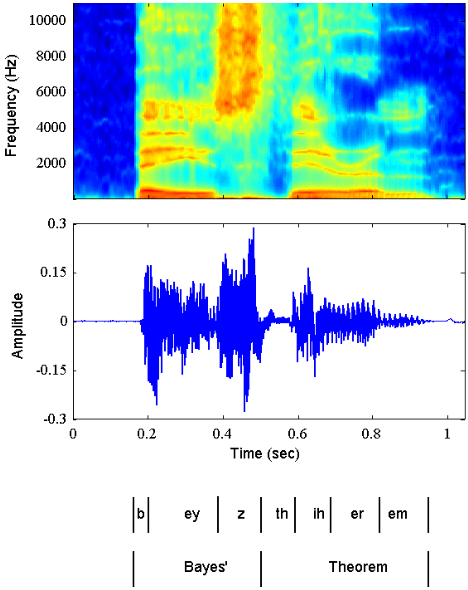
13.2

- Hidden Markov Models
- Parameter estimation

13.3

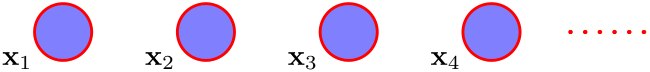
- Linear Dynamical Systems
- Inference in LDS
- Learning in LDS
- Particle filters

Sequential data



Models for sequential data

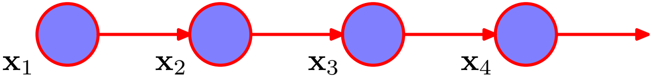
Can model as independent:



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}). \tag{13.1}$$

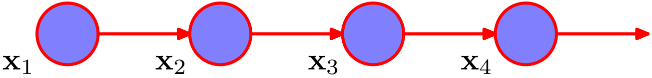
Models for sequential data

Better to link observations, e.g. first-order Markov model conditions on previous observation:

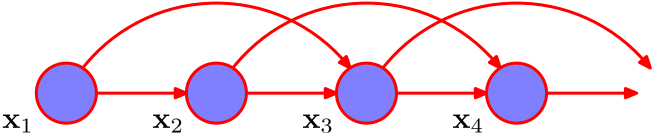


$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n | \mathbf{x}_{n-1}). \tag{13.2}$$

Models for sequential data



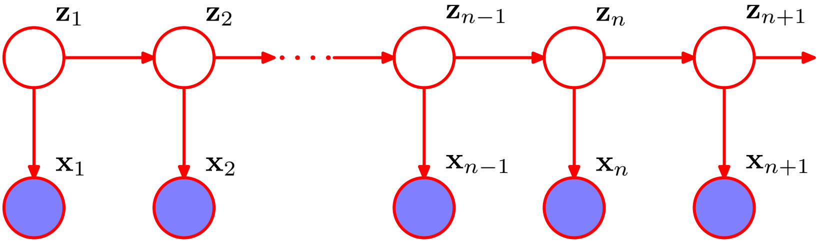
Second-order Markov model conditions on the two previous observations:



$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2|\mathbf{x}_1)\prod_{n=3}^{N} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{x}_{n-2}). \tag{13.4}$$

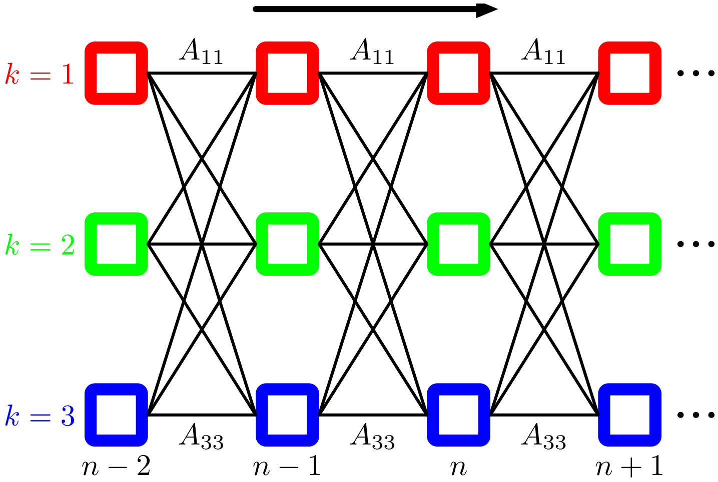
Models for sequential data

Hidden Markov model adds unobserved state variables:

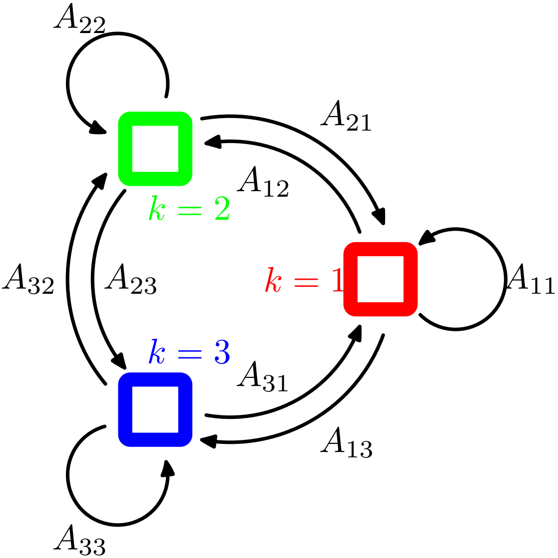


$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{z}_n). \qquad (13.6)$$

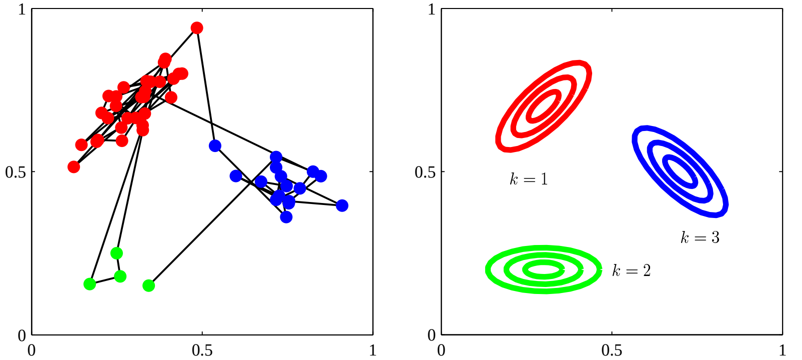
HMM latent state example



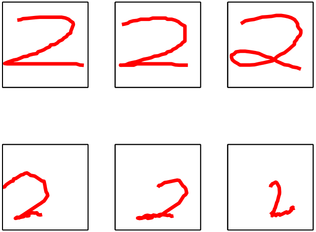
HMM latent state example

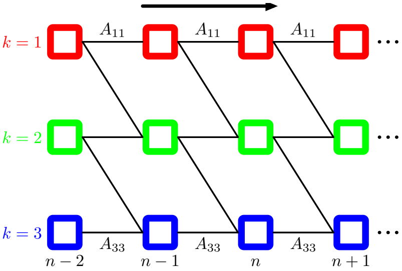


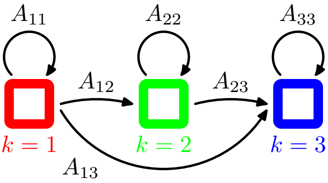
Outputs need not be discrete states



Transitions may be constrained







Finding the most probable sequence of latent states

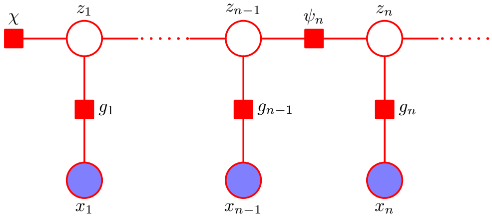
- Could directly iterate over all possible paths, but this is computationally expensive.

Can use dynamic programming (Viterbi algorithm, 13.2.5):

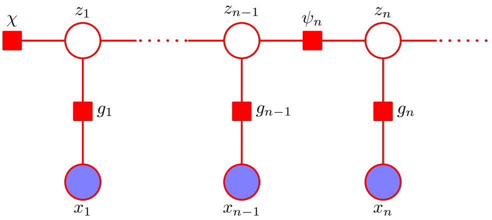
- First work forwards through lattice, summing products of transition and emission probabilities along paths.
- For each latent state, we only need to keep track of the highest probability path that reaches the state.
- With K latent states, at each time step we consider K^2 paths, but only retain K corresponding to the best path for each state at the next time step.
- When we reach the end of the sequence, we can choose the most probable latent state, and trace back through the sequence to retrieve the whole sequence of states.

Learning the model parameters (13.2.1–13.2.3)

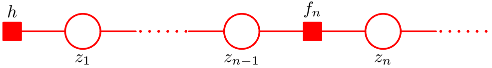
- Variational methods for a fully Bayesian approach (MacKay, 1997).
- Use Baum-Welch algorithm (Baum, 1972) / forwards-backwards algorithm (Rabiner, 1989).
- Use more general sum-product algorithm (8.4.4):



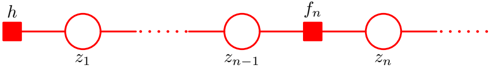
Learning the model parameters (13.2.1–13.2.3)



As always condition on $\boldsymbol{x}_1, \ldots \boldsymbol{x}_N$ for this inference problem, we can absorb the emission probabilities into the transition probability factors:



Learning the model parameters (13.2.1–13.2.3)



- First pass messages along the chain to the root $\boldsymbol{x}_N$:

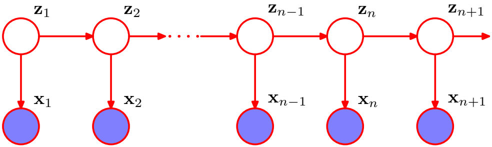
$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}). \qquad (13.36)$$

- Then propagate messages back from the root node to the leaf node:

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n). \qquad (13.38)$$

- By sum-product algorithm, marginal at node $\boldsymbol{z}_n$ is the product of the incoming messages.

Learning the model parameters (13.2.1–13.2.3)



In M-step want to maximise log probability

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) &= \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \ln A_{jk} \\ &\quad + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n | \boldsymbol{\phi}_k) \end{aligned} \tag{13.17}$$

where

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{13.13}$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}). \tag{13.14}$$

Learning the model parameters (13.2.1–13.2.3)

So in M-step we make these assignments:

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{13.18}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})} \tag{13.19}$$

where

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \tag{13.13}$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}, \boldsymbol{\theta}^{\text{old}}). \tag{13.14}$$

Scaling (13.2.4)

- Probabilities along state paths rapidly become very small.
- Simply taking logarithms isn't enough, as we need to compare between sums of small probabilities.
- Store the probabilities normalised over the states for a given timestep, keep track of scaling factors.