STA 414/2104: Machine Learning

Week 9

12 March 2018:
Graphical models,
Sequential models

Delivered by Mark Ebden
With thanks to Russ Salakhutdinov

Examples of other perspectives:

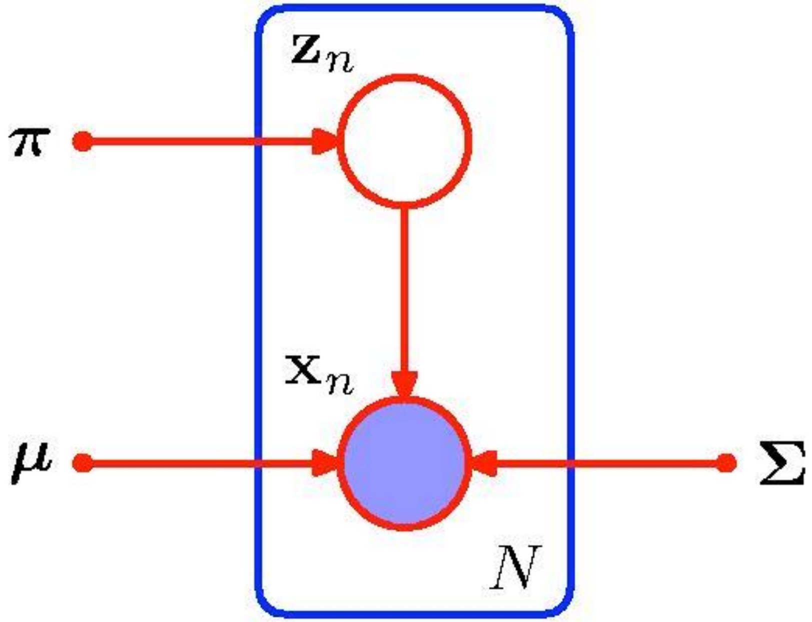
- Bishop 2006: parts of chap. 8 & 13
- Murphy 2012: parts of chap. 10 & 17
- Russell & Norvig, 2009:
  parts of chap. 14 & 15

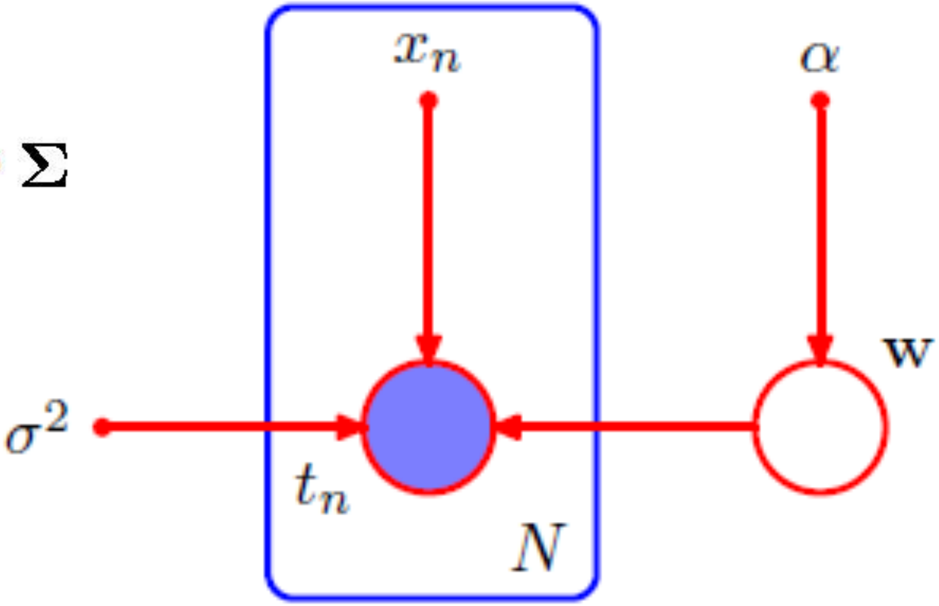
  (Artificial Intelligence:
  A Modern Approach)

Outline

- Review / background of Graphical Models
- Example of a famous Bayesian network
- Inference in Bayesian networks
- Conditional independence in BNs
- Sequential models

Graphical models





ML Example Uses of Conditional Independence

- Naive Bayes assumption: all dimensions are independent given the label *z*

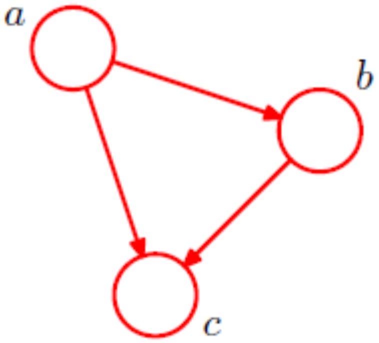
$$p(\mathbf{x}|z=k) = p(x_1, \ldots, x_D|z=k)$$
$$= \prod_{d=1}^{D} p(x_d|z=k)$$

- Markov assumption: the future is independent of the past given the present

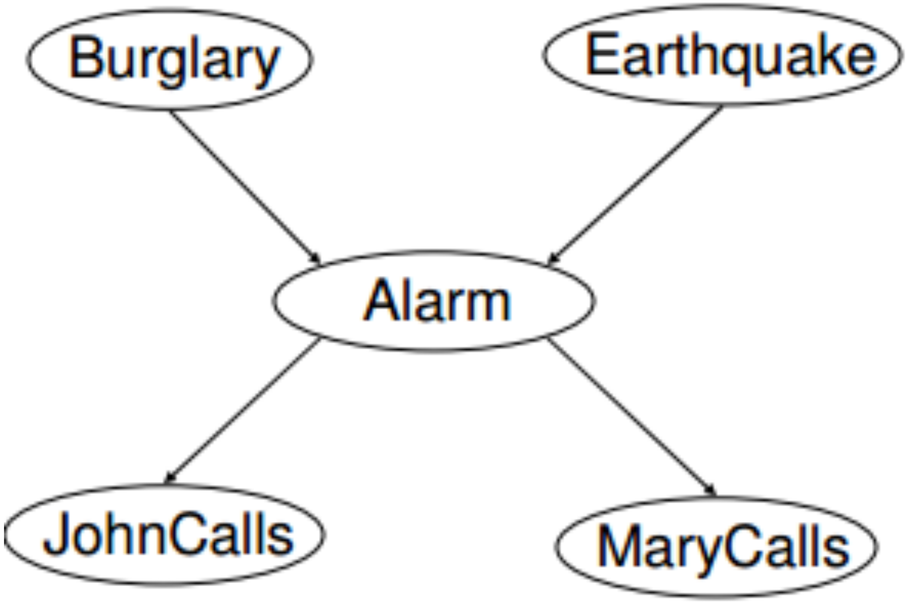
$$p(x_d|x_1, \ldots, x_{d-1}) = p(x_d|x_{d-1})$$

Bayesian Networks

- Simple and visual: you can put conditional probability tables next to nodes
- Can be a compact representation of the full joint distribution, for locally structured (sparse) cases



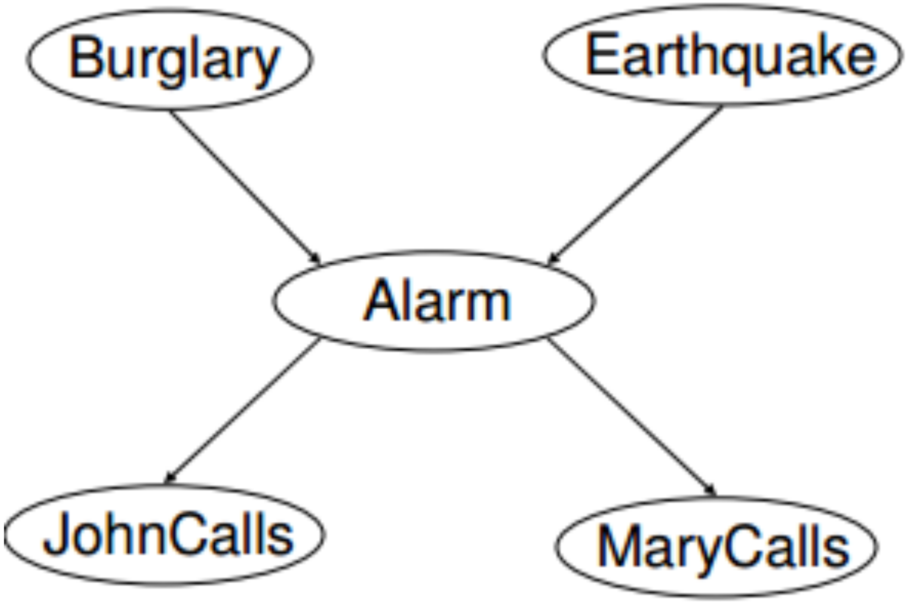
Bayesian Network Example

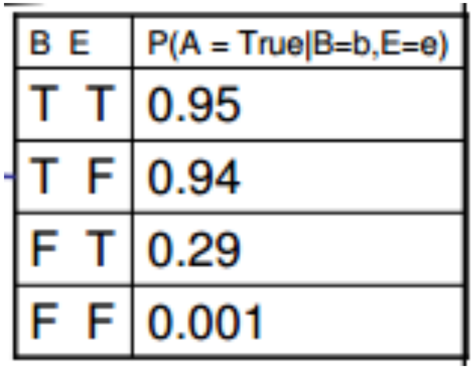


Bayesian Network Example

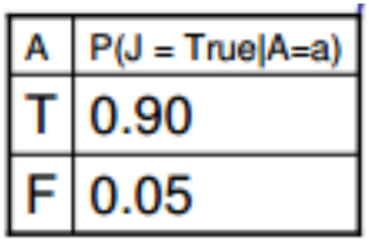
P(B=True) = 0.001

P(E=True) = 0.002

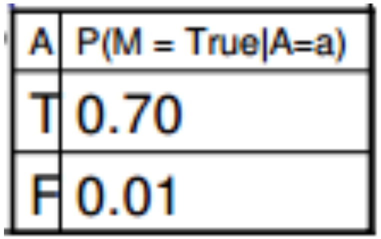




B	E	P(A = True|B=b,E=e)
T	T	0.95
T	F	0.94
F	T	0.29
F	F	0.001



A	P(J = True|A=a)
T	0.90
F	0.05



A	P(M = True|A=a)
T	0.70
F	0.01

Calculating on a Bayesian Network

We can say $$p(x_1, \ldots x_D) = \prod_{d=1}^{D} p(x_d | X_{\mathcal{A}_d})$$

ancestor

P(B=1) P(E=1) P(A=1|B=1,E=1)P(J=1|A=1)P(M=1|A=1) = 0.001*0.002*0.95*0.9*0.7

So, $P(B=1, E=1, A=1, J=1, M=1) = ?$

Answer: ~1.2 x 10^{-6}

$P(B=1 \mid J=1, M=1) = ?$

Answer: Trickier

Outline

- Review / background of Graphical Models
- Example of a famous Bayesian network
- **Inference in Bayesian networks**
- Conditional independence in BNs
- Sequential models

Exact Inference in BNs

$$P(B = 1 \mid J = 1, M = 1)$$

$$= \frac{P(B = 1, J = 1, M = 1)}{P(B = 1, J = 1, M = 1) + P(B = 0, J = 1, M = 1)}$$

where $P(B = 1, J = 1, M = 1)$

$$= \sum_E \sum_A P(B = 1, E, A, J = 1, M = 1)$$

$$= P(B = 1) \sum_E P(E) \sum_A P(A|B = 1, E)\, P(J = 1|A)\, P(M = 1|A)$$

Leading to: $P(B = 1 \mid J = 1,\ M = 1) \approx 0.284$

Direct Sampling: simulating a graphical model

- Put the nodes in ancestral order (parents coming before children)
- Sample each variable given its parents
- The probability of an event can be estimated as the fraction of all complete events generated that match the partially specified event. e.g. if 6 out of 2000 samples have A=1, $P(A)$≈0.003

Rejection Sampling:

- Helps us to estimate
  $$P(X|E) = P(X,E) / P(E)$$
  for a query variable X and evidence E
- e.g. we can estimate $P(J=1 \mid M=1)$
- e.g. sample 1000 times and reject all samples in which $M = 0$. From the remaining N samples (M = 1), estimate: $P(J=1 \mid M=1) \approx N_{J=1} / N$

Who is conditionally independent of whom?

ball goes through
chain: yes unless y observed
fork : yes unless y observed
V : no unless y or any of y’s descendents are oberved

- *Bayes ball algorithm*
- Quickly determines whether $X \perp\!\!\!\perp Z \mid Y$

P(X | Z,Y) = P(X | Y)
or P(X,Y | C) = P(X | Y) P(Z | Y)

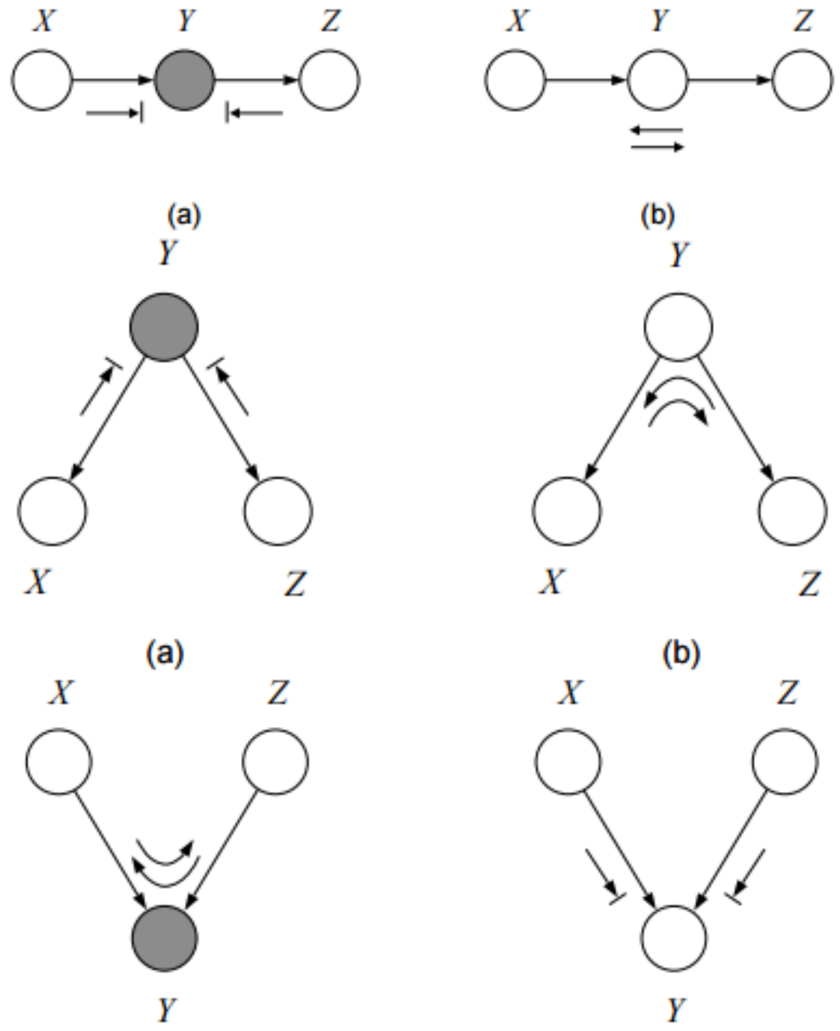
independent relationship (blocked)

dependent relationship (goes through)

chain

fork

V

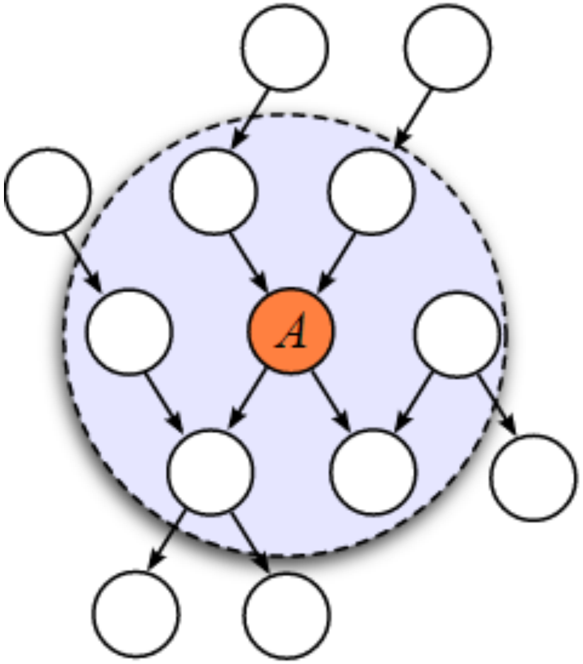


Comparing one node vs another

Conditional independence relations in Bayesian Networks

There are two, equivalent specifications:

- A node is C.I. of its non-descendants given its parents
- A node is C.I. of all other nodes, given its Markov blanket (parents, children, and co-parents)



Comparing one node vs rest of network

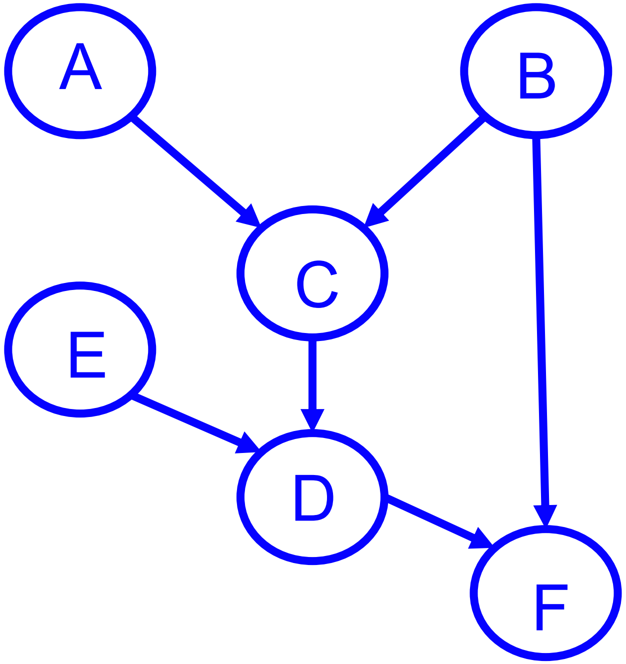
How does this relate?

- For a given node, *A*, the Markov blanket of *A* is the minimal set of nodes which *Bayes-ball*-separates (renders C.I.) node *A* from all other nodes in the network

- True/false:

	bayes-ball	C.I. of non-descendents	Markov blancket
– $M \perp\!\!\!\perp J \mid A$	True	True	True
– $B \perp\!\!\!\perp E \mid A$	False		
– $M \perp\!\!\!\perp E \mid A$	True		

An online true-false quiz



F 1. $A \perp\!\!\!\perp D \mid C$

F 2. $A \perp\!\!\!\perp D \mid C, F$

3. $A \perp\!\!\!\perp D \mid C, F, B$

if C, CF, CFB shaded, TFT

To vote: visit pollev.com/MARKEBDEN209 *or*

- Text MARKEBDEN209 to short code 37607
- Then text FFF, TFT, etc

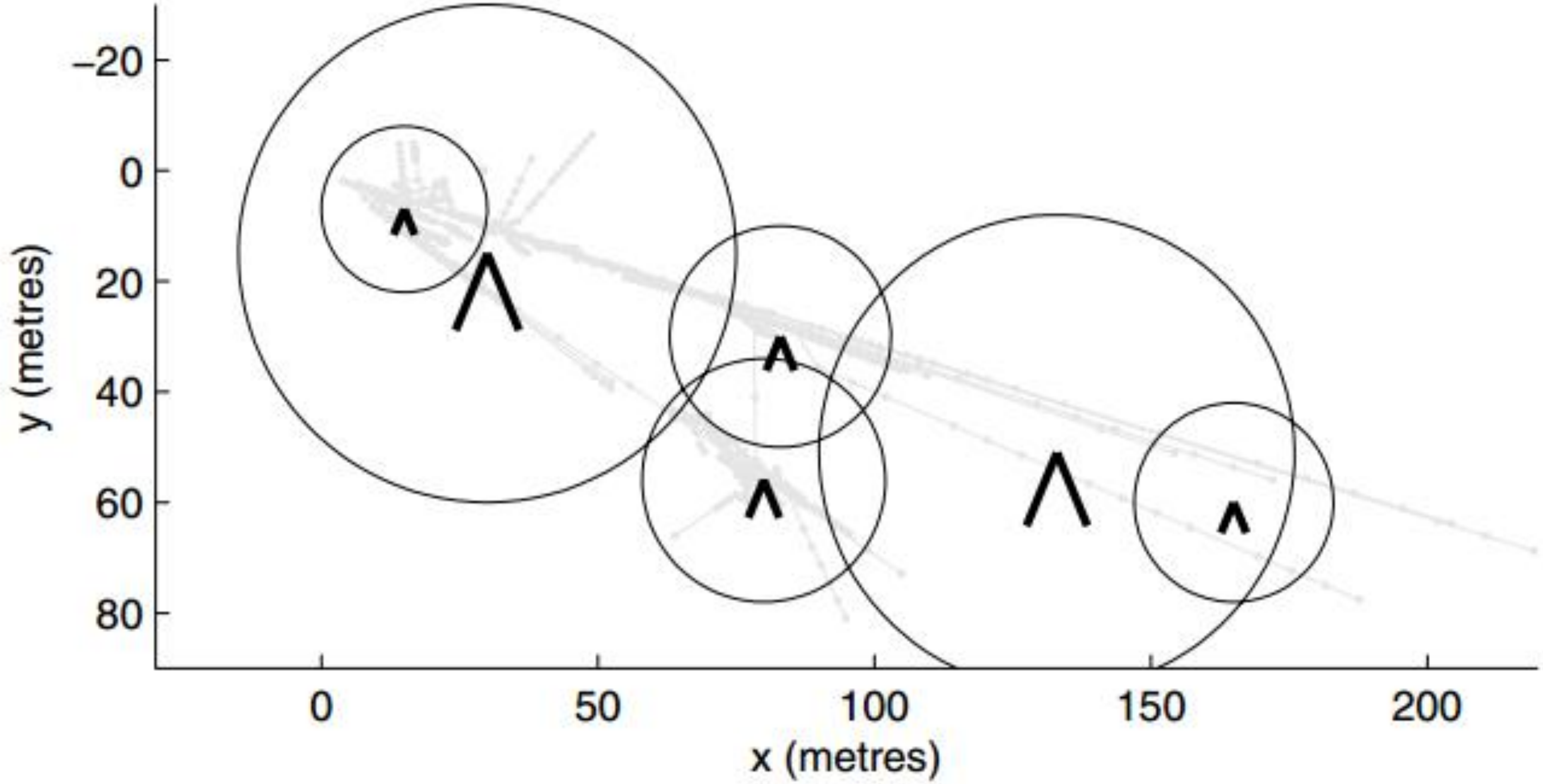
Extra practice question

- Show that #1 would be implied by $A \perp\!\!\!\perp D, E \mid C$
- Hint: no graph required

Bonus

Example application: surveillance

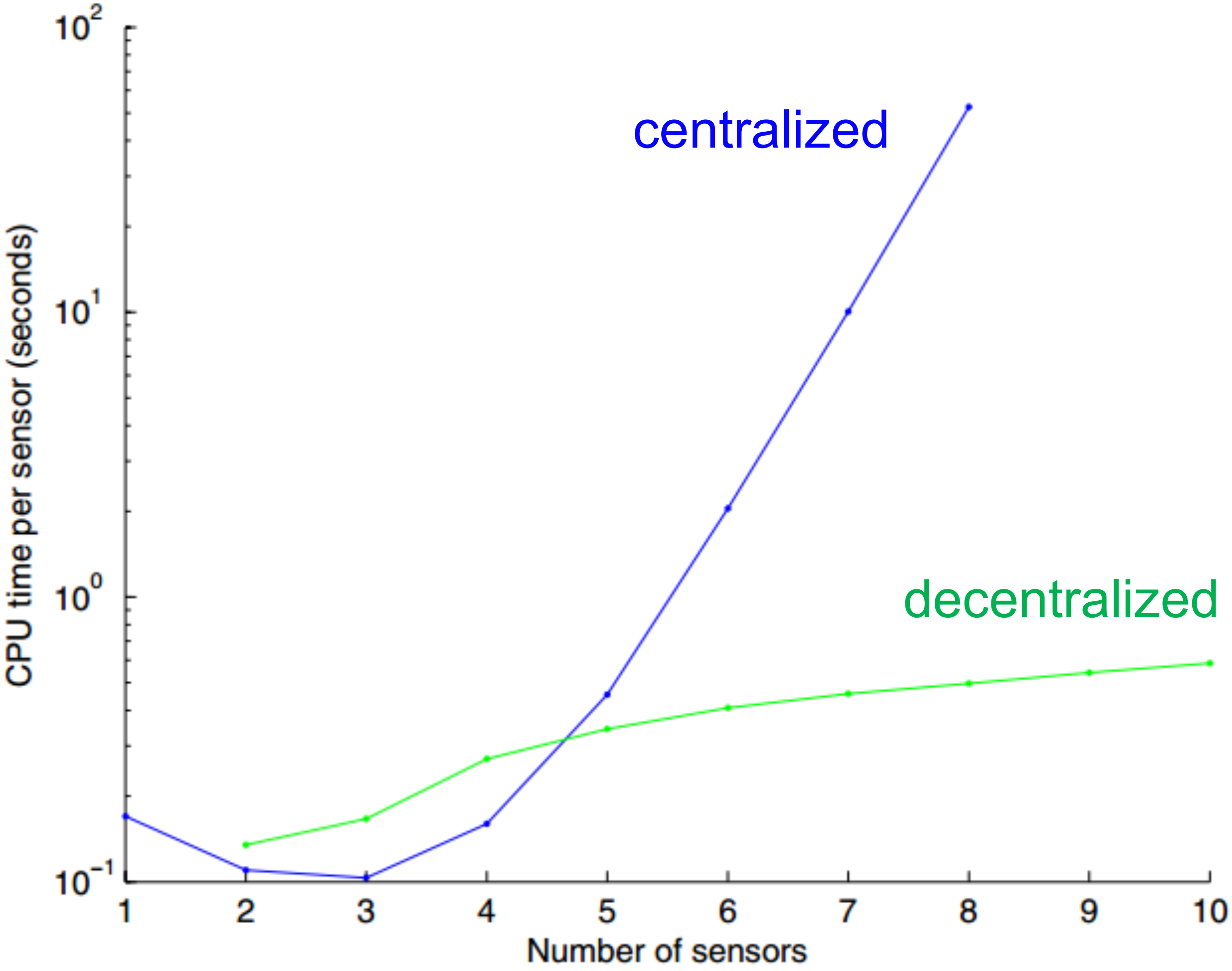
6 sensors



http://mebden.com/reports/AdHocNetworks.pdf

Bonus

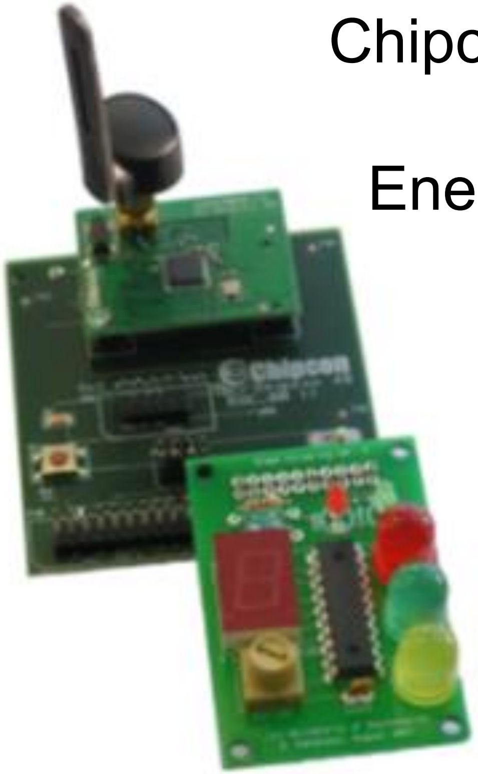
Example application: surveillance



http://mebden.com/reports/AdHocNetworks.pdf

Bonus

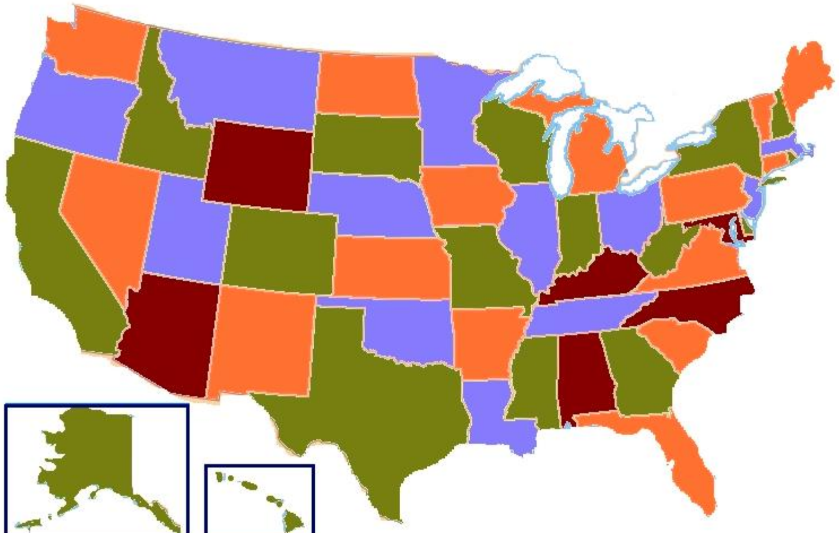
Hardware toys for decentralized coordination



Chipcon CC2431 System-on-Chip (SoC)

Energy function penalizes neighbours who have the same colour

Four-colour theorem:



Outline

- Review / background of Graphical Models
- Example of a famous Bayesian network
- Inference in Bayesian networks
- Conditional independence in BNs
- **Sequential models**

Sequential Data

- So far we've focused on problems that assumed that the data points were independent and identically distributed (i.i.d. assumption)

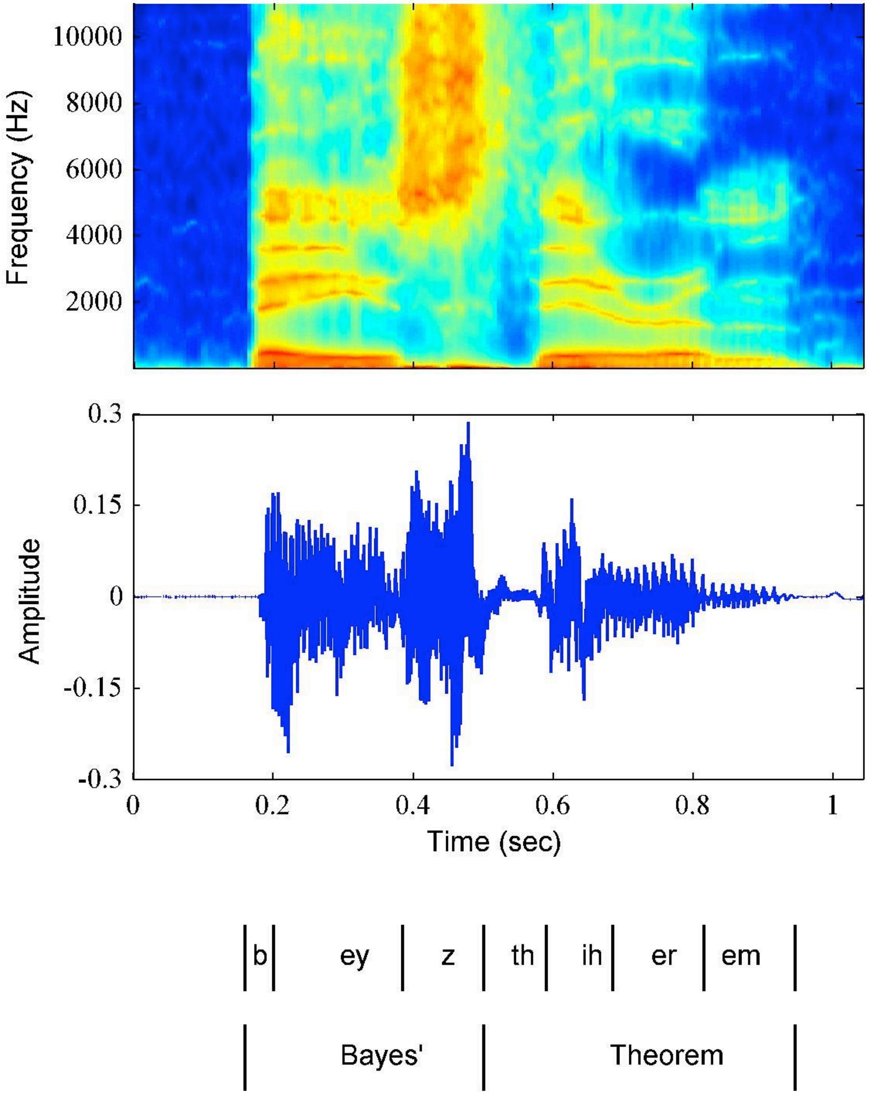
- Expressing the likelihood function as a product over all data points of the probability distribution evaluated at each data point is generally unsuitable when working with sequential data

- For many applications, e.g. financial forecasting, we want to predict the next value in a time series, given past values

- Intuitively, the recent observations are likely to be more informative in predicting the future

- Markov models: future predictions are independent of all but the most recent observations

Example of a Spectrogram

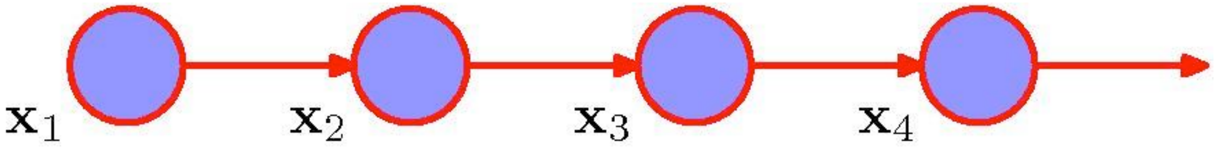


- Example of a spectrogram of a spoken word ‘Bayes theorem’

- Successive observations are highly correlated

Markov Models

- The simplest model is the first-order Markov chain:



- The joint distribution for a sequence of *N* observations under this model is:

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

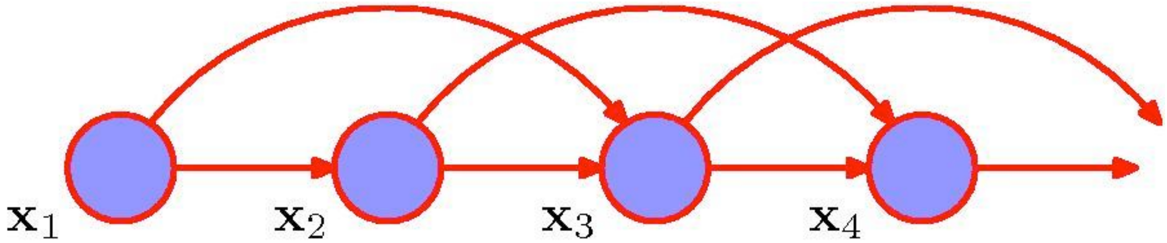
- The conditionals are given by:

$$p(\mathbf{x}_n | \mathbf{x}_1, ..., \mathbf{x}_{n-1}) = p(\mathbf{x}_n | \mathbf{x}_{n-1}).$$

- For many applications, these conditional distributions that define the model will be constrained to be equal
- This corresponds to the assumption of a stationary time series
- The model is known as a homogeneous Markov chain

Second-Order Markov Models

- We can also consider a second-order Markov chain:



- The joint distribution for a sequence of *N* observations under this model is:

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

- We can similarly consider extensions to an M^{th}-order Markov chain
- Increased flexibility → Exponential growth in the number of parameters
- Markov models need big orders to remember past “events”

Learning Markov Models

- The ML parameter estimates for a simple Markov model are easy. Consider a *k*th-order model:

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

- Each window of *k* + 1 outputs is a training case for the model.

$p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{x}_{n-2}, .., \mathbf{x}_{n-k}).$

- Example: for discrete outputs (symbols) and a second-order Markov model we can use the multinomial model:

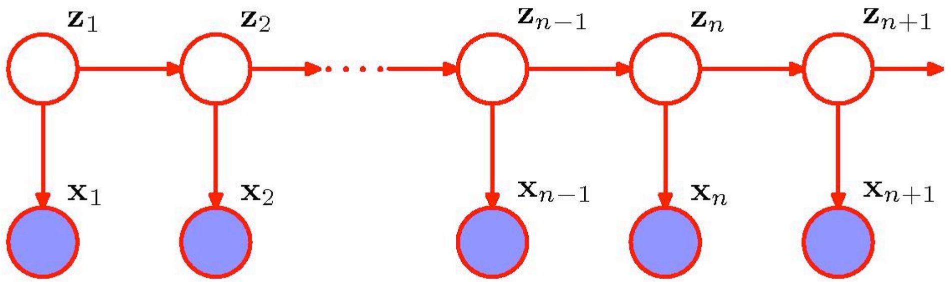
$$p(\mathbf{x}_n = m | \mathbf{x}_{n-1} = a, \mathbf{x}_{n-2} = b) = \alpha_{mab}.$$

- The maximum likelihood values for *α* are:

$$\alpha^*_{mab} = \frac{\mathrm{num}[n,\ s.t.\ \mathbf{x}_n = m, \mathbf{x}_{n-1} = a, \mathbf{x}_{n-2} = b]}{\mathrm{num}[n,\ s.t.\ \mathbf{x}_{n-1} = a, \mathbf{x}_{n-2} = b]}.$$

State Space Models

- Consider the model that is not limited by the Markov assumption to any order
- Solution: Introduce additional latent variables!



- Graphical structure known as the State Space Model.

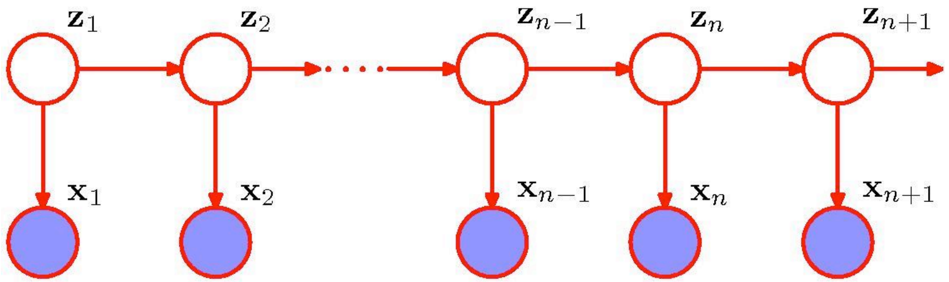
- For each observation $\mathbf{x}_n$, we have a latent variable $\mathbf{z}_n$. Assume that the latent variables form a Markov chain
- If the latent variables are discrete: Hidden Markov Models (HMMs). Observed variables can be discrete or continuous

- If the latent variables and observed variables are Gaussian: Linear Dynamical System

State Space Models

- The joint distribution is given by:

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



- Graphical structure known as the State Space Model.

- There is always a path connecting two observed variables $\mathbf{x}_n$, $\mathbf{x}_m$ via latent variables

- The predictive distribution:

$$p(\mathbf{x}_{n+1}|\mathbf{x}_1, ..., \mathbf{x}_N)$$

does not exhibit any conditional independence properties! And so prediction depends on all previous observations

- Even though the hidden state sequence is first-order Markov, the output process is not Markov of any order!

Hidden Markov Model

- A first-order Markov chain generates a hidden state sequence, through transition probabilities:

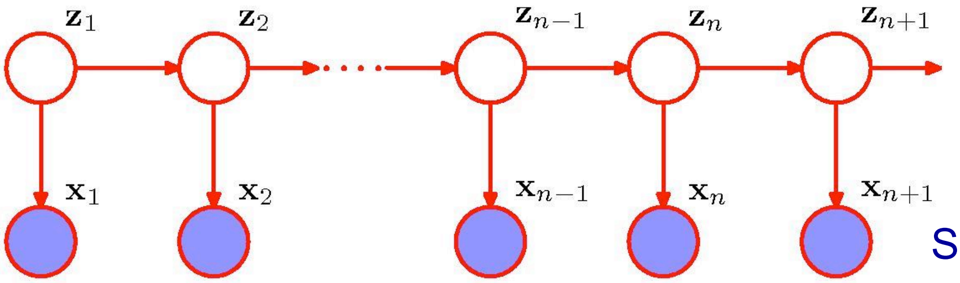
$$p(\mathbf{z}_n = k | \mathbf{z}_{n-1} = j) = A_{jk}, \qquad p(\mathbf{z}_1 = k) = \pi_k.$$

- A set of output probability distributions (**one per state**) converts this state path into a sequence of observable symbols/vectors, through emission probabilities:

$$p(\mathbf{x}_n | \mathbf{z}_n, \phi).$$

Can be e.g. Gaussian if $\mathbf{x}$ is continuous.

Conditional probability table if $\mathbf{x}$ is discrete.



State transition

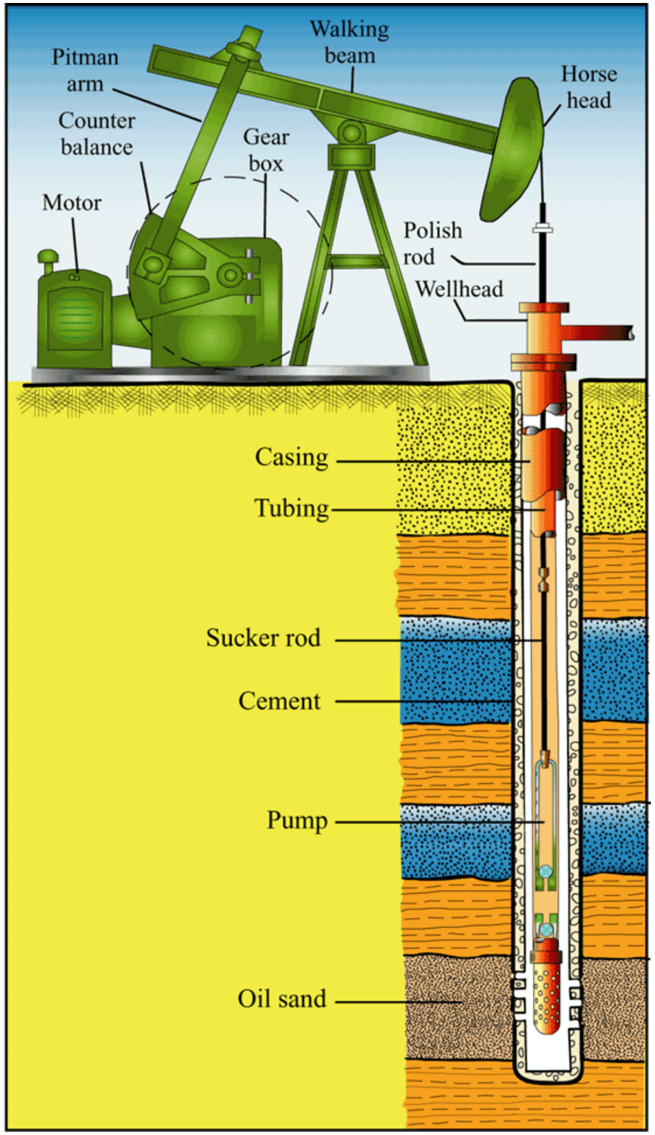
Observation model

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

Bonus

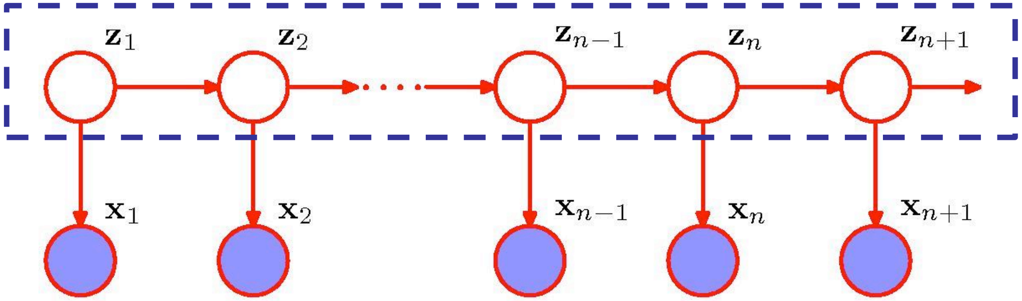
HMM application: Casing Running

www.youtube.com/watch?v=F-HrLO5m_-s

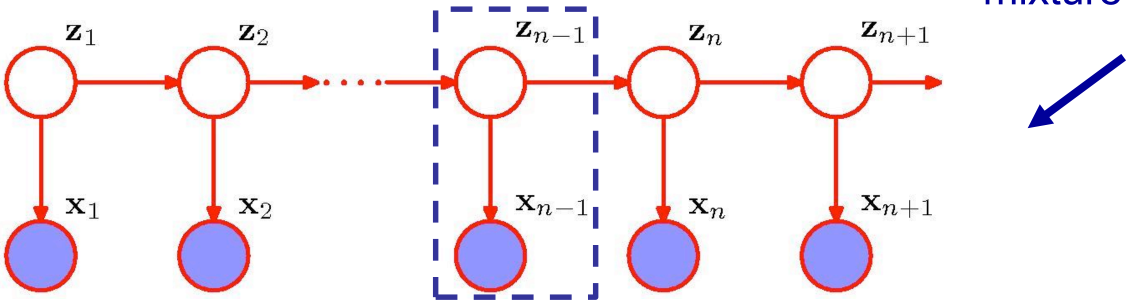


Links to Other Models

- You can view an HMM as a Markov chain with stochastic measurements



- Or as a mixture model with states coupled across time



We will adopt this view, as we worked with mixture models before

Transition Probabilities

- It will be convenient to use a 1-of-*K* encoding for the latent variables
- The matrix of transition probabilities takes the form:

$$p(z_{nk} = 1 | z_{n-1,j} = 1) = A_{jk}, \sum_k A_{jk} = 1.$$

- The conditionals can be written as:

$$p(\mathbf{z}_n | \mathbf{z}_{n-1}, A) = \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j}\ z_{nk}}, \quad p(\mathbf{z}_1 | \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}.$$

- We will focus on homogeneous models: all of the conditional distributions over latent variables share the same parameters A

- Standard mixture model for i.i.d. data:
  - Special case in which all parameters A_{jk} are the same for all j
  - The conditional distribution $p(\mathbf{z}_n | \mathbf{z}_{n\text{-}1})$ is independent of $\mathbf{z}_{n\text{-}1}$

Emission Probabilities

- The emission probabilities take the form:

$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{k=1}^{K} p(\mathbf{x}_n|\phi_k)^{z_{nk}}.$$

- For a continuous $\mathbf{x}$, we commonly have

$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}}.$$

- For the discrete, multinomial observed variable $\mathbf{x}$, using 1-of-*D* encoding, the conditional distribution takes the form:

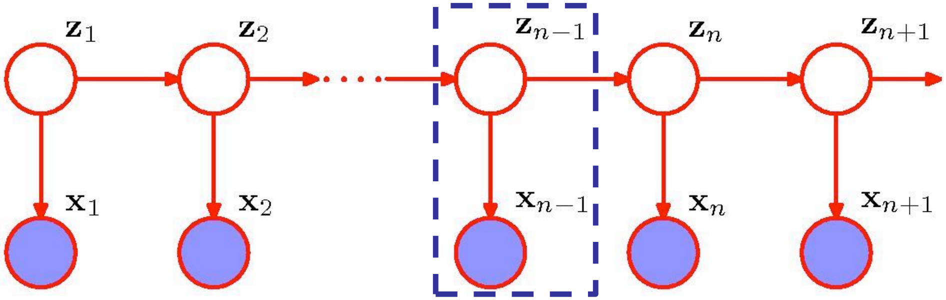
$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{i=1}^{D} \prod_{k=1}^{K} \mu_{ik}^{x_{ni} z_{nk}}.$$

HMM Model Equations

- The joint distribution over the observed- and latent variables is given by:

$$p(\mathbf{X}, \mathbf{Z}|\theta) = p(\mathbf{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}, A) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\phi}),$$

where $\theta = \{\boldsymbol{\pi}, A, \boldsymbol{\phi}\}$ are the model parameters

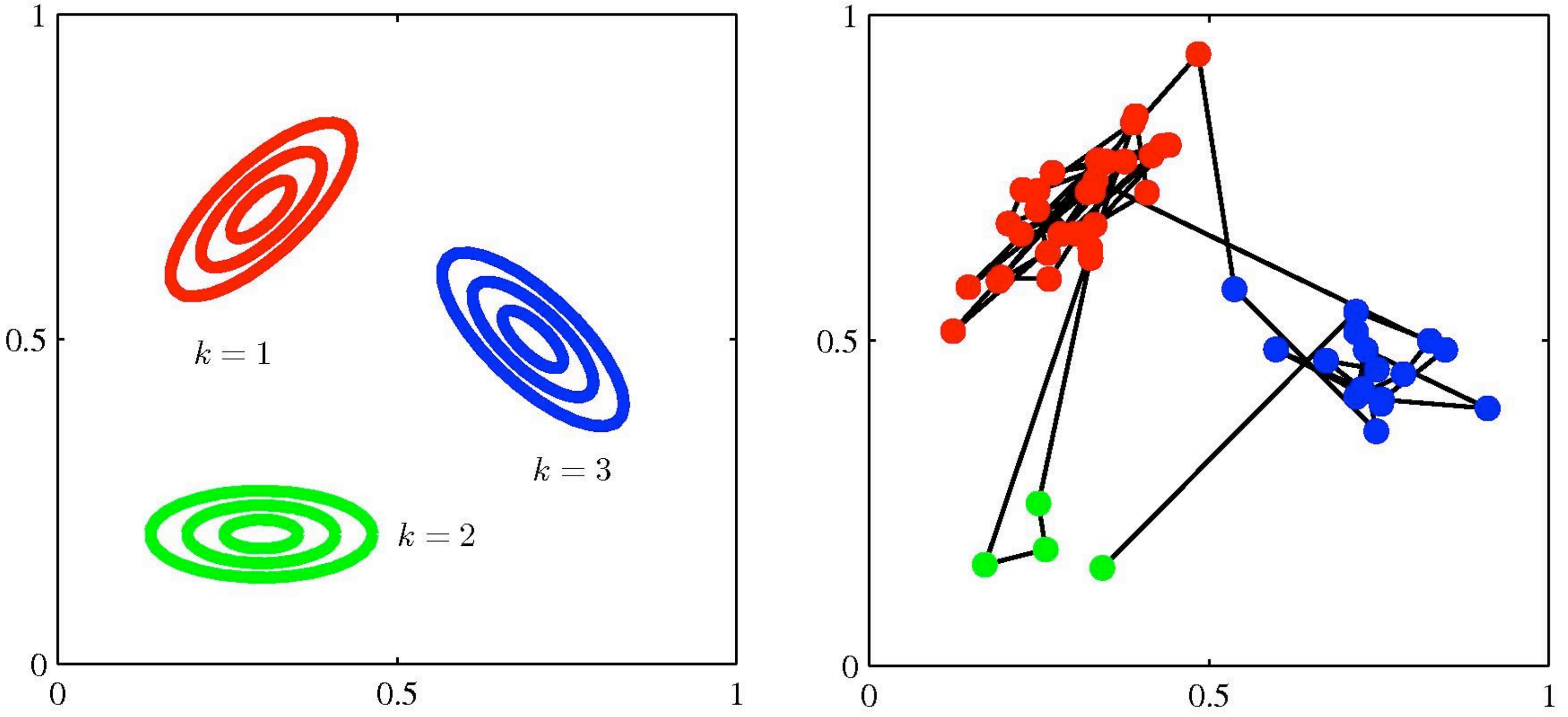


- Data are not i.i.d. Everything is coupled across time

- Three problems and three solutions:
  - Computing probabilities of observed sequences: Forward-backward algorithm
  - Inference of hidden state sequences: Viterbi algorithm
  - Learning of parameters: Baum-Welch algorithm

HMM as a Mixture Through Time

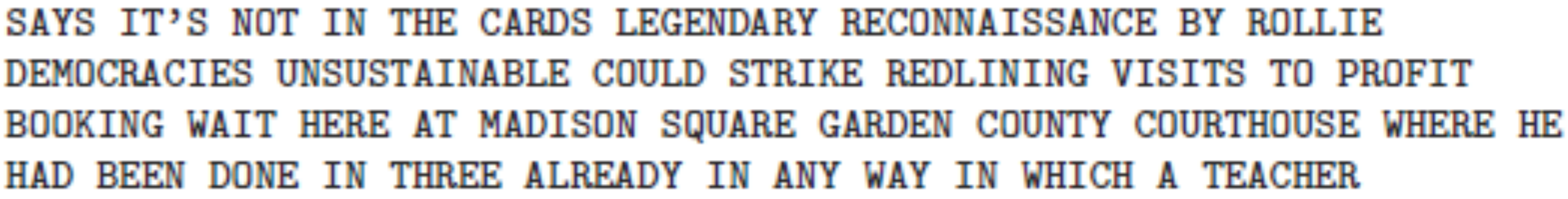
- Sampling from a 3-state HMM with a 2-d Gaussian emission model



- The transition matrix is fixed: $A_{kk}=0.9\ \forall\ k$, and $A_{jk} = 0.05$ for $j \neq k$

Applications of HMMs

- Speech recognition
- Language modeling
- Motion video analysis/tracking
- Protein sequences/genetic sequences: alignment & analysis
- Financial time series prediction



SAYS IT'S NOT IN THE CARDS LEGENDARY RECONNAISSANCE BY ROLLIE
DEMOCRACIES UNSUSTAINABLE COULD STRIKE REDLINING VISITS TO PROFIT
BOOKING WAIT HERE AT MADISON SQUARE GARDEN COUNTY COURTHOUSE WHERE HE
HAD BEEN DONE IN THREE ALREADY IN ANY WAY IN WHICH A TEACHER

Maximum Likelihood for the HMM

- We observe a dataset $\mathbf{X} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$
- The goal is to determine the model parameters $\theta = \{\boldsymbol{\pi}, A, \boldsymbol{\phi}\}.$
- The probability of an observed sequence takes the form:

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta).$$

$$p(\text{observed sequence}) = \sum_{\text{all paths}} p(\text{observed outputs, state paths}).$$

- In contrast to mixture models, the joint distribution $p(\mathbf{X}, \mathbf{Z} \mid \theta)$ does not factorize over *n*

- It looks hard: *N* latent variables, each of which has *K* states. Hence there are

K^N $\cancel{N}^{K}$ total paths

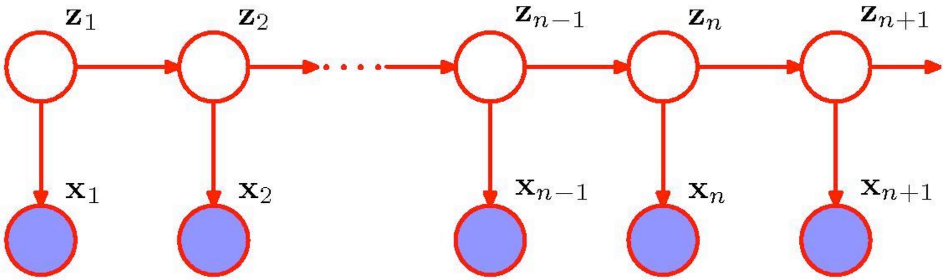
- Remember how we tackled the inference problem on a simple chain (slide 24)

Probability of an Observed Sequence

- The joint distribution can be rearranged:

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta) = \sum_{\mathbf{z}_1,\dots,\mathbf{z}_n} p(\mathbf{z}_1, \mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}) p(\mathbf{x}_n|\mathbf{z}_n)$$

$$= \sum_{\mathbf{z}_1} p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) \sum_{\mathbf{z}_2} p(\mathbf{z}_2|\mathbf{z}_1) p(\mathbf{x}_2|\mathbf{z}_2) \dots$$



$$\sum_{\mathbf{z}_N} p(\mathbf{z}_N|\mathbf{z}_{N-1}) p(\mathbf{x}_N|\mathbf{z}_N).$$

- Dynamic Programming: By moving the summations inside, we can save a lot of work

An EM algorithm

- We cannot perform direct maximization (no closed-form solution):

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta).$$

- EM algorithm: we will derive an efficient algorithm for maximizing the likelihood function in HMMs (and, later, in linear state-space models)

- E-step: Compute the posterior distribution over latent variables:

$$p(\mathbf{Z}|\mathbf{X}, \theta^{old}).$$

- M-step: Maximize the expected complete data log-likelihood:

$$Q(\theta, \theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \log p(\mathbf{X}, \mathbf{Z}|\theta).$$

- If we knew the true state path, then ML parameter estimation would be trivial

- We will first look at the E-step: Computing the true posterior distribution over the state paths

Inference of Hidden States

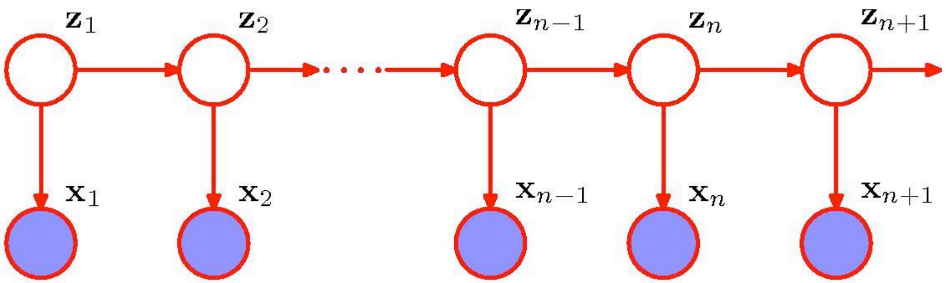
- We want to estimate the hidden states given observations. To start with, let us estimate a single hidden state:

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})}.$$

- Using the conditional-independence property, we obtain:

$$p(\mathbf{z}_n|\mathbf{X}) = \frac{p(\mathbf{x}_1, .., \mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})}$$

$$= \frac{p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}.$$



Inference of Hidden States

- Hence:

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N|\mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}.$$

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, .., \mathbf{x}_n, \mathbf{z}_n)$$

The joint probability of observing all of the data up to time *n*, and $\mathbf{z}_n$.

$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1}, .., \mathbf{x}_N|\mathbf{z}_n).$$

The conditional probability of all future data from time *n*+1 to *N*.

- Each $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ represent a set of *K* numbers, one for each of the possible settings of the 1-of-*K* binary vector $\mathbf{z}_n$

- We will derive an efficient recursive algorithm, known as the alpha-beta recursion, or forward-backward algorithm

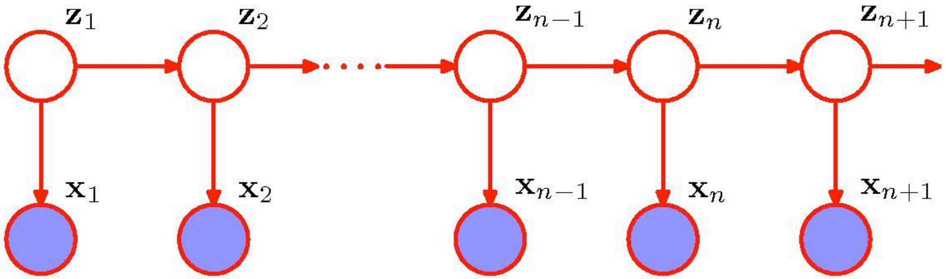
The Forward (α) Recursion

- The forward recursion:

can express alpha on t=n as a function of alpha on t=n-1

$$\begin{aligned}\alpha(\mathbf{z}_n) &= p(\mathbf{x}_1,..,\mathbf{x}_n,\mathbf{z}_n)\\ &= p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{x}_1,..,\mathbf{x}_{n-1},\mathbf{z}_n)\\ &= p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1,..,\mathbf{x}_{n-1},\mathbf{z}_{n-1},\mathbf{z}_n)\\ &= p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1,..,\mathbf{x}_{n-1},\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})\\ &= p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})\end{aligned}$$

Computational cost scales like O(K^2).



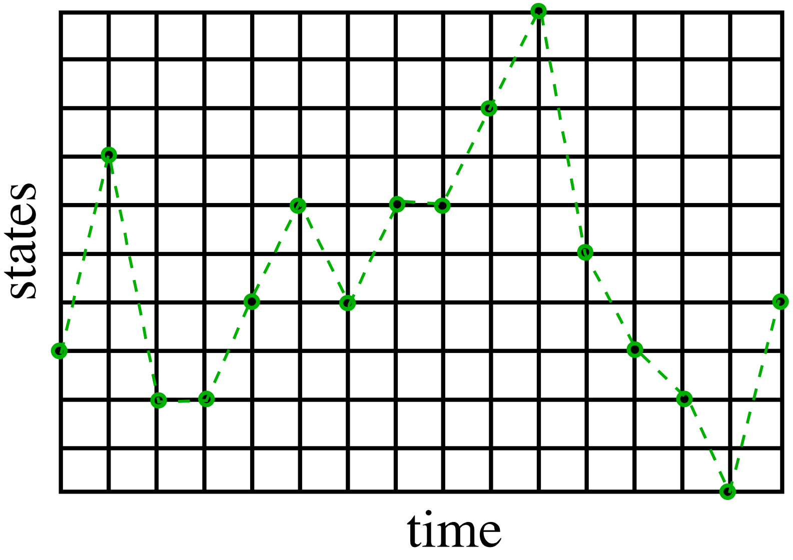
- Note that:

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N).$$

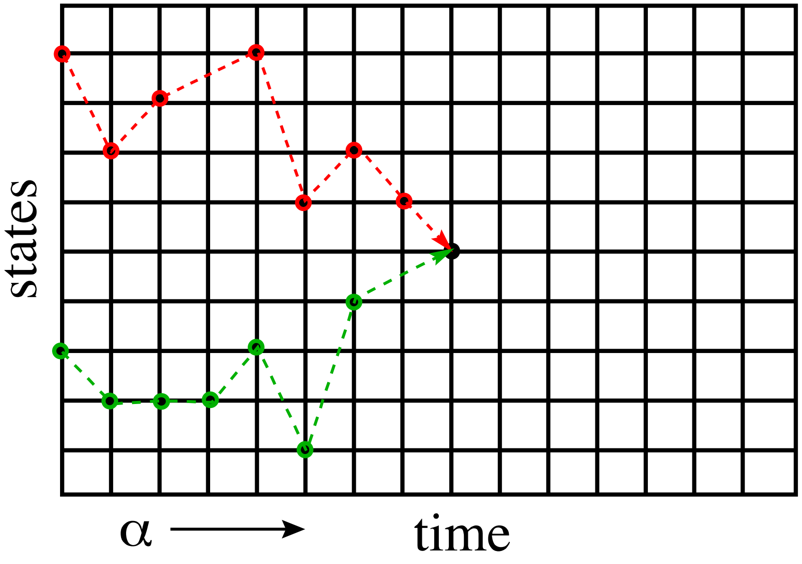
- This enables us to easily (cheaply) compute the desired likelihood

The Forward (α) Recursion

- The forward recursion:



Exponentially many paths.

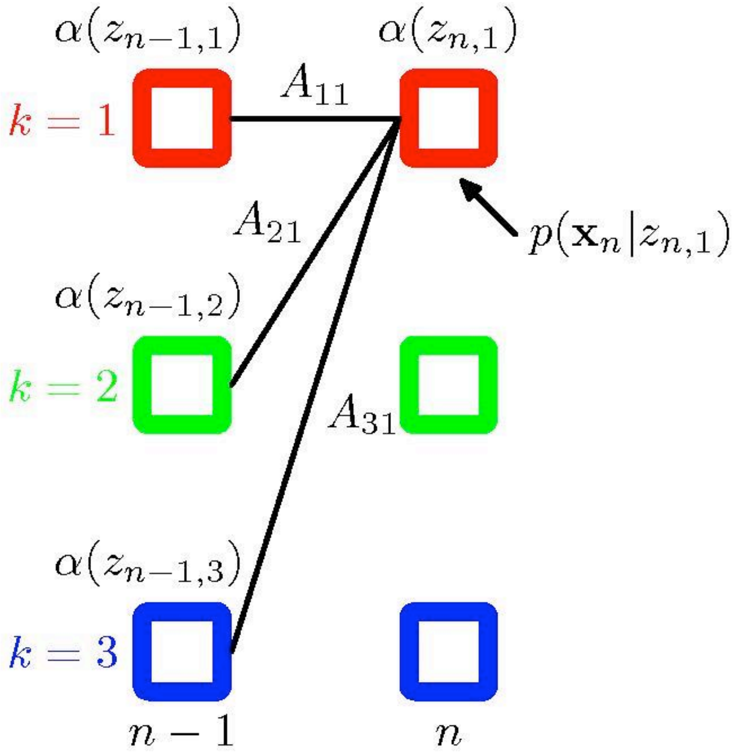


At each node, sum up the values of all incoming paths.

- This is exactly dynamic programming

The Forward (α) Recursion

- Illustration of the forward recursion



Here $\alpha(z_{n,1})$ is obtained by:

- Taking the elements $\alpha(z_{n\text{-}1}, j)$
- Summing them up with weights A_{j1}, corresponding to $p(\mathbf{z}_n \mid \mathbf{z}_{n\text{-}1})$
- Multiplying by the data contribution $p(\mathbf{x}_n \mid z_{n,1})$

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

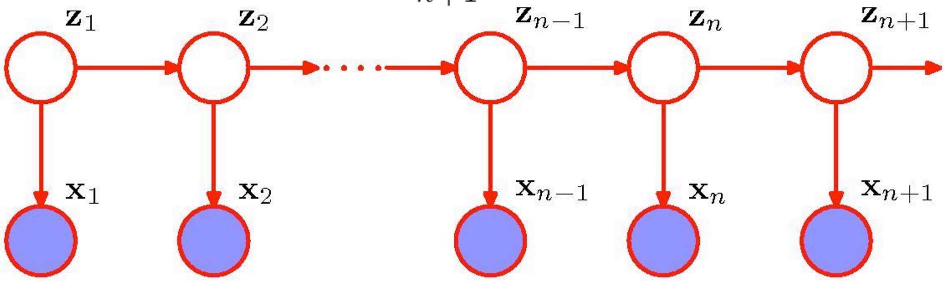
- The initial condition is given by:

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1|\mathbf{z}_1)p(\mathbf{z}_1) = \prod_{k=1}^{K} \left[\pi_k p(\mathbf{x}_1|\phi_k)\right]^{z_{1k}}.$$

The Backward (β) Recursion

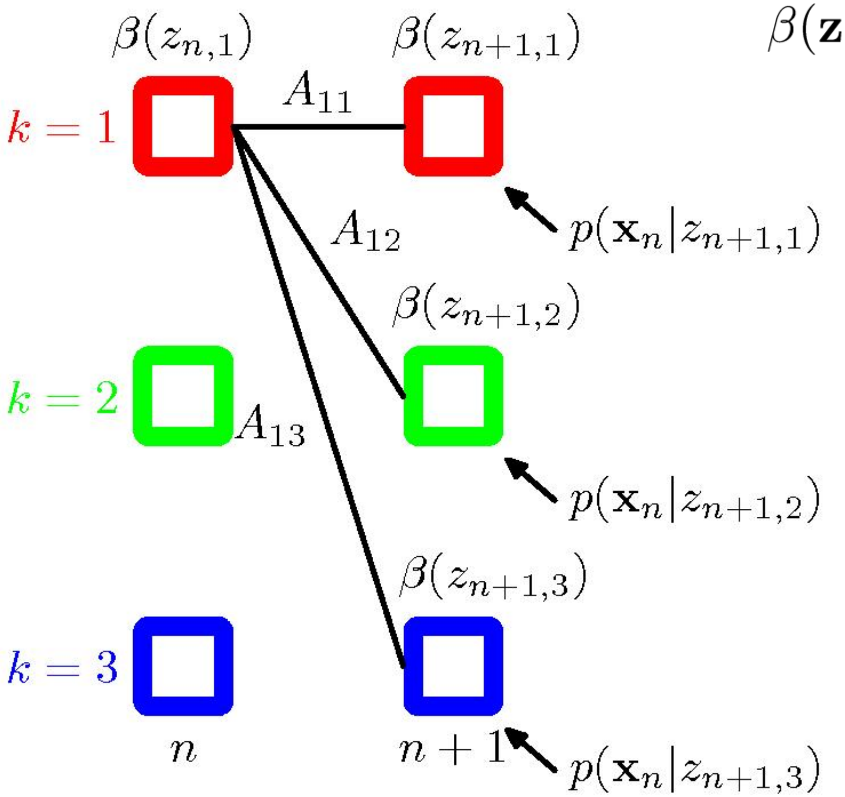
- There is also a simple recursion for $\beta(\mathbf{z}_n)$:

$$\begin{aligned}
\beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1}, .., \mathbf{x}_N | \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N, \mathbf{z}_{n+1} | \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N | \mathbf{z}_{n+1}, \mathbf{z}_n) p(\mathbf{z}_{n+1} | \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, ..., \mathbf{x}_N | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+2}, ..., \mathbf{x}_N | \mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n)
\end{aligned}$$



The Backward (β) Recursion

- Illustration of the backward recursion



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

- Initial condition:

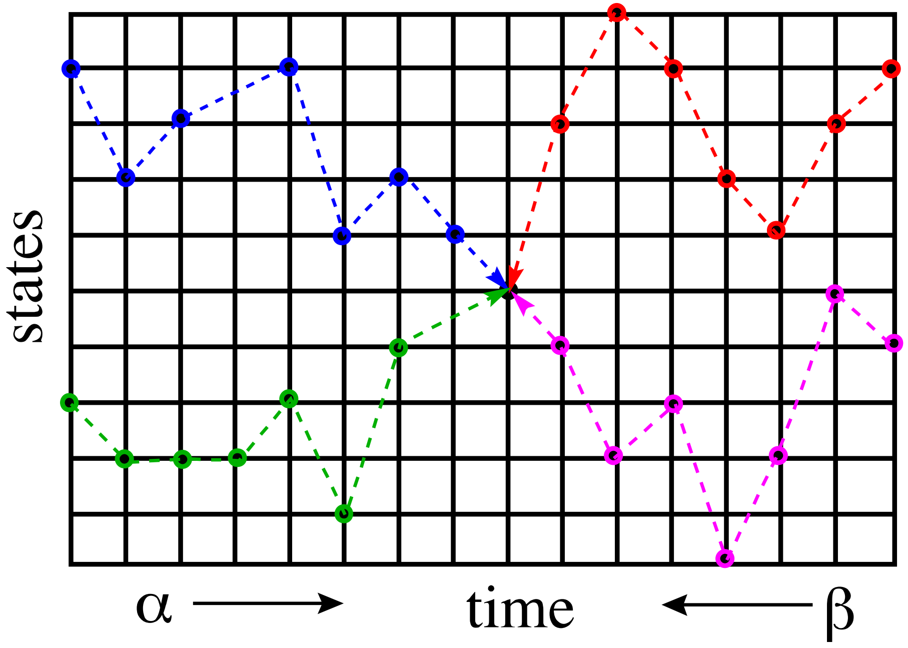
$$\begin{aligned} p(\mathbf{z}_N|\mathbf{X}) &= \frac{\alpha(\mathbf{z}_N)\beta(\mathbf{z}_N)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{X},\mathbf{z}_N)\beta(\mathbf{z}_N)}{p(\mathbf{X})}. \end{aligned}$$

- Hence:

$$\beta(\mathbf{z}_N) = 1.$$

The Backward (β) Recursion

- $\alpha(z_{nk})$ gives total inflow of probability to node (n,k)
- $\beta(z_{nk})$ gives total outflow of probability



- In fact, we can do one forward pass to compute all the $\alpha(\mathbf{z}_n)$ and one backward pass to compute all the $\beta(\mathbf{z}_n)$ and then compute any $\gamma(\mathbf{z}_\mathrm{n})$ we want. Total cost is $\mathcal{O}(K^2N)$

Computing Likelihood

- Note that

$$\sum_{\mathbf{z}_n} \gamma(\mathbf{z}_n) = \sum_{\mathbf{z}_n} p(\mathbf{z}_n|\mathbf{X}) = 1.$$

- We can compute the likelihood at any time using *α*-*β* recursion:

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

- In the forward calculation we proposed originally, we did this at the final time step *n* = *N*

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N).$$

because $\beta(\mathbf{z}_n)$=1

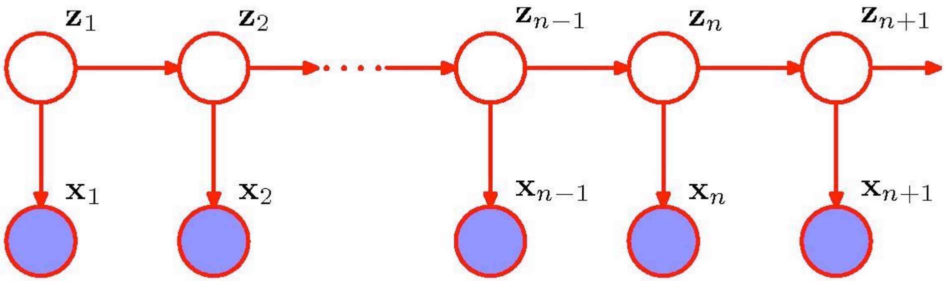
- Monitoring this quantity is a good way to check for convergence during EM

Two-Frame Inference

- We will also need the cross-time statistics for adjacent time steps:

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

- This is a *K* x *K* matrix with elements (*i,j*) representing the expected number of transitions from state *i* to state *j* that begin at time *n*-1, given all the observations.
- Whereas *γ* is the marginal posterior distribution of a latent variable, *ξ* is the joint posterior distribution of two successive latent variables



- It can be computed with the same *α*- and *β* recursions

EM algorithm

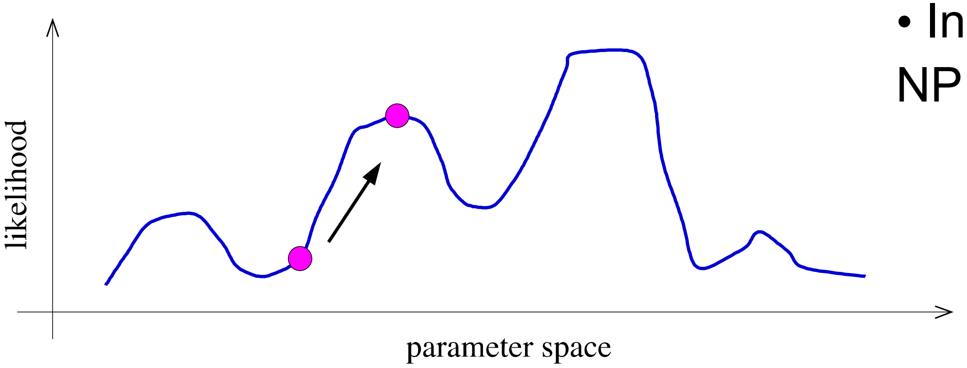
- Intuition: if only we knew the true state path then ML parameter estimation would be trivial
- E-step: Compute the posterior distribution over the state path using *α-β* recursion (dynamic programming):

$$p(\mathbf{Z}|\mathbf{X},\theta^{old}).$$

- M-step: Maximize the expected complete data log-likelihood (parameter re-estimation):

$$Q(\theta,\theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X},\theta^{old}) \log p(\mathbf{X},\mathbf{Z}|\theta).$$

- We then iterate. This is also known as a Baum-Welch algorithm (special case of EM)



- In general, finding the ML parameters is NP hard, so initial conditions matter a lot

Complete Data Log-likelihood

- The complete data log-likelihood takes the form:

$$\log p(\mathbf{X}, \mathbf{Z}|\theta) \quad = \quad \log \left[p(\mathbf{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\mathbf{z}_n|\mathbf{z}_{n-1}, A) \prod_{n=1}^{N} p(\mathbf{x}_n|\mathbf{z}_n, \boldsymbol{\phi}) \right]$$

$$= \quad \log \left[\prod_{k=1}^{K} \pi_k^{z_{1k}} \prod_{n=2}^{N} \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j}, z_{nk}} \prod_{n=1}^{N} \prod_{k=1}^{K} p(\mathbf{x}_n|\mathbf{z}_n)^{z_{nk}} \right]$$

$$= \sum_{k=1}^{K} z_{1k} \log \pi_k + \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} [z_{n-1,j} z_{nk}] \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log p(\mathbf{x}_n|\mathbf{z}_n).$$

transition model

observation model

Expected Complete Data Log-likelihood

- The complete data log-likelihood takes the form:

$$Q(\theta, \theta^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \log p(\mathbf{X}, \mathbf{Z}|\theta).$$

$$= \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} \xi(z_{n-1,j} z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \log p(\mathbf{x}_n|\mathbf{z}_n).$$

- Hence in the E-step, we evaluate:

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

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}).$$

- In the M-step, we optimize *Q* with respect to parameters: $\theta = \{\boldsymbol{\pi}, A, \boldsymbol{\phi}\}.$

Parameter Estimation

- Initial state distribution: Using Lagrange multipliers, the expected number of times in state *k* at time 1 is:

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

- Expected number of transitions from state *j* to *k* which begin at time *n*-1:

$$\xi(\mathbf{z}_{n-1,j}, \mathbf{z}_{n,k}) = p(\mathbf{z}_{n-1,j}, \mathbf{z}_{n,k} | \mathbf{X}),$$

and the estimated transition probabilities work out to be:

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

- The EM algorithm must be initialized by choosing starting values for ***π*** and *A*

- Note that any elements of ***π*** or *A* that initially are set to zero will remain zero in subsequent EM updates

Parameter Estimation: Emission Model

- For the case of discrete multinomial observed variables, the observation model takes the form:

$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{i=1}^{D}\prod_{k=1}^{K} \mu_{ik}^{x_{ni}z_{nk}}.$$

Same as fitting Bernoulli mixture model.

- And the corresponding M-step update: $\mu_{ik}^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^{N} \gamma(z_{nk})}.$

- For the case of the Gaussian emission model:

$$p(\mathbf{x}_n|\mathbf{z}_n, \phi) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_{nk}}.$$

Remember:

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

- And the corresponding M-step updates:

$$\boldsymbol{\mu}_k^{new} = \frac{1}{N_k}\sum_n \gamma(z_{nk})\mathbf{x}_n, \quad N_k = \sum_n \gamma(z_{nk}),$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T,$$

Same as fitting a Gaussian mixture model.

Viterbi Decoding

- The numbers $\gamma(\mathbf{z}_n)$ above gave the probability distribution over all states at any time

- By choosing the state $\gamma^*(\mathbf{z}_n)$ with the largest probability at each time, we can make an "average" state path. This is the path with the maximum expected number of correct states

- To find the single best path, we do Viterbi decoding which is a dynamic programming algorithm applied to this problem

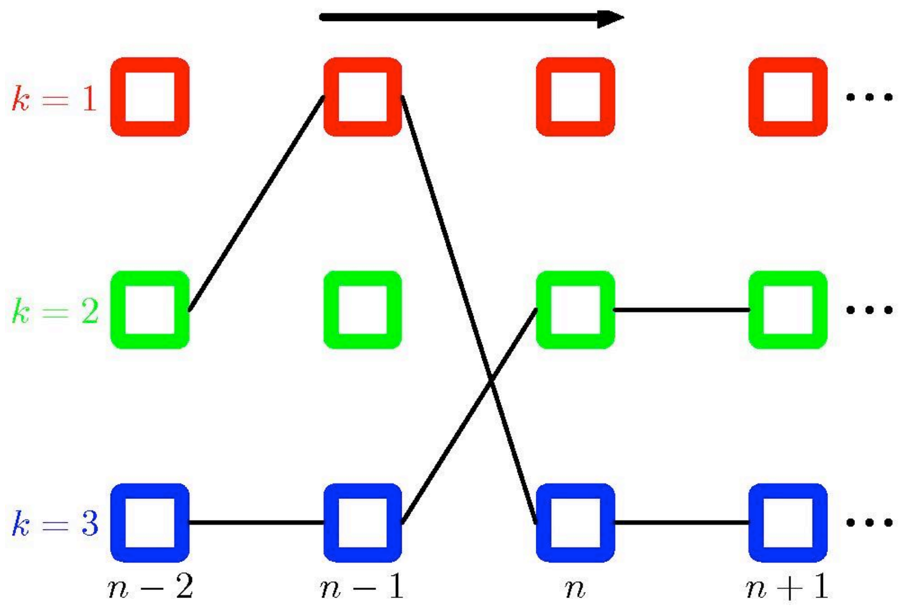
- The recursions look the same, except with 'max' instead of $\sum$

- Same dynamic programming trick: instead of summing, we keep the term with the highest value at each node

- There is also a modified EM (Baum-Welch) training based on the Viterbi decoding. Like K-means instead of mixtures of Gaussians

Viterbi Decoding

- A fragment of the HMM lattice showing two possible paths:

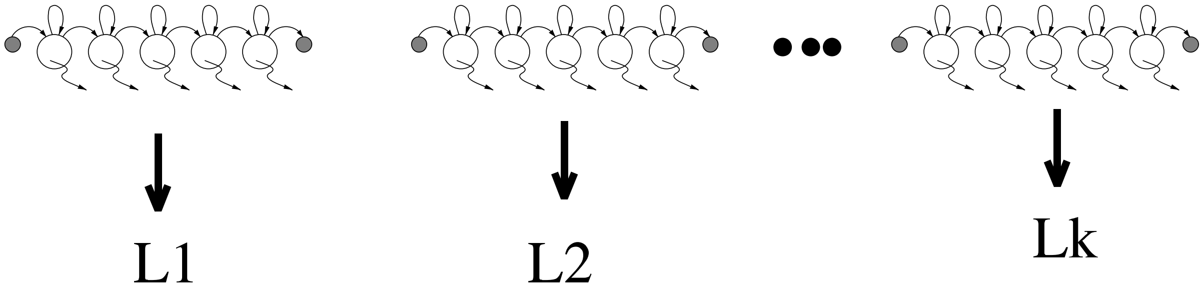


- Viterbi decoding efficiently determines the most probable path from the exponentially many possibilities

- The probability of each path is given by the product of the elements of the transition matrix A_{jk}, along with the emission probabilities associated with each node in the path

Using HMMs for Recognition

- We can use HMMs for recognition by:
  - Training one HMM for each class (requires labelled training data)
  - Evaluating the probability of an unknown sequence under each HMM
  - Classifying the unknown sequence by choosing an HMM with highest likelihood

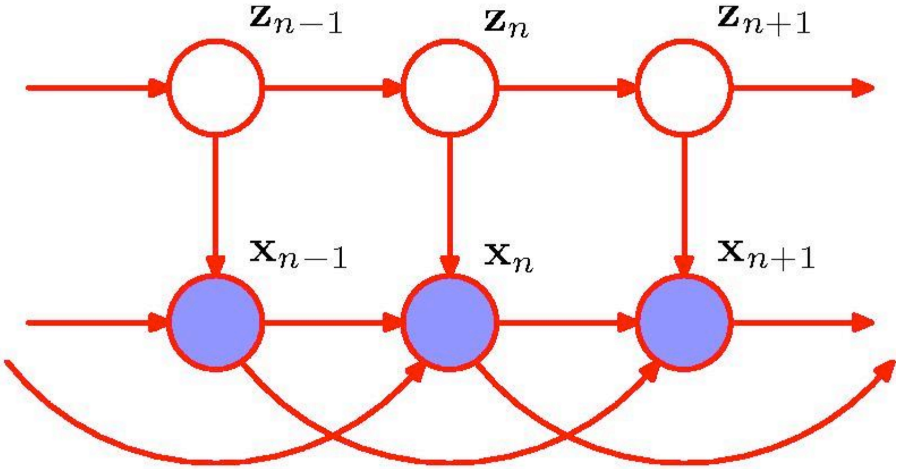


- This requires the solution of two problems:
  - Given a model, evaluate the probability of a sequence. (We can do this exactly and efficiently.)
  - Given some training sequences, estimate the model parameters. (We can find the local maximum using EM.)

Bonus

Autoregressive HMMs

- One limitation of the standard HMM is that it is poor at capturing long-range correlations between observations, as these have to be mediated via the first order Markov chain of hidden states



- Autoregressive HMM: The distribution over $\mathbf{x}_n$ depends also on a subset of previous observations

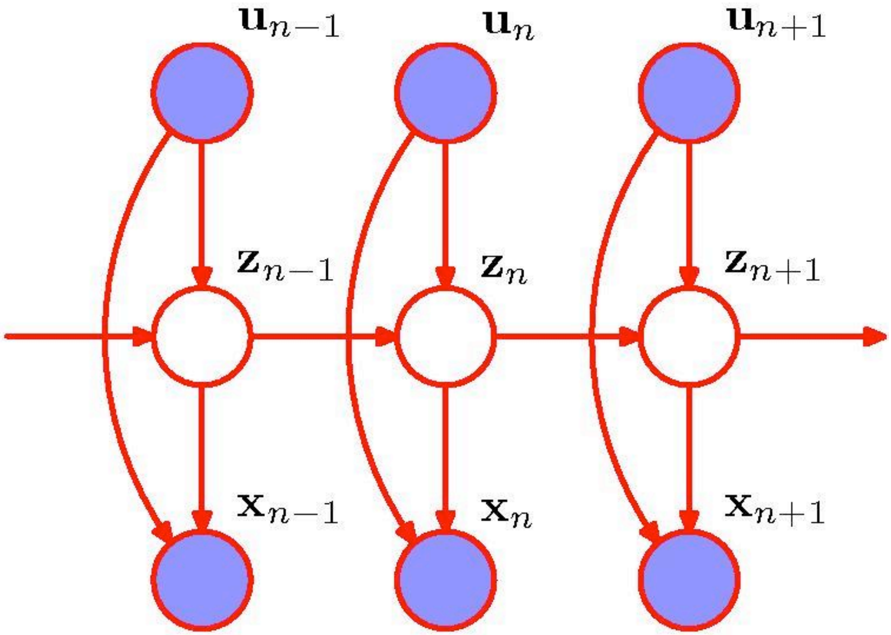
- The number of additional links must be limited to avoid an excessive number of free parameters

- The graphical model framework motivates a number of different models based on HMMs

Bonus

Input-Output HMMs

- Both the emission probabilities and the transition probabilities depend on the values of a sequence of observations $\mathbf{u}_1,\ldots,\mathbf{u}_N$

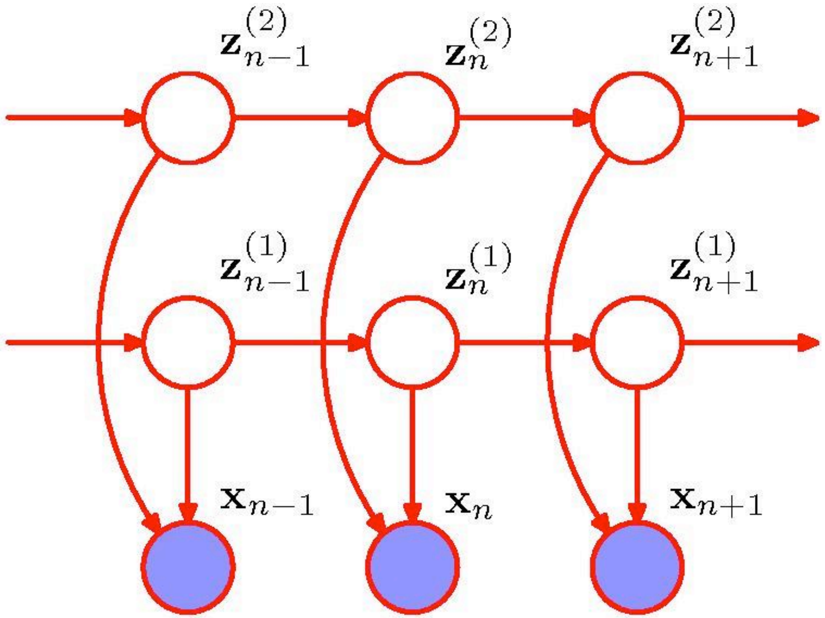


- Model parameters can be efficiently fit using EM, in which the E-step involves forward-backward recursion

Bonus

Factorial HMMs

- Example of Factorial HMM comprising two Markov chains of latent variables:



- Motivation: In order to represent 10 bits of information at a given time step, a standard HMM would need $K=2^{10}=1024$ states

- Factorial HMMs would use 10 binary chains

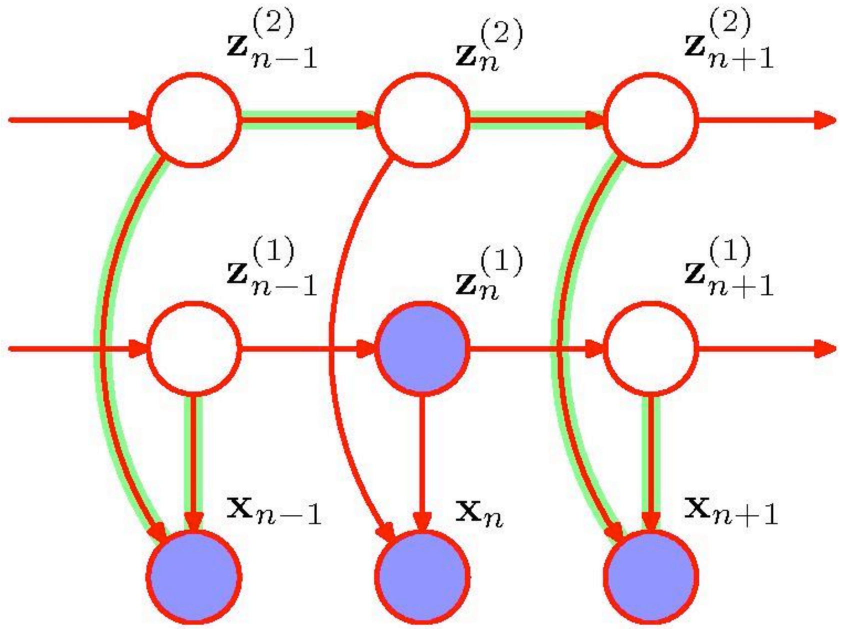
- Much more powerful model

- The key disadvantage: Exact inference is intractable
- Observing the **x** variables introduces dependencies among the latent chains
- Hence E-step for this model **does not** correspond to running forward-backward along the *M* latent chains independently

Bonus

Factorial HMMs

- The conditional independence property: $\mathbf{z}_{n+1} \perp\!\!\!\perp \mathbf{z}_{n-1} \mid \mathbf{z}_n$ does not hold for the individual latent chains



- There is no efficient exact E-step for this model

- One solution would be to use MCMC techniques to obtain an approximate sample from the posterior

- Another alternative is to resort to *variational inference*

- The variational distribution can be described by *M* separate Markov chains corresponding to the latent chains in the original model (structured mean-field approximation)

Outline

- Review / background of Graphical Models
- Example of a famous Bayesian network
- Inference in Bayesian networks
- Conditional independence in BNs
- Sequential models:
  - HMM and its three problems
  - Bonus: ARHMM, IOHMM, FHMM

Bonus

Example conference deadlines

- ICML (February, held in July)
- UAI (March, held in July)
- NIPS (June, held in December)
- AISTATS (October, held in May)