

# **STA 4273H: Statistical Machine Learning**

Russ Salakhutdinov

Department of Statistics

[rsalakhu@utstat.toronto.edu](mailto:rsalakhu@utstat.toronto.edu)

<http://www.utstat.utoronto.ca/~rsalakhu/>

Sidney Smith Hall, Room 6002

## **Lecture 11**

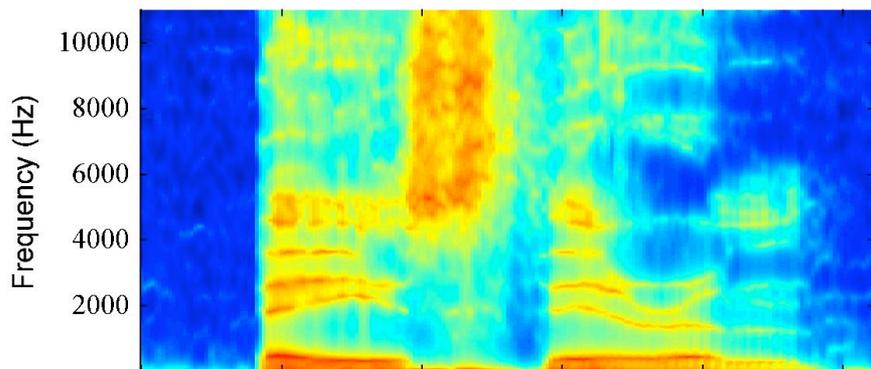
# Project Deadline

- Project Deadline is Dec 15, 2011. This is a hard deadline!
- I will not be able to give any extensions, as I have to turn in the marks by Dec 21.
- Send the projects to me via e-mail to:  
[rsalakhu@utstat.toronto.edu](mailto:rsalakhu@utstat.toronto.edu)

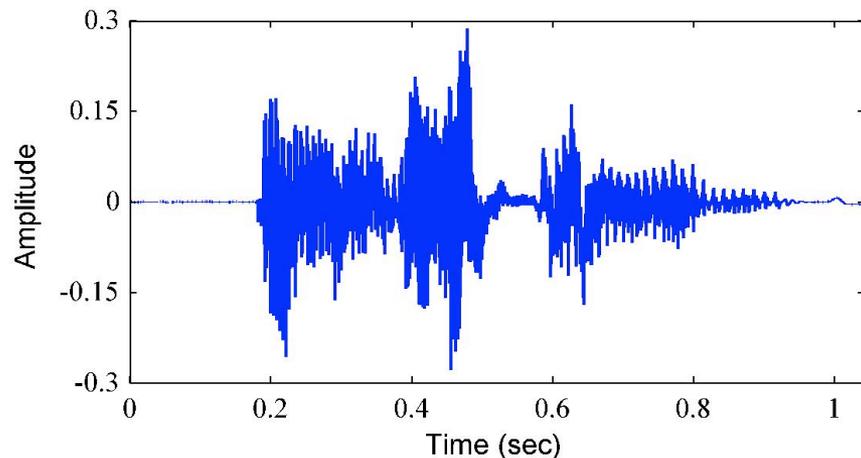
# Sequential Data

- So far we focused on problems that assumed that the data points were **independent and identically distributed** (i.i.d. assumption).
- Express the **likelihood function** as a product over all data points of the probability distribution evaluated at each data point.
- Poor assumption 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



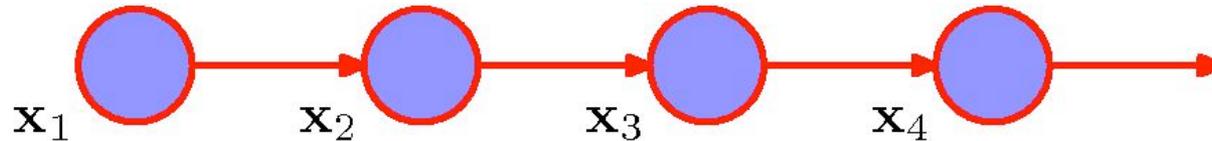
- Example of a spectrogram of a spoken word 'Bayes theorem':
- Successive observations are highly correlated.



b	ey	z	th	ih	er	em
Bayes'		Theorem				

# Markov Models

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



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

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

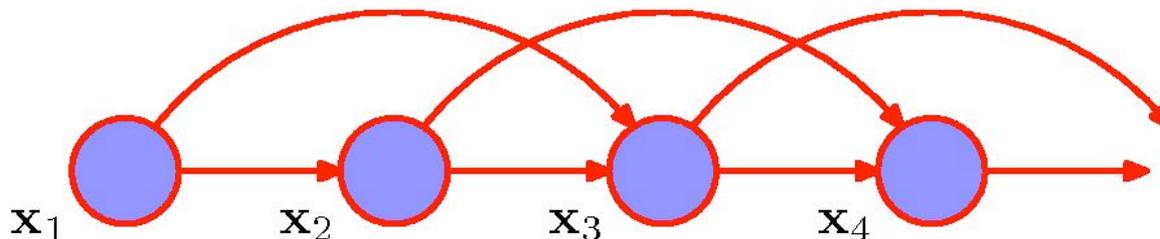
- From the **d-separation property**, the conditionals are given by:

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

- 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 **homogenous 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:

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

- We can similarly consider extensions to an  $M^{\text{th}}$  order Markov chain.
- Increased flexibility  $\rightarrow$  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  $K^{\text{th}}$  order model:

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

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

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

- **Example**: for discrete outputs (symbols) and a **2nd-order Markov model** we can use the multinomial model:

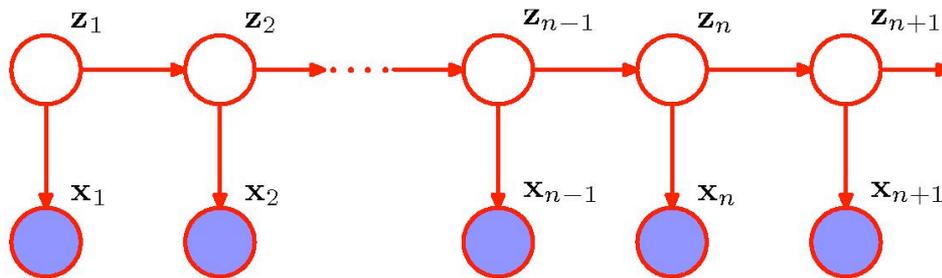
$$p(\mathbf{x}_n = m | \mathbf{x}_{n-1} = a, \mathbf{x}_{n-2} = b) = \alpha_{mab}.$$

- The **maximum likelihood** values for  $\alpha$  are:

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

# State Space Models

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



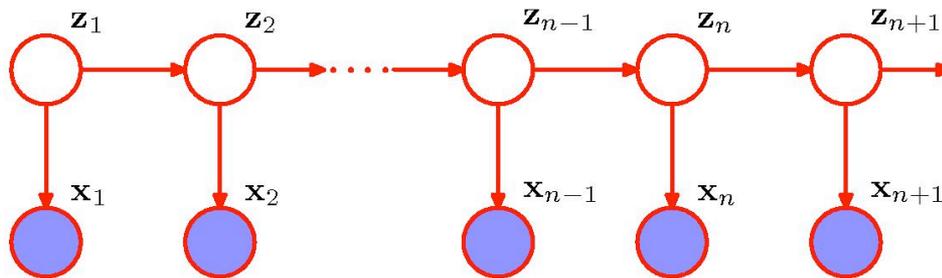
- Graphical structure known as the State Space Model.

- For each observation  $x_n$ , we have a latent variable  $z_n$ . Assume that latent variables form a Markov chain.
- If the latent variables are discrete  $\rightarrow$  Hidden Markov Models (HMMs). Observed variables can be discrete or continuous.
- If the latent and observed variables are Gaussian  $\rightarrow$  Linear Dynamical System.

# State Space Models

- The joint distribution is given by:

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



- Graphical structure known as the **State Space Model**.

- There is always a path connecting two observed variables  $x_n, x_m$  **via latent variables**.

- The **predictive distribution**:

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

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

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

# Hidden Markov Model

- First order Markov chain generates hidden state sequence (known as **transition probabilities**):

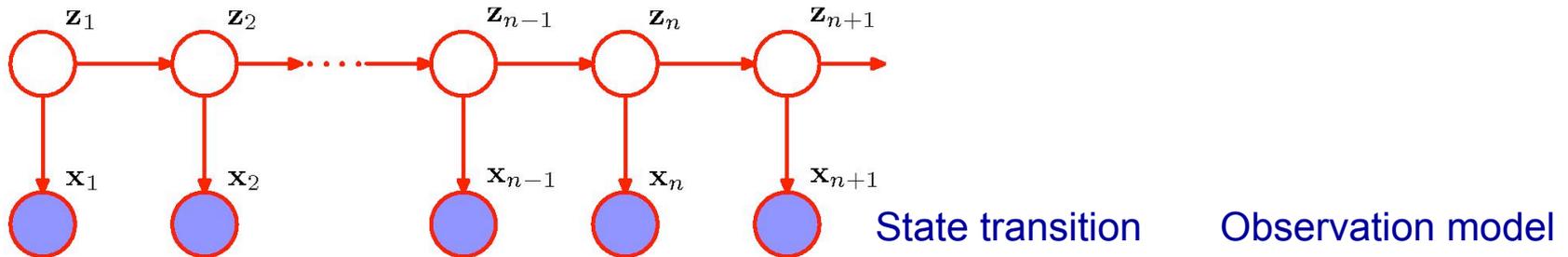
$$p(\mathbf{z}_n = k | \mathbf{z}_{n-1} = j) = A_{jk}, \quad p(\mathbf{z}_1 = k) = \pi_k.$$

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

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

Gaussian, if  $\mathbf{x}$  is continuous.

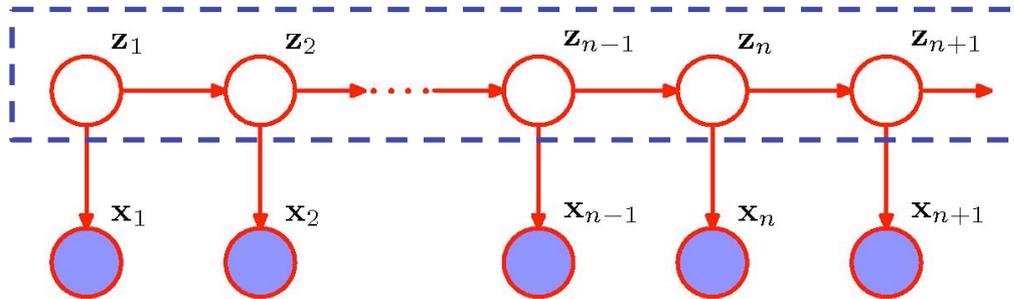
Conditional probability table if  $\mathbf{x}$  is discrete.



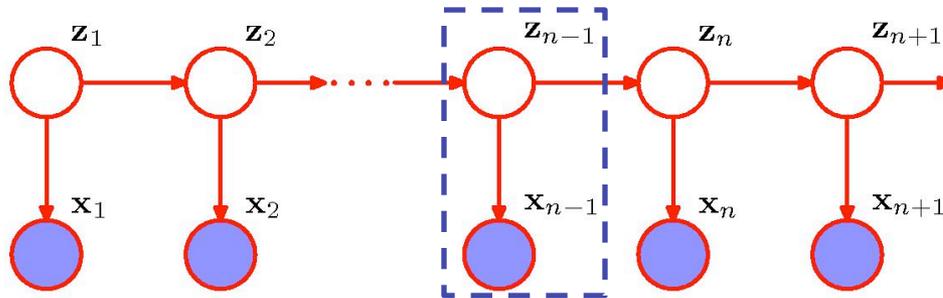
$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}) \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n).$$

# Links to Other Models

- You can view HMM as: A **Markov chain with stochastic measurements**.



- Or a **mixture model with states coupled across time**.



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



# Transition Probabilities

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

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

- 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 **homogenous 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  $j$ .
- Or the **conditional distribution**  $p(z_n | z_{n-1})$  is independent of  $z_{n-1}$ .

# Emission Probabilities

- The **emission probabilities** take form:

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

- For example, for a **continuous**  $\mathbf{x}$ , we 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-K encoding, the conditional distribution takes form:

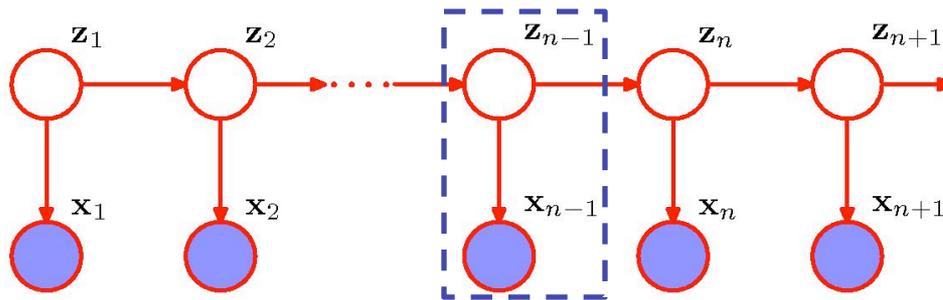
$$p(\mathbf{x}_n | \mathbf{z}_n, \phi) = \prod_{i=1}^D \prod_{k=1}^K \mu_{ik}^{x_{ni} z_{nk}}.$$

# HMM Model Equations

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

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

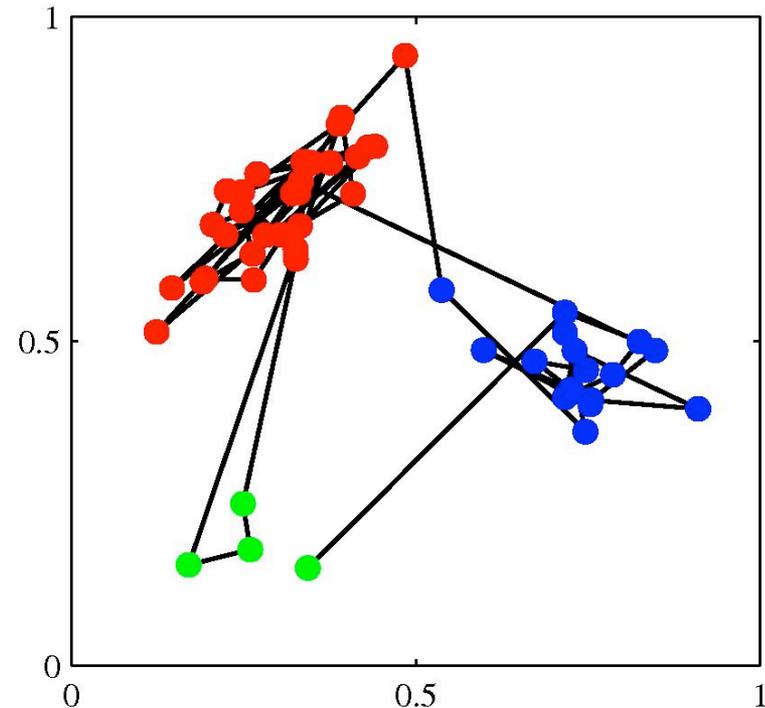
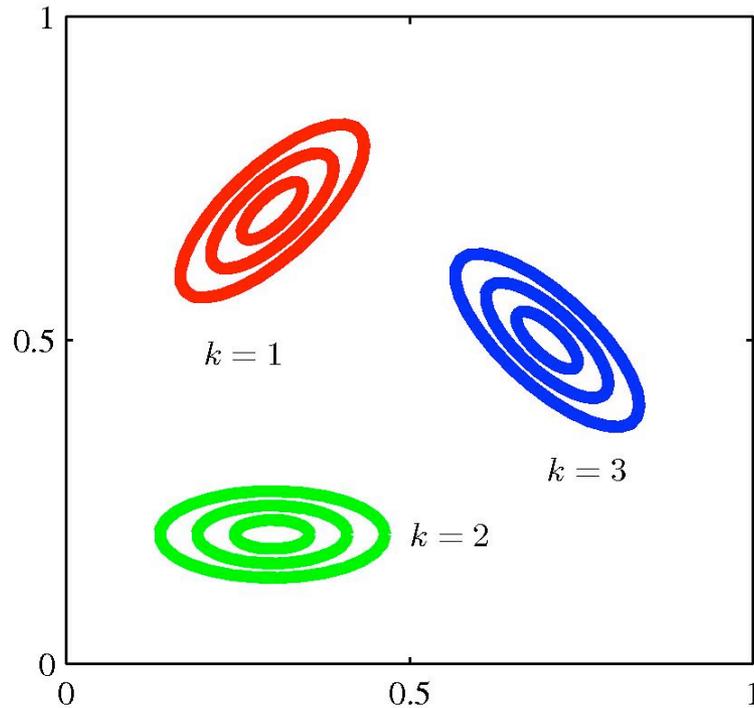
where  $\theta = \{\boldsymbol{\pi}, A, \phi\}$  are the **model parameters**.



- Data are not i.i.d. **Everything is coupled across time.**
- Three problems:** computing probabilities of observed sequences, inference of hidden state sequences, learning of parameters.

# HMM as a Mixture Through Time

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



- The transition matrix is fixed:  $A_{kk}=0.9$  and  $A_{jk} = 0.05$ .

# Applications of HMMs

- Speech recognition.
- Language modeling
- Motion video analysis/tracking.
- Protein sequence and genetic sequence alignment and analysis.
- Financial time series prediction.

# Maximum Likelihood for the HMM

- We observe a dataset  $\mathbf{X} = \{x_1, \dots, x_N\}$ .
- The goal is to determine model parameters  $\theta = \{\pi, A, \phi\}$ .
- The **probability of observed sequence** takes form:

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

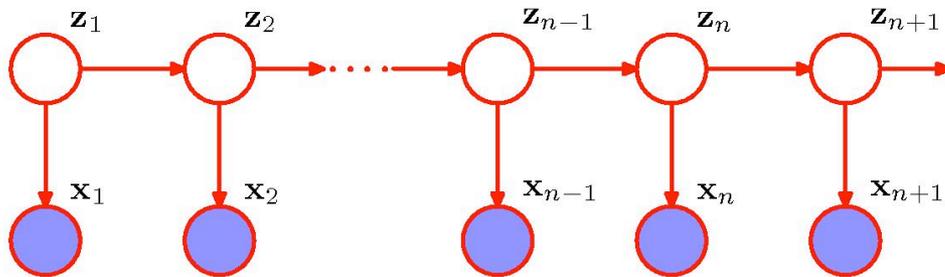
$$p(\text{observed sequence}) = \sum_{\text{all paths}} p(\text{observed outputs, state paths}).$$

- In contrast to mixture models, the joint distribution  $p(\mathbf{X}, \mathbf{Z} | \theta)$  **does not factorize over n**.
- **It looks hard**: N variables, each of which has K states. Hence  $N^K$  total paths.
- Remember inference problem on a simple chain.

# Probability of an Observed Sequence

- The joint distribution factorizes:

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta) = \sum_{\mathbf{z}_1, \dots, \mathbf{z}_n} p(\mathbf{z}_1, \mathbf{x}_1) \prod_{n=2}^N p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)$$
$$= \sum_{\mathbf{z}_1} p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) \sum_{\mathbf{z}_2} p(\mathbf{z}_2|\mathbf{z}_1)p(\mathbf{x}_2|\mathbf{z}_2) \dots$$



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

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

# 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 efficient algorithm for maximizing the likelihood function in HMMs (and later for 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.

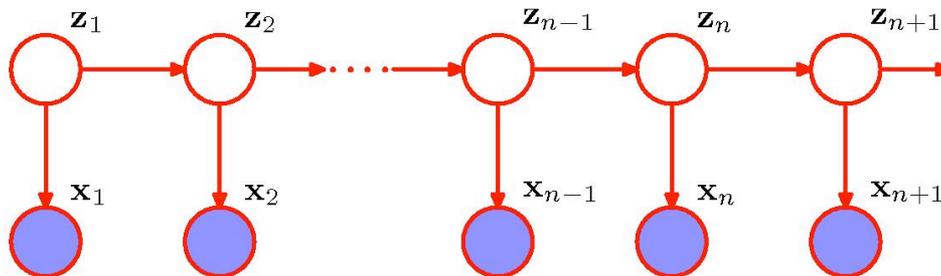
# Inference of Hidden States

- 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 conditional independence property, we obtain:

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



# Inference of Hidden States

- Hence:

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

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

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

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

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

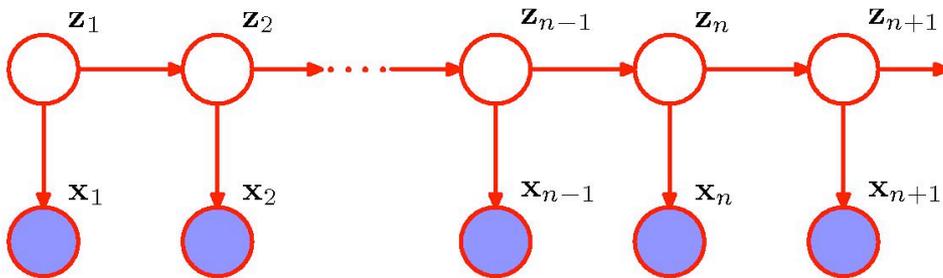
- Each  $\alpha(\mathbf{z}_n)$  and  $\beta(\mathbf{z}_n)$  represent a set of K numbers, one for each of the possible settings of the 1-of-K binary vector  $\mathbf{z}_n$ .
- We will derive efficient recursive algorithm, known as the **alpha-beta recursion**, or **forward-backward algorithm**.
- Remember the sum-product message passing algorithm for tree-structured graphical models.

# The Forward ( $\alpha$ ) Recursion

- The forward recursion:

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

Computational cost scales like  $O(K^2)$ .



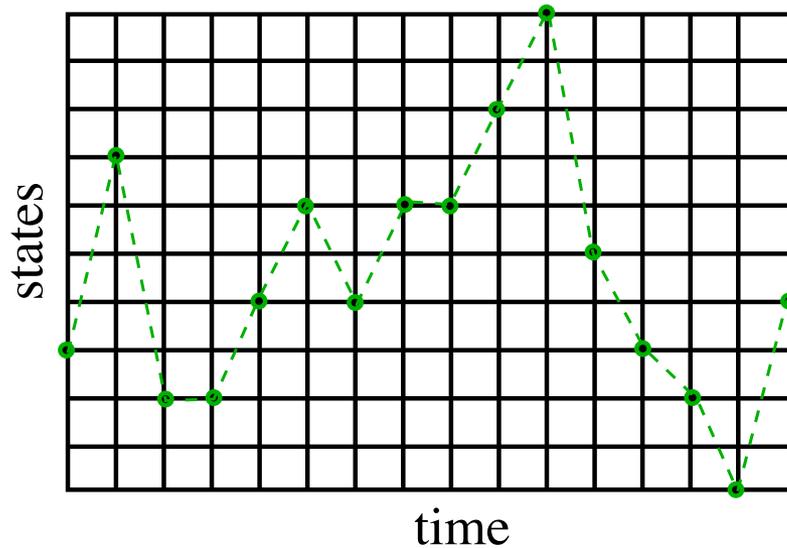
- Observe:

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

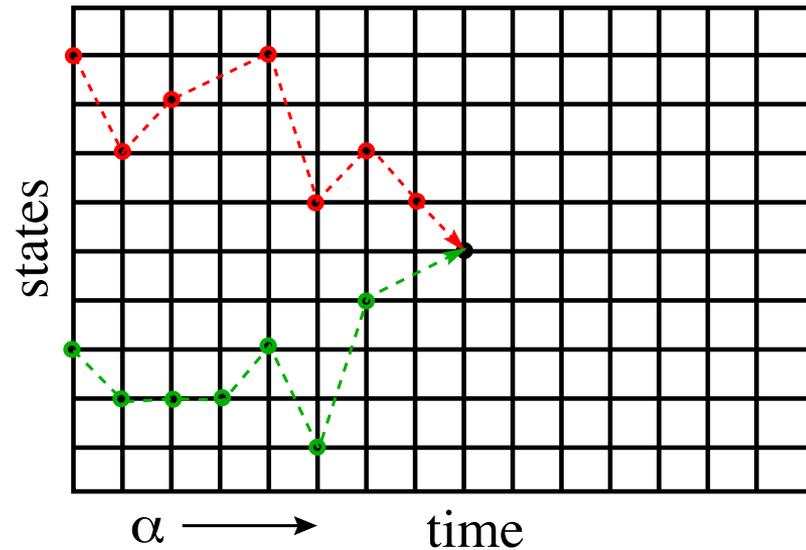
- This enables us to easily (cheaply) compute the desired likelihood.

# The Forward ( $\alpha$ ) Recursion

- The forward recursion:



Exponentially many paths.

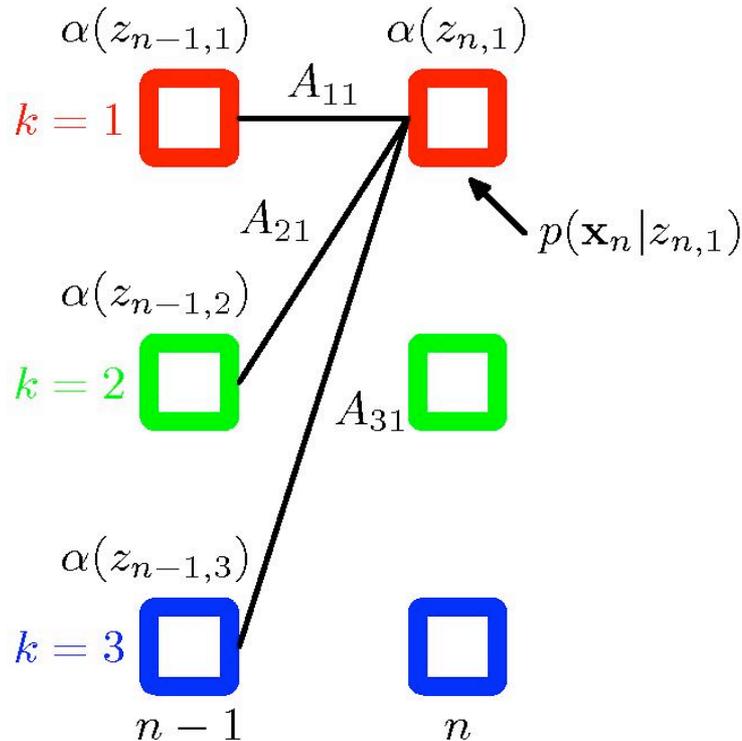


At each node, sum up the values of all incoming paths.

- This is exactly **dynamic programming**.

# The Forward ( $\alpha$ ) Recursion

- Illustration of the forward recursion



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

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

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

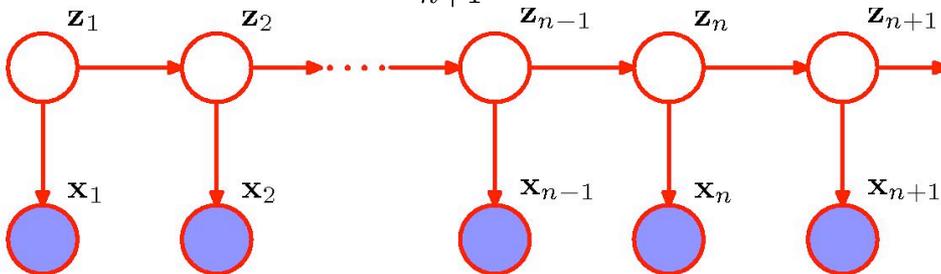
- The initial condition is given by:

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

# The Backward ( $\beta$ ) Recursion

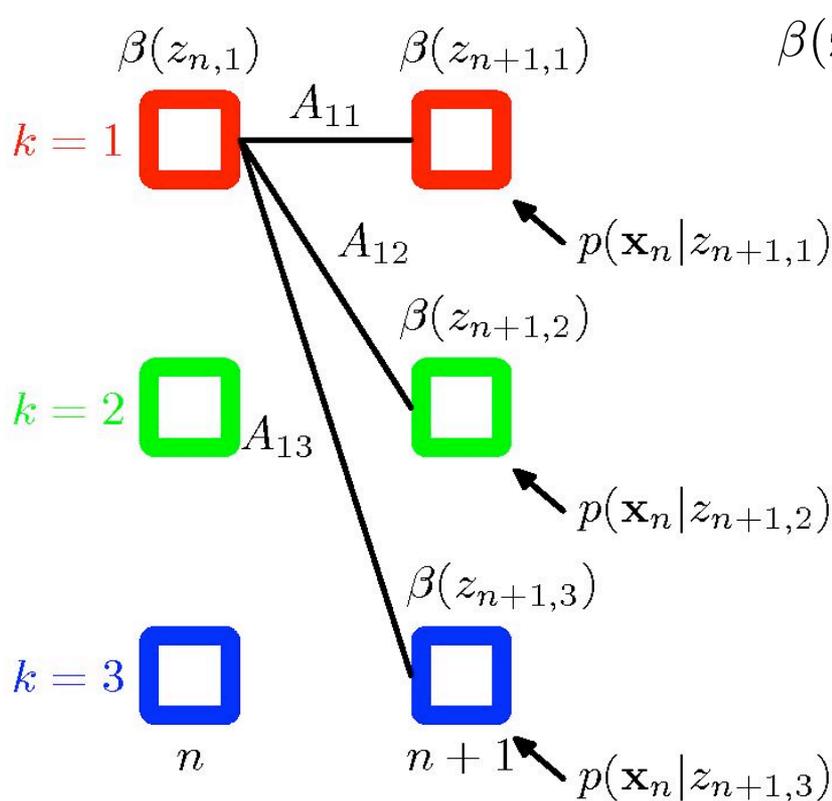
- There is also a simple recursion for  $\beta(\mathbf{z}_n)$ :

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



# The Backward ( $\beta$ ) 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:

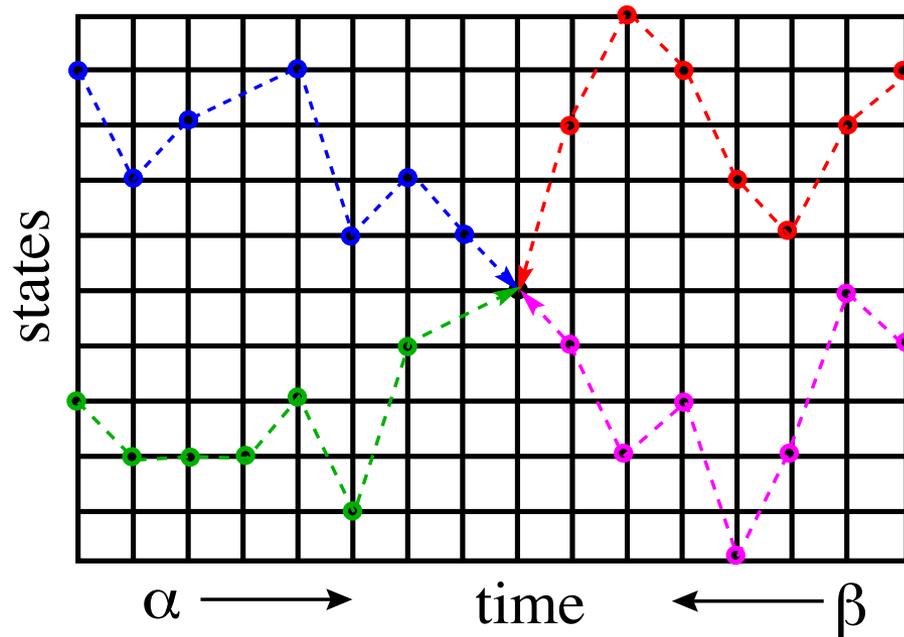
$$\begin{aligned} p(\mathbf{z}_N | \mathbf{X}) &= \frac{\alpha(\mathbf{z}_N) \beta(\mathbf{z}_N)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{X}, \mathbf{z}_N) \beta(\mathbf{z}_N)}{p(\mathbf{X})}. \end{aligned}$$

- Hence:

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

# The Backward ( $\beta$ ) 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  $\alpha(z_n)$  and one backward pass to compute all the  $\beta(z_n)$  and then compute any  $\gamma(z_n)$  we want. Total cost is  $O(K^2N)$ .

# 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  $\alpha - \beta$  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  $\beta(\mathbf{z}_n) = 1$ .

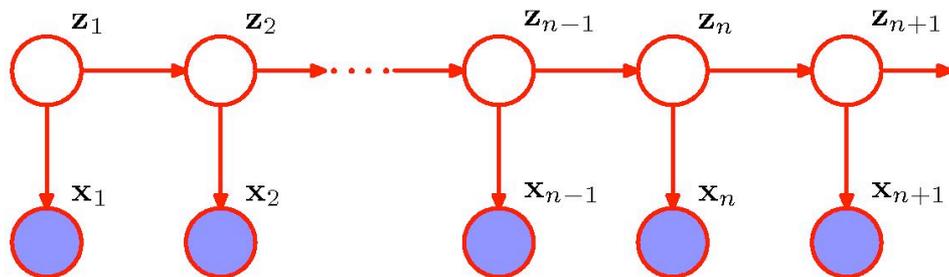
- This is a good way to **check your code!**

# Two-Frame Inference

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

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

- This is a  $K \times K$  matrix with elements  $\xi(i,j)$  representing the **expected number of transitions from state  $i$  to state  $j$  that begin at time  $n-1$** , given all the observations.



- It can be computed with the same  $\alpha$  and  $\beta$  recursions.

# EM algorithm

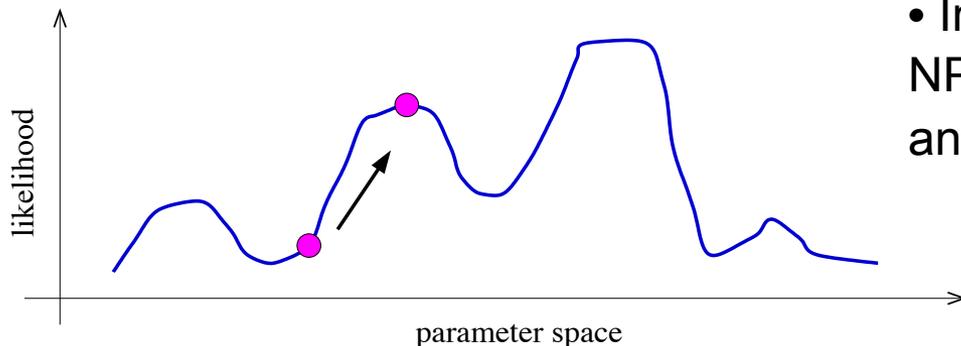
- **Intuition**: if only we knew the true state path then ML parameter estimation would be trivial.
- E-step: Compute the posterior distribution over the state path using  $\alpha - \beta$  recursion (dynamic programming):

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

- 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 and convergence is hard to tell.

# Complete Data Log-likelihood

- Complete data log-likelihood takes form:

$$\begin{aligned}
 \log p(\mathbf{X}, \mathbf{Z} | \theta) &= \log \left[ p(\mathbf{z}_1 | \boldsymbol{\pi}) \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}, A) \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n, \phi) \right] \\
 &= \log \left[ \prod_{k=1}^K \pi_k^{z_{1k}} \prod_{n=2}^N \prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1,j} z_{nk}} \prod_{n=1}^N \prod_{k=1}^K p(\mathbf{x}_n | \mathbf{z}_n)^{z_{nk}} \right] \\
 &= \sum_{k=1}^K z_{1k} \log \pi_k + \sum_{n=2}^N \sum_{k=1}^K \sum_{j=1}^K [z_{n-1,j} z_{nk}] \log A_{jk} + \sum_{n=1}^N \sum_{k=1}^K z_{nk} \log p(\mathbf{x}_n | \mathbf{z}_n).
 \end{aligned}$$



transition model

observation model

- Statistics we need from the E-step are:

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

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

# Expected Complete Data Log-likelihood

- The complete data log-likelihood takes form:

$$\begin{aligned} Q(\theta, \theta^{old}) &= \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \theta^{old}) \log p(\mathbf{X}, \mathbf{Z}|\theta). \\ &= \sum_{k=1}^K \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^N \sum_{k=1}^K \sum_{j=1}^K \xi(z_{n-1,j}, z_{nk}) \log A_{jk} + \sum_{n=1}^N \sum_{k=1}^K \gamma_{nk} \log p(\mathbf{x}_n | \mathbf{z}_n). \end{aligned}$$

- 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:  $\theta = \{\pi, A, \phi\}$ .

# Parameter Estimation

- **Initial state distribution**: expected number of times in state  $k$  at time 1:

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

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

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

so the estimated **transition probabilities** are:

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

- The EM algorithm must be initialized by choosing starting values for  $\pi$  and  $\mathbf{A}$ .
- Note that any elements of  $\pi$  or  $\mathbf{A}$  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 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:

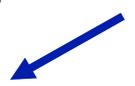
$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}).$$

- And the corresponding M-step updates:

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

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

Same as fitting a Gaussian mixture model.

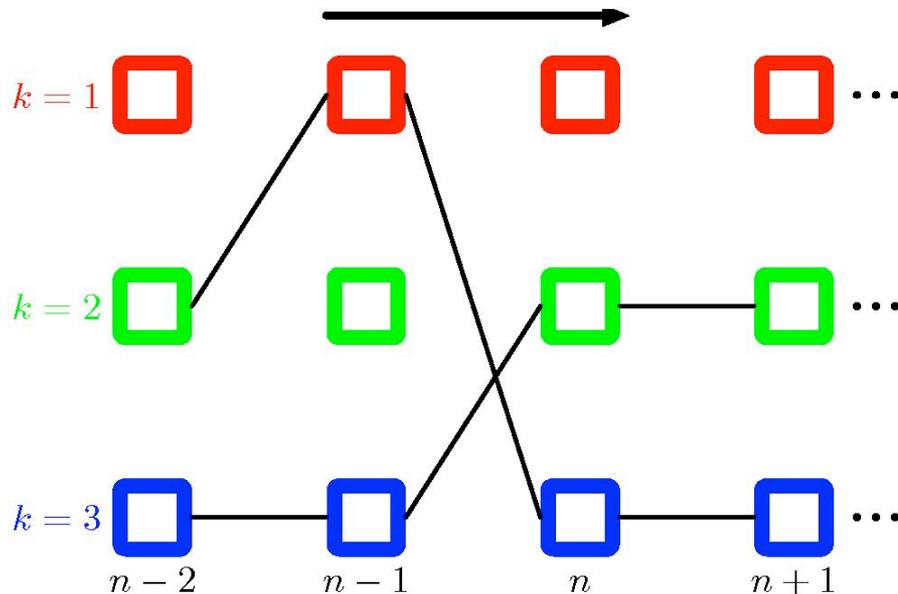


# Viterbi Decoding

- The numbers  $\gamma(z_n)$  above gave the **probability distribution over all states** at any time.
- By choosing the state  $\gamma^*(z_n)$  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 **Bellman’s 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.
- Remember the max-sum algorithm for **tree structured graphical models**.

# Viterbi Decoding

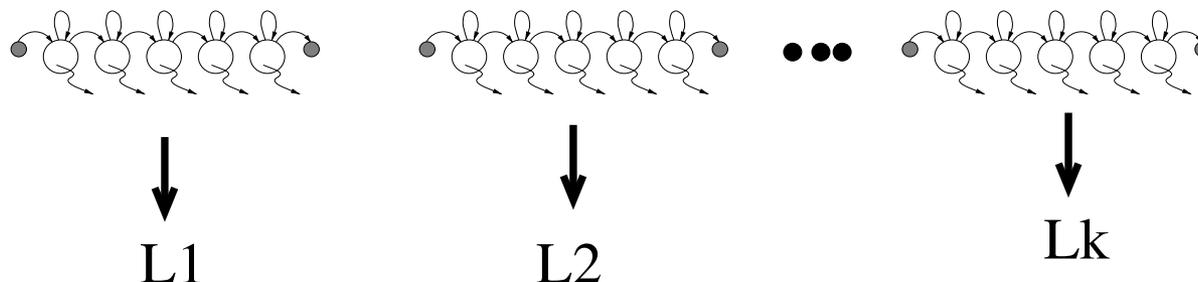
- A fragment of the HMM lattice showing two possible paths:



- **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

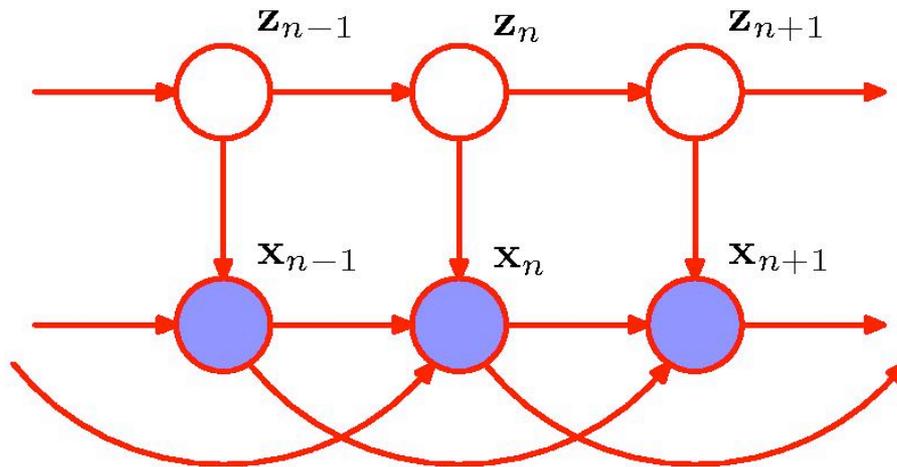
- We can use HMMs for recognition by:
  - training one HMM for each class (requires **labeled training data**)
  - evaluating probability of an unknown sequence under each HMM
  - classifying unknown sequence by choosing an HMM with highest likelihood



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

# Autoregressive HMMs

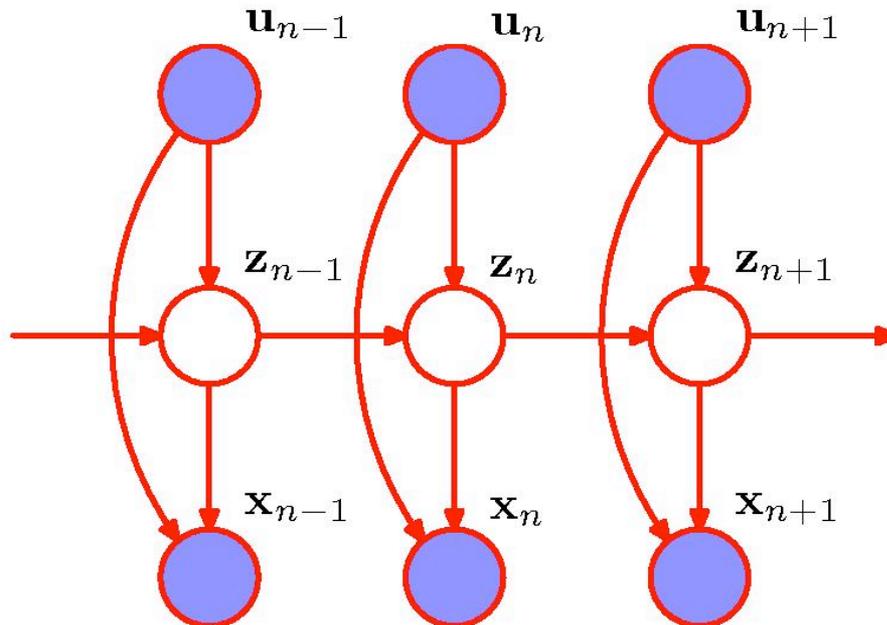
- One limitation of the standard HMM is that it is poor at capturing long-range correlations between observations, as these have to be mediated via the first order Markov chain of hidden states.



- **Autoregressive HMM:** The distribution over  $x_n$  depends 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.

# Input-Output HMMs

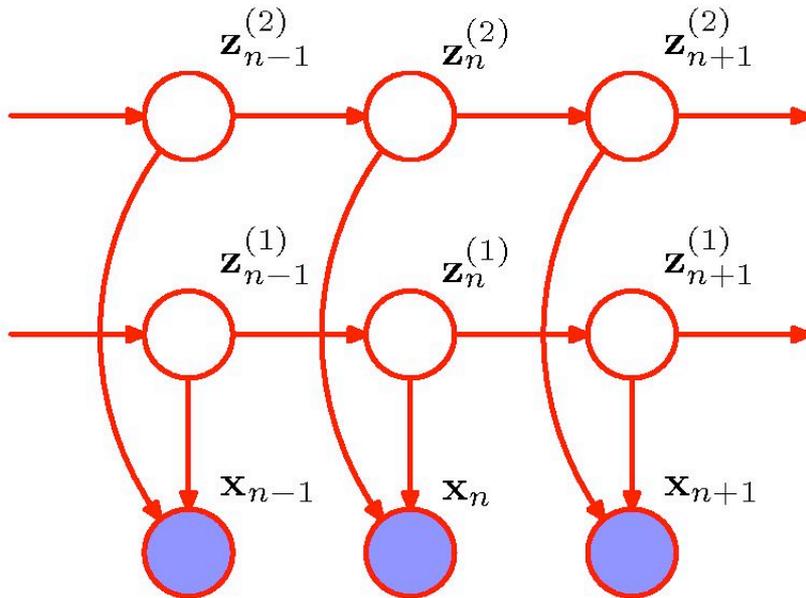
- Both the emission probabilities and the transition probabilities depend on the values of a sequence of observations  $u_1, \dots, u_N$ .



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

# Factorial HMMs

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

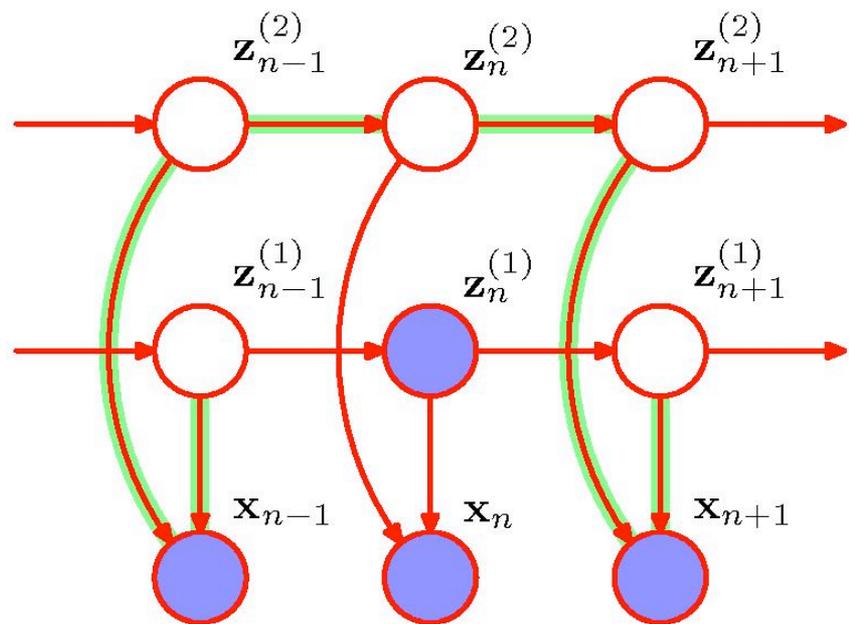


- **Motivation:** In order to represent 10 bits of information at a given time step, a standard HMM would need  $K=2^{10}=1024$  states.
- Factorial HMMs would use 10 binary chains.
- Much more powerful model.

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

# Factorial HMMs

- The conditional independence property:  $z_{n+1} \perp z_{n-1} \mid z_n$  does not hold for the individual latent chains.



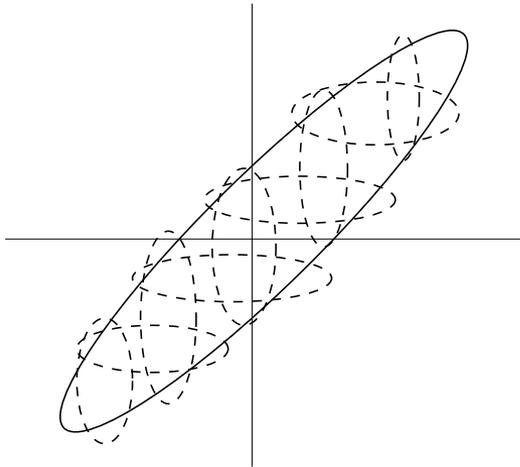
- There is no efficient exact E-step for this model.

- One solution would be to use MCMC techniques to obtain 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**).

# Regularizing HMMs

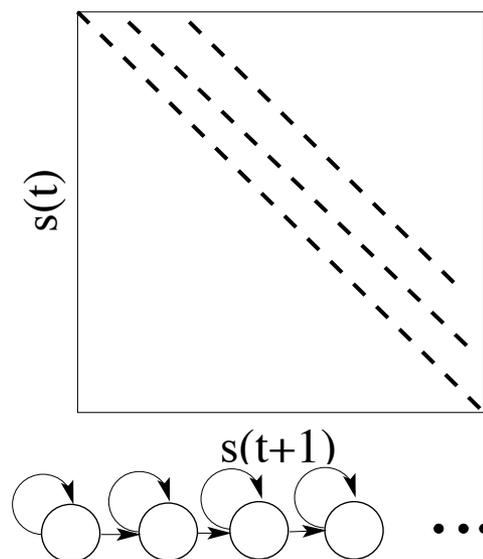
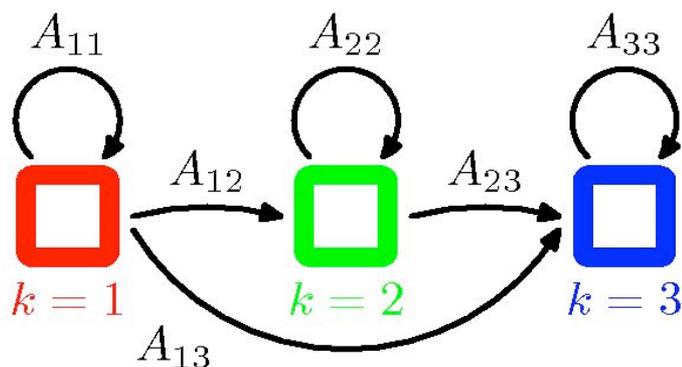
- There are **two problems**:
  - for high dimensional outputs, **lots of parameters in the emission model**
  - with many states, **transition matrix has many (squared) elements**
- First problem: full covariance matrices in high dimensions or discrete symbol models with many symbols have lots of parameters. To estimate these accurately requires a lot of training data.



- We can use **mixtures of diagonal covariance Gaussians**.
  - For discrete data, we can use **mixtures of base rates**.
- We can also tie parameters across states.

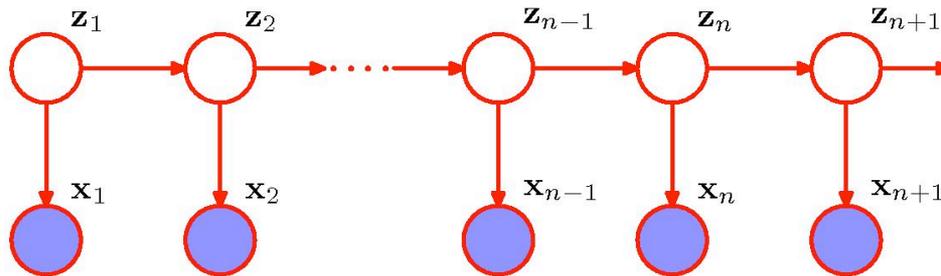
# Regularizing Transition Matrices

- One way to regularize large transition matrices is to constrain them to be **sparse**: instead of being allowed to transition to any other state, each state has only a few possible successor states.
- A very effective way to constrain the transitions is to order the states in the HMM and allow **transitions only to states that come later in the ordering**.
- Such models are known as “**linear HMMs**”, “**chain HMMs**” or “**left-to-right HMMs**”. Transition matrix is upper-diagonal (usually only has a few bands).



# Linear Dynamical Systems

- In HMMs, latent variables are discrete but with arbitrary emission probability distributions.



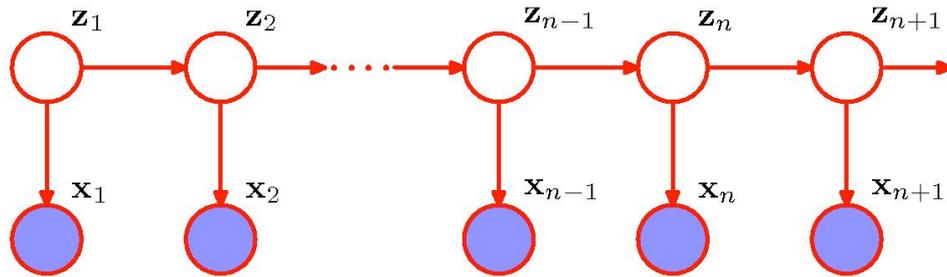
- We now consider a **linear-Gaussian state-space model**, so that latent variables and observed variables are multivariate Gaussian distributions.
- An HMM can be viewed as an extension of the mixture models to allow for sequential correlations in the data
- Similarly, the linear dynamical system (LDS) can be viewed as a **generalization of the continuous latent variable models**, such as probabilistic PCA.

# Linear Dynamical Systems

- The model is represented by a tree-structured directed graph, so inference can be solved efficiently using the sum-product algorithm.
- The forward recursions, analogous to the  $\alpha$ -messages of HMMs are known as the **Kalman filter equations**.
- The backward recursions, analogous to the  $\beta$ -messages, are known as the **Kalman smoother equations**.
- The Kalman filter is used in many real-time tracking applications.
- Because the LDS is a linear-Gaussian model, the joint distribution over all variables, as well as marginals and conditionals, will be Gaussian.
- This leads to **tractable inference and learning**.

# The Model

- We can write the transition and emission distributions in the general form:



$$p(\mathbf{z}_n | \mathbf{z}_{n-1}) = \mathcal{N}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}, \mathbf{\Gamma}),$$

$$p(\mathbf{x}_n | \mathbf{z}_n) = \mathcal{N}(\mathbf{x}_n | \mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}),$$

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1 | \boldsymbol{\mu}_0, \mathbf{V}_0).$$

- These can be expressed in terms of **noisy linear equations**:

$$\mathbf{z}_n = \mathbf{A}\mathbf{z}_{n-1} + \mathbf{w}_n, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{w} | \mathbf{0}, \mathbf{\Gamma}),$$

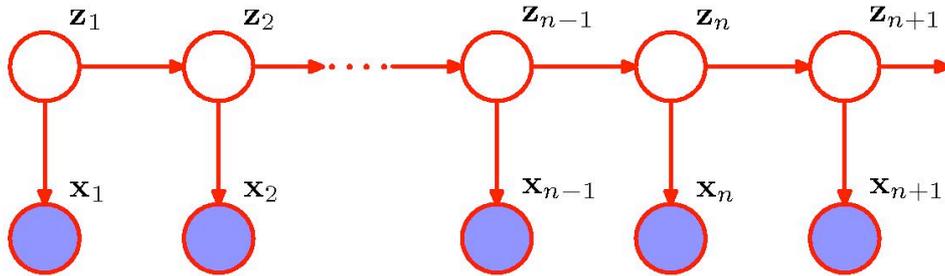
$$\mathbf{x}_n = \mathbf{C}\mathbf{z}_n + \mathbf{v}_n, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{v} | \mathbf{0}, \mathbf{\Sigma}),$$

$$\mathbf{z}_1 = \boldsymbol{\mu}_0 + \mathbf{u}, \quad \mathbf{u} \sim \mathcal{N}(\mathbf{u} | \mathbf{0}, \mathbf{V}_0).$$

- Model parameters  $\theta = \{\mathbf{A}, \mathbf{\Gamma}, \mathbf{C}, \mathbf{\Sigma}, \boldsymbol{\mu}_0, \mathbf{V}_0\}$  can be learned using EM algorithm (similar to standard HMM case).

# Inference in LDS

- Consider forward equations. The initial message is Gaussian, and since each of the factors is Gaussian, **all subsequent messages will also be Gaussians**.



$$\begin{aligned} \mathbf{z}_n &= \mathbf{A}\mathbf{z}_{n-1} + \mathbf{w}_n, & \mathbf{w} &\sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{\Gamma}), \\ \mathbf{x}_n &= \mathbf{C}\mathbf{z}_n + \mathbf{v}_n, & \mathbf{v} &\sim \mathcal{N}(\mathbf{v}|\mathbf{0}, \mathbf{\Sigma}), \\ \mathbf{z}_1 &= \boldsymbol{\mu}_0 + \mathbf{u}, & \mathbf{u} &\sim \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{V}_0). \end{aligned}$$

- Similar to HMMs, let us define the normalized version of  $\alpha(\mathbf{z}_n)$ :

$$\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_N) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \dots, \mathbf{x}_n)} = \mathcal{N}(\mathbf{z}_n | \boldsymbol{\mu}_n, \mathbf{V}_n).$$

Remember: for HMMs

- Using forward recursion, we get:

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

$$c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n | \mathbf{z}_n) \int \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n | \mathbf{z}_{n-1}) d\mathbf{z}_{n-1}$$

$$= \mathcal{N}(\mathbf{x}_n | \mathbf{C}\mathbf{z}_n, \mathbf{\Sigma}) \int \mathcal{N}(\mathbf{z}_{n-1} | \boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1}) \mathcal{N}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}, \mathbf{\Gamma}) d\mathbf{z}_{n-1}.$$

# Inference in LDS

- Hence we obtain:

$$\begin{aligned}c_n \hat{\alpha}(\mathbf{z}_n) &= p(\mathbf{x}_n | \mathbf{z}_n) \int \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n | \mathbf{z}_{n-1}) d\mathbf{z}_{n-1} \\ &= \mathcal{N}(\mathbf{x}_n | \mathbf{C}\mathbf{z}_n, \Sigma) \int \mathcal{N}(\mathbf{z}_{n-1} | \boldsymbol{\mu}_{n-1}, \mathbf{V}_{n-1}) \mathcal{N}(\mathbf{z}_n | \mathbf{A}\mathbf{z}_{n-1}, \Gamma) d\mathbf{z}_{n-1}.\end{aligned}$$

in which case  $\alpha(\mathbf{z}_n)$  is Gaussian:

$$c_n \hat{\alpha}(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n | \boldsymbol{\mu}_n, \mathbf{V}_n).$$

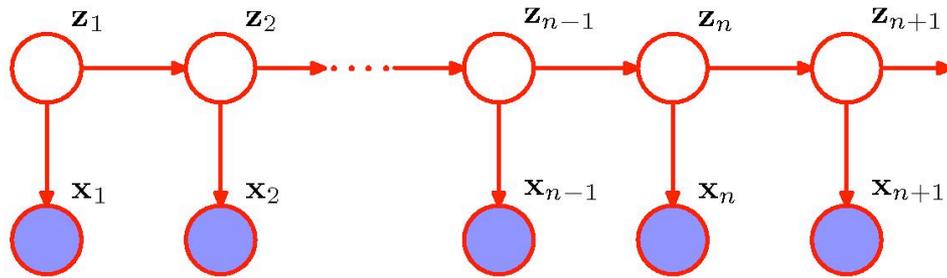
$$\begin{aligned}\boldsymbol{\mu}_n &= \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{K}_n(\mathbf{x}_n - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}) \\ \mathbf{V}_n &= (\mathbf{I} - \mathbf{K}_n\mathbf{C})\mathbf{P}_{n-1} \\ \mathbf{P}_{n-1} &= \mathbf{A}\mathbf{V}_{n-1}\mathbf{A}^T + \Gamma.\end{aligned}$$

and we have also defined the **Kalman gain matrix**:

$$\mathbf{K}_n = \mathbf{P}_{n-1}\mathbf{C}^T (\mathbf{C}\mathbf{P}_{n-1}\mathbf{C}^T + \Sigma)^{-1}.$$

# Kalman Filter

- Let us examine the evolution of the mean:



$$\mathbf{z}_n = \mathbf{A}\mathbf{z}_{n-1} + \mathbf{w}_n,$$

$$\mathbf{x}_n = \mathbf{C}\mathbf{z}_n + \mathbf{v}_n,$$

$$\mathbf{z}_1 = \boldsymbol{\mu}_0 + \mathbf{u},$$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \boldsymbol{\Gamma}),$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{v}|\mathbf{0}, \boldsymbol{\Sigma}),$$

$$\mathbf{u} \sim \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{V}_0).$$

Prediction of the mean over  $\mathbf{z}_n$ .

Predicted observation for  $x_n$ .

$$\boldsymbol{\mu}_n = \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{K}_n(\mathbf{x}_n - \mathbf{C}\mathbf{A}\boldsymbol{\mu}_{n-1}).$$

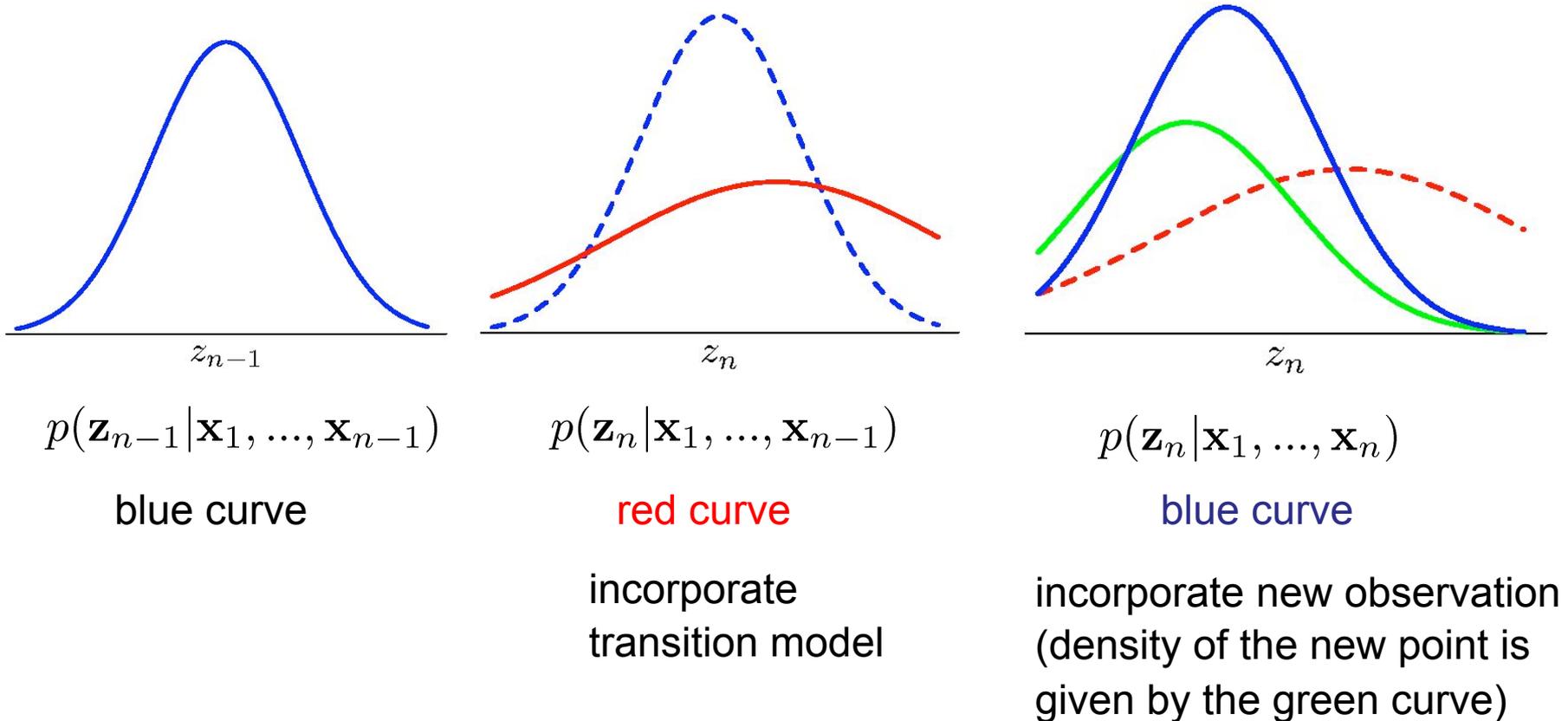
Predicted mean plus the correction term controlled by the Kalman gain matrix.

Error between the predicted observation  $x_n$  and the actual observation  $x_n$ .

- We can view the Kalman filter as a process of making subsequent predictions and then correcting these predictions in the light of the new observations.

# Kalman Filter

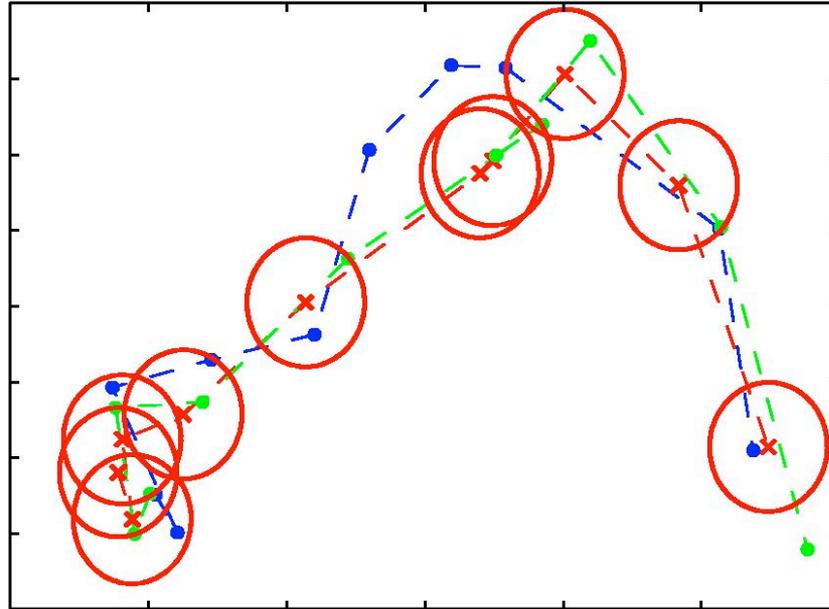
- Example:



- The new observation has shifted and narrowed the distribution compared to (see red curve)  $p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_{n-1})$ .

# Tracking Example

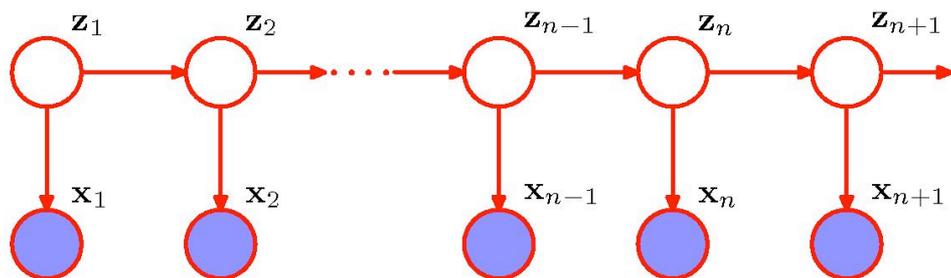
- LDS that is being used to track a moving object in 2-D space:



- **Blue points** indicate the true position of the object.
- **Green points** denote the noisy measurements.
- **Red crosses** indicate the means of the inferred posterior distribution of the positions inferred by the Kalman filter.

# Particle Filters

- For dynamical systems that are non-Gaussian (e.g. emission densities are non-Gaussian), we can use sampling methods to find a tractable solution to the inference problem.
- Consider a **class of distributions represented by the graphical model:**



- Suppose we observed  $\mathbf{X}_n = \{x_1, \dots, x_n\}$ , and we wish to approximate:

$$\begin{aligned}\mathbb{E}[f(\mathbf{z}_n)] &= \int f(\mathbf{z}_n) p(\mathbf{z}_n | \mathbf{X}_n) d\mathbf{z}_n \\ &= \int f(\mathbf{z}_n) p(\mathbf{z}_n | x_n, \mathbf{X}_{n-1}) d\mathbf{z}_n \\ &= \frac{\int f(\mathbf{z}_n) p(x_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{X}_{n-1}) d\mathbf{z}_n}{\int p(x_n | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{X}_{n-1}) d\mathbf{z}_n} \approx \sum_{l=1}^L w_n^{(l)} f(\mathbf{z}_n^{(l)}),\end{aligned}$$

# Particle Filters

- Hence

$$\mathbb{E}[f(\mathbf{z}_n)] = \int f(\mathbf{z}_n) p(\mathbf{z}_n | \mathbf{X}_n) d\mathbf{z}_n \approx \sum_{l=1}^L w_n^{(l)} f(\mathbf{z}_n^{(l)}), \quad \mathbf{z}_n^{(l)} \sim p(\mathbf{z}_n | \mathbf{X}_{n-1}),$$

with importance weights:

$$w_n^{(l)} = \frac{p(\mathbf{x}_n | \mathbf{z}_n^{(l)})}{\sum_{m=1}^L p(\mathbf{x}_n | \mathbf{z}_n^{(m)})}.$$

- Hence the posterior  $p(\mathbf{z}_n | \mathbf{X}_n)$  is represented by the set of **L samples together with the corresponding importance weights**.
- We would like to define a sequential algorithm.
- Suppose that a set of samples and weights have been obtained at time step  $n$ .
- We wish to find the set of new samples and weights at time step  $n+1$ .

# Particle Filters

- From our previous result, let  $f(\mathbf{z}_n) = p(\mathbf{z}_{n+1}|\mathbf{z}_n)$ ,

$$p(\mathbf{z}_{n+1}|\mathbf{X}_n) = \int p(\mathbf{z}_{n+1}|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{X}_n)d\mathbf{z}_n$$

$$\approx \sum_{l=1}^L w_n^{(l)} p(\mathbf{z}_{n+1}|\mathbf{z}_n^{(l)}), \quad \mathbf{z}_n^{(l)} \sim p(\mathbf{z}_n|\mathbf{X}_{n-1}),$$

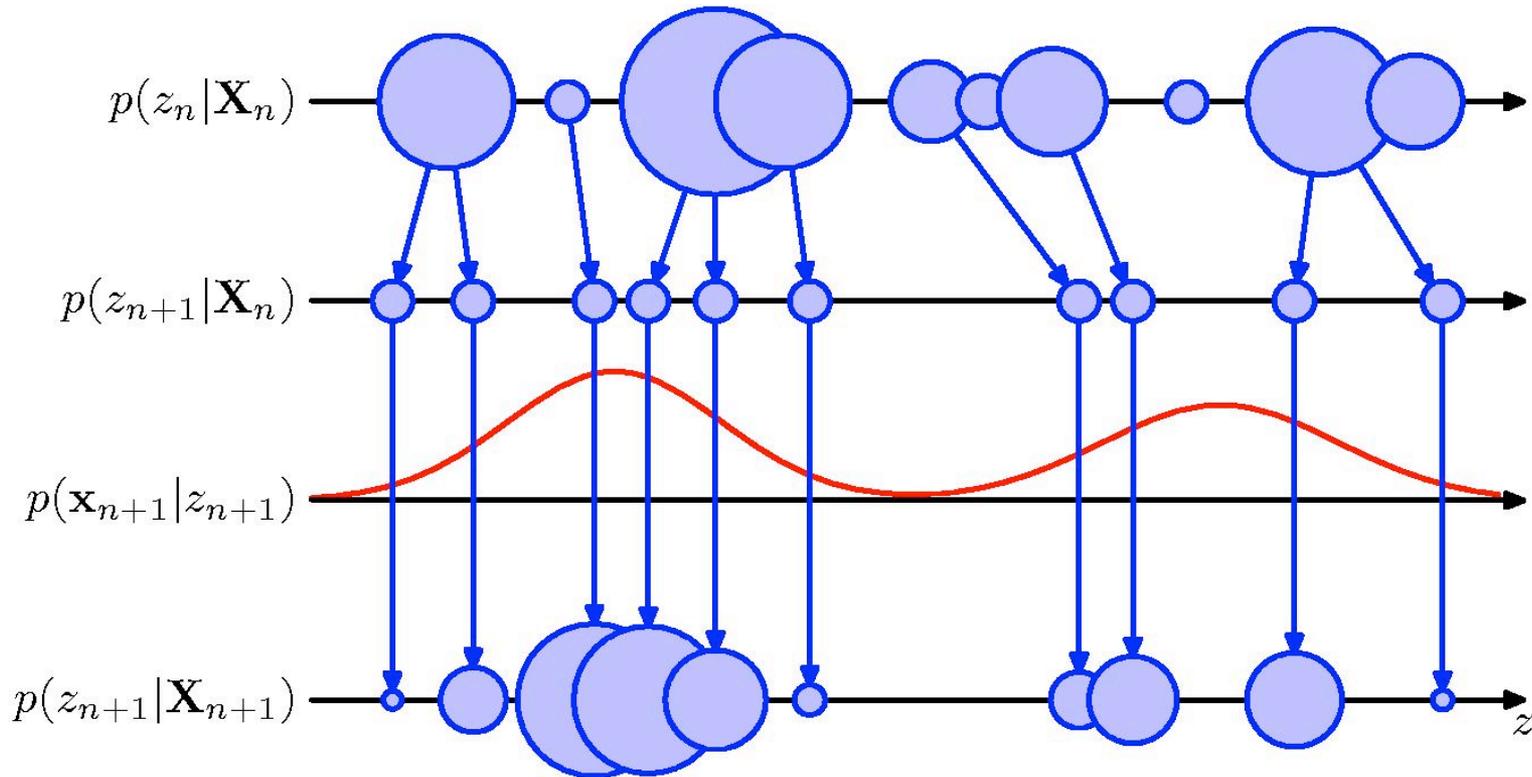
$$w_n^{(l)} = \frac{p(\mathbf{x}_n|\mathbf{z}_n^{(l)})}{\sum_{m=1}^L p(\mathbf{x}_n|\mathbf{z}_n^{(m)})}.$$

- Summary of the **particle filter algorithm**:

- At time  $n$ , we have a sample representation of the posterior distribution  $p(\mathbf{z}_n | \mathbf{X}_n)$  expressed as  $L$  samples with corresponding weights.
- We next draw  $L$  samples from the mixture distribution (above).
- For each sample, we then use the new observation to re-evaluate the weights:

$$w_{n+1}^{(l)} = \frac{p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}^{(l)})}{\sum_{m=1}^L p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}^{(m)})}.$$

# Example



- At time  $n$ , the posterior  $p(z_n | \mathbf{X}_n)$  is represented as a mixture distribution.
- We draw a set of  $L$  samples from this distribution (incorporating the transition model).
- The new weights evaluated by incorporating the **new observation**  $x_{n+1}$ .