

STAD68: Machine Learning

Russ Salakhutdinov

Department of Statistics

rsalakhu@utstat.toronto.edu

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

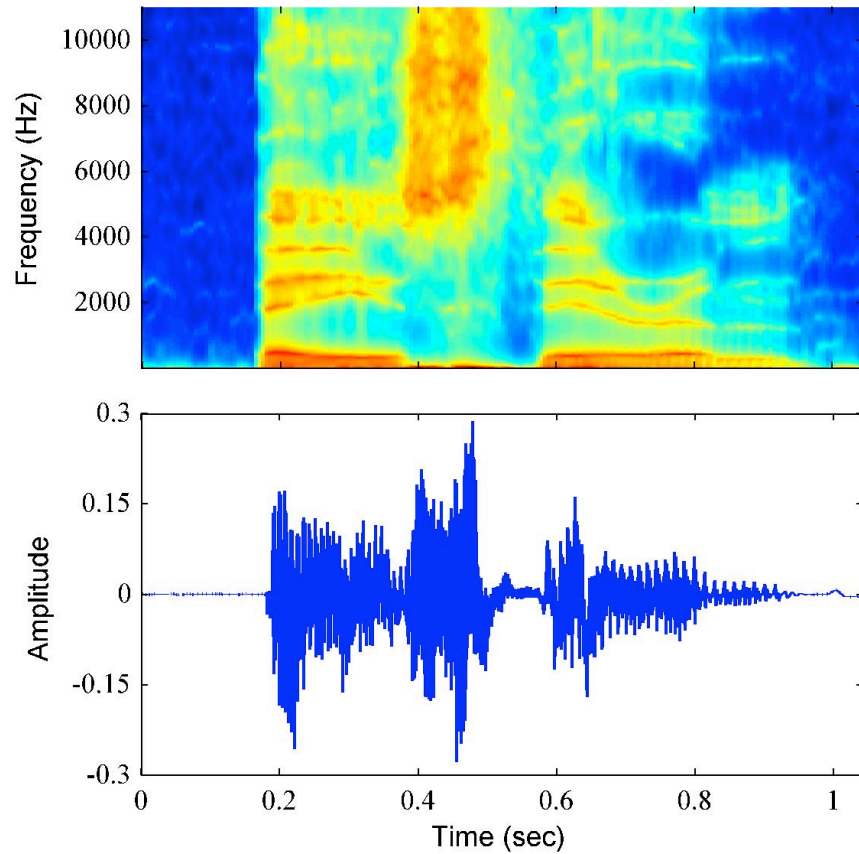
Sidney Smith Hall, Room 6002

Lecture 10

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

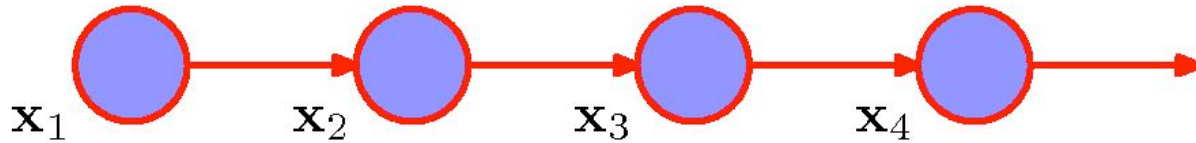


- Example of a spectrogram of a spoken word 'Bayes theorem':
- Successive observations are highly correlated.

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:

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

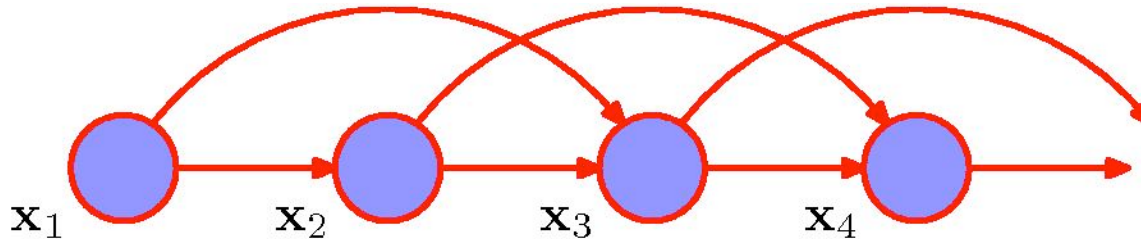
- From the **d-separation property**, the conditionals are given by:

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

- 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^{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^{th} order model:

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

- 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:

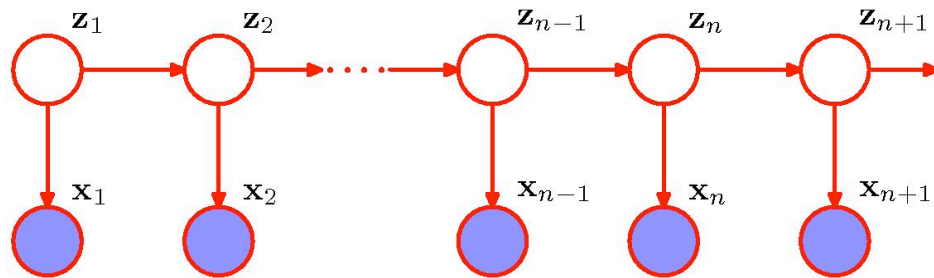
$$p(\mathbf{x}_n = m | \mathbf{x}_{n-1} = a, \mathbf{x}_{n-2} = b) = \alpha_{mab}.$$

- The **maximum likelihood** values for α 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!



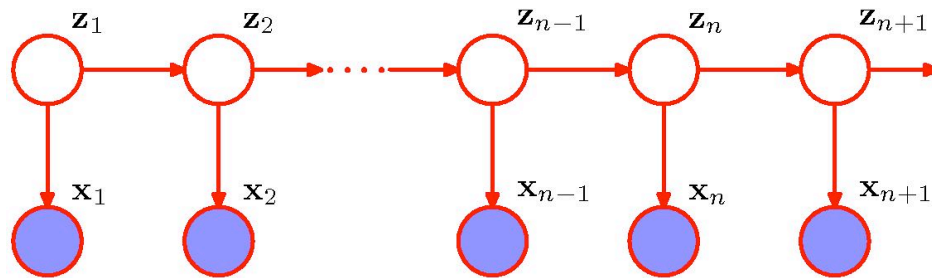
- Graphical structure known as the State Space Model.

- 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**):

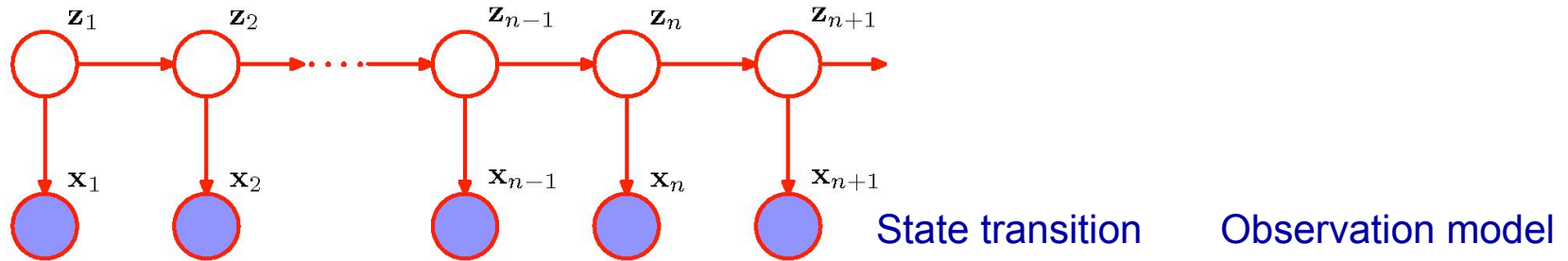
$$p(\mathbf{z}_n = k | \mathbf{z}_{n-1} = j) = A_{jk}, \quad p(\mathbf{z}_1 = k) = \pi_k.$$

- 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.

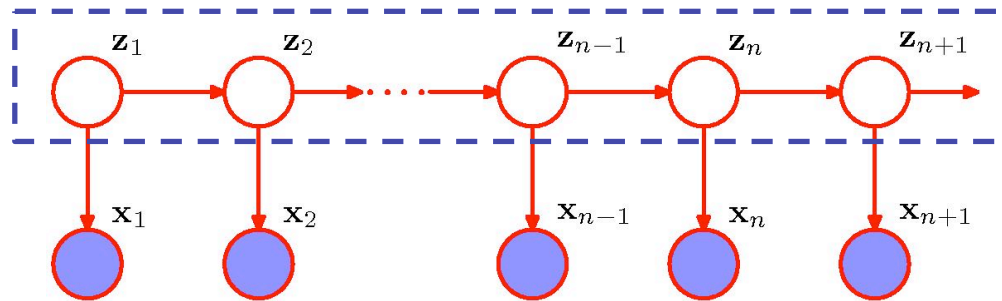
Conditional probability table if \mathbf{x} is discrete.



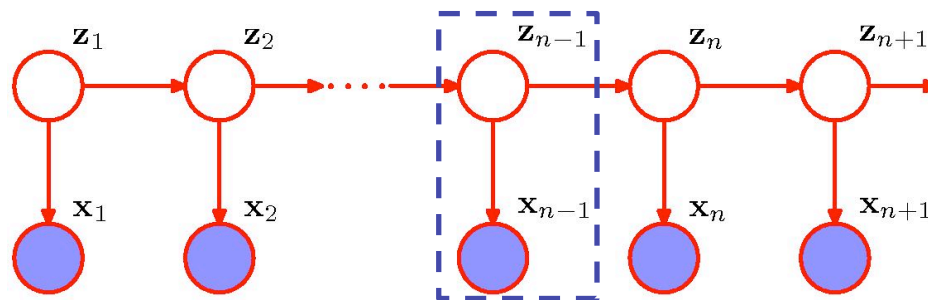
$$p(\mathbf{x}_1, \dots, \mathbf{x}_N, \mathbf{z}_1, \dots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}) \prod_{n=1}^N p(\mathbf{x}_n | \mathbf{z}_n).$$

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 \mathbf{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** $p(\mathbf{z}_n | \mathbf{z}_{n-1})$ is independent of \mathbf{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:

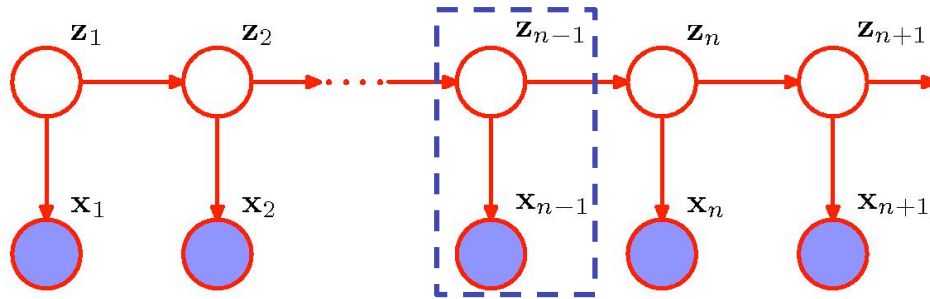
$$p(\mathbf{x}_n | \mathbf{z}_n, \phi) = \prod_{i=1}^D \prod_{k=1}^K \mu_{ik}^{x_{ni} z_{nk}}.$$

HMM Model Equations

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

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

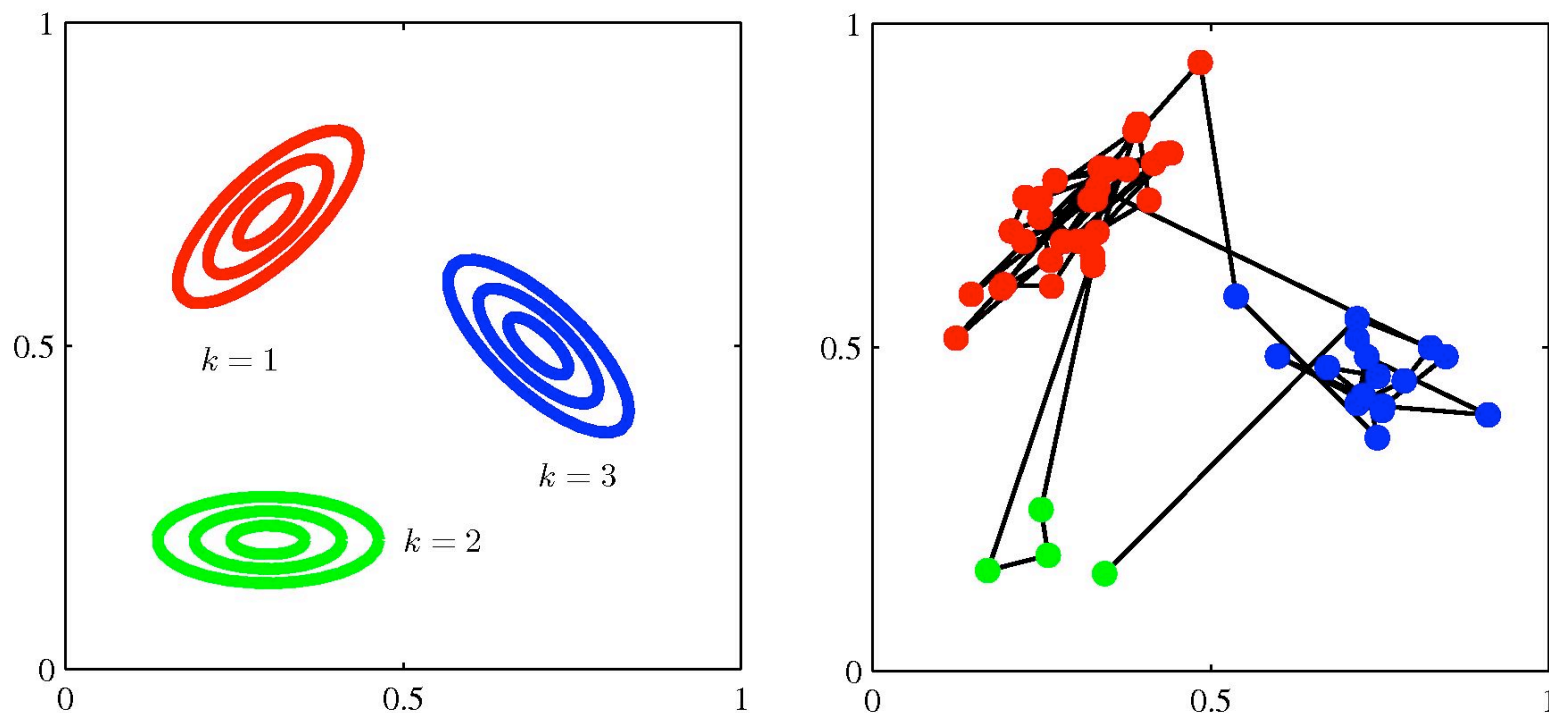
where $\theta = \{\boldsymbol{\pi}, A, \phi\}$ are the **model parameters**.



- 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:

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

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

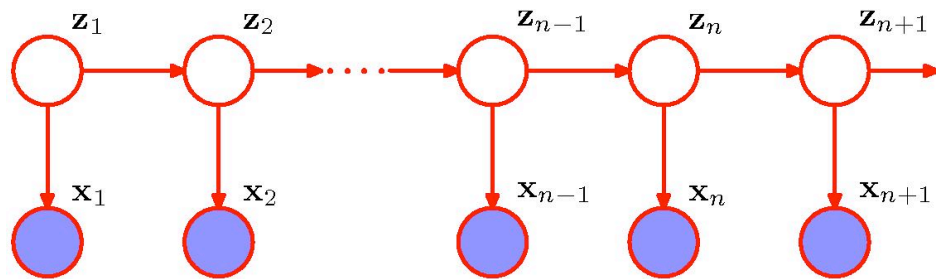
- In contrast to mixture models, the joint distribution $p(\mathbf{X}, \mathbf{Z} | \theta)$ **does not factorize over n**.
- **It looks hard**: N variables, each of which has K states. Hence N^K total paths.
- Remember inference problem on a simple chain.

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.

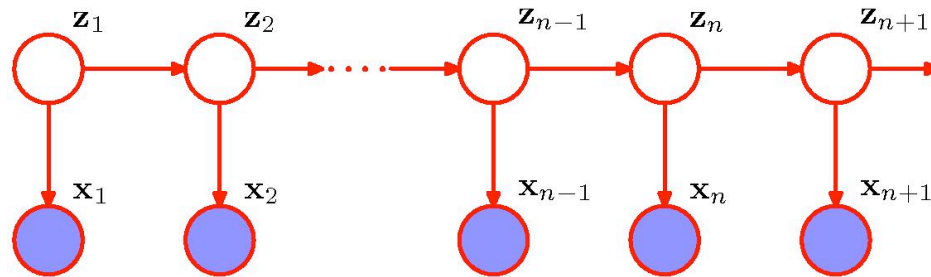
Inference of Hidden States

- 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.

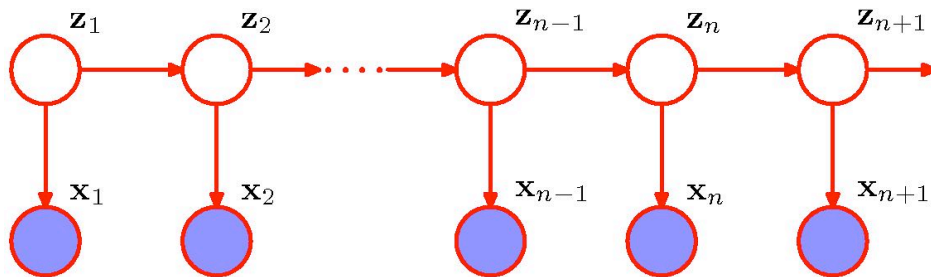
- Each $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$ represent a set of K numbers, one for each of the possible settings of the 1-of-K binary vector \mathbf{z}_n .
- We will derive efficient recursive algorithm, known as the **alpha-beta recursion**, or **forward-backward algorithm**.
- Relates to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

The Forward (α) 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 $O(K^2)$.



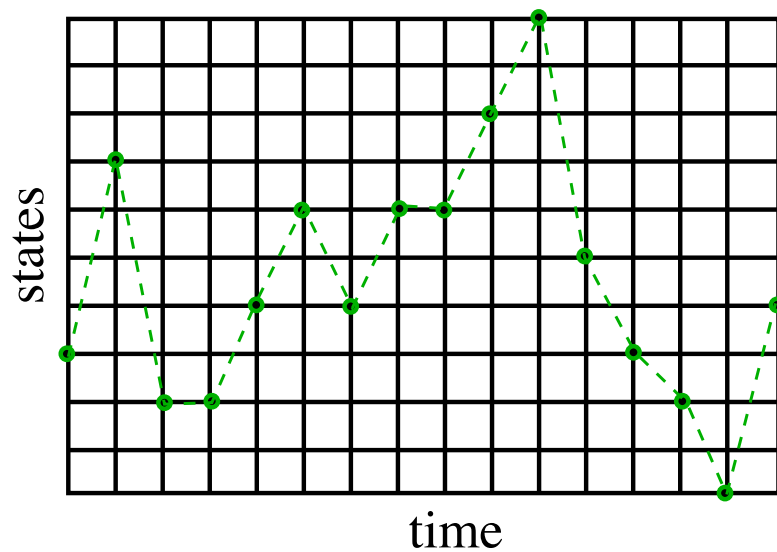
- Observe:

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

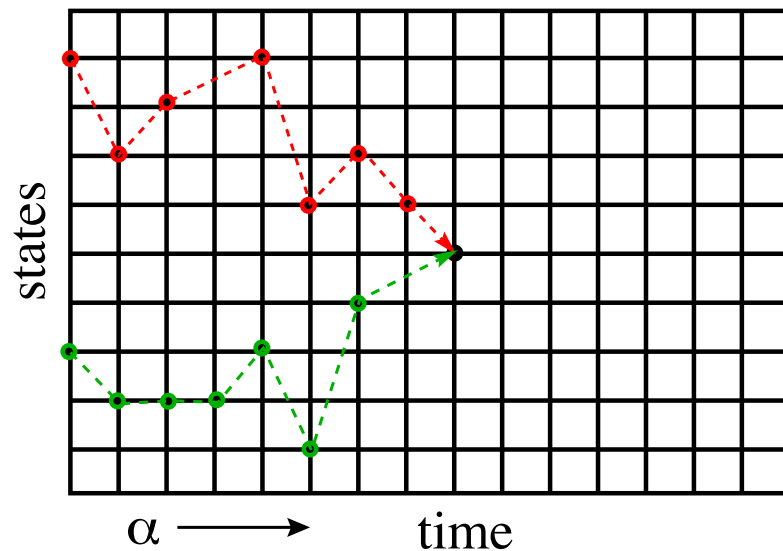
- This enables us to easily (cheaply) compute the desired likelihood.

The Forward (α) Recursion

- The forward recursion:



Exponentially many paths.

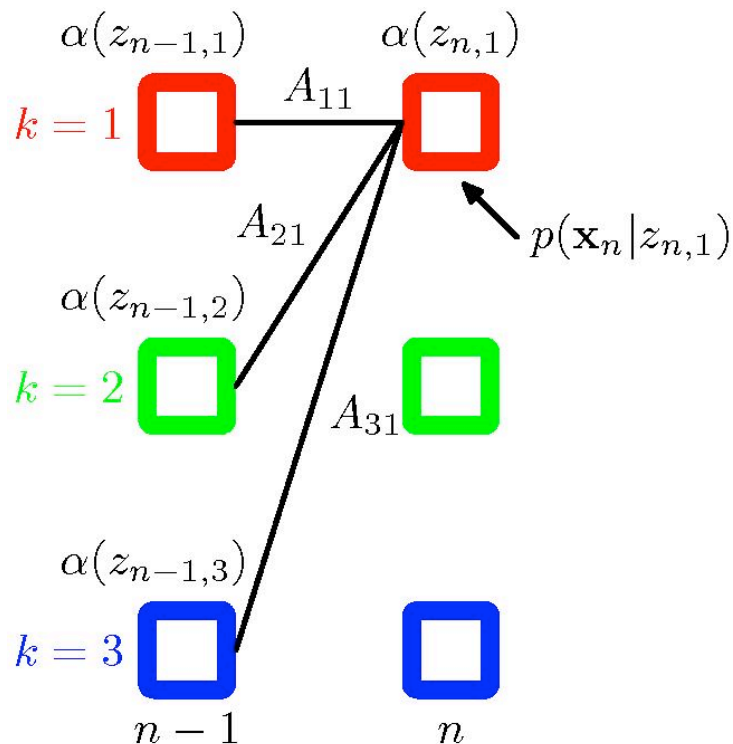


At each node, sum up the values of all incoming paths.

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

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

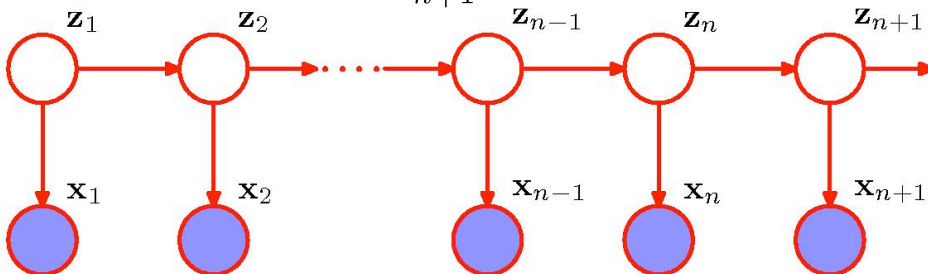
- The initial condition is given by:

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

The Backward (β) Recursion

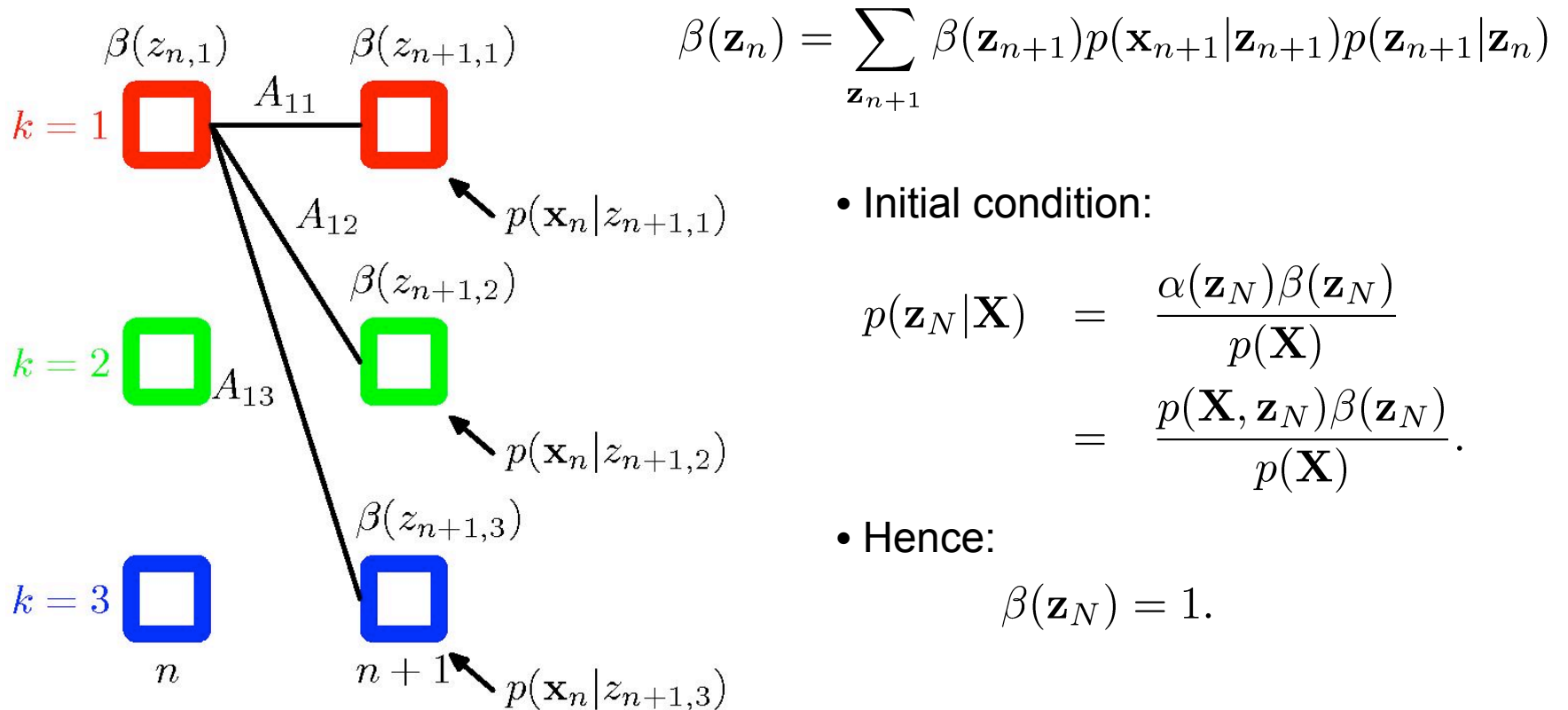
- There is also a simple recursion for $\beta(\mathbf{z}_n)$:

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



The Backward (β) Recursion

- Illustration of the backward recursion



- Initial condition:

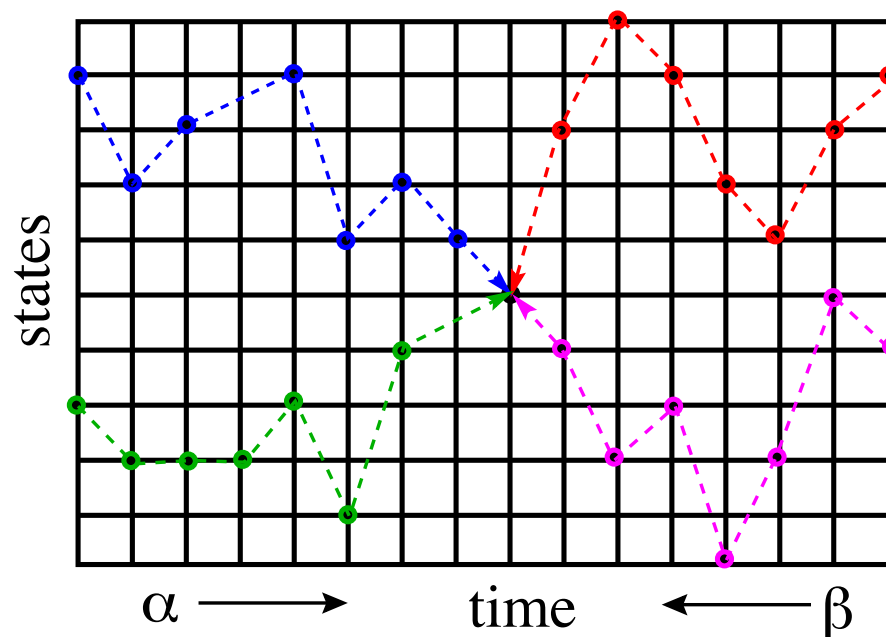
$$\begin{aligned} p(\mathbf{z}_N | \mathbf{X}) &= \frac{\alpha(\mathbf{z}_N) \beta(\mathbf{z}_N)}{p(\mathbf{X})} \\ &= \frac{p(\mathbf{X}, \mathbf{z}_N) \beta(\mathbf{z}_N)}{p(\mathbf{X})}. \end{aligned}$$

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

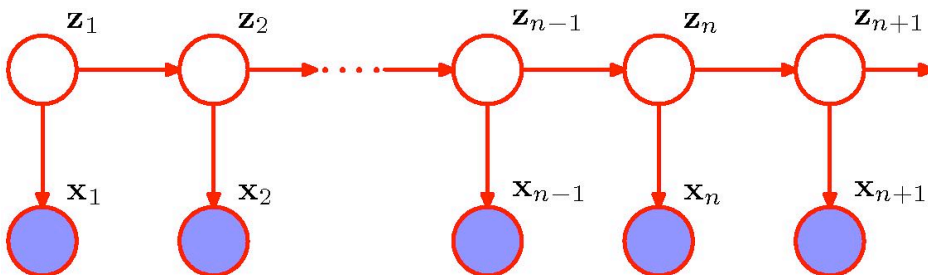
- This is a good way to **check your code!**

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 α and β recursions.

EM algorithm

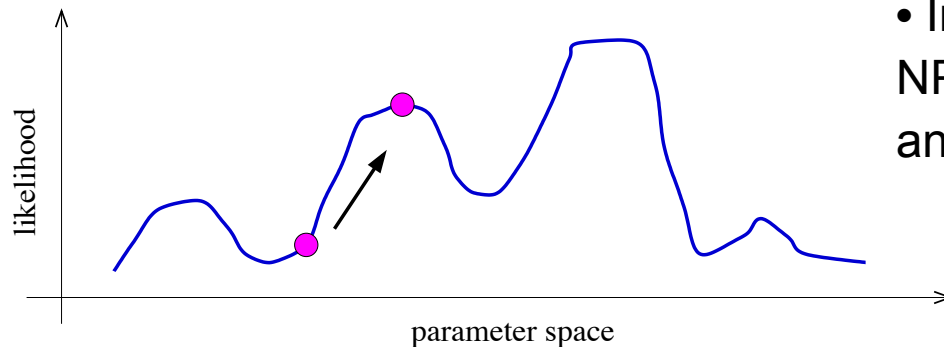
- **Intuition:** if only we knew the true state path then ML parameter estimation would be trivial.
- **E-step:** Compute the posterior distribution over the state path using $\alpha - \beta$ recursion (dynamic programming):

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

- **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:

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

- 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 π and \mathbf{A} .
- Note that any elements of π 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.

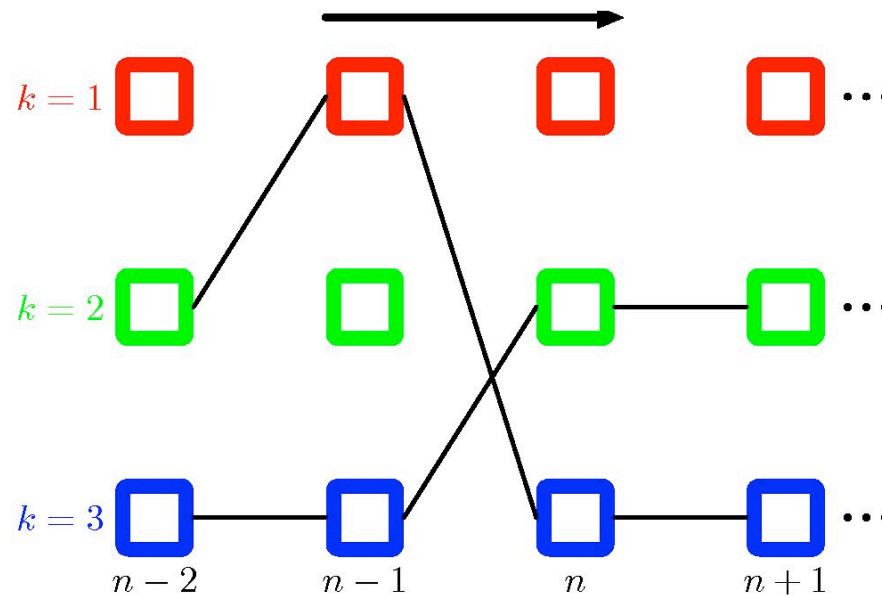


Viterbi Decoding

- 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.
- Relates to the max-sum algorithm for **tree structured graphical models**.

Viterbi Decoding

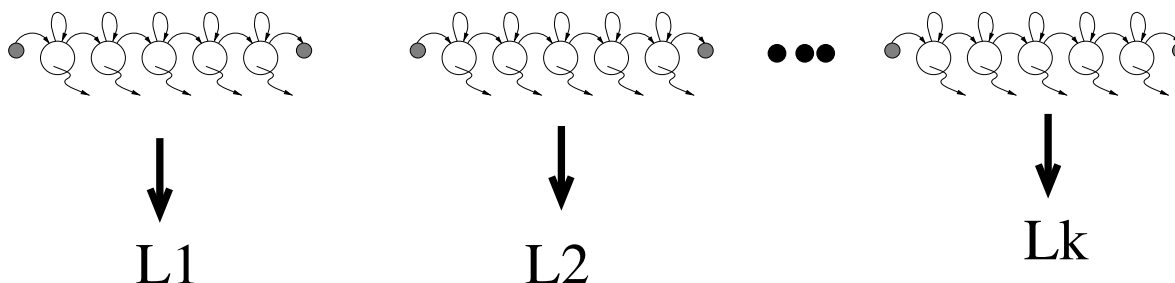
- A fragment of the HMM lattice showing two possible paths:



- **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

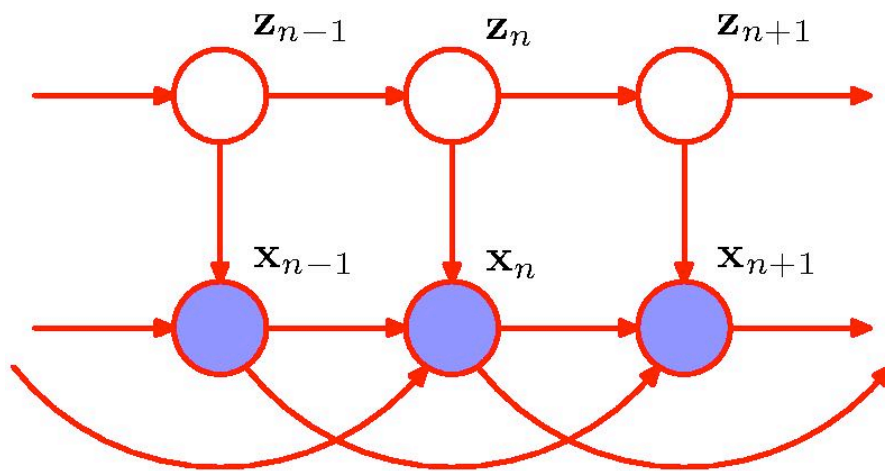
- We can use HMMs for recognition by:
 - training one HMM for each class (requires **labeled training data**)
 - evaluating probability of an unknown sequence under each HMM
 - classifying unknown sequence by choosing an HMM with highest likelihood



- 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

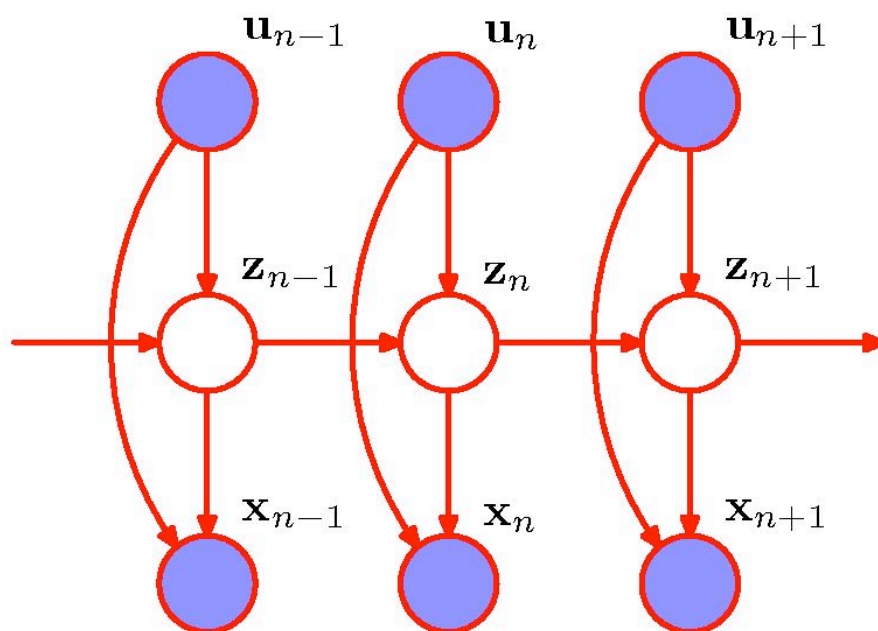
- One limitation of the standard HMM is that it is poor at capturing long-range correlations between observations, as these have to be mediated via the first order Markov chain of hidden states.



- **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

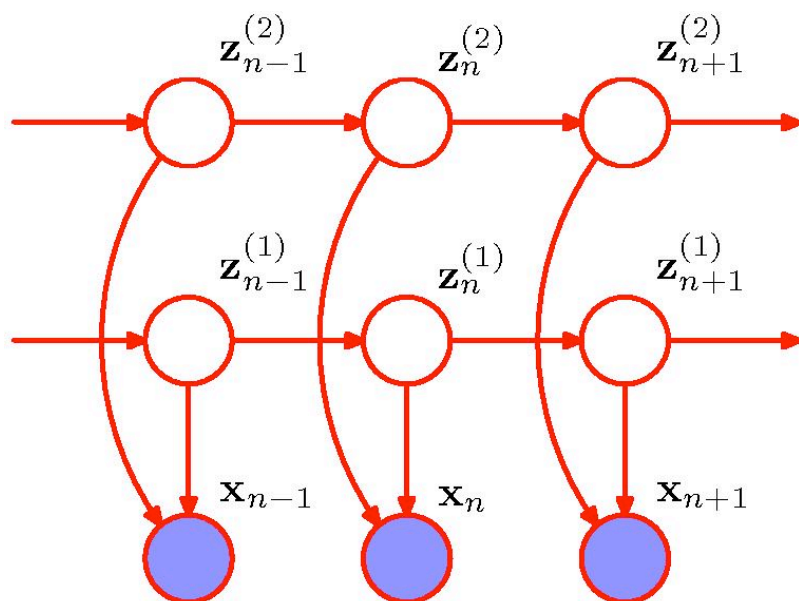
- Both the emission probabilities and the transition probabilities depend on the values of a sequence of observations u_1, \dots, u_N .



- 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:

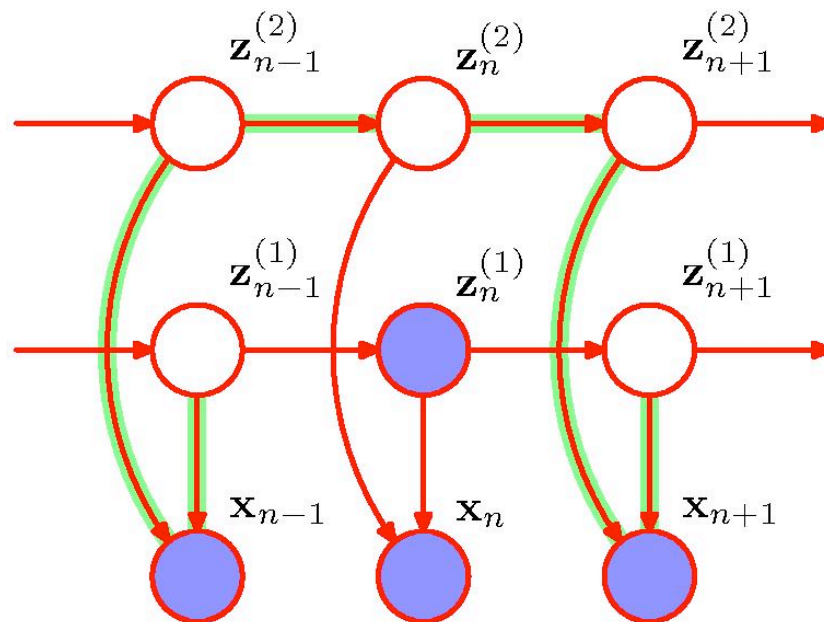


- **Motivation:** In order to represent 10 bits of information at a given time step, a standard HMM would need $K=2^{10}=1024$ states.
- Factorial HMMs would use 10 binary chains.
- Much more powerful model.

- The key disadvantage: **Exact inference is intractable.**
- Observing the \mathbf{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.

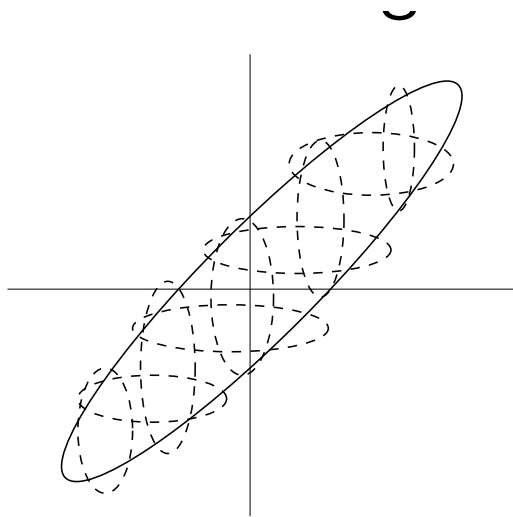


- There is no efficient exact E-step for this model.
- One solution would be to use MCMC techniques to obtain approximate sample from the posterior.

- 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.