

CS5340

Uncertainty Modeling in AI

Lecture 8: Hidden Markov Models (HMM)

Asst. Prof. Lee Gim Hee

AY 2020/21

Semester 1

Course Schedule

Week	Date	Topic	Remarks
1	12 Aug	Introduction to probabilistic reasoning	1830hrs: MS Teams (Live Introduction)
2	19 Aug	Bayesian networks (Directed graphical models)	
3	26 Aug	Markov random Fields (Undirected graphical models)	1830hrs: Zoom discussions
4	02 Sep	Variable elimination and belief propagation	Assignment 1: Belief propagation and maximal probability (15%)
5	09 Sep	Factor graph and the junction tree algorithm	
6	16 Sep	Parameter learning with complete data	Assignment 1: Due Assignment 2: Junction tree and parameter learning (15%) 1830hrs: Zoom discussions
-	23 Sep	Recess week	No lecture
7	30 Sep	Mixture models and the EM algorithm	Assignment 2: Due Online quiz 1 (20%)
8	07 Oct	Hidden Markov Models (HMM)	Assignment 3: Hidden Markov model (15%)
9	14 Oct	Monte Carlo inference (Sampling)	1830hrs: Zoom discussions
10	21 Oct	Variational inference	Assignment 3: Due Assignment 4: MCMC Sampling (15%)
11	28 Oct	Variational Auto-Encoder and Mixture Density Networks	
12	04 Nov	Graph-cut and alpha expansion	Assignment 4: Due 1830hrs: Zoom discussions
-	11 Nov	--	Online quiz 2 (20%)

Acknowledgements

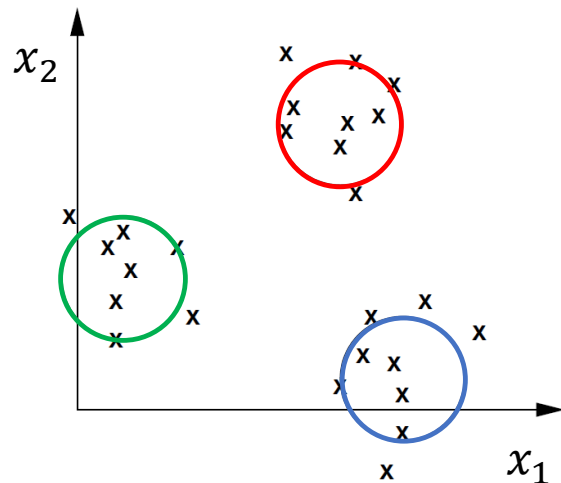
- A lot of slides and content of this lecture are adopted from:
 1. “Pattern Recognition and Machine Learning”, Christopher Bishop, Chapter 13.
 2. “Machine Learning – A Probabilistic Perspective”, Kevin Murphy, Chapter 17.
 3. “An Introduction to Probabilistic Graphical Models”, Michael I. Jordan, Chapters 12.
<http://people.eecs.berkeley.edu/~jordan/prelims/chapter12.pdf>
 4. “Computer Vision: Models, Learning and Inference”, Simon Prince, Chapter 11.

Learning Outcomes

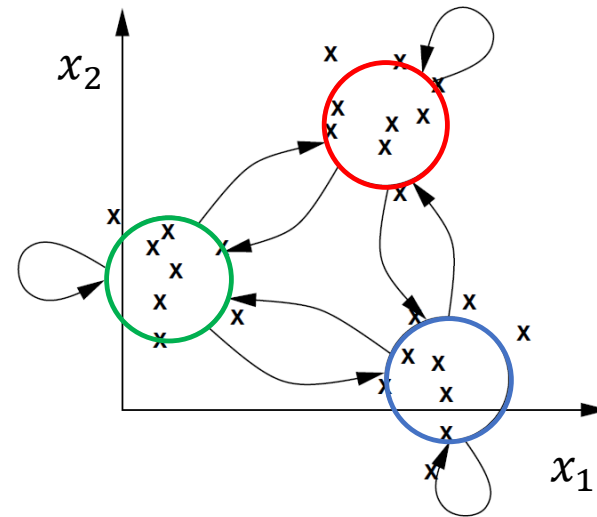
- Students should be able to:
 1. Describe the joint distribution of a HMM with the **transition and emission probabilities**.
 2. Use the **EM algorithm** for maximum likelihood estimation of the latent variables and unknown parameters in the HMM.
 3. Use the **forward-backward algorithm** to evaluate the EM algorithm.
 4. Apply the **Viterbi algorithm** to find the maximal probability and its configuration.

Sequential Data

Independent choice of mixture component



Choice of current mixture component is dependent on the previous observation



- In EM, we focused on mixture models, where the choice of mixture component for each observation is **independent**.
- HMM is an extension of the mixture model, where choice of mixture component **depends on** the choice of component for the **previous observation**.

Image Source: "An introduction to probabilistic graphical models", Michael I. Jordan, 2002.

Hidden Markov Model (HMM)

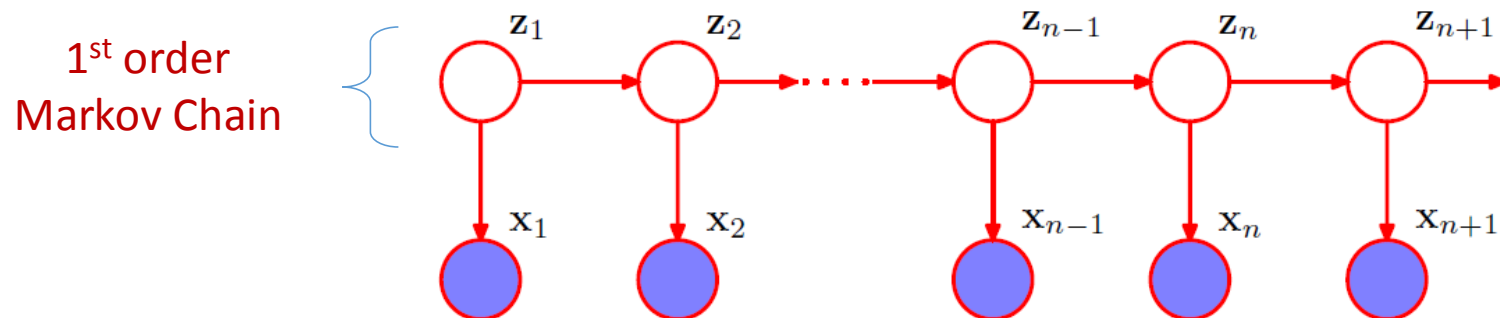
- An HMM is a natural generalization of a mixture model – can be viewed as a “dynamic” mixture model.
- For each **observed variable** X_n , there is corresponding **latent variable** Z_n (which may be of different type or dimensionality to X_n).
- The latent variables $\{Z_1, \dots, Z_{n-1}, Z_n, \dots\}$ form a **Markov Chain**, where the current state Z_n is **dependent** on the previous state Z_{n-1} .

HMM: State-Space Model

- The Markov chain of latent variables gives rise to the graphical structure known as a **state space model**.
- It satisfies the key **conditional independence** property that Z_{n-1} and Z_{n+1} are independent given Z_n , so that:

$$\mathbf{Z}_{n+1} \perp\!\!\!\perp \mathbf{Z}_{n-1} \mid \mathbf{Z}_n.$$

HMM: Joint Distribution

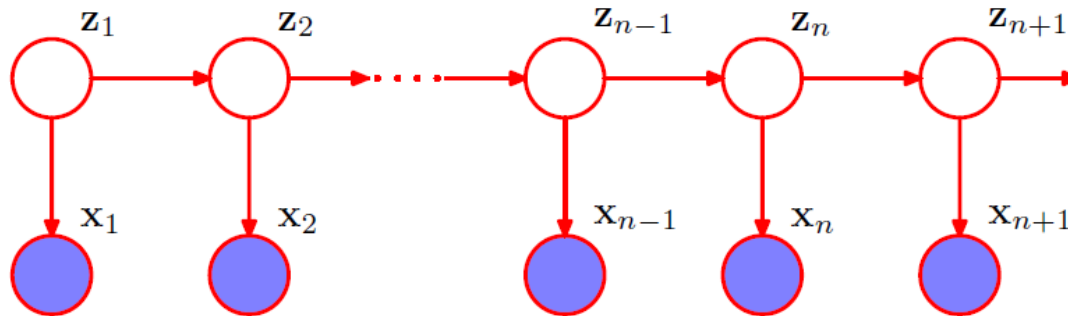


- The **joint distribution** is given by:

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

Image source: "Pattern recognition and machine learning", Christopher Bishop

HMM: Joint Distribution



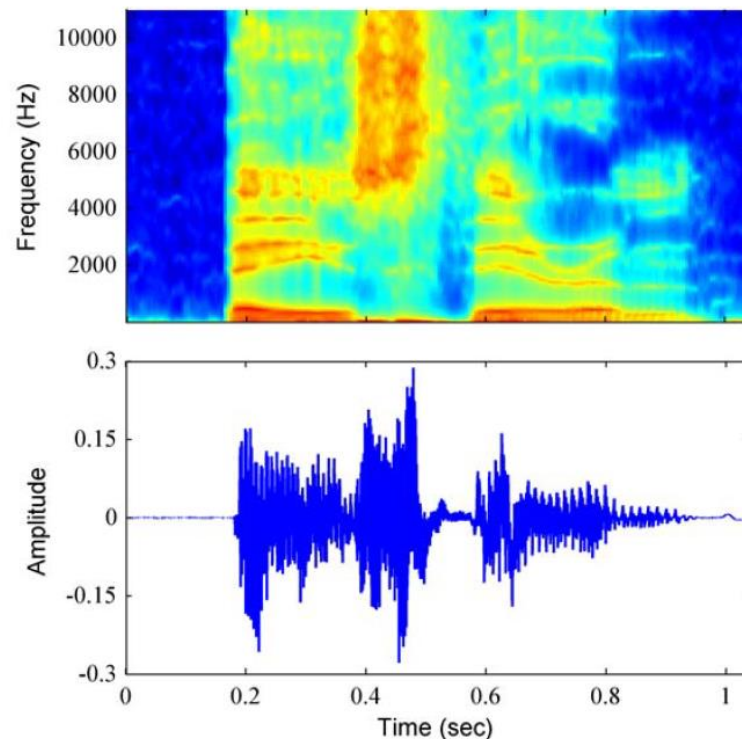
- **HMM** (covered in this lecture): Latent variables **must be discrete**, observed variable can be either discrete or continuous.
- **Linear dynamic system** (not covered in this course): Latent and observed variables are both **continuous**; linear-Gaussian if both are Gaussian.

Image source: "Pattern recognition and machine learning", Christopher Bishop

Hidden Markov Model (HMM)

Example: Speech Recognition

- Given an audio waveform, the goal is to robustly extract and recognize any spoken words



b	ey	z	th	ih	er	em
Bayes'			Theorem			

Image source: "Pattern recognition and machine learning", Christopher Bishop

Hidden Markov Model (HMM)

Example: Target Tracking

- Estimate motion of targets in 3D world from indirect, potentially noisy measurements

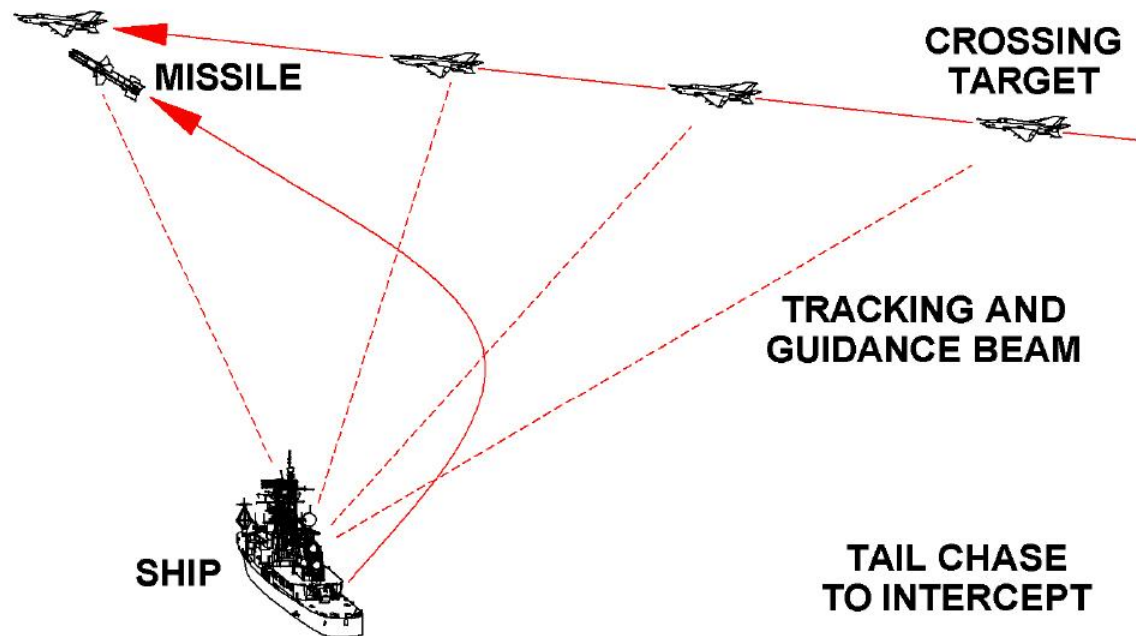
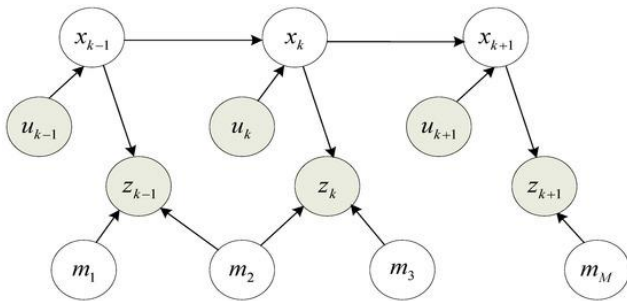


Image source: <http://www.okieboat.com/History%20guidance%20and%20homing.html>

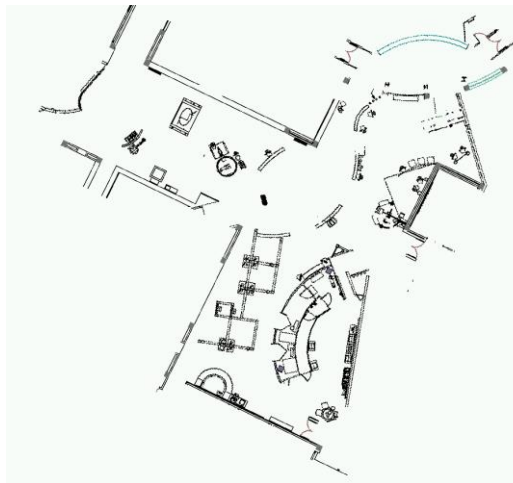
Hidden Markov Model (HMM)

Example: Robotic SLAM

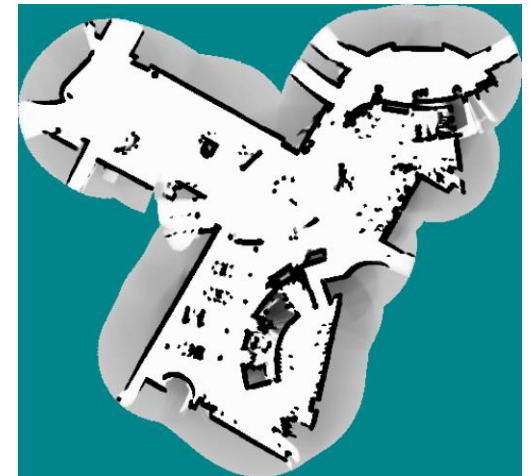
Simultaneous Localization and Mapping (SLAM) – Pose and world geometry estimation as robot moves.



CAD Map



Estimated Map



(S. Thrun, San Jose Tech Museum)

Image source: <http://www.mdpi.com/1424-8220/17/5/1174>

Hidden Markov Model (HMM)

Example: Financial Forecasting

- Predict future market behavior from historical data.

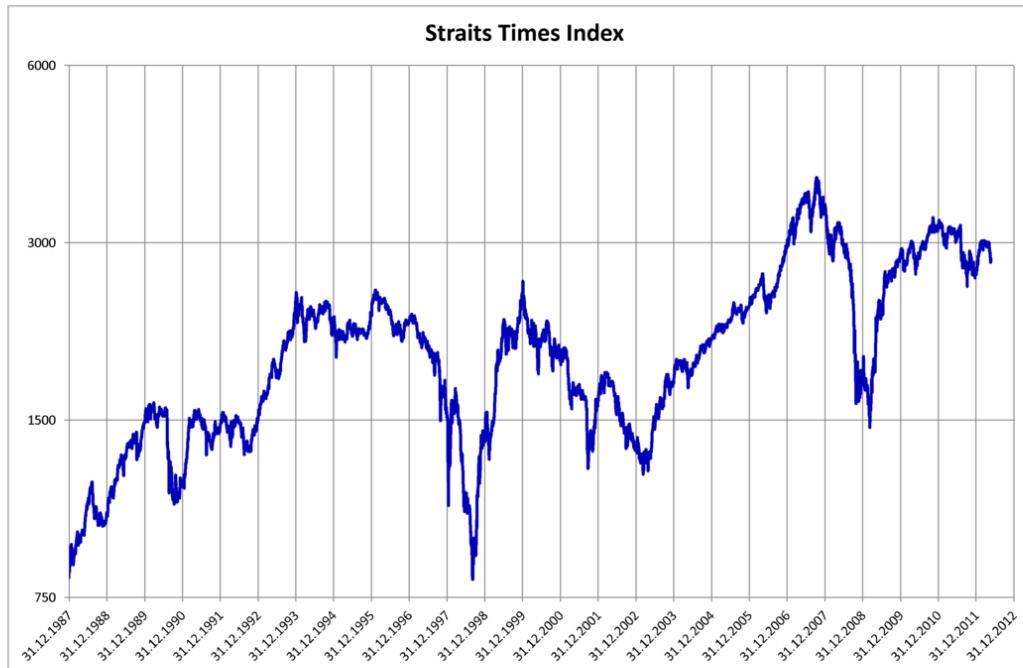


Image source: https://en.wikipedia.org/wiki/Straits_Times_Index

Transition Probabilities

- **1-of- K coding scheme** for the discrete latent variables Z_n .
- This is to describe which component of the mixture is **responsible for generating** the corresponding observation X_n .
- Z_n depends on the state of the previous latent variable Z_{n-1} through a **conditional distribution**:

$$p(z_n | z_{n-1}).$$

Transition Probabilities

- Since $z_n \in \{0,1\}^K$, $p(z_n|z_{n-1})$ corresponds to a $K \times K$ matrix A , where the elements are known as **transition probabilities**.
- Properties of the **state transition matrix** $A \in \mathbb{R}^{K \times K}$:
 1. $A_{jk} \equiv p(z_{nk} = 1 | z_{n-1,j} = 1)$
 2. $0 \leq A_{jk} \leq 1$, with $\sum_k A_{jk} = 1$
 3. $K(K - 1)$ independent parameters.

Transition Probabilities

- **Transition diagram** showing a model whose latent variables have three possible states corresponding to the three boxes.
- The black lines denote the elements of the transition matrix A_{jk} .

This is NOT a graphical model!!!

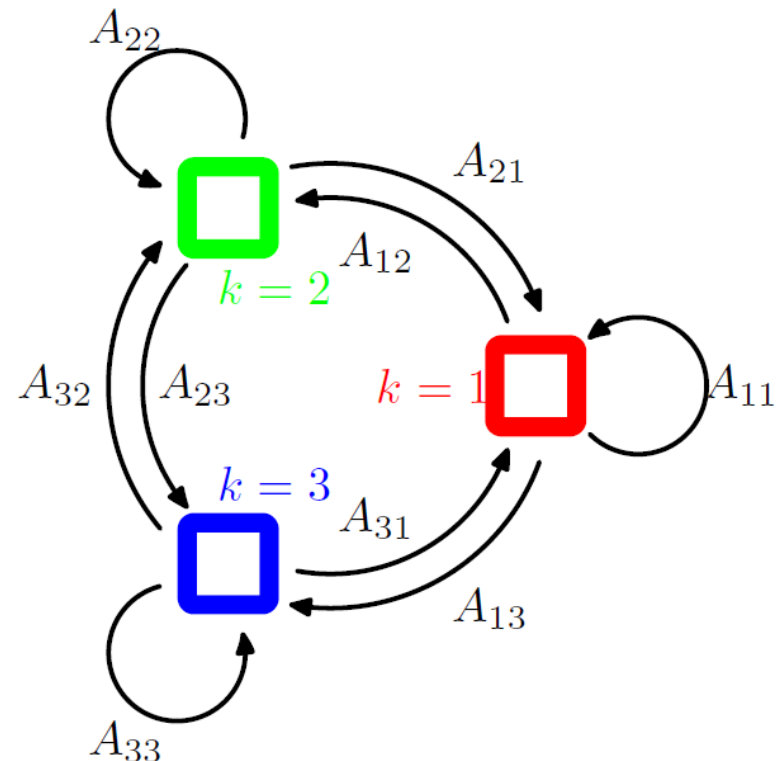


Image source: "Pattern recognition and machine learning", Christopher Bishop

Transition Probabilities

- We obtain a **lattice or trellis representation** of the latent states by unfolding the state transition diagram.
- Each column in the diagram corresponds to one of the latent variables Z_n .

This is NOT a graphical model!!!

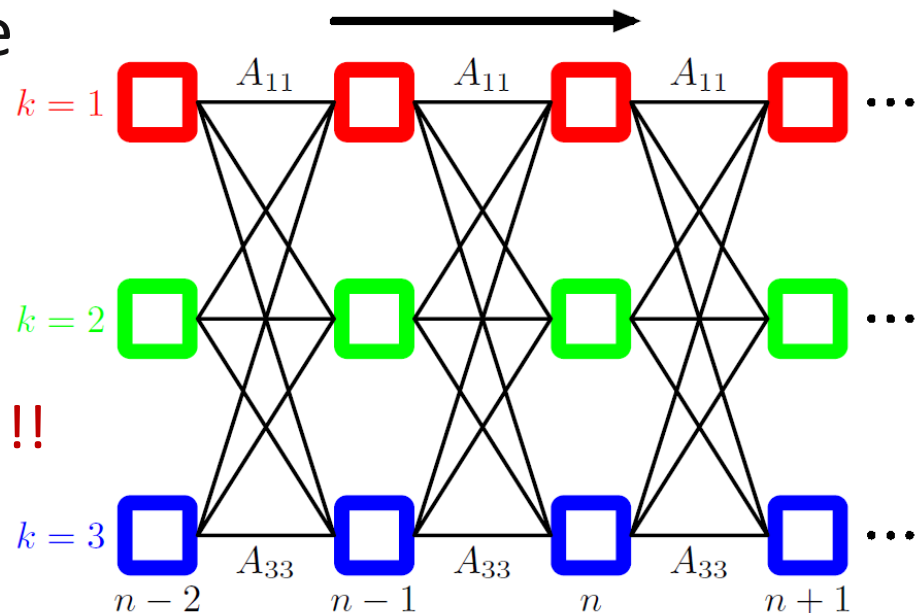


Image source: "Pattern recognition and machine learning", Christopher Bishop

Transition Probabilities

- We can then write the **conditional distribution** explicitly in the form:

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

- **Initial latent variable** Z_1 does not have a parent node, it is represented as a **categorical distribution**:

$$p(\mathbf{z}_1 | \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{z_{1k}}, \text{ where } \sum_k \pi_k = 1.$$

Emission Probabilities

- **Emission probabilities**: conditional distributions of the observed variable X_n , given the latent variable Z_n ,

$$p(x_n | z_n, \phi).$$

- $\phi = \{\phi_1, \dots, \phi_K\}$ is a **set of parameters** governing the distribution.
- Can be Gaussians if X_n is **continuous**, or conditional probability tables if X_n is **discrete**.

Emission Probabilities

- The emission probabilities is given by:

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

- For a given value of ϕ , $p(x_n | z_n, \phi)$ consists of **a vector of K numbers** corresponding to the K possible states of the binary vector Z_n .

Homogenous Model

- All of the conditional distributions governing the latent variables share the **same parameters** A .
- Similarly, all of the emission distributions share the **same parameters** ϕ .
- **Note:** Mixture model for an **i.i.d. data set** if all rows in A are the same, i.e., $p(z_n|z_{n-1})$ is independent of z_{n-1} .

HMM: Joint Probability Revisited

- The **joint probability** distribution over both latent and observed variables is given by:

$$p(\mathbf{X}, \mathbf{Z} | \boldsymbol{\theta}) = p(\mathbf{z}_1 | \boldsymbol{\pi}) \underbrace{\left[\prod_{n=2}^N p(\mathbf{z}_n | \mathbf{z}_{n-1}, \mathbf{A}) \right]}_{\text{Transition probabilities}} \underbrace{\prod_{m=1}^N p(\mathbf{x}_m | \mathbf{z}_m, \boldsymbol{\phi})}_{\text{Emission probabilities}}$$

- where $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$, and $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\phi}\}$ denotes the set of parameters governing the model.

Maximum Likelihood for HMM

- The **likelihood function** is obtained from the joint distribution by **marginalizing over the latent variables**:

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

- Cannot treat each of the sum over Z_n independently because $p(X, Z|\theta)$ **does not factorize** over Z_1, \dots, Z_n .
- Performing the sums over all N variables results in a **exponential complexity** of $O(K^N)$.
- Moreover, direct maximization will lead to **no closed-form** solutions!

MLE for HMM: EM Algorithm

- We turn to the **EM algorithm** to find an efficient framework for maximizing the likelihood function in hidden Markov models.
- The EM algorithm starts **with initialization of the model parameters**, which we denote by θ^{old} .
- In the **E step**, we take these parameter values and find the **posterior distribution of the latent variables** $p(Z|X, \theta^{old})$.

MLE for HMM: EM Algorithm

- Use $p(Z|X, \theta^{old})$ to evaluate the **expectation of the log complete-data likelihood** defined by:

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

- Let us now denote the **marginal posterior distribution** of a latent variable Z_n as:

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

- And the **joint posterior distribution** of two successive latent variables (Z_{n-1}, Z_n) as:

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

MLE for HMM: EM Algorithm

- For each value of n , we can store $\gamma(z_n)$ using a set of K nonnegative numbers that sum to unity.
- Similarly, we can store $\xi(z_{n-1}, z_n)$ using a $K \times K$ matrix of nonnegative numbers that again sum to unity.
- We shall also use $\gamma(z_{nk})$ to denote the conditional probability of $z_{nk} = 1$, with a similar use of notation for $\xi(z_{n-1,j}, z_{nk})$.

MLE for HMM: EM Algorithm

- Furthermore, the **expectation of a binary random variable** is just the probability that it takes the value of 1, we have:

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \sum_{\mathbf{z}_n} \gamma(\mathbf{z}_n) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = \mathbb{E}[z_{n-1,j} z_{nk}] = \sum_{\mathbf{z}_{n-1}, \mathbf{z}_n} \xi(\mathbf{z}_n) z_{n-1,j} z_{nk}.$$

MLE for HMM: EM Algorithm

- Putting everything together:

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

where

$$p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) = \underbrace{p(\mathbf{z}_1|\boldsymbol{\pi})}_{\prod_{k=1}^K \pi_k^{z_{1k}}} \left[\prod_{n=2}^N \underbrace{p(\mathbf{z}_n|\mathbf{z}_{n-1}, \mathbf{A})}_{\prod_{k=1}^K \prod_{j=1}^K A_{jk}^{z_{n-1,j} z_{nk}}} \right] \prod_{m=1}^N \underbrace{p(\mathbf{x}_m|\mathbf{z}_m, \boldsymbol{\phi})}_{\prod_{k=1}^K p(\mathbf{x}_m|\phi_k)^{z_{mk}}}$$

MLE for HMM: EM Algorithm

- We get:

$$Q(\theta, \theta^{old}) = \sum_Z p(Z|X, \theta^{old}) \left\{ \sum_{k=1}^K z_{1k} \ln \pi_k + \sum_{n=2}^N \sum_{k=1}^K \sum_{j=1}^K z_{n-1,j} z_{nk} \ln A_{jk} + \sum_{m=1}^N \sum_{k=1}^K z_{mk} \ln p(x_m | \phi_k) \right\}$$

- Which evaluates to:

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

MLE for HMM: EM Algorithm

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

- The **goal of the E step** will be to evaluate the quantities $\gamma(z_n)$ and $\xi(z_{n-1,j}, z_{nk})$ efficiently!
- We shall discuss this shortly in the **Forward-backward algorithm**.

MLE for HMM: EM Algorithm

- In the **M step**, we **maximize** $Q(\theta, \theta^{old})$ w.r.t. the parameters $\theta = \{\pi, A, \phi\}$ in which we treat $\gamma(z_n)$ and $\xi(z_{n-1}, z_n)$ **as constant**.
- Maximization with respect to π and A is easily achieved using appropriate **Lagrange multipliers** with:

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

MLE for HMM: EM Algorithm

Notes:

1. EM algorithm must be **initialized** by choosing starting values for π and A with the **summation constraints**.
2. Initial values for π and A **CANNOT be set to zero**, they will remain zero in subsequent EM updates.
3. Typically, we set **random starting values** for these parameters s.t. the summation and non-negativity constraints.

MLE for HMM: EM Algorithm

- Assuming that $p(x_n|\phi_k) = \mathcal{N}(x_n|\mu_k, \Sigma_k)$, maximizing of $Q(\theta, \theta^{old})$ w.r.t. $\phi_k = \{\mu_k, \Sigma_k\}$ gives:

$$\mu_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^N \gamma(z_{nk})}, \quad \Sigma_k = \frac{\sum_{n=1}^N \gamma(z_{nk}) (\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^T}{\sum_{n=1}^N \gamma(z_{nk})}.$$

MLE for HMM: EM Algorithm

- For **discrete multinomial** X_n , i.e. $x_n \in \mathbb{R}^D$ and $x_{ni} \in \{0,1\}$ and $\sum_{i=1}^D x_{ni} = 1$, the **conditional distribution** of the observations takes the form:

$$p(\mathbf{x}|\mathbf{z}) = \prod_{i=1}^D \prod_{k=1}^K \mu_{ik}^{x_i z_k}$$

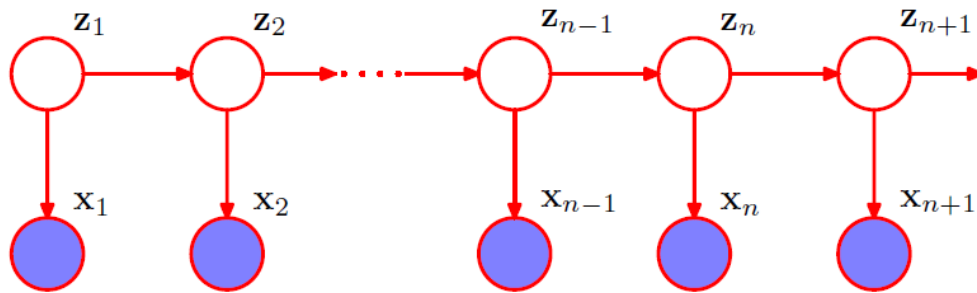
- and the corresponding **M-step equations** are given by:

$$\mu_{ik} = \frac{\sum_{n=1}^N \gamma(z_{nk}) x_{ni}}{\sum_{n=1}^N \gamma(z_{nk})}.$$

The Forward-Backward Algorithm

- We use the **forward-backward algorithm** to compute $\gamma(z_n)$ and $\xi(z_{n-1}, z_n)$ efficiently.
- Also known as the **Baum-Welch algorithm**.
- Many variants of the algorithm, we shall focus on the most widely used of these, known as the **alpha-beta algorithm**.

Conditional Independence Properties



$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

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

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

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

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

$$p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N|\mathbf{z}_{n+1}, \mathbf{x}_{n+1}) = p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N|\mathbf{z}_{n+1})$$

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

$$p(\mathbf{x}_{N+1}|\mathbf{X}, \mathbf{z}_{N+1}) = p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1})$$

$$p(\mathbf{z}_{N+1}|\mathbf{z}_N, \mathbf{X}) = p(\mathbf{z}_{N+1}|\mathbf{z}_N)$$

Using **d-separation**, we get these C.I. from the DGM of HMM.

The Forward-Backward Algorithm

- **Evaluating $\gamma(\mathbf{z}_{nk})$** : we are interested in finding the posterior distribution $p(\mathbf{z}_n | x_1, \dots, x_N)$.

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{X}) = \frac{p(\mathbf{X} | \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})} \quad \text{(Bayes' Rule)}$$

$$= \frac{p(x_1, \dots, x_n | \mathbf{z}_n) p(x_{n+1}, \dots, x_N | \mathbf{z}_n) p(\mathbf{z}_n)}{p(\mathbf{X})} \quad \text{(Conditional Independence)}$$

$$= \frac{\frac{p(x_1, \dots, x_n, \mathbf{z}_n)}{p(\mathbf{z}_n)} \cancel{p(\mathbf{z}_n)} p(x_{n+1}, \dots, x_N | \mathbf{z}_n)}{p(\mathbf{X})} \quad \text{(Product Rule)}$$

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

The Forward-Backward Algorithm

We have defined:

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

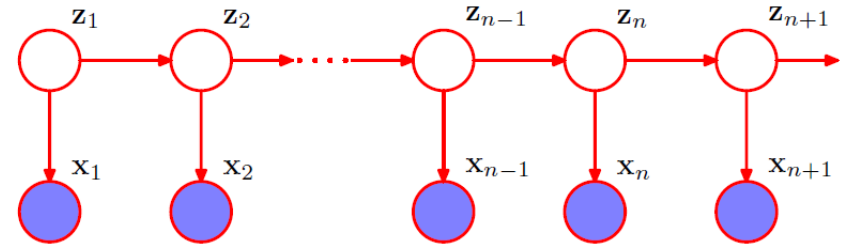
$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n)$$

- $\alpha(z_n)$ represents the **joint probability** of observing all of the given data up to time n and value of Z_n .
- $\beta(z_n)$ represents the **conditional probability** of all future data from time $n + 1$ up to N given the value of Z_n .
- $\alpha(z_n)$ and $\beta(z_n)$ each represent **set of K numbers**, one for each of the possible settings of the 1-of- K coded binary vector Z_n .

The Forward-Backward Algorithm

Forward recursion:

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



Proof:

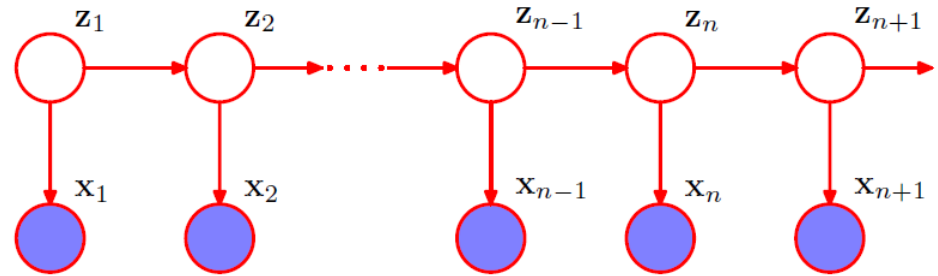
$$\begin{aligned}
 \alpha(\mathbf{z}_n) &= p(\mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{z}_n) \\
 &= p(\mathbf{x}_1, \dots, \mathbf{x}_n | \mathbf{z}_n) p(\mathbf{z}_n) && \text{(product rule)} \\
 &= p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_n) p(\mathbf{z}_n)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

Image source: "Pattern recognition and machine learning", Christopher Bishop

The Forward-Backward Algorithm

Forward recursion:

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



- There are K terms in the summation over Z_{n-1} .
- RHS has to be evaluated for K values of Z_n .
- So each step of the forward recursion has **computational cost** of $O(K^2)$.

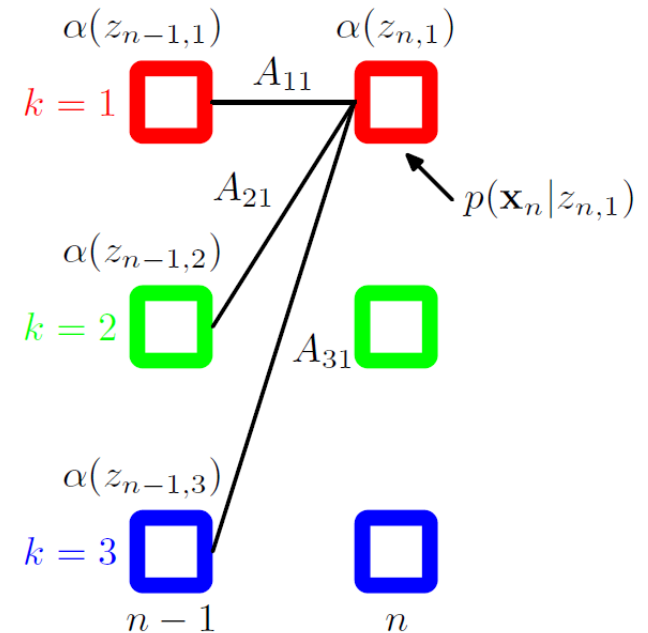
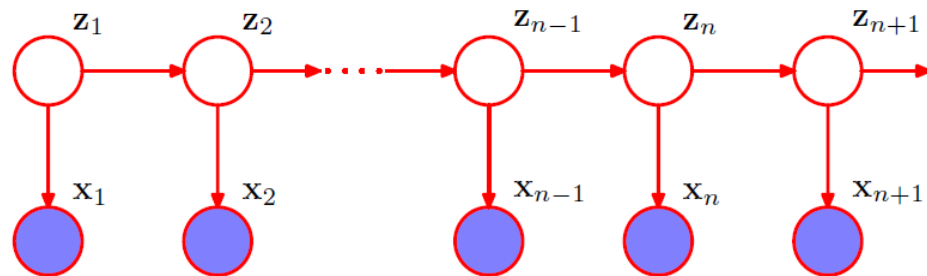


Image source: "Pattern recognition and machine learning", Christopher Bishop

The Forward-Backward Algorithm

Forward recursion:

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



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

1. elements $\alpha(z_{n-1,j})$ of $\alpha(z_{n-1})$ at step $n - 1$
2. **summing** them up with weights given by A_{j1} , i.e. values of $p(z_n | z_{n-1})$
3. and then **multiplying** by the data contribution $p(x_n | z_{n,1})$.

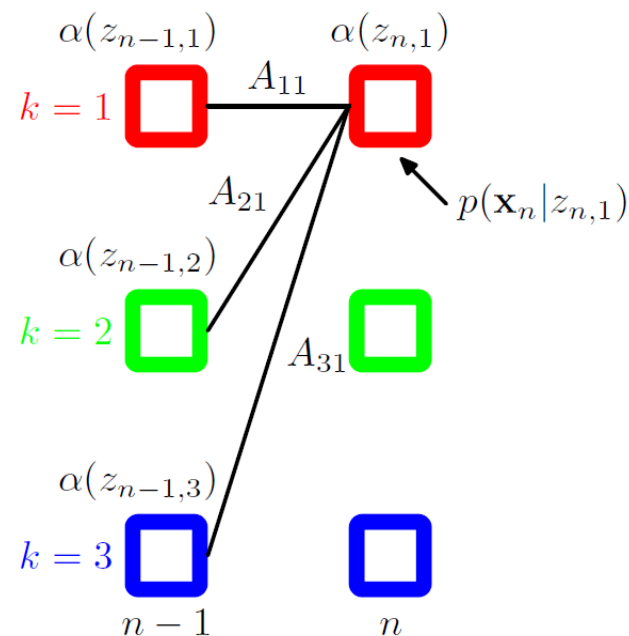
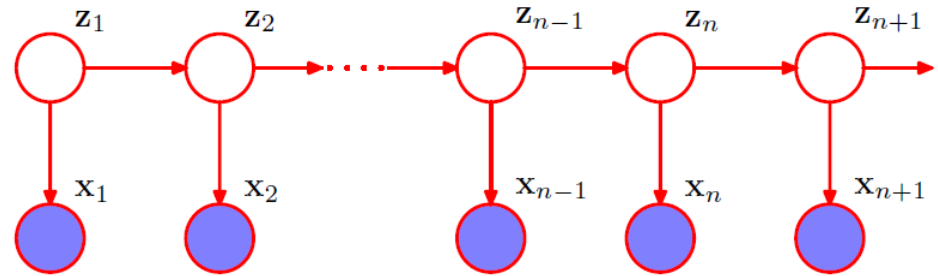


Image source: "Pattern recognition and machine learning", Christopher Bishop

The Forward-Backward Algorithm

Forward recursion:

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



- **Initialization:**

$\alpha(z_{1k})$ for $k = 1, \dots, K$ takes the value $\pi_k p(x_1 | \phi_k)$

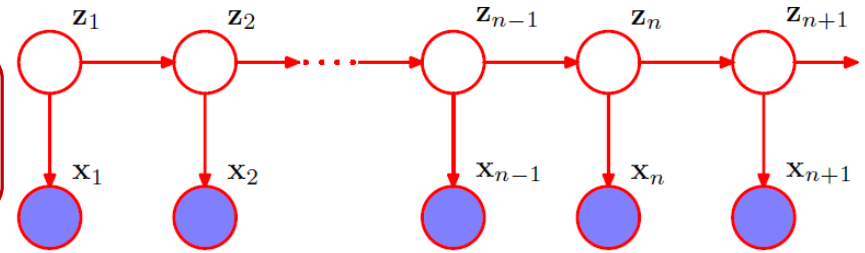
$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1) p(\mathbf{x}_1 | \mathbf{z}_1) = \prod_{k=1}^K \{\pi_k p(\mathbf{x}_1 | \phi_k)\}^{z_{1k}}$$

- **Total complexity** for the whole chain: $O(K^2 N)$.

The Forward-Backward Algorithm

Backward recursion:

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



Proof:

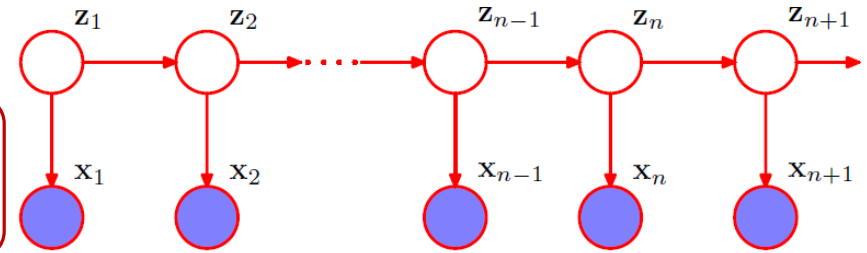
$$\begin{aligned}
 \beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n) \\
 &= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N, \mathbf{z}_{n+1} | \mathbf{z}_n) && \text{(marginalization)} \\
 &= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n, \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) && \text{(product rule)} \\
 &= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) && \text{(conditional independence)} \\
 &= \sum_{\mathbf{z}_{n+1}} \underbrace{p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N | \mathbf{z}_{n+1})}_{\beta(\mathbf{z}_{n+1})} p(\mathbf{x}_{n+1} | \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} | \mathbf{z}_n) && \text{(conditional independence)}
 \end{aligned}$$

Image source: "Pattern recognition and machine learning", Christopher Bishop

The Forward-Backward Algorithm

Backward recursion:

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



$\beta(z_{n1})$ is obtained by taking:

1. components $\beta(z_{n+1,k})$ of $\beta(z_{n+1})$ at step $n + 1$
2. **summing** them up with weights given by the **products** of A_{1k} , i.e. values of $p(z_{n+1} | z_n)$
3. and the corresponding values of the **emission density** $p(x_{n+1} | z_{n+1,k})$.

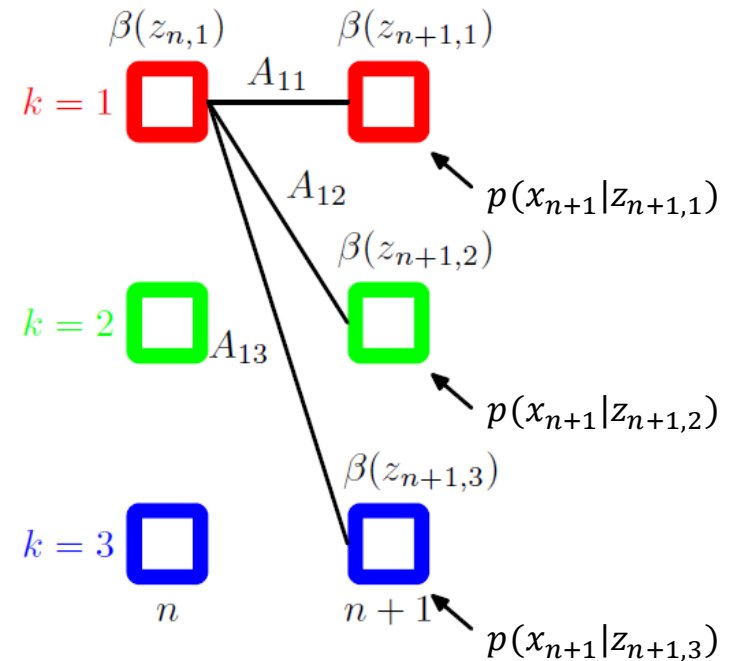


Image source: "Pattern recognition and machine learning", Christopher Bishop

The Forward-Backward Algorithm

- **Initialization:** $\beta(z_N) = 1$ for all settings of Z_N .

Proof:

Recall that we have:

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

Marginalizing both sides over \mathbf{z}_n gives us:

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

In the case of $n = N$, we have:

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N) = \sum_{\mathbf{z}_N} p(x_1, \dots, x_N, z_N) \Rightarrow \beta(z_N) = 1$$

The Forward-Backward Algorithm

- **Evaluating $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$** : which corresponds to the values of the conditional probabilities $p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X})$ for each of the $K \times K$ settings for $(\mathbf{z}_{n-1}, \mathbf{z}_n)$.

$$\begin{aligned}
 \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n | \mathbf{X}) \\
 &= \frac{p(\mathbf{X} | \mathbf{z}_{n-1}, \mathbf{z}_n) p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} && \text{(Bayes' rule)} \\
 &= \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_{n-1} | \mathbf{z}_{n-1}) p(\mathbf{x}_n | \mathbf{z}_n) p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{z}_n) p(\mathbf{z}_n | \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})}{p(\mathbf{X})} && \text{(Conditional Independence)}
 \end{aligned}$$

Forward recursion

Backward recursion

$$= \frac{\overbrace{\alpha(\mathbf{z}_{n-1})} \text{Forward recursion} \quad p(\mathbf{x}_n | \mathbf{z}_n) \quad p(\mathbf{z}_n | \mathbf{z}_{n-1}) \quad \overbrace{\beta(\mathbf{z}_n)} \text{Backward recursion}}{p(\mathbf{X})}$$

$\sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N)$

Predictive Distribution

- Given the observed data is $X = \{x_1, \dots, x_N\}$, **predict x_{N+1}** , e.g. financial forecasting.

$$p(\mathbf{x}_{N+1}|\mathbf{X}) = \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}, \mathbf{z}_{N+1}|\mathbf{X}) \quad \text{(marginalization)}$$

$$= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1})p(\mathbf{z}_{N+1}|\mathbf{X}) \quad \text{(product rule)}$$

$$= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}, \mathbf{z}_N|\mathbf{X}) \quad \text{(marginalization)}$$

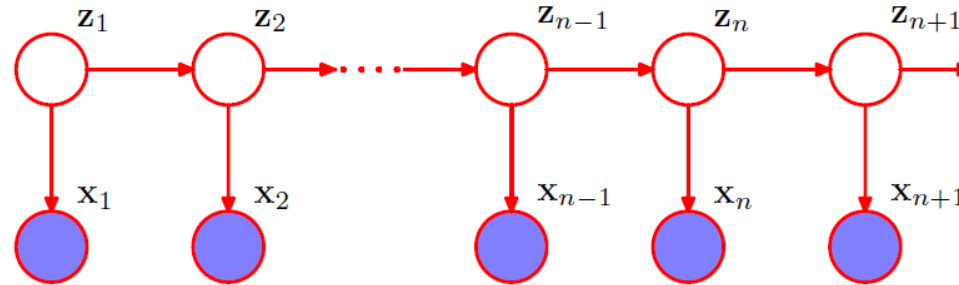
$$= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N)p(\mathbf{z}_N|\mathbf{X}) \quad \text{(product rule)}$$

$$= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N) \frac{p(\mathbf{z}_N, \mathbf{X})}{p(\mathbf{X})} \quad \text{(product rule)}$$

$$= \frac{1}{p(\mathbf{X})} \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}|\mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}|\mathbf{z}_N) \underbrace{\alpha(\mathbf{z}_N)}_{\sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N)}$$

Forward recursion

HMM: Sum-of-Product Algorithm



Convert the Bayesian network into a **factor graph**

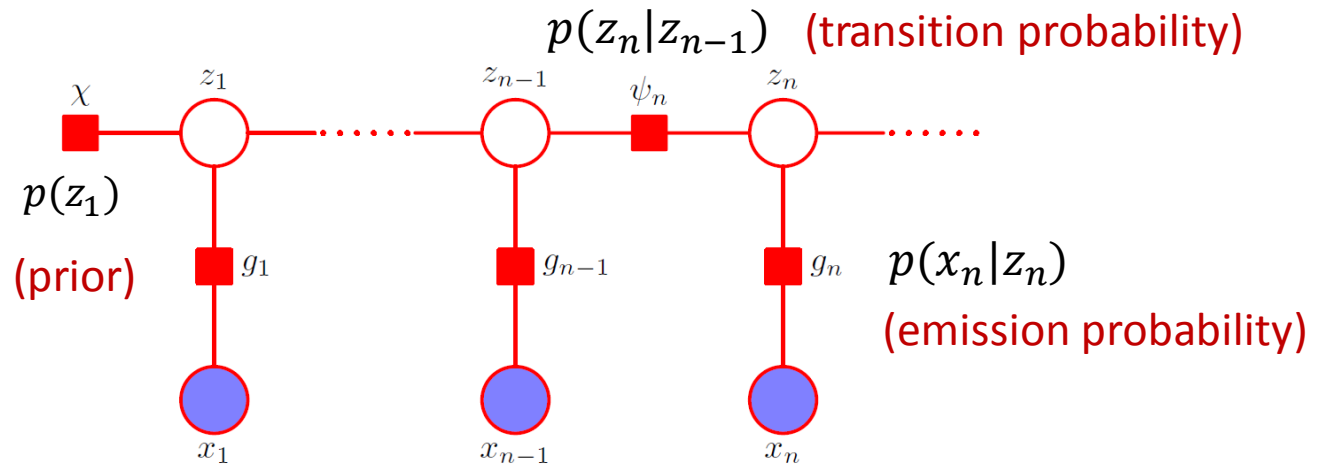
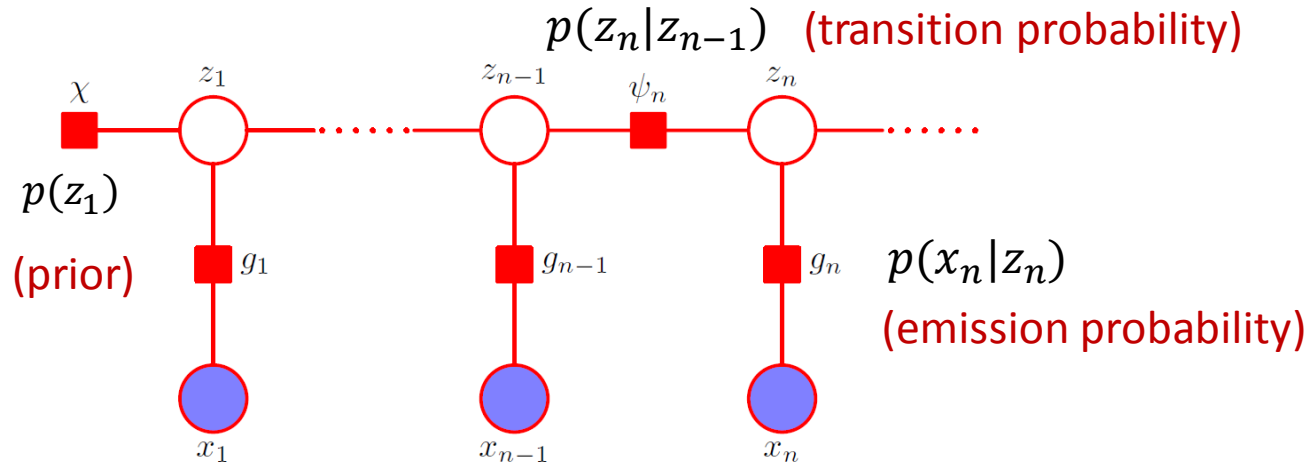


Image source: "Pattern recognition and machine learning", Christopher Bishop

HMM: Sum-of-Product Algorithm



Since we are always conditioning on x_1, \dots, x_N , we can simplify the factor graph by **absorbing the emission probabilities** into the transition probability factors.



$$h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)$$

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

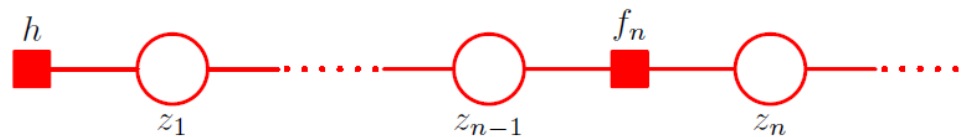
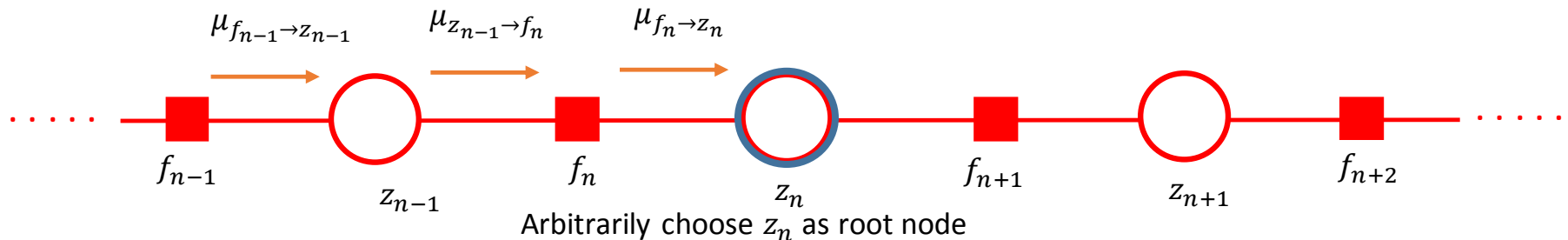


Image source: "Pattern recognition and machine learning", Christopher Bishop

HMM: Sum-of-Product Algorithm

$$h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)$$

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



- Messages from the **left towards the root node**:

Node-to-factor: $\mu_{z_{n-1} \rightarrow f_n}(\mathbf{z}_{n-1}) = \mu_{f_{n-1} \rightarrow z_{n-1}}(\mathbf{z}_{n-1})$

NO computation since there is only two neighbor nodes!

Factor-to-node: $\mu_{f_n \rightarrow z_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) \mu_{z_{n-1} \rightarrow f_n}(\mathbf{z}_{n-1})$

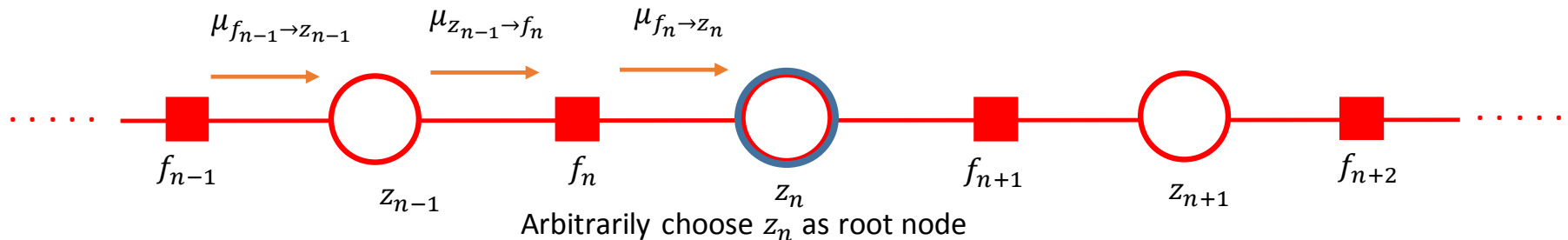


$$\mu_{f_n \rightarrow z_n}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n-1}} f_n(\mathbf{z}_{n-1}, \mathbf{z}_n) \mu_{f_{n-1} \rightarrow z_{n-1}}(\mathbf{z}_{n-1})$$

HMM: Sum-of-Product Algorithm

$$h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)$$

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



- Messages from the **left towards the root node**:

$$\underbrace{\mu_{f_n \rightarrow z_n}(\mathbf{z}_n)}_{\alpha(\mathbf{z}_n)} = \sum_{\mathbf{z}_{n-1}} \underbrace{f_n(\mathbf{z}_{n-1}, \mathbf{z}_n)}_{\alpha(\mathbf{z}_{n-1})} \underbrace{\mu_{f_{n-1} \rightarrow z_{n-1}}(\mathbf{z}_{n-1})}_{\alpha(\mathbf{z}_{n-1})}$$



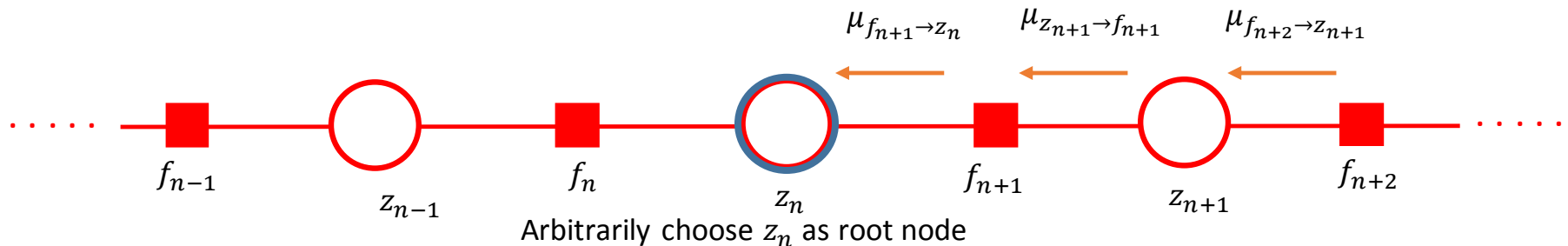
$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Same as forward recursion!

HMM: Sum-of-Product Algorithm

$$h(\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1)$$

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



- Messages from the **right towards the root node**:

Node-to-factor: $\mu_{z_{n+1} \rightarrow f_{n+1}}(z_{n+1}) = \mu_{f_{n+2} \rightarrow z_{n+1}}(z_{n+1})$ NO computation since there is only two neighbor nodes!

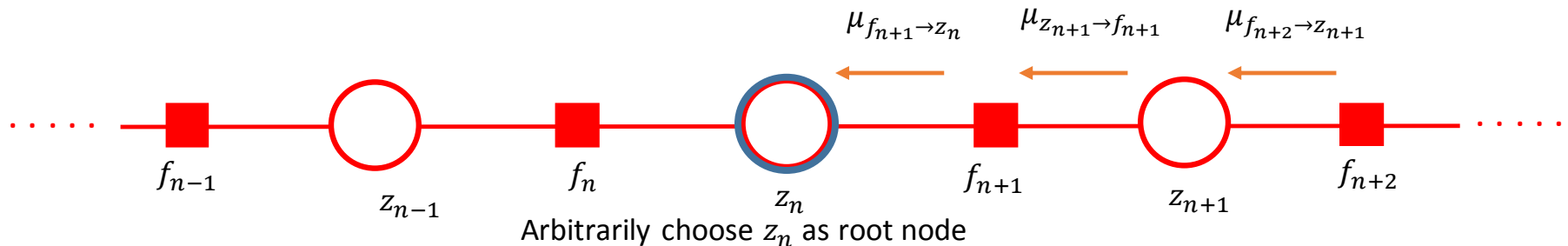
Factor-to-node: $\mu_{f_{n+1} \rightarrow z_n}(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1}) \mu_{z_{n+1} \rightarrow f_{n+1}}(z_{n+1})$



$$\mu_{f_{n+1} \rightarrow z_n}(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1}) \mu_{f_{n+2} \rightarrow z_{n+1}}(z_{n+1})$$

HMM: Sum-of-Product Algorithm

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



- Messages from the **right towards the root node**:

$$\underbrace{\mu_{f_{n+1} \rightarrow z_n}(z_n)}_{\beta(z_n)} = \sum_{\mathbf{z}_{n+1}} \underbrace{f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1})}_{p(z_{n+1}|\mathbf{z}_n)p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})} \underbrace{\mu_{f_{n+2} \rightarrow z_{n+1}}(z_{n+1})}_{\beta(z_{n+1})}$$



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

Same as backward recursion!

Scaling Factors

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

- Each step the new value $\alpha(z_n)$ is obtained from the previous value $\alpha(z_{n-1})$ by multiplying by quantities $p(z_n | z_{n-1})$ and $p(x_n | z_n)$.
- These probabilities are often significantly less than unity as we work our way forward along the chain, the **values of $\alpha(z_n)$ can go to zero exponentially quickly.**

Scaling Factors

- Taking logarithm **does not help** because we are forming sums of products of small numbers.

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

$$\log \alpha(z_n) = \log p(x_n | z_n) + \log \underbrace{\sum_{z_{n-1}} \alpha(z_{n-1}) p(z_n | z_{n-1})}_{\text{Small number}}$$

- We therefore work with **re-scaled versions** of $\alpha(z_n)$ whose values remain of order unity.

Scaling Factors

- We define a **normalized version** of α given by:

$$\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{x}_1, \dots, \mathbf{x}_n) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \dots, \mathbf{x}_n)}$$

- Which is **well behaved numerically** because it is a probability distribution over K variables for any value of n .

Scaling Factors

- Rewriting the **forward-recursion** into the normalized form:

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

$$\underbrace{\frac{p(x_1, \dots, x_n)}{p(x_1, \dots, x_{n-1})}}_{c_n} \underbrace{\frac{\alpha(z_n)}{p(x_1, \dots, x_n)}}_{\hat{\alpha}(z_n)} = p(x_n | z_n) \sum_{z_{n-1}} \underbrace{\frac{\alpha(z_{n-1})}{p(x_1, \dots, x_{n-1})}}_{\hat{\alpha}(z_{n-1})} p(z_n | z_{n-1})$$

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

Scaling Factors

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

$\tilde{\alpha}(\mathbf{z}_n)$

- c_n can be recursively computed as: $c_n = \sum_{\mathbf{z}_n} \tilde{\alpha}(\mathbf{z}_n)$

Sum over all k
entries in Z_n

Proof:

$$\begin{aligned} c_n &= \sum_{\mathbf{z}_n} \tilde{\alpha}(\mathbf{z}_n) \\ &= \sum_{\mathbf{z}_n} p(x_n | z_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(z_{n-1}) p(z_n | z_{n-1}) \\ &= \sum_{\mathbf{z}_n} p(x_n | z_n) \sum_{\mathbf{z}_{n-1}} \frac{\alpha(z_{n-1})}{p(x_1, \dots, x_{n-1})} p(z_n | z_{n-1}) \\ &= \sum_{\mathbf{z}_n} \frac{p(x_1, \dots, x_n, z_n)}{p(x_1, \dots, x_{n-1})} = \frac{p(x_1, \dots, x_n)}{p(x_1, \dots, x_{n-1})} \end{aligned}$$

Scaling Factor

- We can do similar normalization for β :

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

$\beta(\mathbf{z}_n)$

- And re-writing the **backward-recursion** into the normalized form:

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

- c_{n+1} is stored and reused from the forward-recursion, i.e.

$$c_{n+1} = \frac{p(\mathbf{x}_{n+1}, \dots, \mathbf{x}_N | \mathbf{x}_1, \dots, \mathbf{x}_n)}{p(\mathbf{x}_{n+2}, \dots, \mathbf{x}_N | \mathbf{x}_1, \dots, \mathbf{x}_{n+1})} = \frac{p(\mathbf{X})/p(\mathbf{x}_1, \dots, \mathbf{x}_n)}{p(\mathbf{X})/p(\mathbf{x}_1, \dots, \mathbf{x}_{n+1})} = \frac{p(\mathbf{x}_1, \dots, \mathbf{x}_{n+1})}{p(\mathbf{x}_1, \dots, \mathbf{x}_n)}.$$

Scaling Factor

- As a result, we get:

$$\begin{aligned}\gamma(\mathbf{z}_n) &= \hat{\alpha}(\mathbf{z}_n)\hat{\beta}(\mathbf{z}_n) \\ \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= c_n^{-1} \hat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\hat{\beta}(\mathbf{z}_n)\end{aligned}$$

Proof:

$$\begin{aligned}\gamma(z_n) = \hat{\alpha}(z_n)\hat{\beta}(z_n) &= \frac{\alpha(z_n)}{p(x_1, \dots, x_n)} \frac{\beta(z_n)}{p(x_{n+1}, \dots, x_N | x_1, \dots, x_n)} \\ &= \frac{\alpha(z_n)}{p(x_1, \dots, x_n)} \frac{\beta(z_n)}{\frac{p(x_1, \dots, x_N)}{p(x_1, \dots, x_n)}} \\ &= \frac{\alpha(z_n)\beta(z_n)}{p(X)}\end{aligned}$$

Scaling Factor

- As a result, we get:

$$\begin{aligned}\gamma(\mathbf{z}_n) &= \hat{\alpha}(\mathbf{z}_n)\hat{\beta}(\mathbf{z}_n) \\ \xi(\mathbf{z}_{n-1}, \mathbf{z}_n) &= c_n^{-1} \hat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{x}_n|\mathbf{z}_n)p(z_n|z_{n-1})\hat{\beta}(\mathbf{z}_n)\end{aligned}$$

Proof:

$$\begin{aligned}\xi(z_{n-1}, z_n) &= \frac{\cancel{p(x_1, \dots, x_{n-1})}}{p(x_1, \dots, x_n)} \frac{\alpha(z_{n-1})}{\cancel{p(x_1, \dots, x_{n-1})}} p(x_n|z_n)p(z_n|z_{n-1}) \frac{\beta(z_n)}{p(x_{n+1}, \dots, x_N|x_1, \dots, x_n)} \\ &= \frac{\alpha(z_{n-1})}{\cancel{p(x_1, \dots, x_n)}} p(x_n|z_n)p(z_n|z_{n-1}) \frac{\beta(z_n)}{\frac{p(x_1, \dots, x_N)}{\cancel{p(x_1, \dots, x_n)}}} \\ &= \frac{\alpha(z_{n-1})p(x_n|z_n)p(z_n|z_{n-1})\beta(z_n)}{p(X)}\end{aligned}$$

Scaling Factor: Initialization

- $\hat{\alpha}(\mathbf{z}_1) = \frac{\alpha(\mathbf{z}_1)}{p(\mathbf{x}_1)} = \frac{p(\mathbf{x}_1, \mathbf{z}_1)}{\sum_{\mathbf{z}_1} p(\mathbf{x}_1, \mathbf{z}_1)}$ ← Sum of all values in $\alpha(\mathbf{z}_1)$!

- We initialize $\hat{\beta}(\mathbf{z}_N) = 1$.

Proof:

$$\gamma(\mathbf{z}_n) = \hat{\alpha}(\mathbf{z}_n) \hat{\beta}(\mathbf{z}_n) \Rightarrow \gamma(\mathbf{z}_N) = \hat{\alpha}(\mathbf{z}_N) \frac{\beta(\mathbf{z}_N)}{n} \quad \leftarrow \text{normalizer}$$

Marginalizing over \mathbf{z}_N on both sides, we get

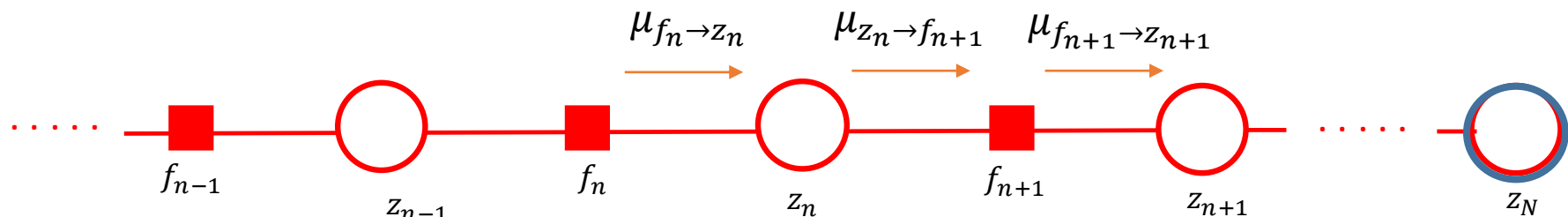
$$\sum_{\mathbf{z}_N} \gamma(\mathbf{z}_N) = \sum_{\mathbf{z}_N} \hat{\alpha}(\mathbf{z}_N) \frac{\beta(\mathbf{z}_N)}{n}$$

$$\Rightarrow n = \sum_{\mathbf{z}_N} \hat{\alpha}(\mathbf{z}_N) = 1$$

The Viterbi Algorithm

- We are now interested in finding the **max probability** and **most probable sequence of hidden states** for a given observation sequence.
- **Recall**: finding the **most probable sequence** of latent states \neq finding the set of states that are **individually** the most probable.
- Finding the most probable *sequence* of states can be solved efficiently using the **max-sum algorithm**, i.e. **Viterbi algorithm** for HMM.

The Viterbi Algorithm



Choose Z_N as root node

- **Messages** in the max-sum algorithm:

Node-to-Factor: $\mu_{\mathbf{z}_n \rightarrow f_{n+1}}(\mathbf{z}_n) = \mu_{f_n \rightarrow \mathbf{z}_n}(\mathbf{z}_n)$

Factor-to-Node:
$$\mu_{f_{n+1} \rightarrow \mathbf{z}_{n+1}}(\mathbf{z}_{n+1}) = \max_{\mathbf{z}_n} \left\{ \underbrace{\ln f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1})}_{p(\mathbf{z}_{n+1}|\mathbf{z}_n)} + \mu_{\mathbf{z}_n \rightarrow f_{n+1}}(\mathbf{z}_n) \right\}$$

- Similar to forward recursion, except summation of Z_n is **replaced with** max of Z_n .
- No backward passing since the max probability is **the same** regardless of the choice of root node.

The Viterbi Algorithm

- Denote $\omega(z_n) \equiv \mu_{f_n \rightarrow z_n}(z_n)$, we can rewrite the message as:

$$\omega(z_{n+1}) = \ln p(x_{n+1}|z_{n+1}) + \max_{z_n} \{ \ln p(z_{n+1}|z_n) + \omega(z_n) \},$$

$K \times 1$

entries which can be **computed recursively!**

Proof:

$$\underbrace{\mu_{f_{n+1} \rightarrow \mathbf{z}_{n+1}}(\mathbf{z}_{n+1})}_{\omega(z_{n+1})} = \max_{\mathbf{z}_n} \left\{ \underbrace{\ln f_{n+1}(\mathbf{z}_n, \mathbf{z}_{n+1})}_{p(z_{n+1}|z_n)p(x_{n+1}|z_{n+1})} + \underbrace{\mu_{\mathbf{z}_n \rightarrow f_{n+1}}(\mathbf{z}_n)}_{\mu_{f_n \rightarrow z_n}(z_n)} \right\}$$

$$\Rightarrow \omega(z_{n+1}) = \max_{z_n} \{ \ln p(z_{n+1}|z_n) + \ln p(x_{n+1}|z_{n+1}) + \underbrace{\mu_{z_n \rightarrow f_{n+1}}(z_n)}_{\omega(z_n)} \}$$

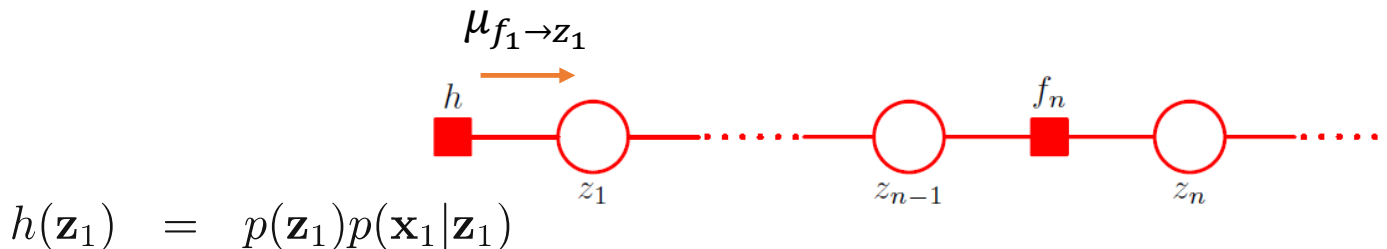
$$\Rightarrow \omega(z_{n+1}) = \ln p(x_{n+1}|z_{n+1}) + \max_{z_n} \{ \ln p(z_{n+1}|z_n) + \omega(z_n) \}$$

The Viterbi Algorithm

$K \times 1$
entries

$$\omega(z_{n+1}) = \ln p(x_{n+1}|z_{n+1}) + \max_{z_n} \{\ln p(z_{n+1}|z_n) + \omega(z_n)\},$$

- Note that **no need for scaling** since the Viterbi algorithm works with log probabilities.
- **Initialization:** factor-to-node message



$K \times 1$
entries

$$\omega(\mathbf{z}_1) = \ln p(\mathbf{z}_1) + \ln p(\mathbf{x}_1|\mathbf{z}_1).$$

The Viterbi Algorithm

- **Root Node:** The **maximal probability** of the joint distribution $p(x_1, \dots, x_N, z_1, \dots, z_N)$ is given by the max of $\omega(z_N)$ at the root node.

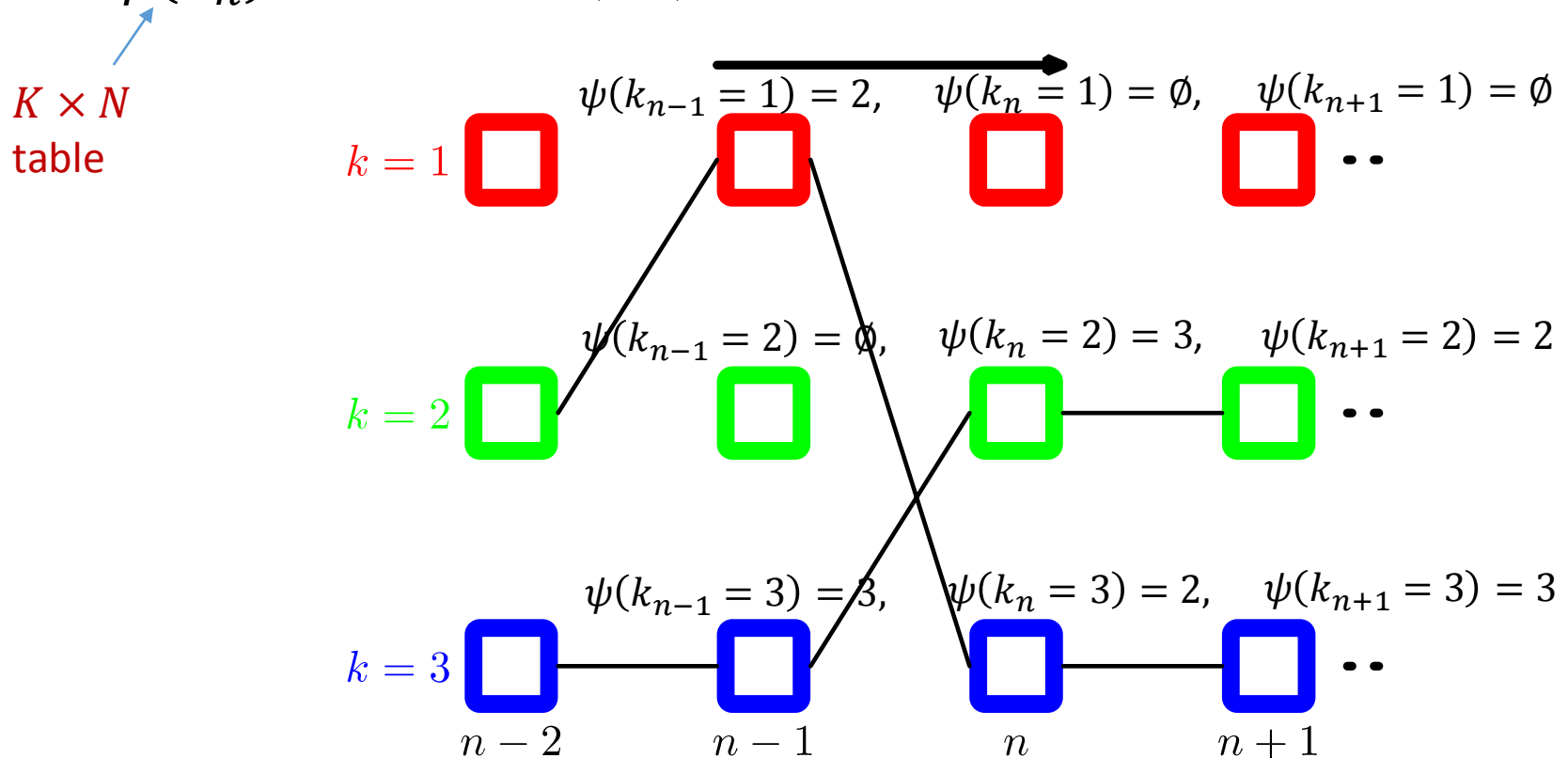
$$\max_{z_1, \dots, z_N} p(x_1, \dots, x_N, z_1, \dots, z_N) = \max_{z_N} \omega(z_N)$$

The Viterbi Algorithm

- The forward recursion to Z_n will give us the **maximal probability** of the joint distribution $p(X, Z)$.
- We also wish to find the **sequence of latent variable values** that corresponds to the maximal probability.
- We will use the **back-tracking procedure** described in Lecture 6 to do this.

The Viterbi Algorithm

- Keep a record of the values of Z_n that correspond to the maxima for each value of the K values of Z_{n+1} , denoted by $\psi(k_n)$ where $k = 1, \dots, K$.



The Viterbi Algorithm

- We get $\psi(k_N)$ when we reach the end of the chain, i.e. root node Z_N .
- The **sequence of latent variable values** that corresponds to the maximal probability can then be obtained by backtracking the chain recursively:

$$k_n^{\max} = \psi(k_{n+1}^{\max}).$$

Summary

- We have looked at how to:
 1. Describe the joint distribution of a HMM with the **transition and emission probabilities**.
 2. Use the **EM algorithm** for maximum likelihood estimation of the latent variables and unknown parameters in the HMM.
 3. Use the **forward-backward algorithm** to evaluate the EM algorithm.
 4. Apply the **Viterbi algorithm** to find the maximal probability and its configuration.