

DS-GA 3001.001
Probabilistic time series analysis
Lecture 6. Hidden Markov Models

Instructor: Cristina Savin
NYU, CNS & CDS

1. Project proposals due next Tue (10/20, 6pm)

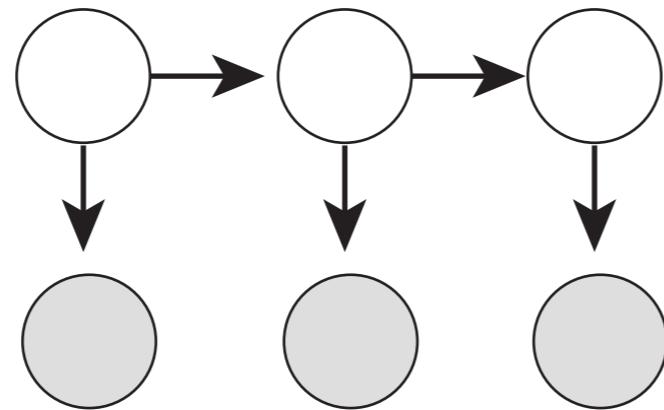
1. Project proposals due next Tue (10/20, 6pm)

2. HW2 (LDS), due Oct 16; CS office hours (Thu)

Latent state models

In the latent space the temporal dependencies are simple:

1st order Markov



Goal: find a **latent** variable
that summarizes
the relevant **history**
while keeping math simple

continuous z , linear gaussian: **Kalman filtering**

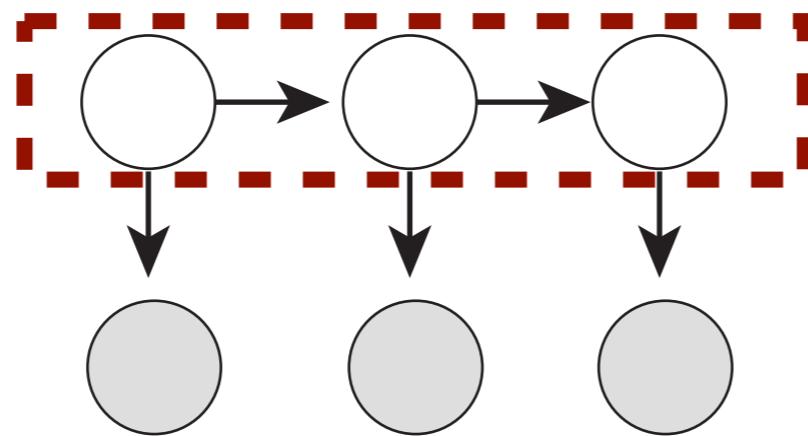
continuous z , (non)linear/(non)gaussian: **particle filtering**

discrete z : **hidden Markov models (HMMs)**

Latent state models

In the latent space the temporal dependencies are simple:

1st order Markov



Goal: find a **latent** variable
that summarizes
the relevant **history**
while keeping math simple

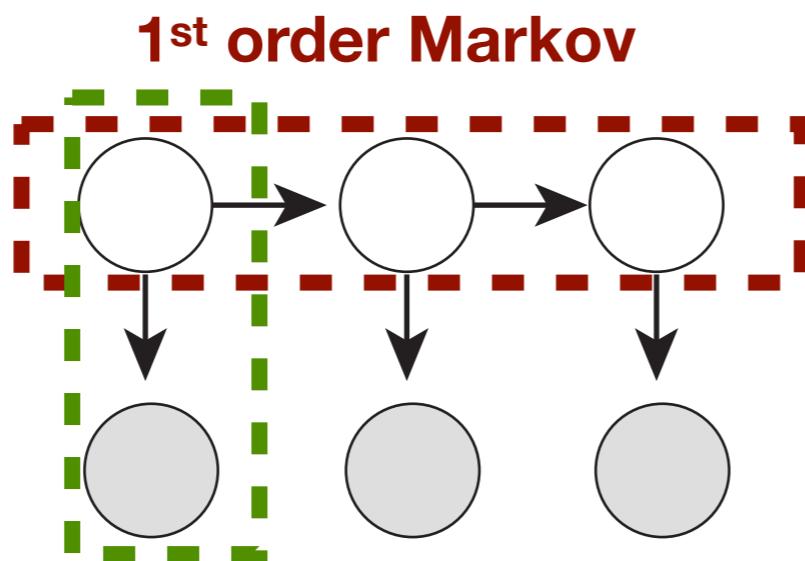
continuous z , linear gaussian: **Kalman filtering**

continuous z , (non)linear/(non)gaussian: **particle filtering**

discrete z : **hidden Markov models (HMMs)**

Latent state models

In the latent space the temporal dependencies are simple:



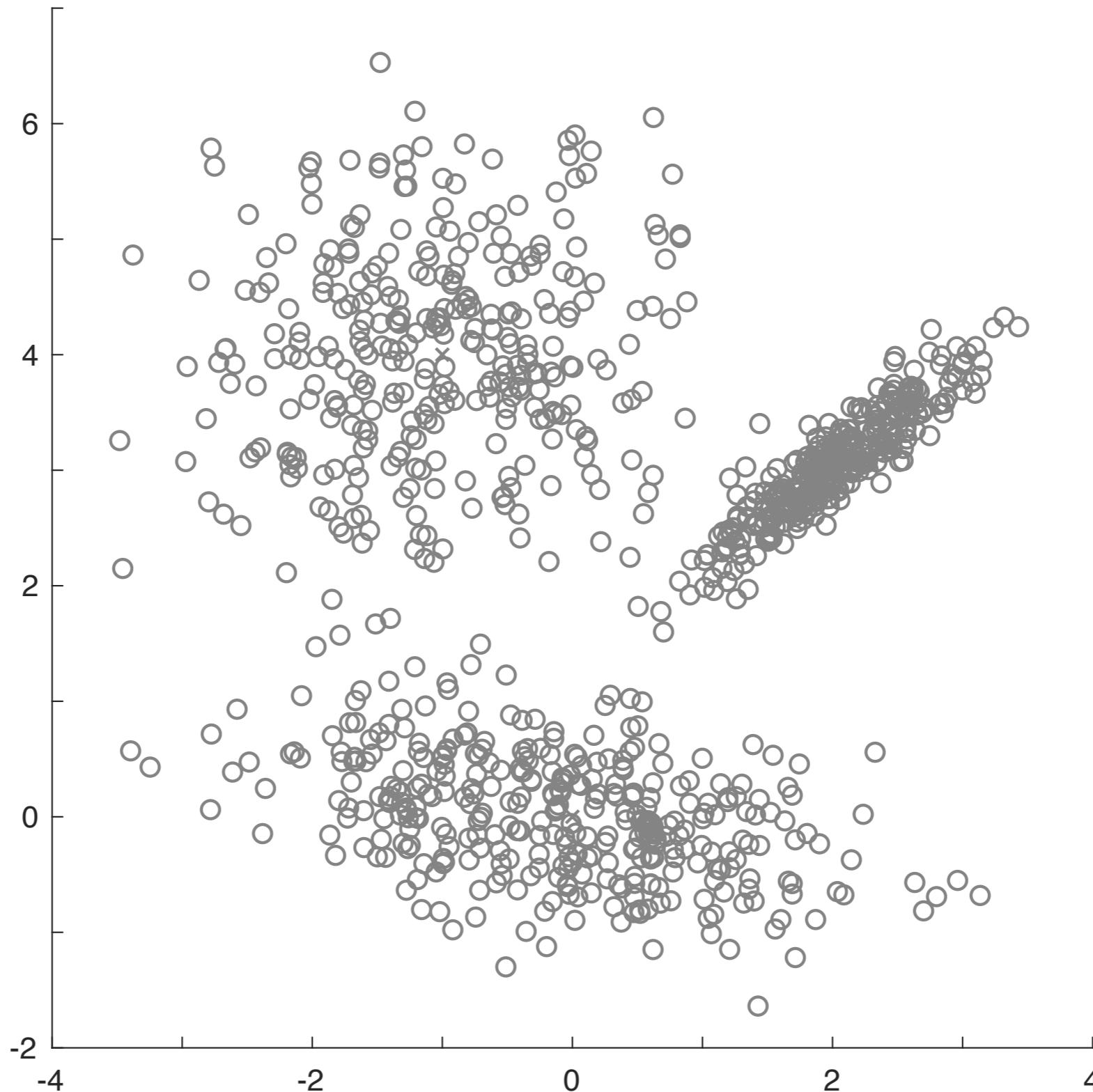
Goal: find a **latent** variable
that summarizes
the relevant **history**
while keeping math simple

continuous z , linear gaussian: **Kalman filtering**

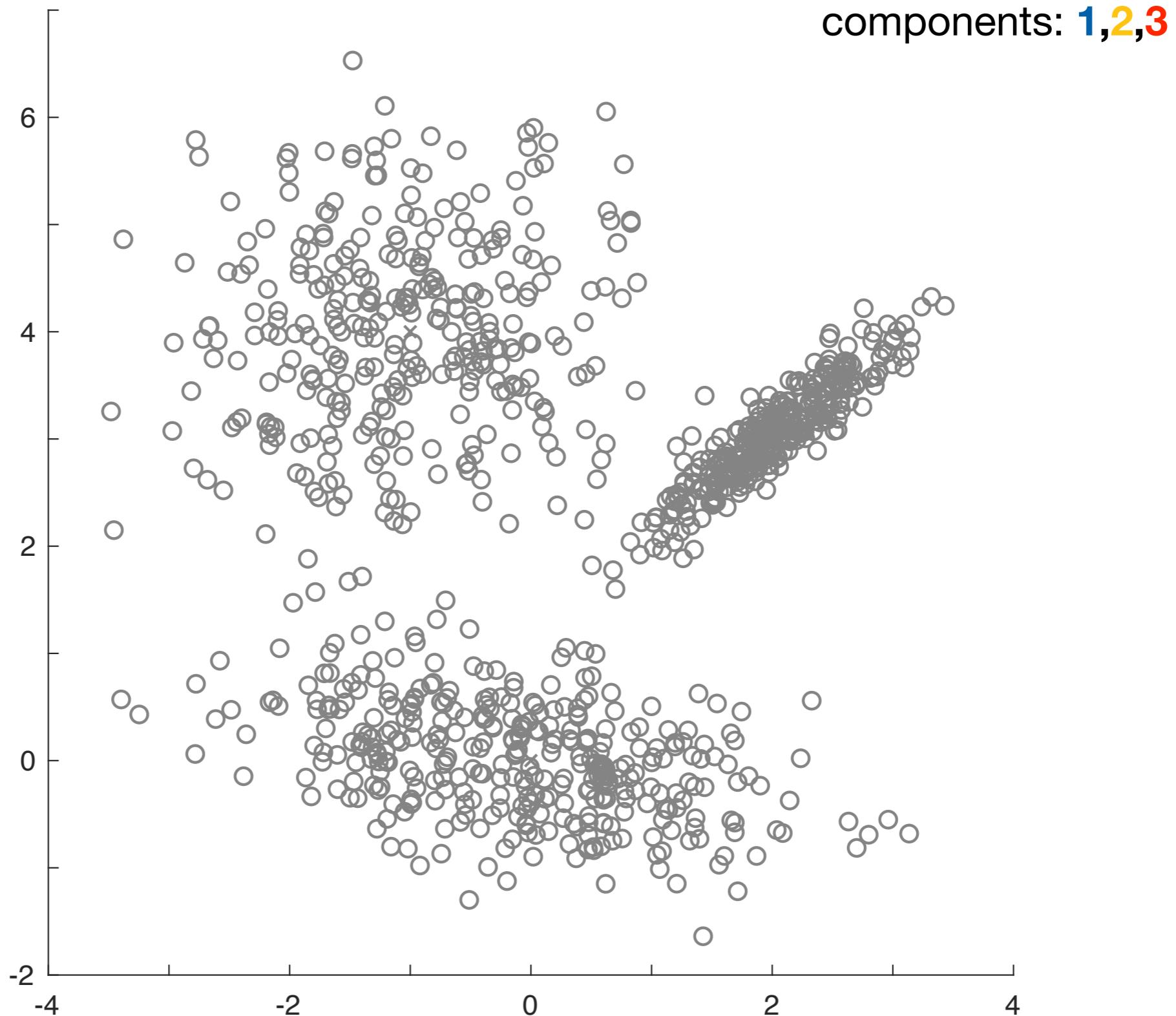
continuous z , (non)linear/(non)gaussian: **particle filtering**

discrete z : **hidden Markov models (HMMs)**

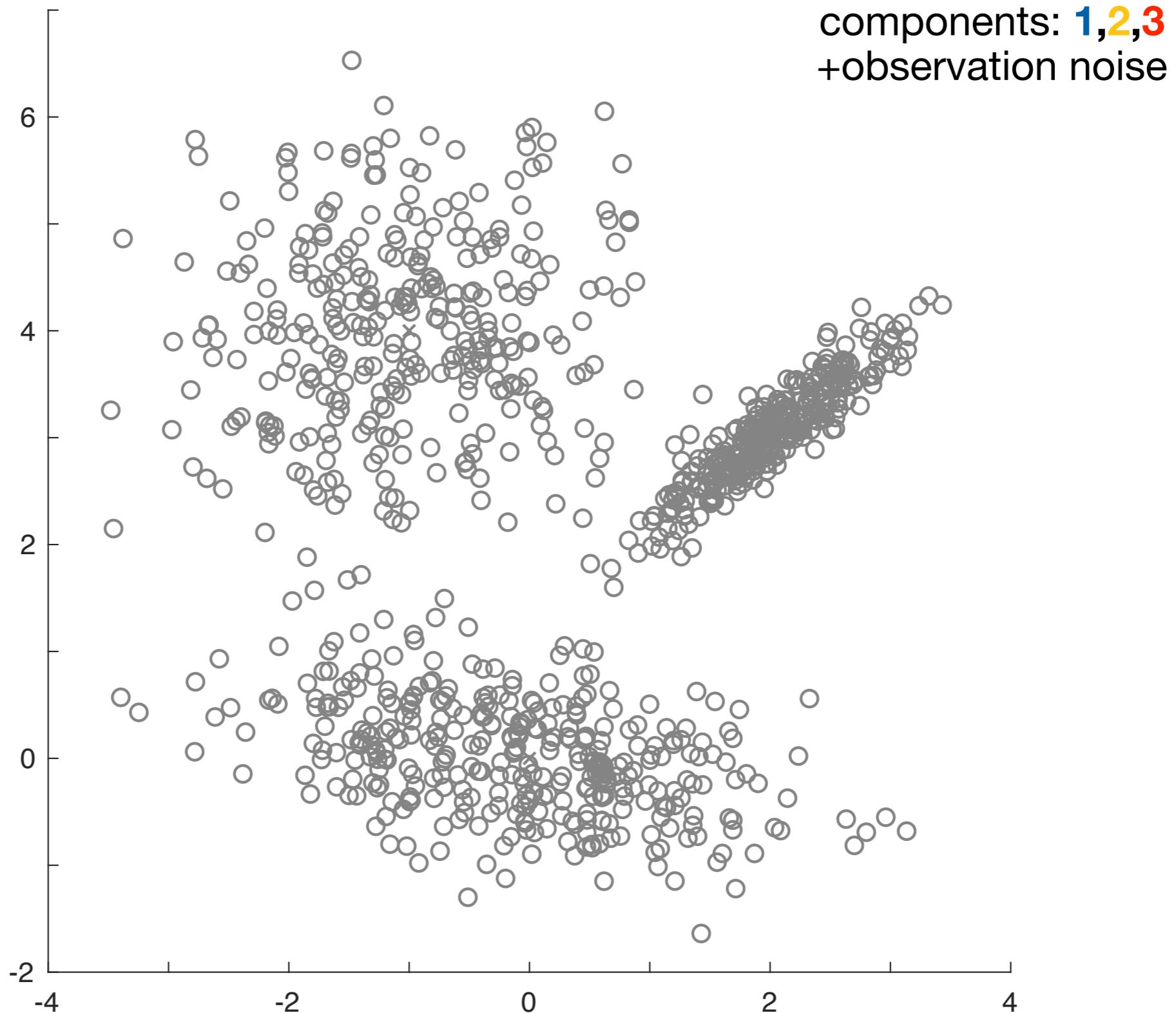
Reminder: mixture models



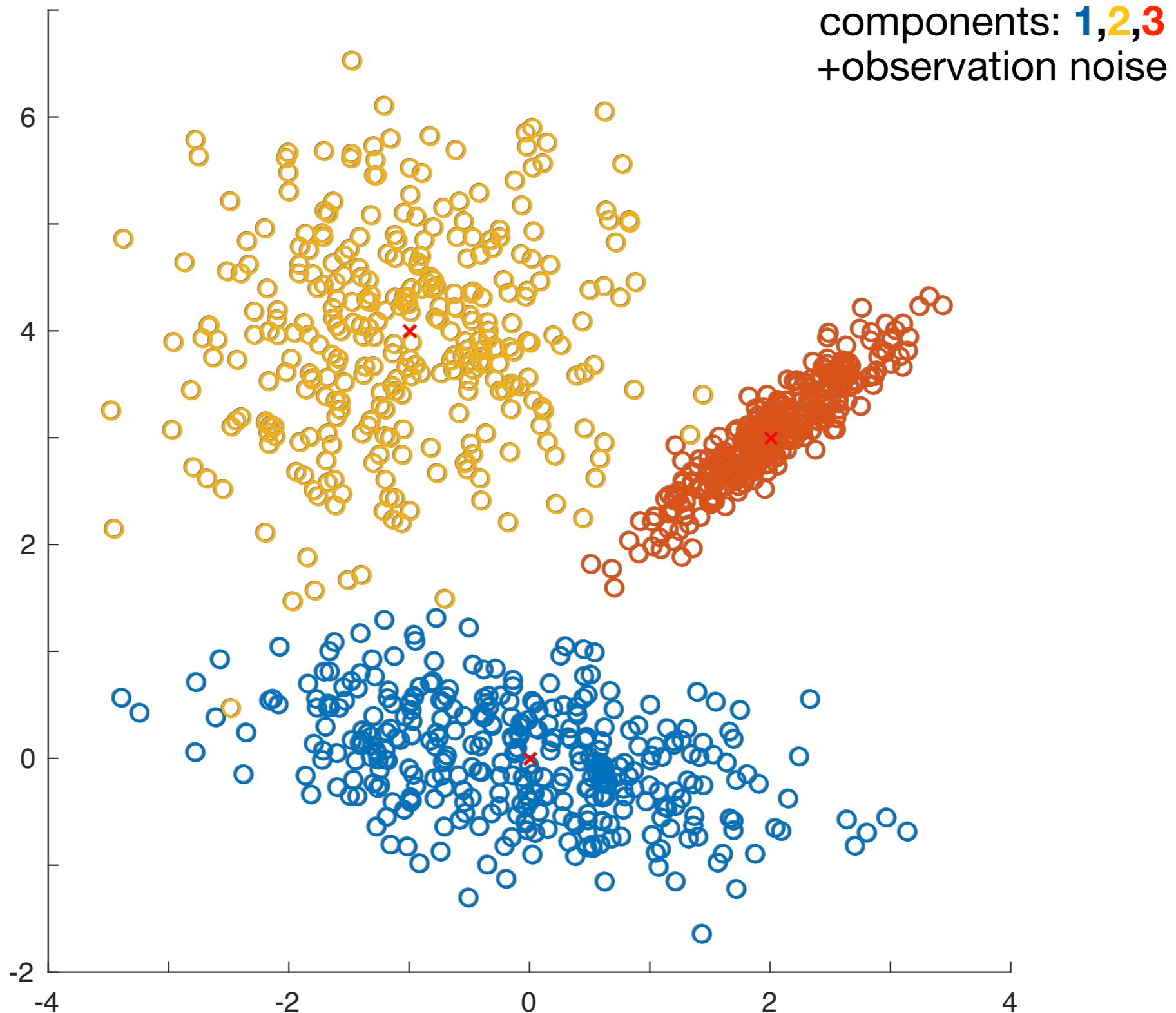
Reminder: mixture models



Reminder: mixture models

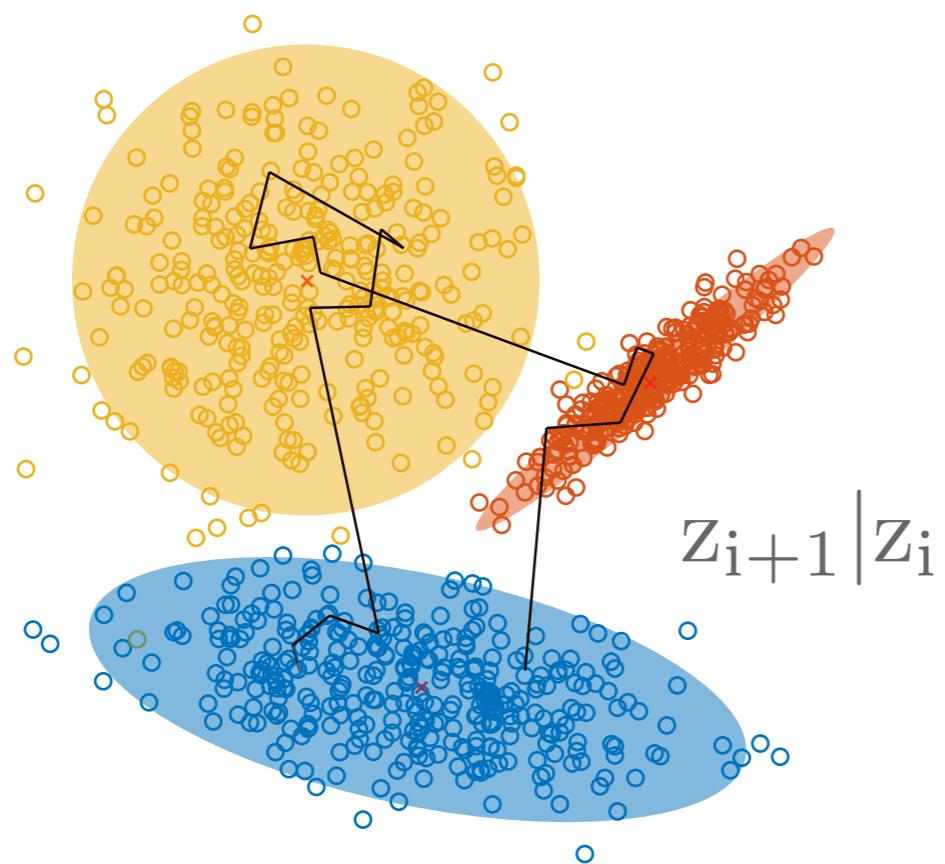


Reminder: mixture models

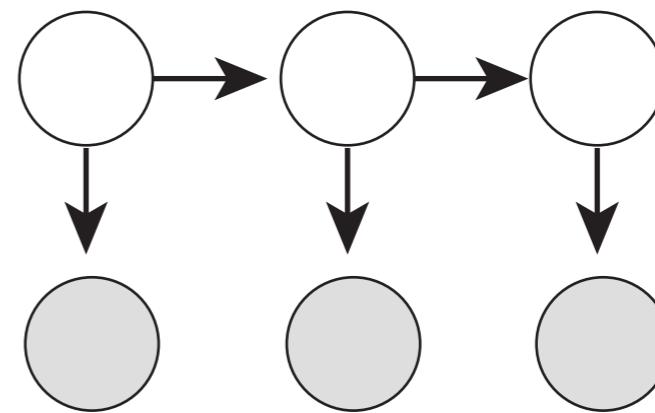
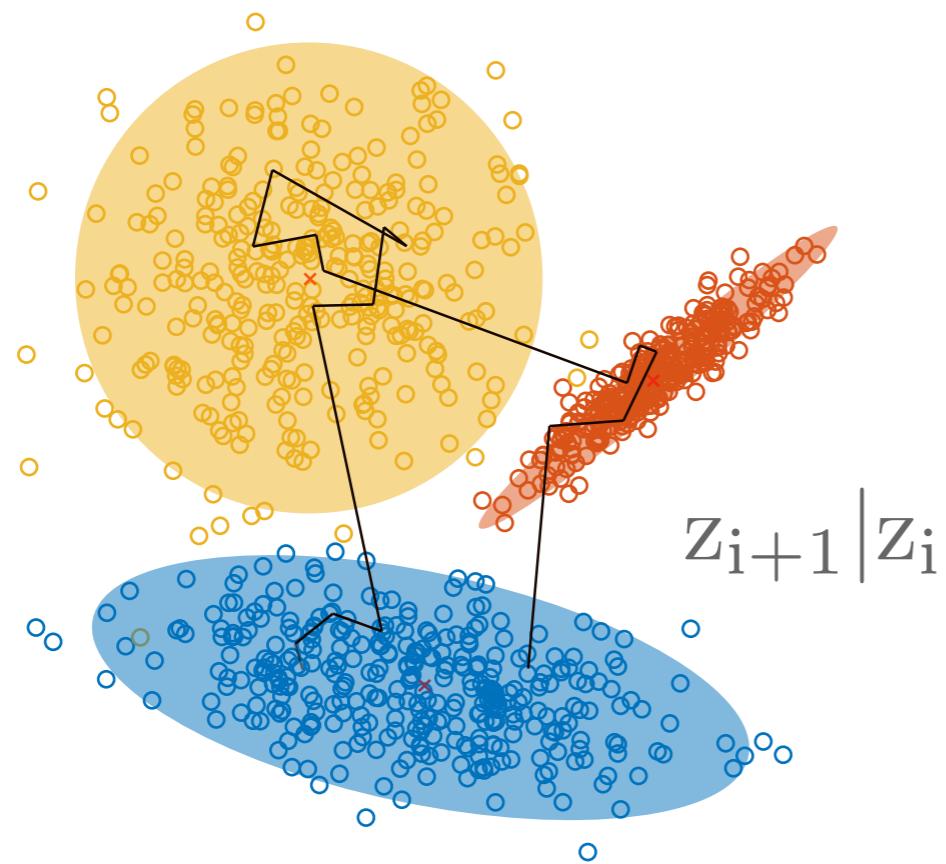


Hidden Markov models: mixture models, with **temporal dependencies** for components

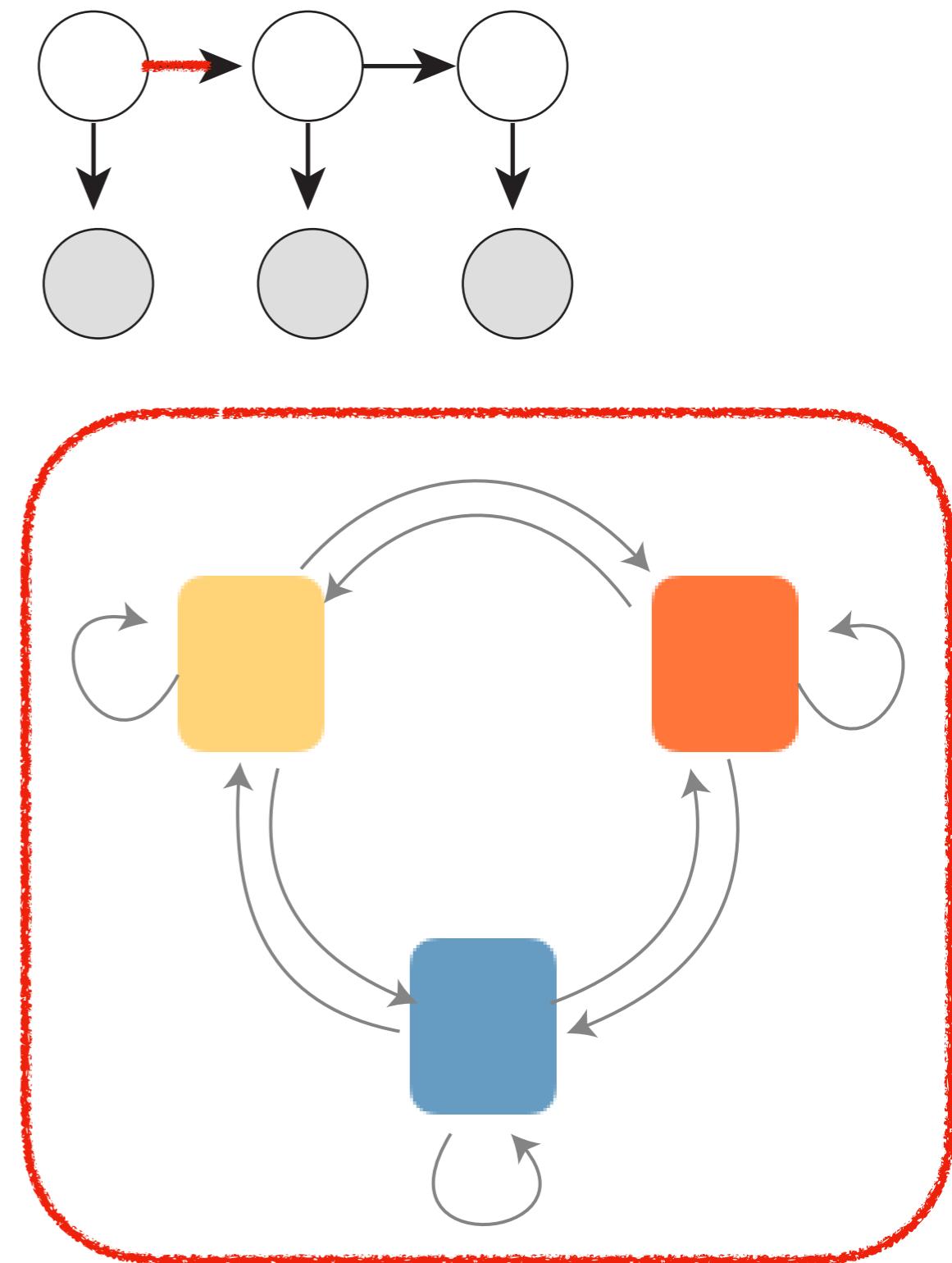
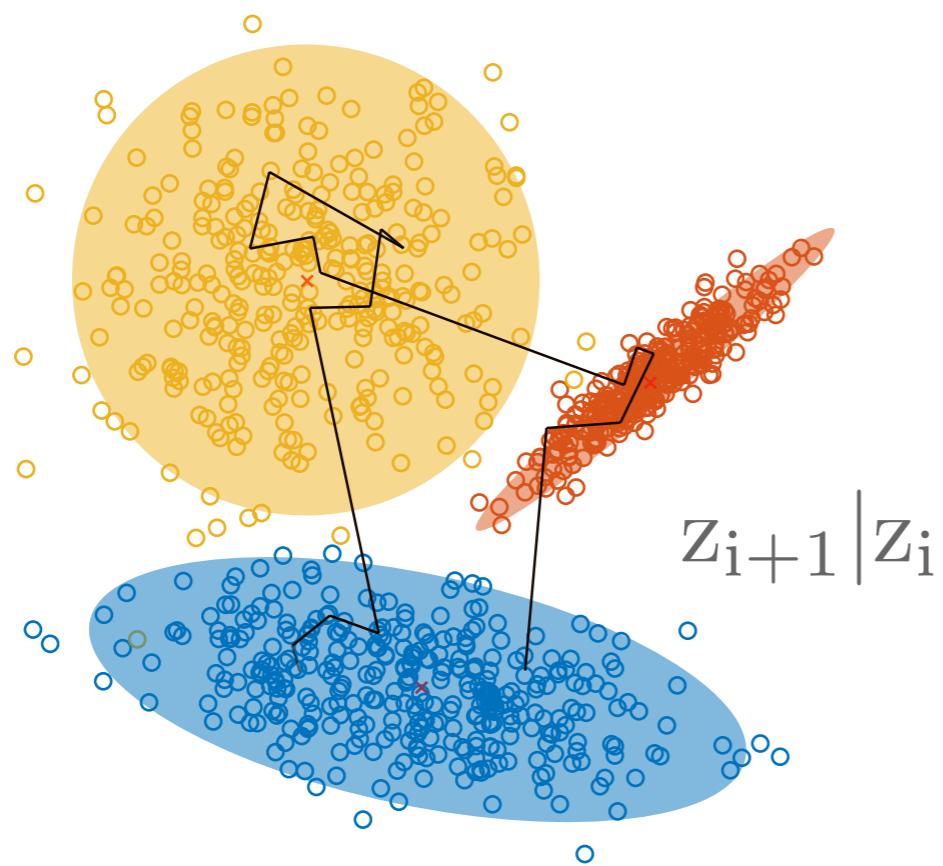
Hidden Markov models: mixture models, with temporal dependencies for components



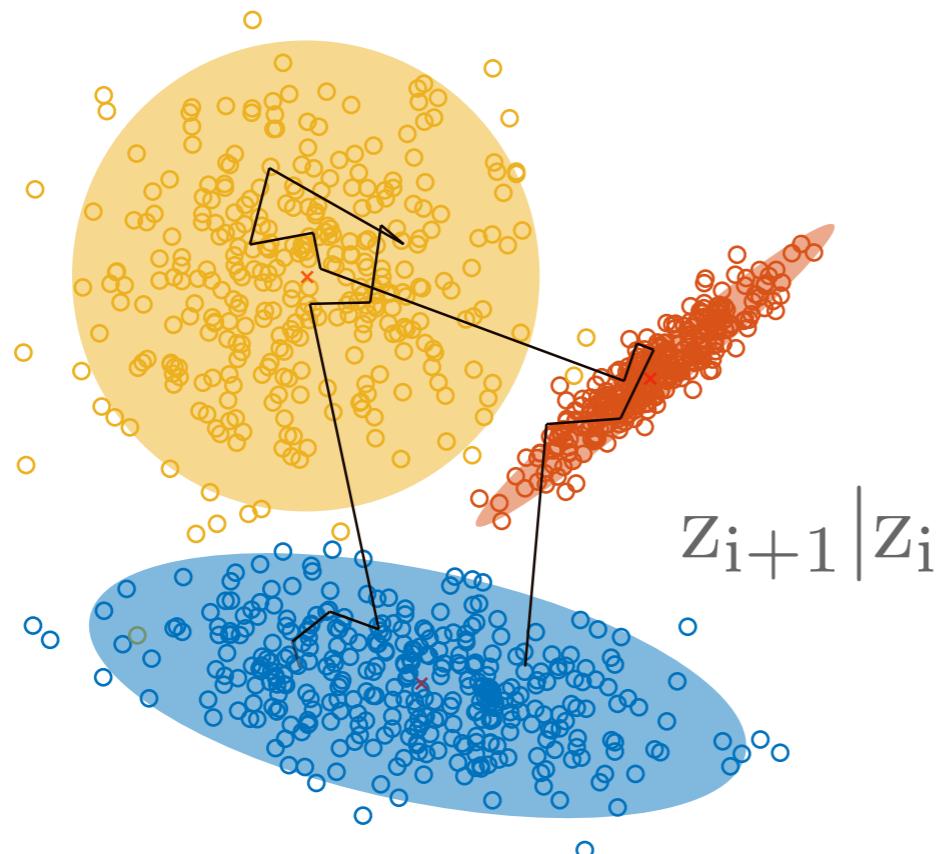
Hidden Markov models: mixture models, with temporal dependencies for components



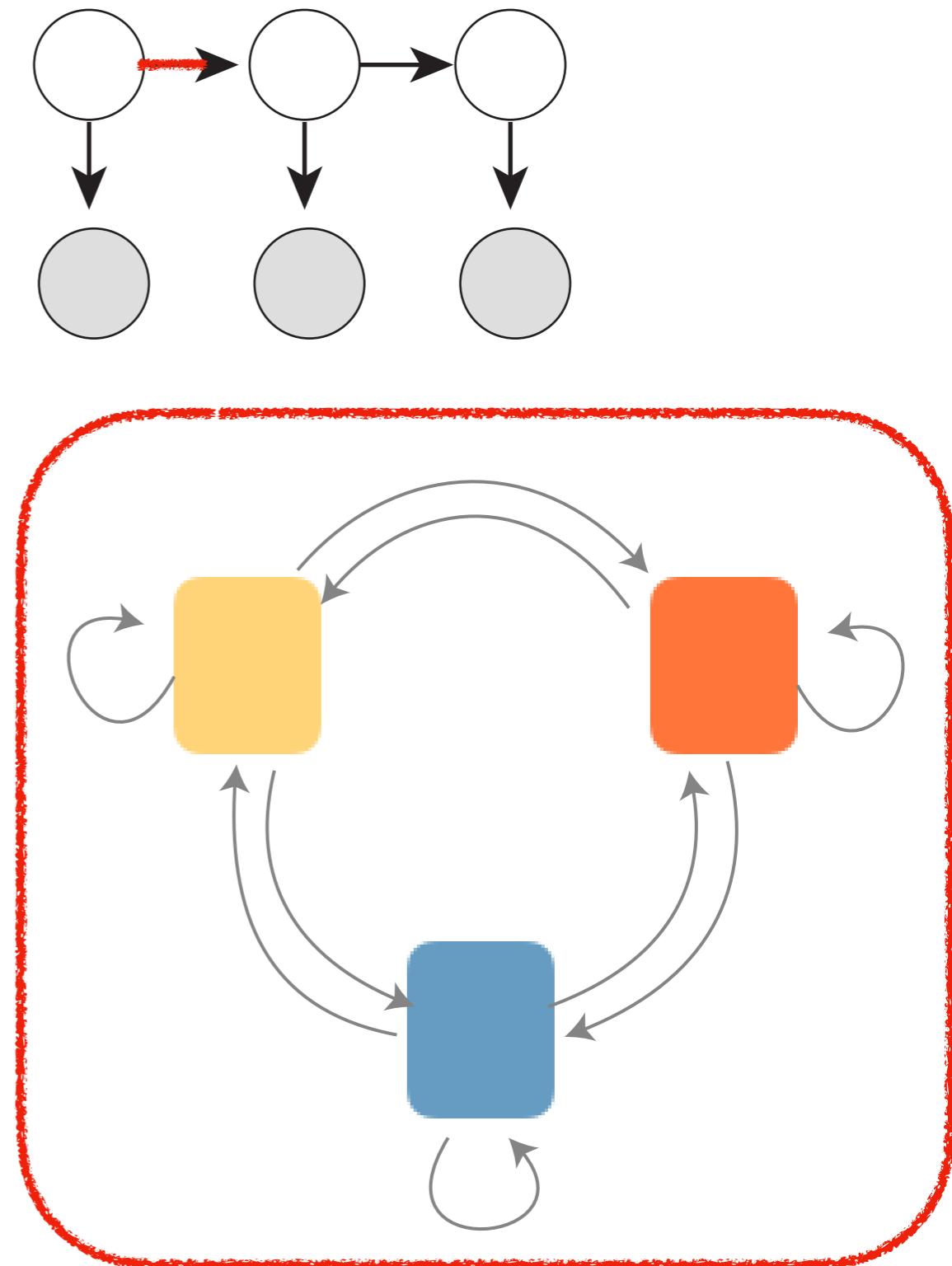
Hidden Markov models: mixture models, with temporal dependencies for components



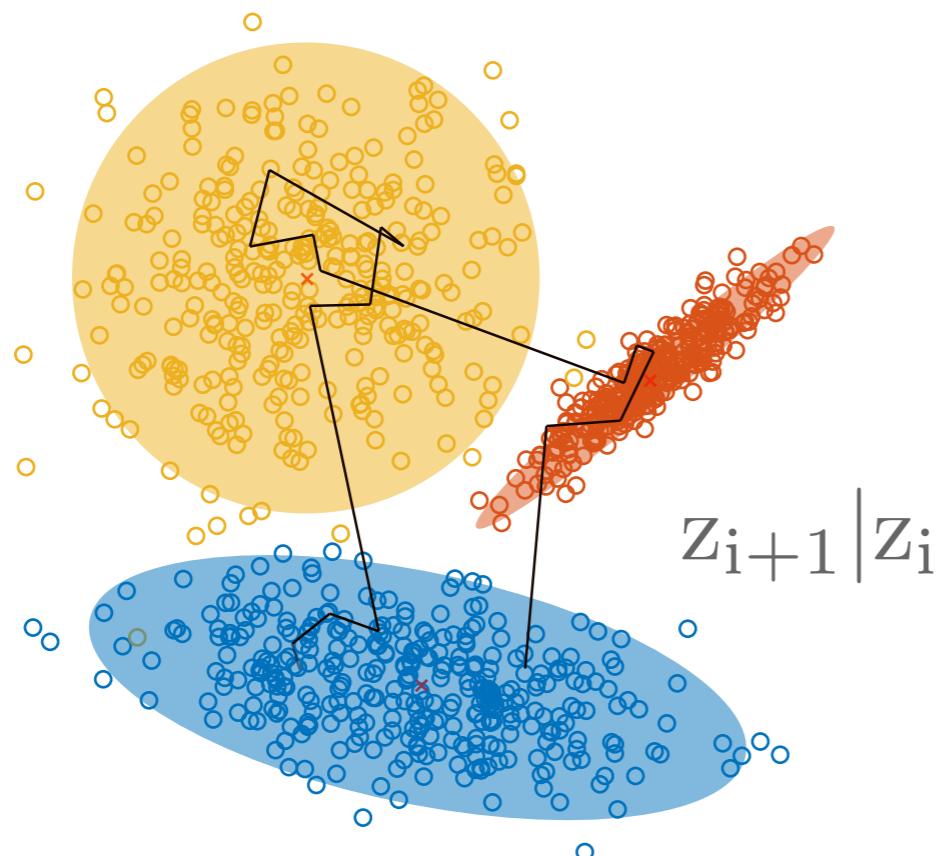
Hidden Markov models: mixture models, with temporal dependencies for components



Markov(1913) used frequency counts to compute the probability that the next letter is a vowel?



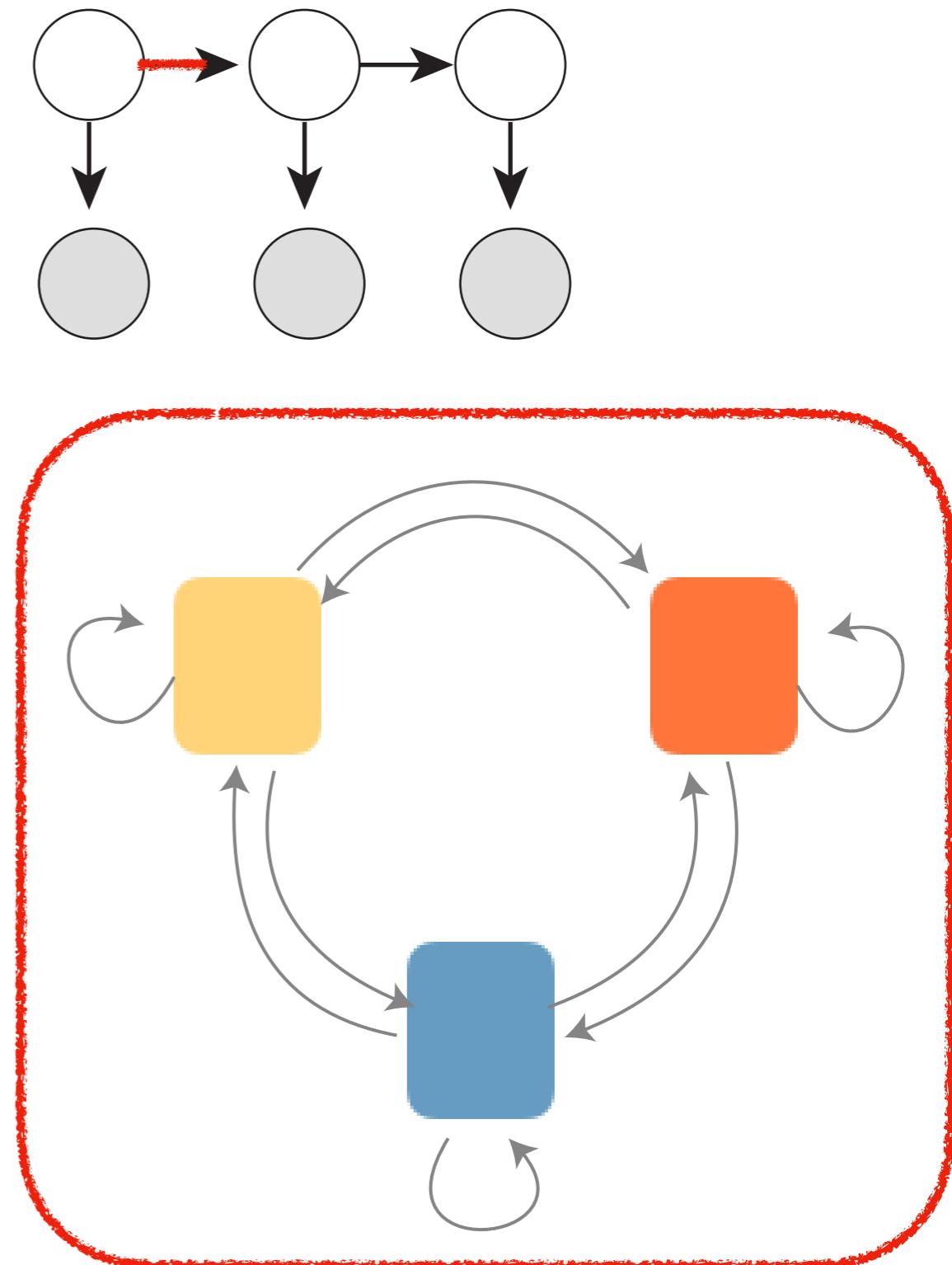
Hidden Markov models: mixture models, with temporal dependencies for components

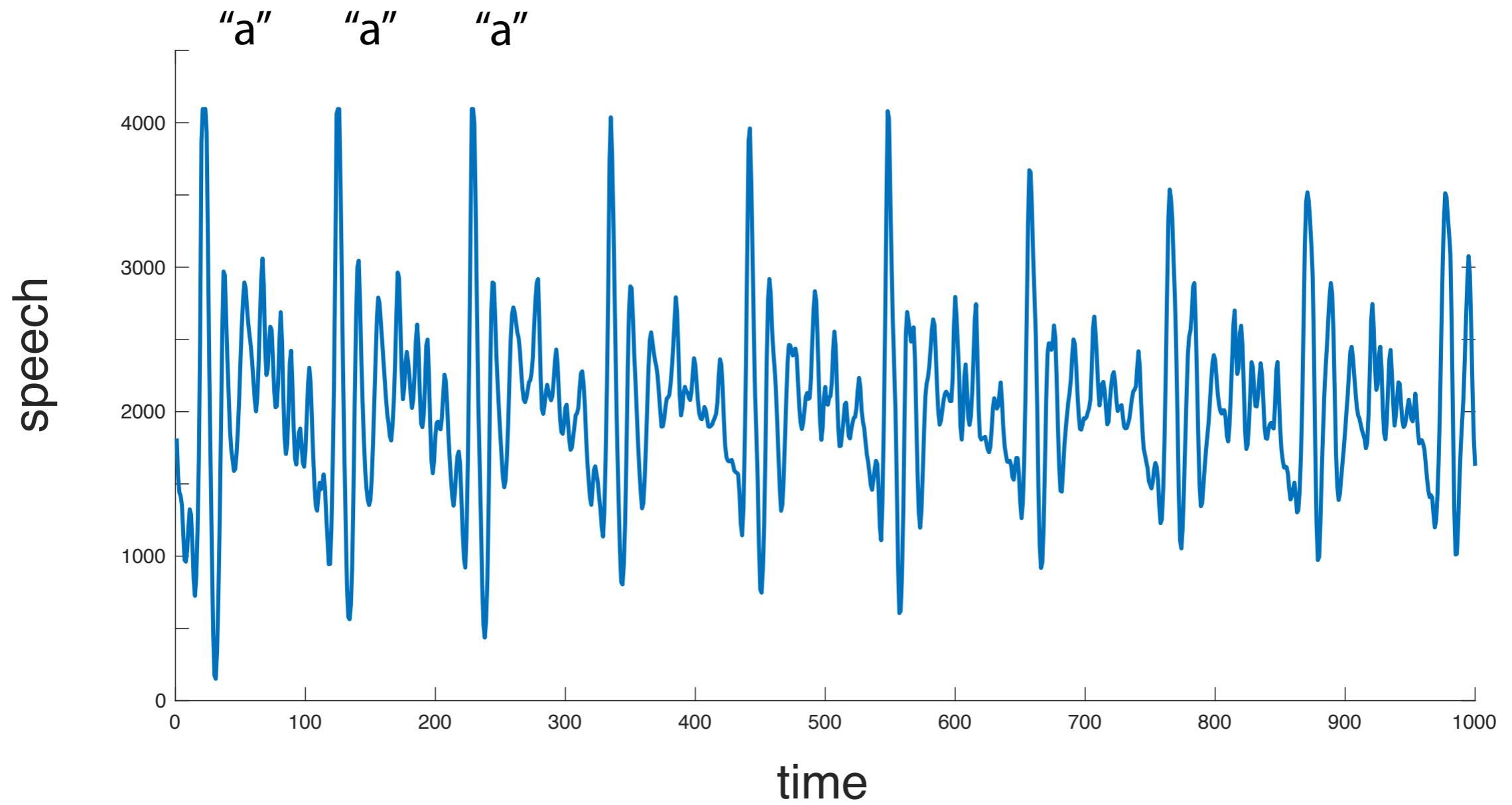


Markov(1913) used frequency counts to compute the probability that the next letter is a vowel?

Applications:

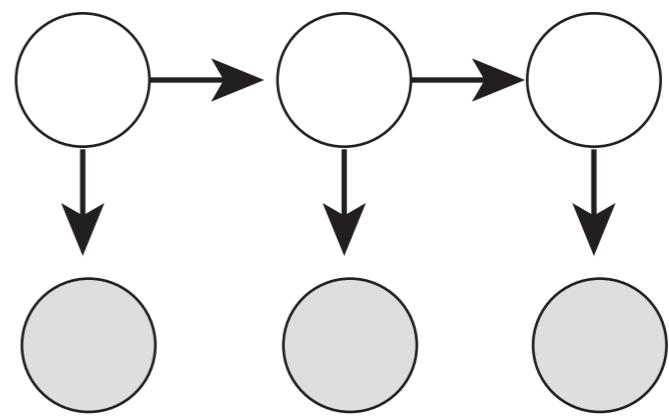
- speech recognition
- natural language modeling
- online handwritten recognition
- structure of proteins, DNA





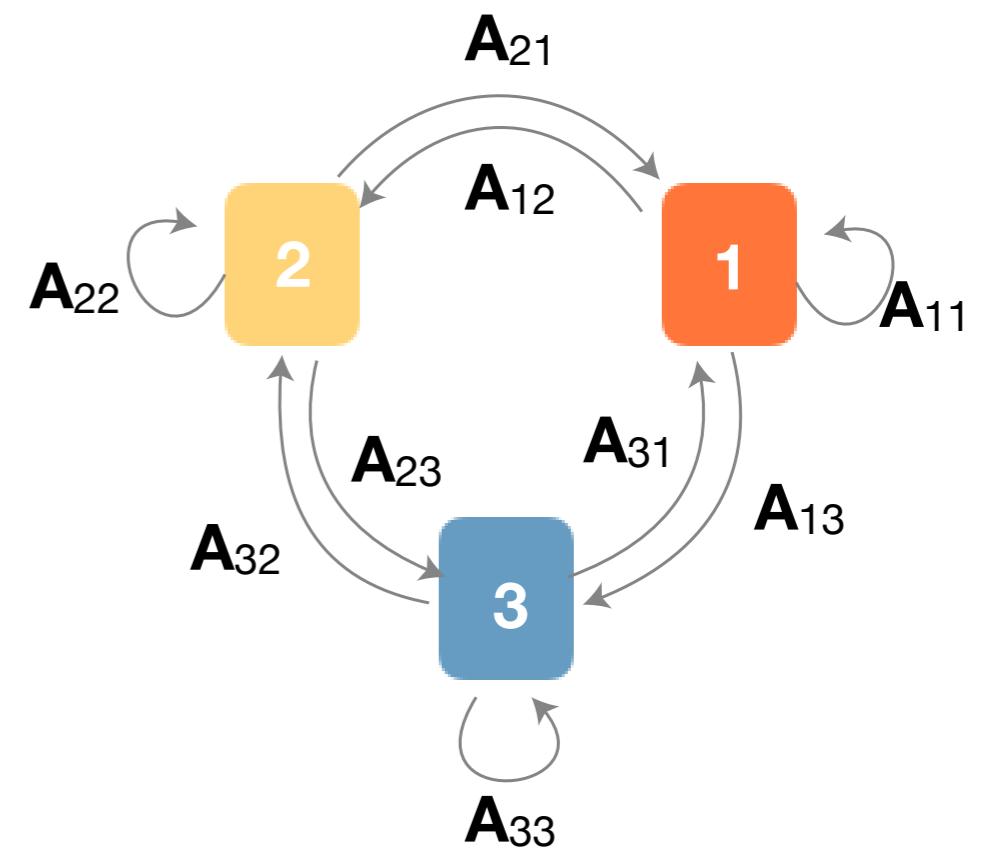
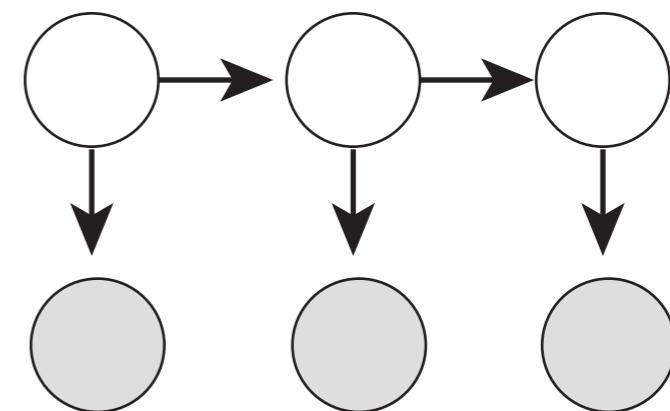
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



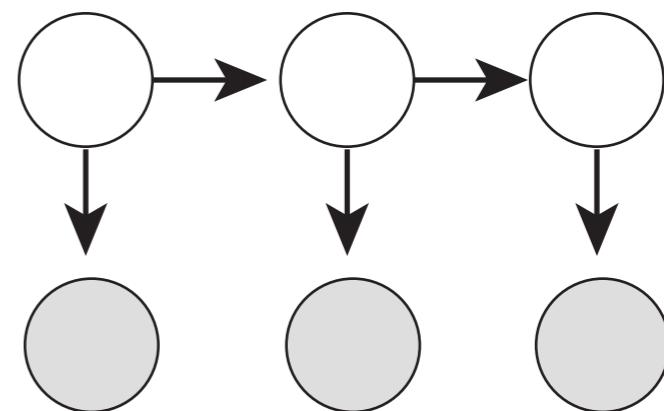
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



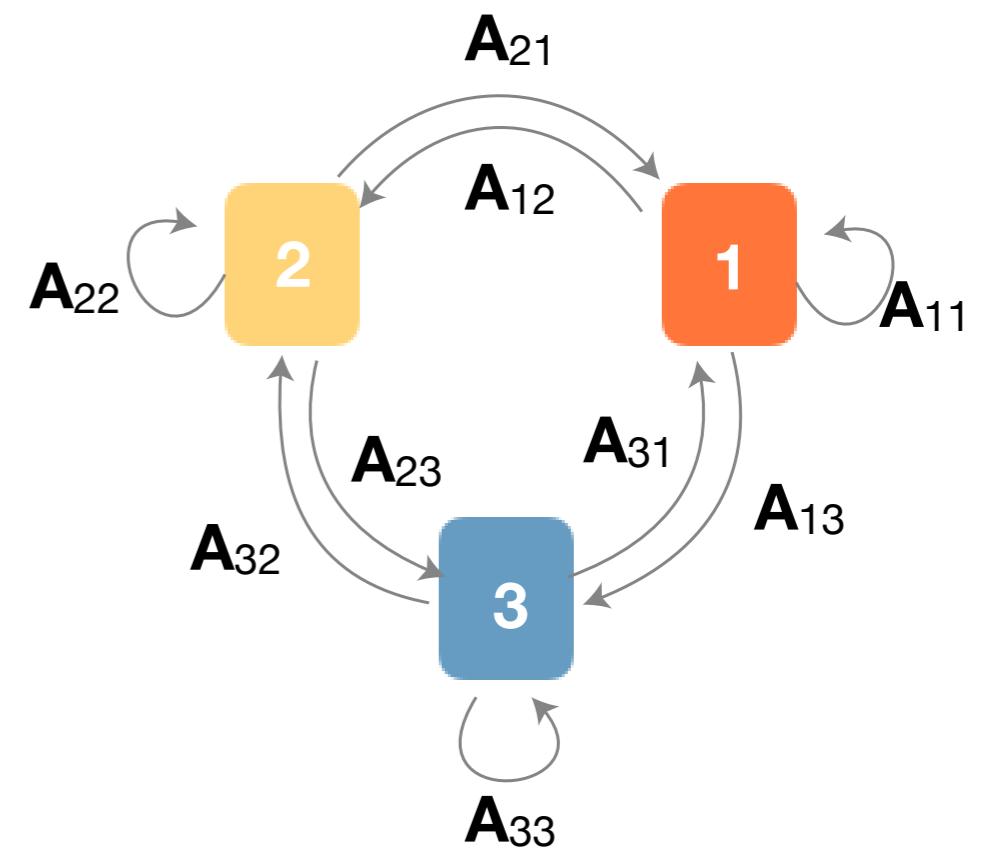
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



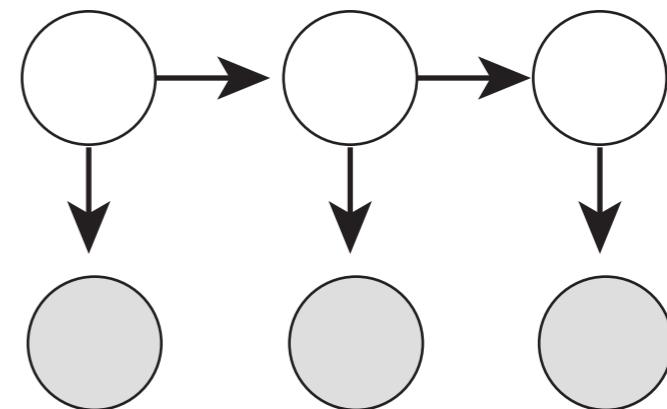
Transition probabilities: A

$$A_{jk} = P(z_{i+1} = k | z_i = j)$$



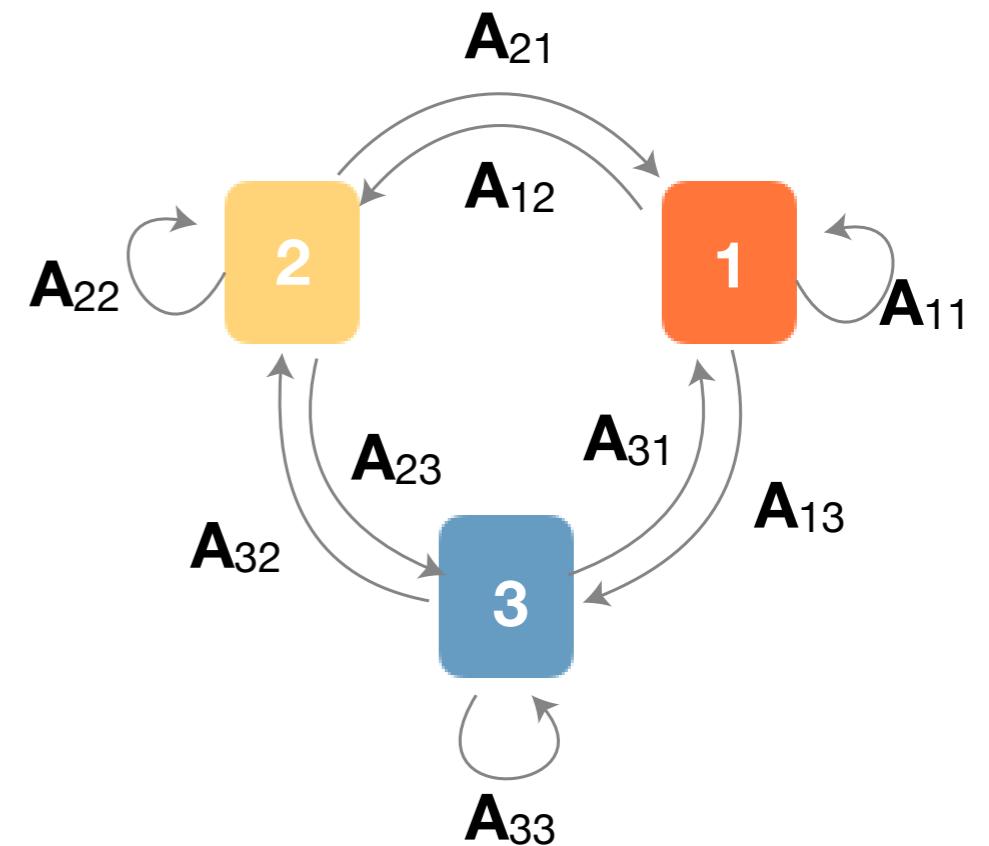
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



Transition probabilities: A

$$A_{jk} = P(z_{i+1} = k | z_i = j)$$

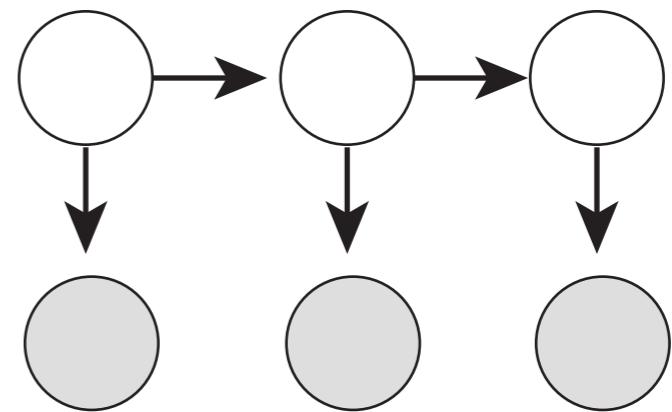


$$0 \leq A_{j,k} \leq 1$$

$$\sum_k A_{j,k} = 1$$

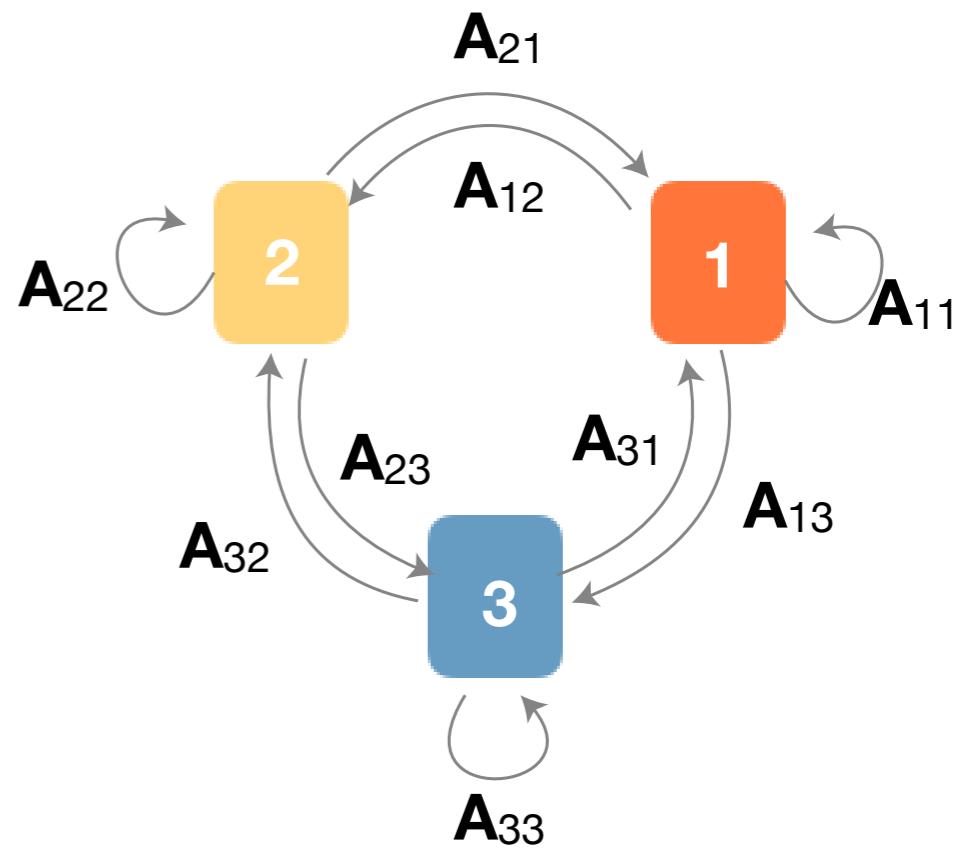
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



Transition probabilities: A

$$A_{jk} = P(z_{i+1} = k | z_i = j)$$



Emissions model for observations x_i

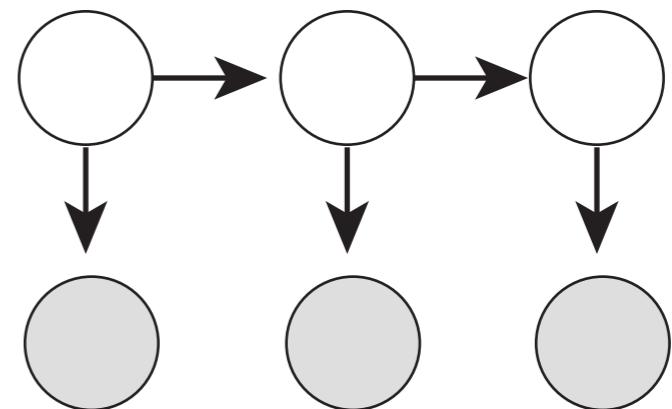
$$P(x_i | z_i)$$

$$0 \leq A_{j,k} \leq 1$$

$$\sum_k A_{j,k} = 1$$

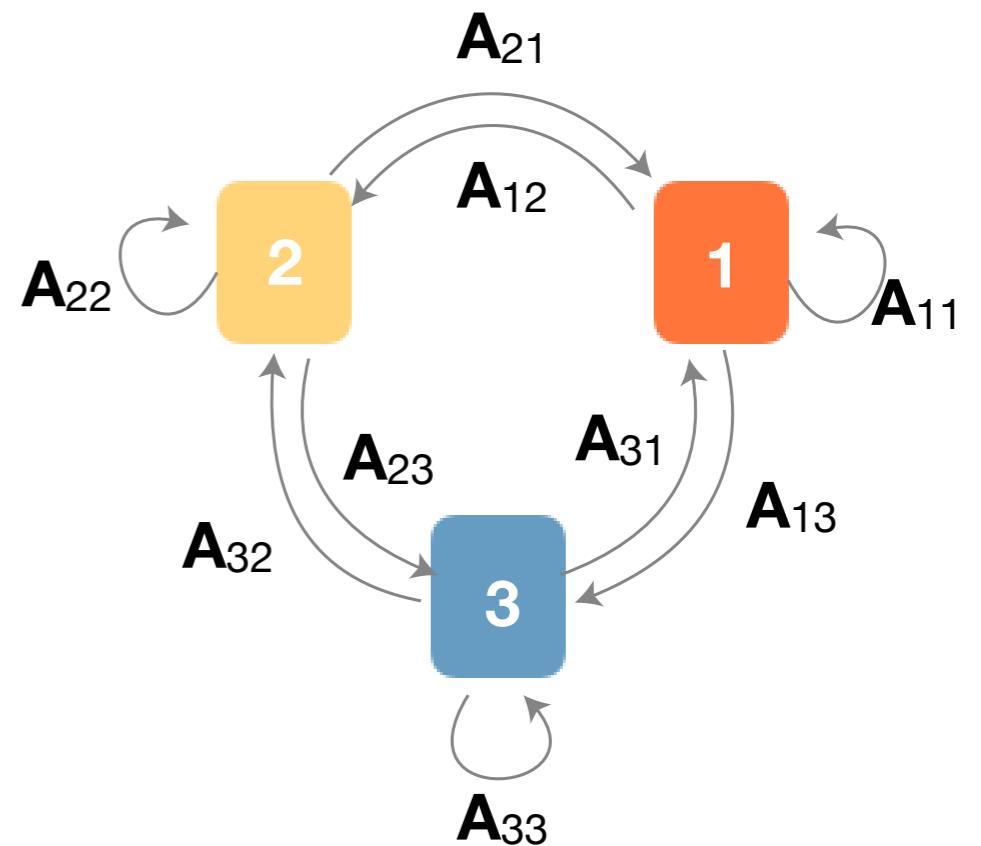
Hidden Markov models: details

K district latents, $z_i = \{1, 2, \dots, K\}$



Transition probabilities: \mathbf{A}

$$A_{jk} = P(z_{i+1} = k | z_i = j)$$



Emissions model for observations \mathbf{x}_i

$$P(\mathbf{x}_i | z_i)$$

Initial state, z_0

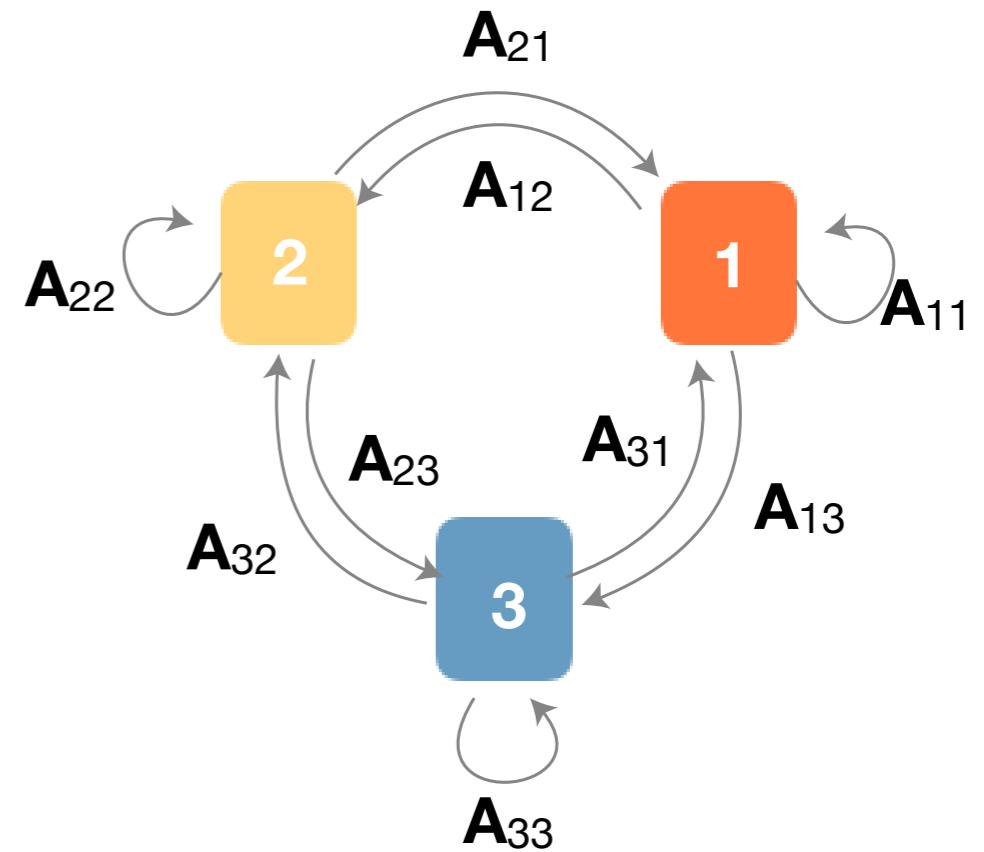
$$P(z_1)$$

$$0 \leq A_{j,k} \leq 1$$

$$\sum_k A_{j,k} = 1$$

Alternative **1-in-K** (one hot) representation*:

\mathbf{z}_i - K-dimensional binary vector



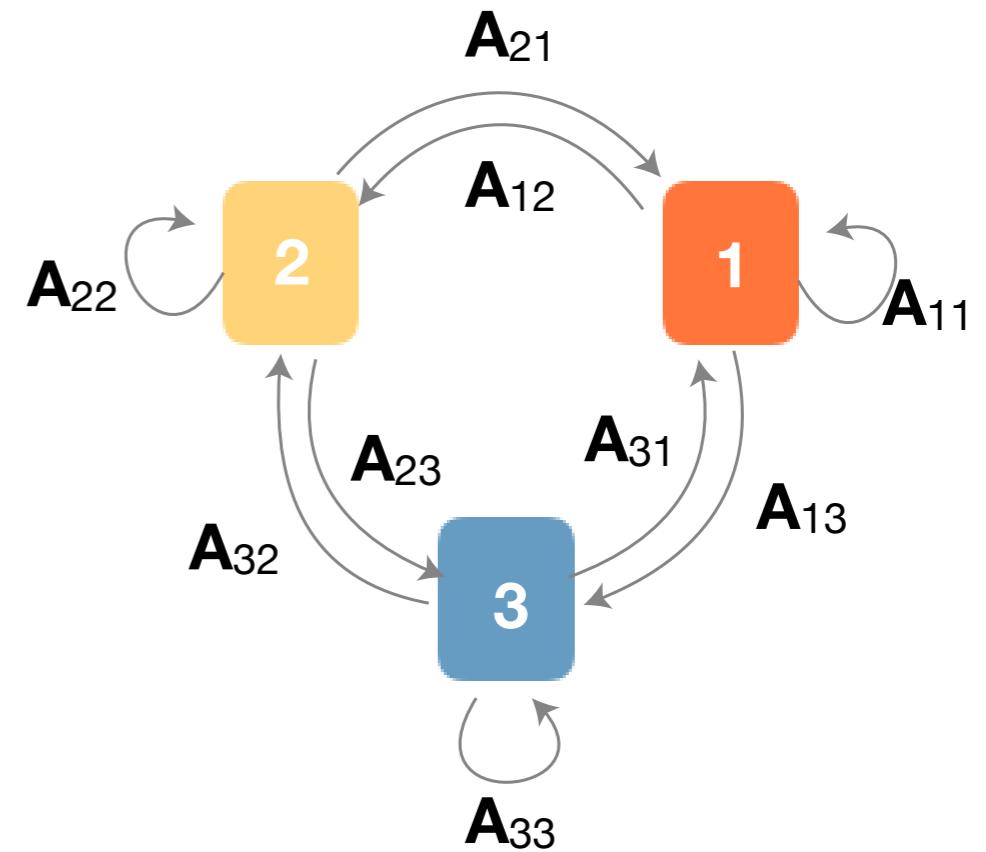
Initial state, \mathbf{z}_0 $P(\mathbf{z}_1) =$

*Note: This seems unnecessarily complicated at this point, but will turn out convenient when we want to represent beliefs about the state of the latent variables

Alternative **1-in-K** (one hot) representation*:

\mathbf{z}_i - K-dimensional binary vector

$$A_{jk} = P(\mathbf{z}_{i+1,k} = 1 | \mathbf{z}_{i,j} = 1)$$



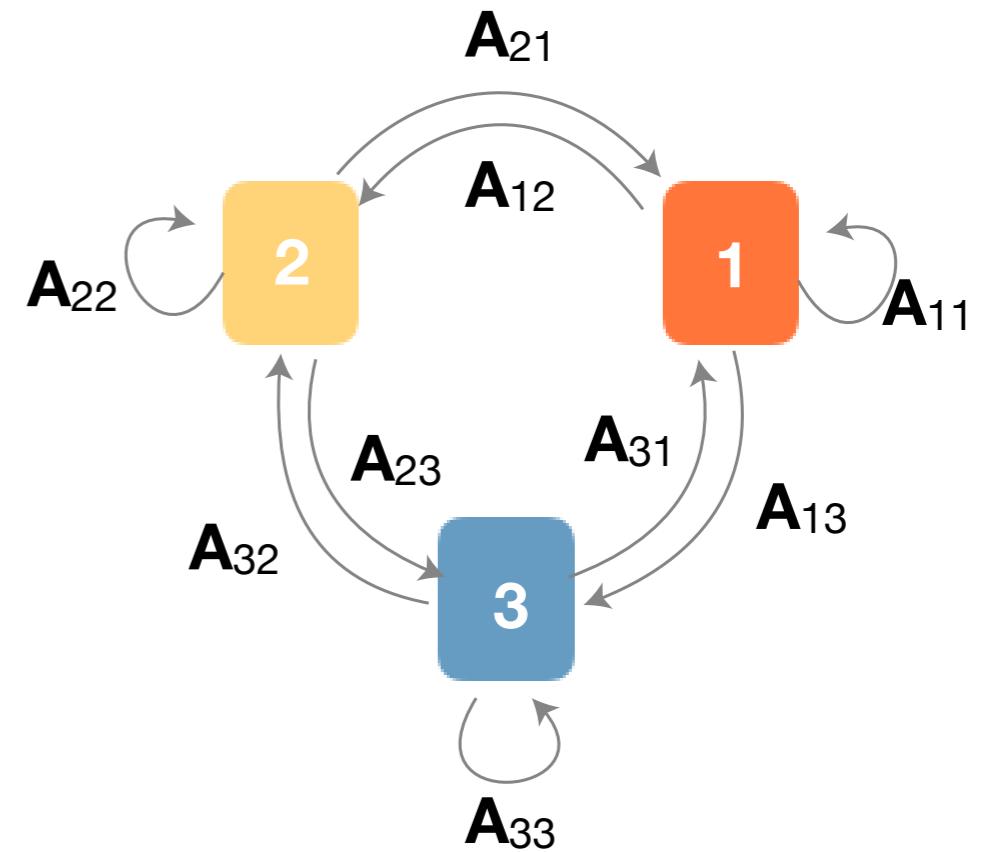
Initial state, \mathbf{z}_0 $P(\mathbf{z}_1) =$

*Note: This seems unnecessarily complicated at this point, but will turn out convenient when we want to represent beliefs about the state of the latent variables

Alternative **1-in-K** (one hot) representation*:

\mathbf{z}_i - K-dimensional binary vector

$$A_{jk} = P(\mathbf{z}_{i+1,k} = 1 | \mathbf{z}_{i,j} = 1)$$



Initial state, \mathbf{z}_0 $P(\mathbf{z}_1) =$

*Note: This seems unnecessarily complicated at this point, but will turn out convenient when we want to represent beliefs about the state of the latent variables

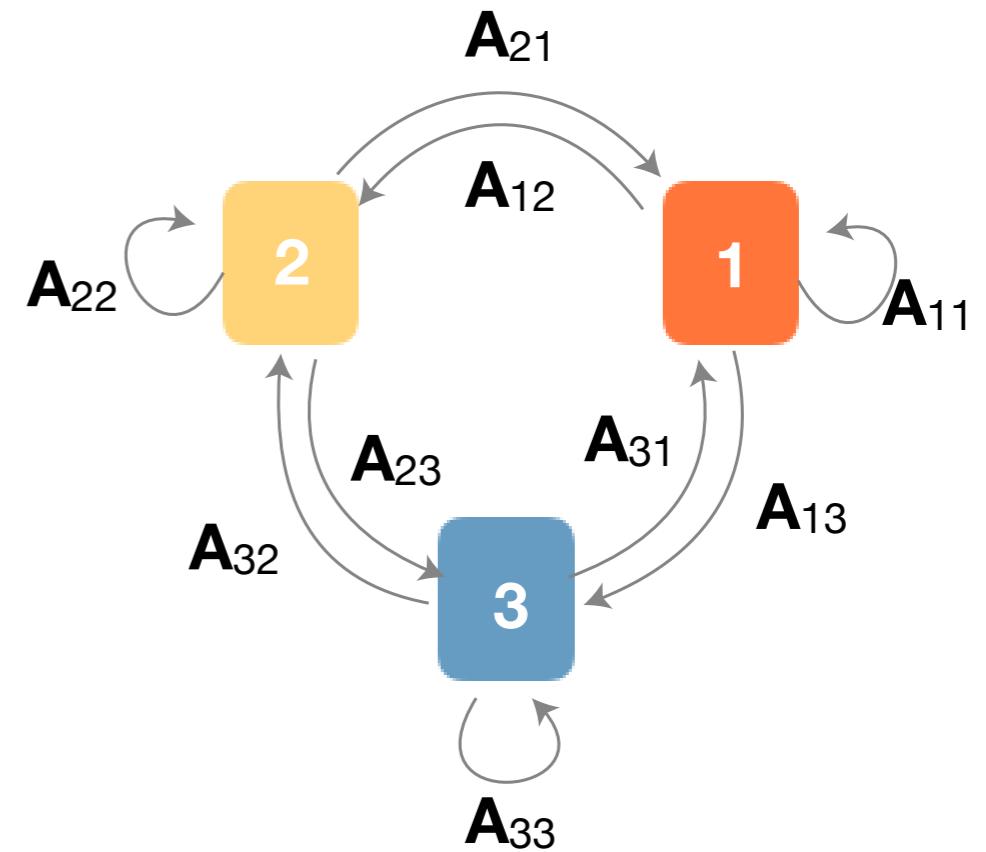
Alternative **1-in-K** (one hot) representation*:

\mathbf{z}_i - K-dimensional binary vector

$$A_{jk} = P(z_{i+1,k} = 1 | z_{i,j} = 1)$$

$$P(z_{i+1} | z_i) = \prod_{j,k} A_{j,k}^{z_{i,j} \cdot z_{i+1,k}}$$

Initial state, \mathbf{z}_0 $P(z_1) =$



*Note: This seems unnecessarily complicated at this point, but will turn out convenient when we want to represent beliefs about the state of the latent variables

Alternative **1-in-K** (one hot) representation*:

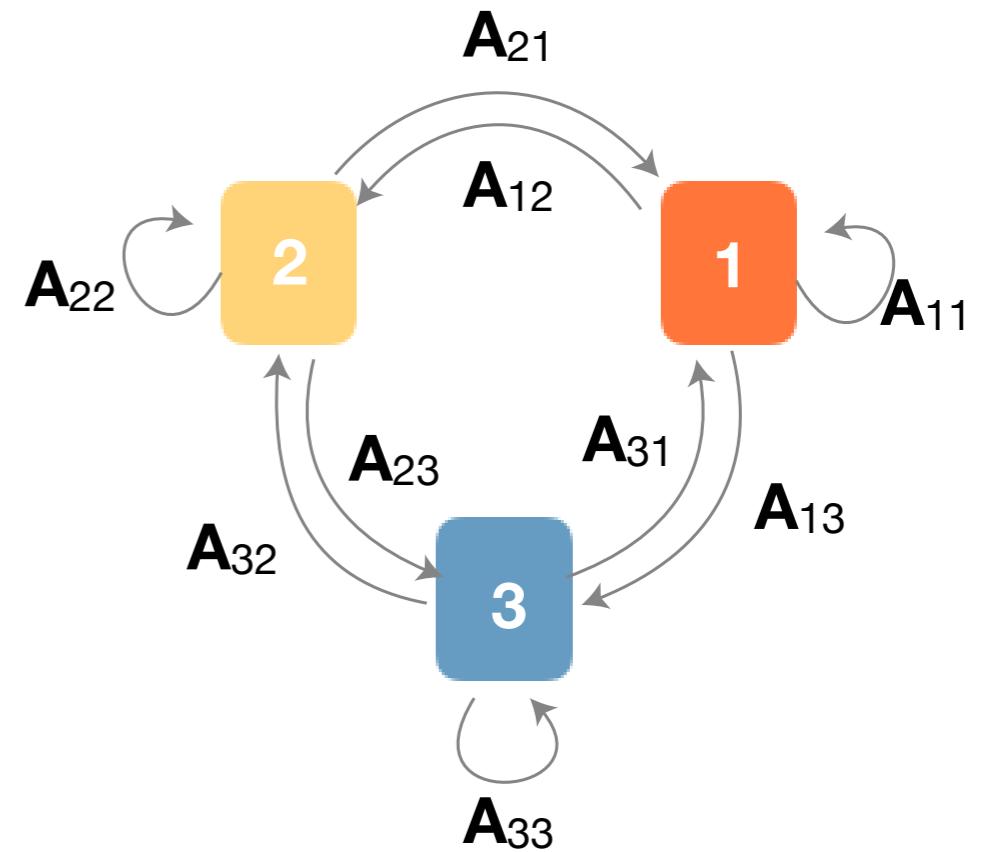
\mathbf{z}_i - K-dimensional binary vector

$$A_{jk} = P(z_{i+1,k} = 1 | z_{i,j} = 1)$$

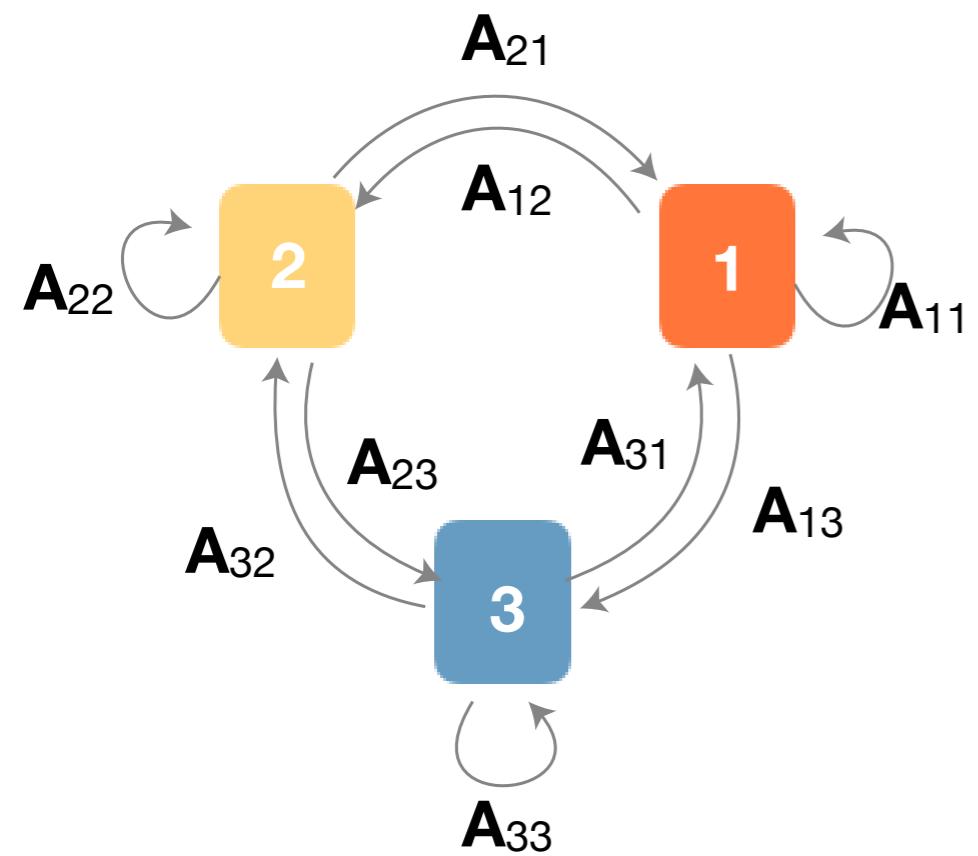
$$P(z_{i+1} | z_i) = \prod_{j,k} A_{j,k}^{z_{i,j} \cdot z_{i+1,k}}$$

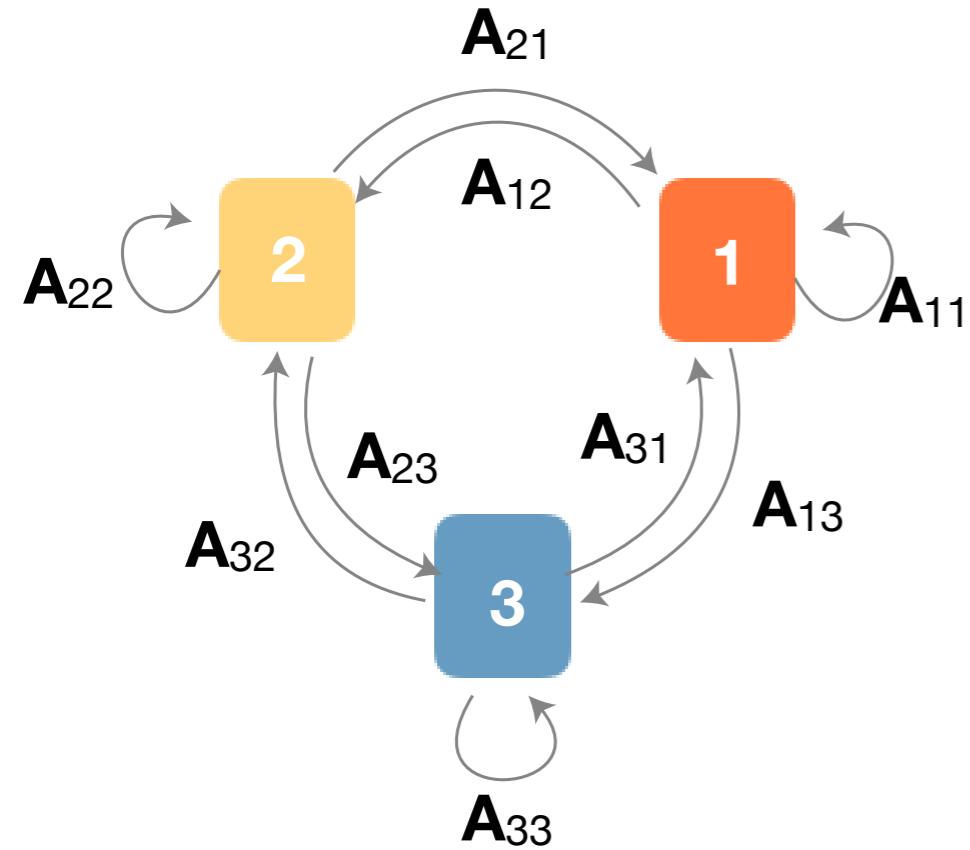
Initial state, \mathbf{z}_0

$$P(z_1) = \prod_k \pi_k^{z_{1,k}}$$

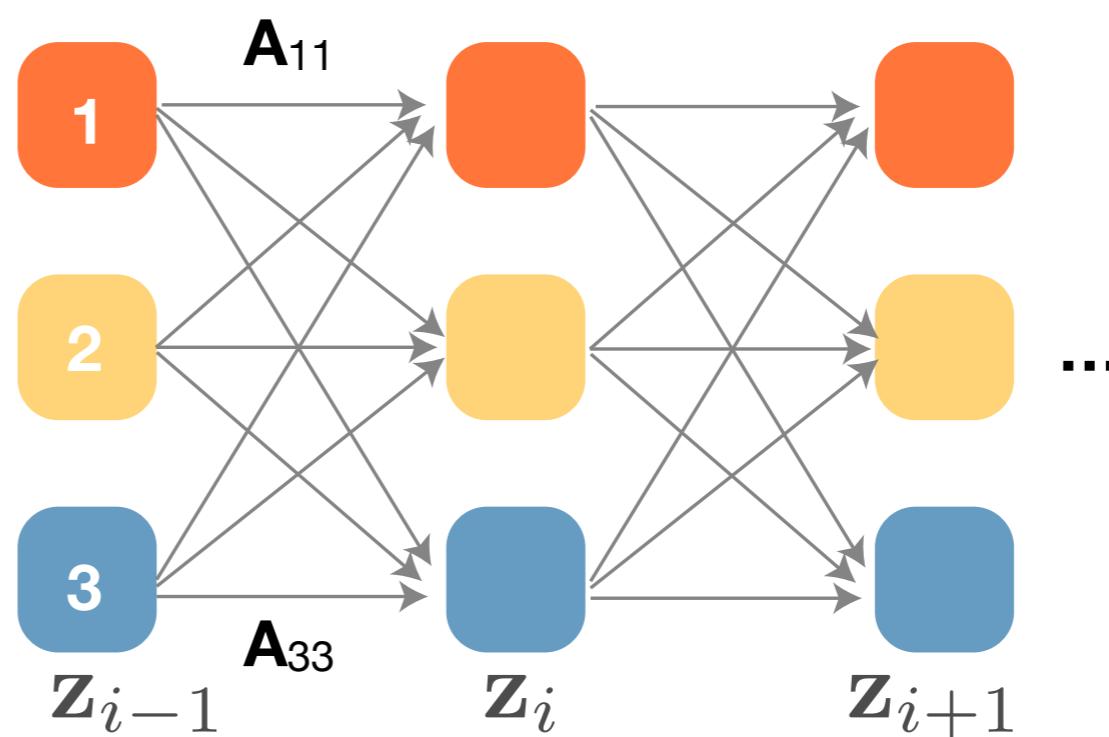


*Note: This seems unnecessarily complicated at this point, but will turn out convenient when we want to represent beliefs about the state of the latent variables



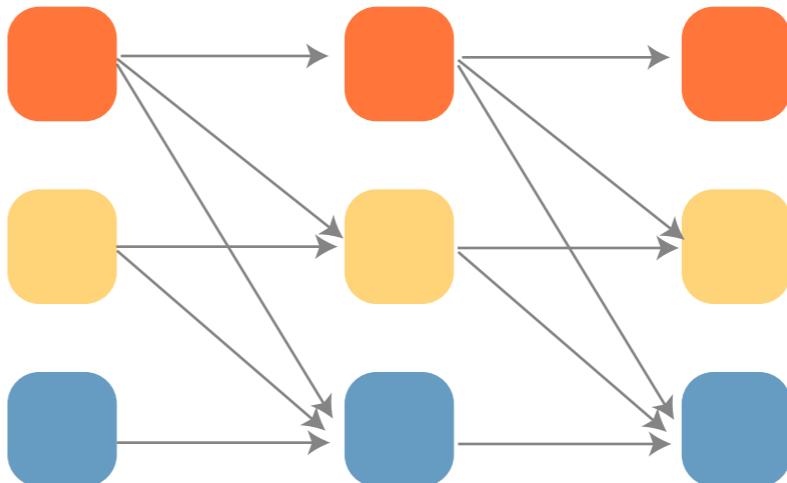


unfold over time:

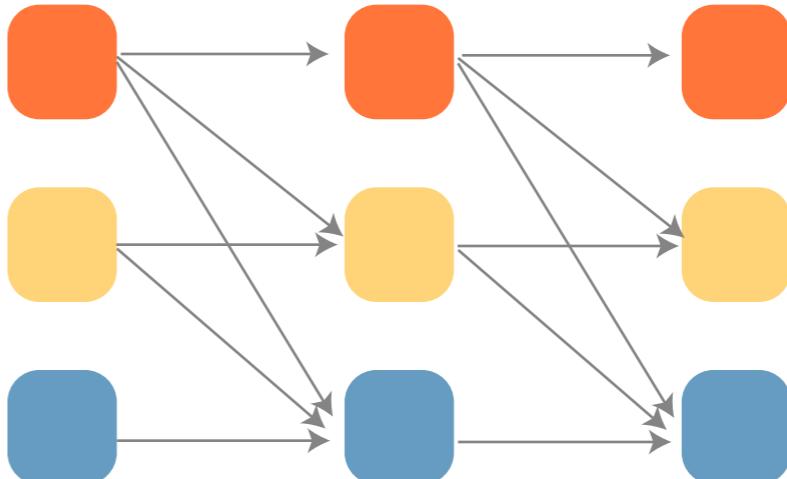


trellis representation

natural ordering of
components:
left-to-right HMMs

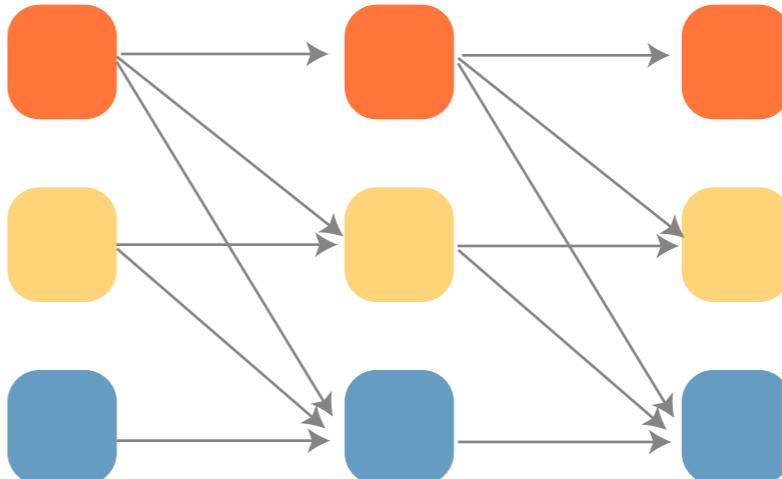


natural ordering of
components:
left-to-right HMMs



Example: real time digit recognition

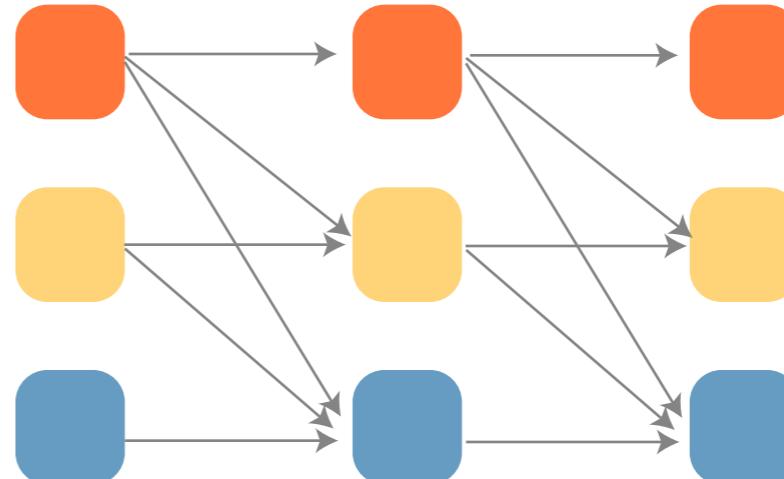
natural ordering of
components:
left-to-right HMMs



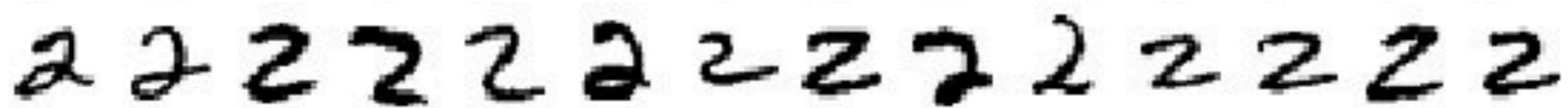
Example: real time digit recognition

2 2 2 2 2 2 2 2 2 2 2 2

natural ordering of
components:
left-to-right HMMs

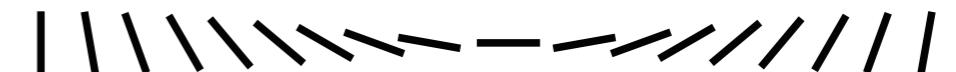


Example: real time digit recognition



probabilistic transition between components: rescaling, variation in length

z: lines fixes length, different orientations



1. Inference

2. Parameter learning

1. Inference

the full posterior is $P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t})$

2. Parameter learning

1. Inference

the full posterior is $P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t})$

$\mathbf{z}_{1:t}$ has K^t possible configurations, so computationally intractable

2. Parameter learning

1. Inference

the full posterior is $P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t})$

$\mathbf{z}_{1:t}$ has K^t possible configurations, so computationally intractable

What we can do is to compute exact **marginals**: **alpha-beta algorithm** (Baum-Welch)
(as we did in LDS with Kalman smoother)

1. Inference

the full posterior is $P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t})$

$\mathbf{z}_{1:t}$ has K^t possible configurations, so computationally intractable

What we can do is to compute exact **marginals: alpha-beta algorithm** (Baum-Welch)
(as we did in LDS with Kalman smoother)

We may also want to determine the **most likely sequence (MAP)**
Viterbi algorithm

1. Inference

the full posterior is $P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t})$

$\mathbf{z}_{1:t}$ has K^t possible configurations, so computationally intractable

What we can do is to compute exact **marginals**: **alpha-beta algorithm** (Baum-Welch)
(as we did in LDS with Kalman smoother)

We may also want to determine the **most likely sequence (MAP)**
Viterbi algorithm

2. Parameter learning

This is just more **EM**

1. Inference: marginals

We represent **posterior marginals** and joints using

$$\begin{aligned}\gamma(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_*, \theta^{\text{old}}) \\ \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) &= P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_*, \theta^{\text{old}})\end{aligned}$$

1. Inference: marginals

We represent **posterior marginals** and joints using

$$\begin{aligned}\gamma(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_*, \theta^{\text{old}}) && \text{dimensionality?} \\ \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) &= P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_*, \theta^{\text{old}})\end{aligned}$$

1. Inference: marginals

We represent **posterior marginals** and joints using

$$\begin{aligned}\gamma(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_*, \theta^{\text{old}}) && \text{dimensionality?} \\ \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) &= P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_*, \theta^{\text{old}})\end{aligned}$$

With the corresponding **expectations**:

1. Inference: marginals

We represent **posterior marginals** and joints using

$$\begin{aligned}\gamma(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_*, \theta^{\text{old}}) && \text{dimensionality?} \\ \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) &= P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_*, \theta^{\text{old}})\end{aligned}$$

With the corresponding **expectations**:

$$\begin{aligned}\gamma(z_{ik}) &= \mathbb{E}[z_{i,k} | \mathbf{x}_*, \theta^{\text{old}}] = \sum_{\mathbf{z}_i} \gamma(\mathbf{z}_i) z_{i,k} \\ \xi(z_{i,j}, z_{i+1,k}) &= \mathbb{E}[z_{i,j} z_{i+1,k} | \mathbf{x}_*, \theta^{\text{old}}] = \sum_{\mathbf{z}_i, \mathbf{z}_{i+1}} \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) z_{i,j} z_{i+1,k}\end{aligned}$$

1. Inference: marginals

We represent **posterior marginals** and joints using

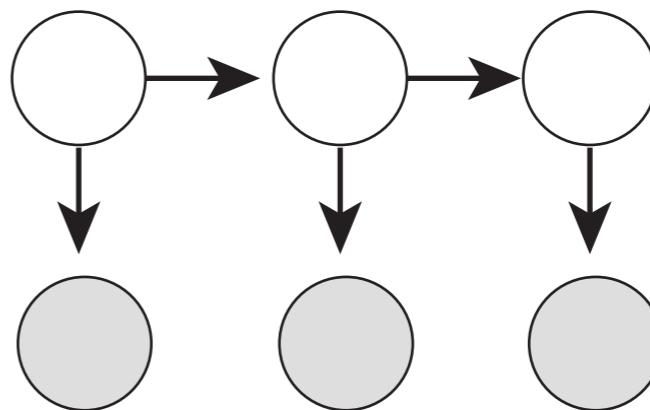
$$\begin{aligned}\gamma(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_*, \theta^{\text{old}}) && \text{dimensionality?} \\ \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) &= P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_*, \theta^{\text{old}})\end{aligned}$$

With the corresponding **expectations**:

$$\begin{aligned}\gamma(z_{ik}) &= \mathbb{E}[z_{i,k} | \mathbf{x}_*, \theta^{\text{old}}] = \sum_{\mathbf{z}_i} \gamma(\mathbf{z}_i) z_{i,k} \\ \xi(z_{i,j}, z_{i+1,k}) &= \mathbb{E}[z_{i,j} z_{i+1,k} | \mathbf{x}_*, \theta^{\text{old}}] = \sum_{\mathbf{z}_i, \mathbf{z}_{i+1}} \xi(\mathbf{z}_i, \mathbf{z}_{i+1}) z_{i,j} z_{i+1,k}\end{aligned}$$

We seek an efficient way of computing these by using recursions
(dynamic programming, as for LDS)

**Reminder: useful
conditional independencies**



$$P(\mathbf{x}_{1:t} | \mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)$$

$$P(\mathbf{x}_{1:i} | \mathbf{z}_{i+1}, \mathbf{x}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_{i+1})$$

$$P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1}, \mathbf{x}_{i+1}) = P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1})$$

$$P(\mathbf{x}_{1:i} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_i)$$

$$P(\mathbf{x}_{i+1:t} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_{i+1})$$

$$P(\mathbf{x}_{1:t} | \mathbf{z}_i, \mathbf{z}_{i+1}) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1})$$

First, let's look at **marginals**

$$P(\mathbf{z}_i | \mathbf{x}_{1:t}, \theta^{\text{old}}) = \frac{P(\mathbf{x}_{1:t} | \mathbf{z}_i) P(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{P(\mathbf{x}_{1:i}, \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{\alpha(\mathbf{z}_i) \beta(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})}$$

First, let's look at **marginals**

$$P(\mathbf{z}_i | \mathbf{x}_{1:t}, \theta^{\text{old}}) = \frac{P(\mathbf{x}_{1:t} | \mathbf{z}_i) P(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{P(\mathbf{x}_{1:i}, \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{\alpha(\mathbf{z}_i) \beta(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})}$$

where we defined the ‘messages’

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i}, \mathbf{z}_i)$$

$$\beta(\mathbf{z}_i) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)$$

First, let's look at **marginals**

$$P(\mathbf{z}_i | \mathbf{x}_{1:t}, \theta^{\text{old}}) = \frac{P(\mathbf{x}_{1:t} | \mathbf{z}_i) P(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{P(\mathbf{x}_{1:i}, \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{\alpha(\mathbf{z}_i) \beta(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})}$$

where we defined the ‘messages’

$$\begin{aligned}\alpha(\mathbf{z}_i) &= P(\mathbf{x}_{1:i}, \mathbf{z}_i) \\ \beta(\mathbf{z}_i) &= P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)\end{aligned}$$

And the normalizing constant is computed by marginalizing \mathbf{z}_i :

First, let's look at **marginals**

$$P(\mathbf{z}_i | \mathbf{x}_{1:t}, \theta^{\text{old}}) = \frac{P(\mathbf{x}_{1:t} | \mathbf{z}_i) P(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{P(\mathbf{x}_{1:i}, \mathbf{z}_i) P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)}{P(\mathbf{x}_{1:t})} = \frac{\alpha(\mathbf{z}_i) \beta(\mathbf{z}_i)}{P(\mathbf{x}_{1:t})}$$

where we defined the ‘messages’

$$\begin{aligned}\alpha(\mathbf{z}_i) &= P(\mathbf{x}_{1:i}, \mathbf{z}_i) \\ \beta(\mathbf{z}_i) &= P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)\end{aligned}$$

And the normalizing constant is computed by marginalizing \mathbf{z}_i :

$$P(\mathbf{x}_{1:t}) = \sum_{\mathbf{z}_k} \alpha(\mathbf{z}_k) \beta(\mathbf{z}_k)$$

Recursion equations:

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \alpha(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$\beta(\mathbf{z}_i) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \beta(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

With initial conditions:

$$\begin{aligned}\alpha(\mathbf{z}_1) &= P(\mathbf{x}_1 | \mathbf{z}_1) P(\mathbf{z}_1) = \prod_k (\pi_k P(\mathbf{x}_1 | \phi_k))^{z_{1k}} \\ \beta(\mathbf{z}_t) &= 1\end{aligned}$$

Recursion equations:

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \alpha(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$\beta(\mathbf{z}_i) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \beta(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

With initial conditions:

$$\begin{aligned}\alpha(\mathbf{z}_1) &= P(\mathbf{x}_1 | \mathbf{z}_1) P(\mathbf{z}_1) = \prod_k (\pi_k P(\mathbf{x}_1 | \phi_k))^{z_{1k}} \\ \beta(\mathbf{z}_t) &= 1\end{aligned}$$

These quantities are enough to also compute the joint:

Recursion equations:

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \alpha(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$\beta(\mathbf{z}_i) = P(\mathbf{x}_{i+1:t} | \mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \beta(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

With initial conditions:

$$\begin{aligned}\alpha(\mathbf{z}_1) &= P(\mathbf{x}_1 | \mathbf{z}_1) P(\mathbf{z}_1) = \prod_k (\pi_k P(\mathbf{x}_1 | \phi_k))^{z_{1k}} \\ \beta(\mathbf{z}_t) &= 1\end{aligned}$$

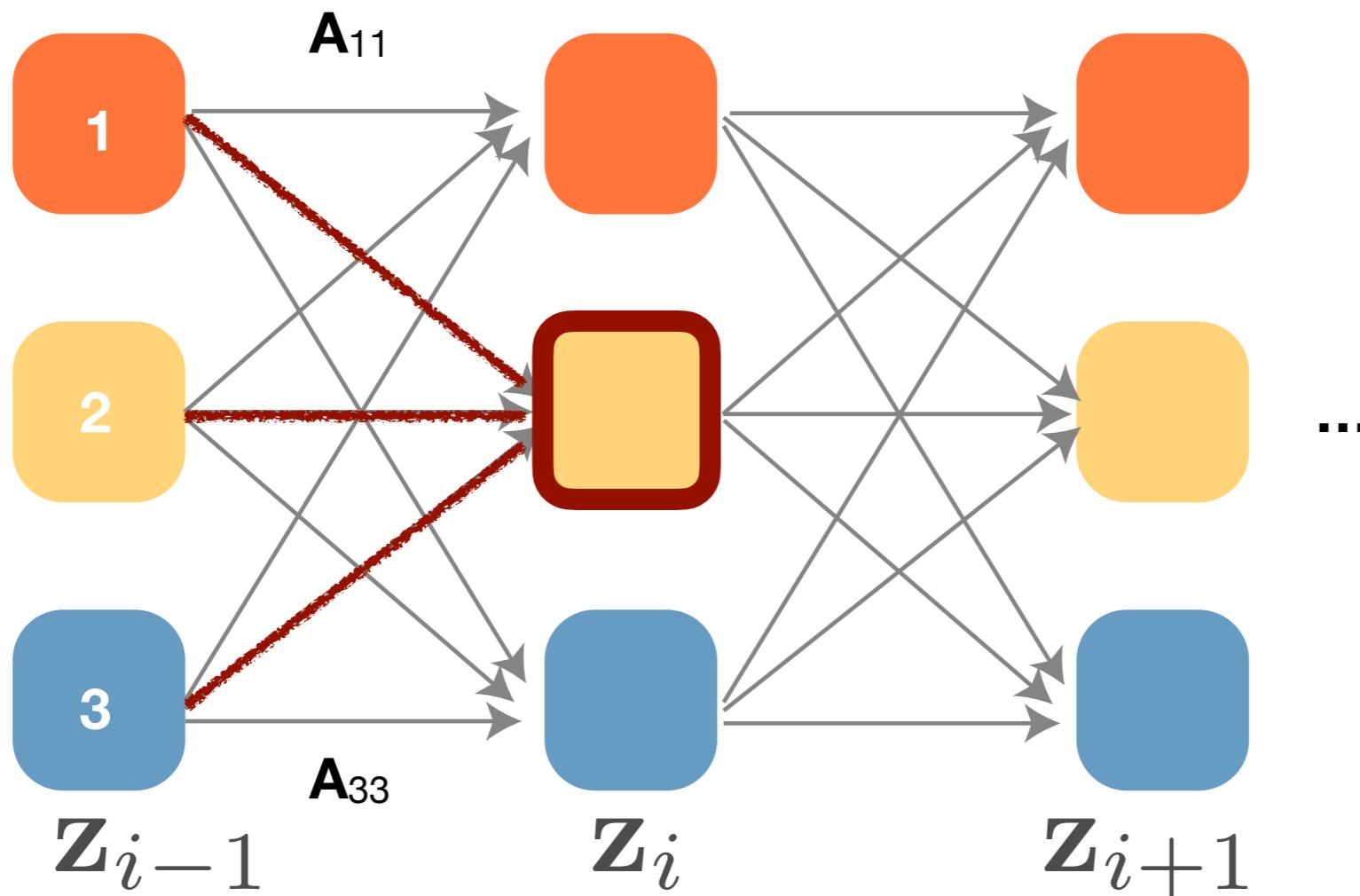
These quantities are enough to also compute the joint:

$$P(\mathbf{z}_i, \mathbf{z}_{i+1} | \mathbf{x}_{1:t}) = \frac{\alpha(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \beta(\mathbf{z}_{i+1})}{P(\mathbf{x}_{1:t})}$$

$$\begin{aligned}
\alpha(\mathbf{z}_i) &= P(\mathbf{x}_{1:i}, \mathbf{z}_i) = P(\mathbf{x}_{1:i} | \mathbf{z}_i) P(\mathbf{z}_i) \\
&= P(\mathbf{x}_{1:i-1} | \mathbf{z}_i) P(\mathbf{x}_i | \mathbf{z}_i) P(\mathbf{z}_i) \\
&= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} P(\mathbf{x}_{1:i-1}, \mathbf{z}_{i-1}, \mathbf{z}_i) \\
&= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} P(\mathbf{x}_{1:i-1} | \mathbf{z}_{i-1}, \mathbf{z}_i) P(\mathbf{z}_i, \mathbf{z}_{i-1}) \\
&= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} P(\mathbf{x}_{1:i-1} | \mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1}) P(\mathbf{z}_{i-1}) \\
&= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} P(\mathbf{x}_{1:i-1}, \mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1}) \\
&= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \alpha(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})
\end{aligned}$$

$$\begin{aligned}
\beta(\mathbf{z}_i) &= P(\mathbf{x}_{i+1:t} | \mathbf{z}_i) \\
&= \sum_{\mathbf{z}_{i+1}} P(\mathbf{x}_{i+1:t}, \mathbf{z}_{i+1} | \mathbf{z}_i) \\
&= \sum_{\mathbf{z}_{i+1}} P(\mathbf{x}_{i+1:t} | \mathbf{z}_{i+1}, \mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) \\
&= \sum_{\mathbf{z}_{i+1}} P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{x}_{i+2:t} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i) \\
&= \sum_{\mathbf{z}_{i+1}} \beta(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)
\end{aligned}$$

Interpretation: sum-product



$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \alpha(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

consider **all possible ways** to reach state $\mathbf{z}_{i,k}$ and **sum** them up,
then combine with local evidence

Implementational caveat (a rather important one)

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i}, \mathbf{z}_i)$$

joint distribution over increasingly many things,
the probability of any particular configuration is vanishingly small,
easy to underflow machine precision

Implementational caveat (a rather important one)

$$\alpha(\mathbf{z}_i) = P(\mathbf{x}_{1:i}, \mathbf{z}_i)$$

joint distribution over increasingly many things,
the probability of any particular configuration is vanishingly small,
easy to underflow machine precision

Solution: rescale messages to keep things in sensible range

$$\begin{aligned}\hat{\alpha}(\mathbf{z}_i) &= P(\mathbf{z}_i | \mathbf{x}_{1:i}) = \frac{\alpha(\mathbf{z}_i)}{P(\mathbf{x}_{1:i})} \\ \hat{\beta}(\mathbf{z}_i) &= \frac{P(\mathbf{x}_{i+1:t} | \mathbf{z}_i)}{P(\mathbf{x}_{i+1:t} | \mathbf{x}_{1:i})} = \frac{\beta(\mathbf{z}_i)}{\prod_{j=i+1:t} c_j}\end{aligned}$$

where we've introduced
an intermediate variable

$$c_i = P(\mathbf{x}_i | \mathbf{x}_{1:i-1})$$
$$P(\mathbf{x}_{1:i}) = \prod_{j=1:i} c_j$$

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$\begin{aligned} c_i \hat{\alpha}(\mathbf{z}_i) &= P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1}) \\ c_{i+1} \hat{\beta}(\mathbf{z}_i) &= \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i) \end{aligned}$$

Finally the new expressions for the posterior marginals are:

$$\begin{aligned} \gamma(\mathbf{z}_i) &= \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i) \\ \xi(z_{i,j}, z_{i+1,k}) &= {c_{i+1}}^{-1} \hat{\alpha}(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \hat{\beta}(\mathbf{z}_{i+1}) \end{aligned}$$

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$c_i \hat{\alpha}(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

check at home!

$$c_{i+1} \hat{\beta}(\mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

Finally the new expressions for the posterior marginals are:

$$\begin{aligned} \gamma(\mathbf{z}_i) &= \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i) \\ \xi(z_{i,j}, z_{i+1,k}) &= c_{i+1}^{-1} \hat{\alpha}(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \hat{\beta}(\mathbf{z}_{i+1}) \end{aligned}$$

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$c_i \hat{\alpha}(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$c_{i+1} \hat{\beta}(\mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

check at home!

Finally the new expressions for the posterior marginals are:

$$\gamma(\mathbf{z}_i) = \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i)$$

$$\xi(z_{i,j}, z_{i+1,k}) = c_{i+1}^{-1} \hat{\alpha}(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \hat{\beta}(\mathbf{z}_{i+1})$$

Formal link to LDS:

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$c_i \hat{\alpha}(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$c_{i+1} \hat{\beta}(\mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

check at home!

Finally the new expressions for the posterior marginals are:

$$\gamma(\mathbf{z}_i) = \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i)$$

$$\xi(z_{i,j}, z_{i+1,k}) = c_{i+1}^{-1} \hat{\alpha}(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \hat{\beta}(\mathbf{z}_{i+1})$$

Formal link to LDS:

Kalman filtering: $\hat{\alpha}(\mathbf{z}_i)$

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$c_i \hat{\alpha}(\mathbf{z}_i) = P(\mathbf{x}_i | \mathbf{z}_i) \sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1}) P(\mathbf{z}_i | \mathbf{z}_{i-1})$$

$$c_{i+1} \hat{\beta}(\mathbf{z}_i) = \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1}) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) P(\mathbf{z}_{i+1} | \mathbf{z}_i)$$

check at home!

Finally the new expressions for the posterior marginals are:

$$\gamma(\mathbf{z}_i) = \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i)$$

$$\xi(z_{i,j}, z_{i+1,k}) = c_{i+1}^{-1} \hat{\alpha}(\mathbf{z}_i) P(\mathbf{z}_{i+1} | \mathbf{z}_i) P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) \hat{\beta}(\mathbf{z}_{i+1})$$

Formal link to LDS:

Kalman filtering: $\hat{\alpha}(\mathbf{z}_i)$

Kalman smoothing: $\gamma(\mathbf{z}_i) = \hat{\alpha}(\mathbf{z}_i) \hat{\beta}(\mathbf{z}_i)$

1. Inference: MAP

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

The Viterbi algorithm

(this is also an instance of dynamic programming)

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

The Viterbi algorithm

(this is also an instance of dynamic programming)

General idea: somebody told you the probability of the most likely configuration is up to point i , and its end point, can you use that to figure out the most likely configuration up to \mathbf{z}_{i+1}

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

The Viterbi algorithm

(this is also an instance of dynamic programming)

General idea: somebody told you the probability of the most likely configuration is up to point i , and its end point, can you use that to figure out the most likely configuration up to \mathbf{z}_{i+1}

Yet another instance of forward message passing:

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

The Viterbi algorithm

(this is also an instance of dynamic programming)

General idea: somebody told you the probability of the most likely configuration is up to point i , and its end point, can you use that to figure out the most likely configuration up to \mathbf{z}_{i+1}

Yet another instance of forward message passing:

why not conditional?
why log?

1. Inference: MAP

$$\hat{\mathbf{z}}_{1:t} = \operatorname{argmax}_{\mathbf{z}_{1:t}} \{ P(\mathbf{z}_{1:t} | \mathbf{x}_{1:t}) \}$$

The Viterbi algorithm

(this is also an instance of dynamic programming)

General idea: somebody told you the probability of the most likely configuration is up to point i , and its end point, can you use that to figure out the most likely configuration up to \mathbf{z}_{i+1}

Yet another instance of forward message passing:

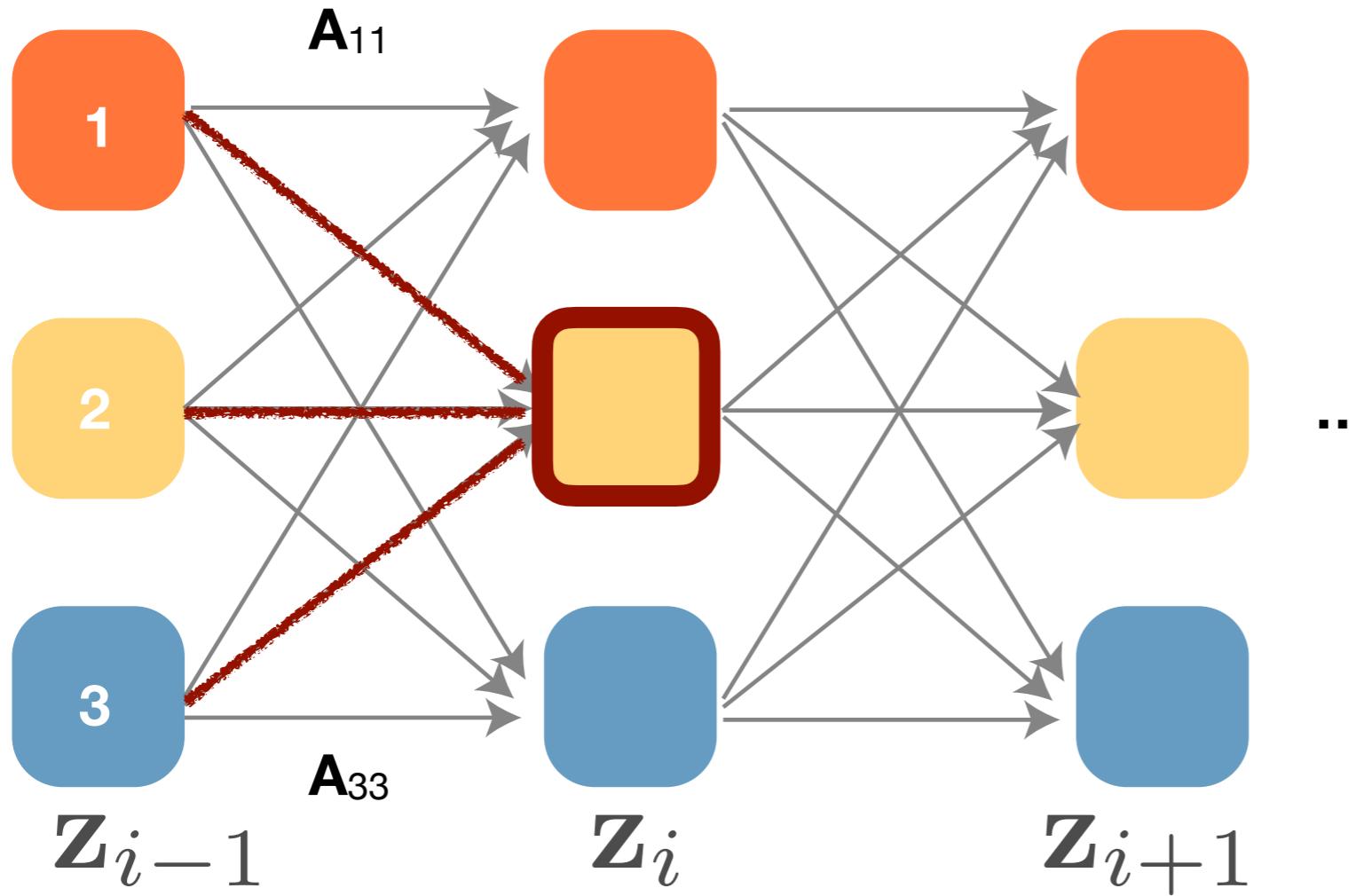
why not conditional?
why log?

With recursion:

$$w_{i+1} = \log P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) + \max_{\mathbf{z}_i} \{ \log P(\mathbf{z}_{i+1} | \mathbf{z}_i) + w_i(\mathbf{z}_i) \}$$

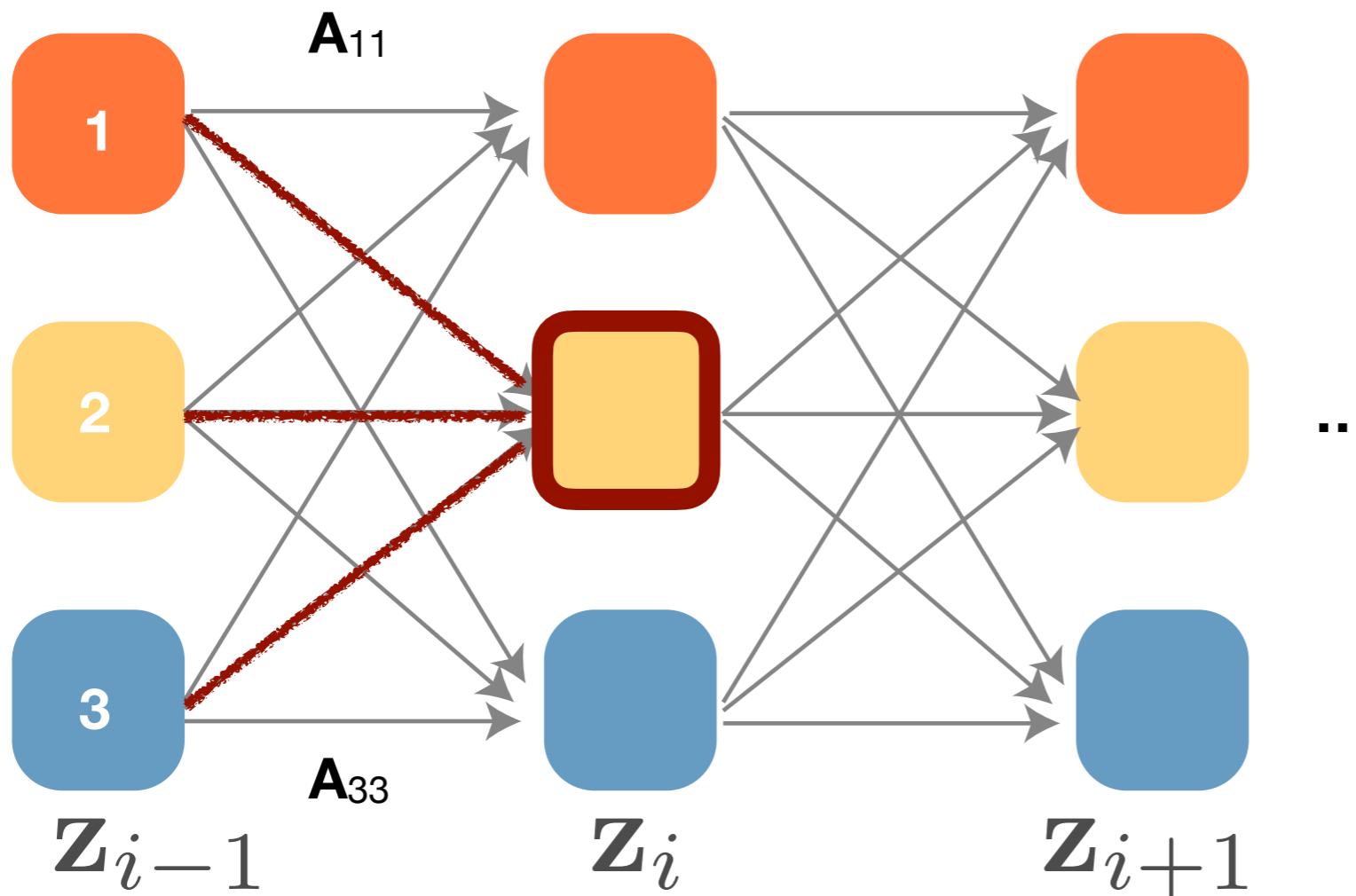
And initial condition: $w_1 = \log P(\mathbf{z}_1) + \log P(\mathbf{x}_1 | \mathbf{z}_1)$

Interpretation: max-product



$$w_{i+1} = \log P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) + \max_{\mathbf{z}_i} \{ \log P(\mathbf{z}_{i+1} | \mathbf{z}_i) + w_i(\mathbf{z}_i) \}$$

Interpretation: max-product

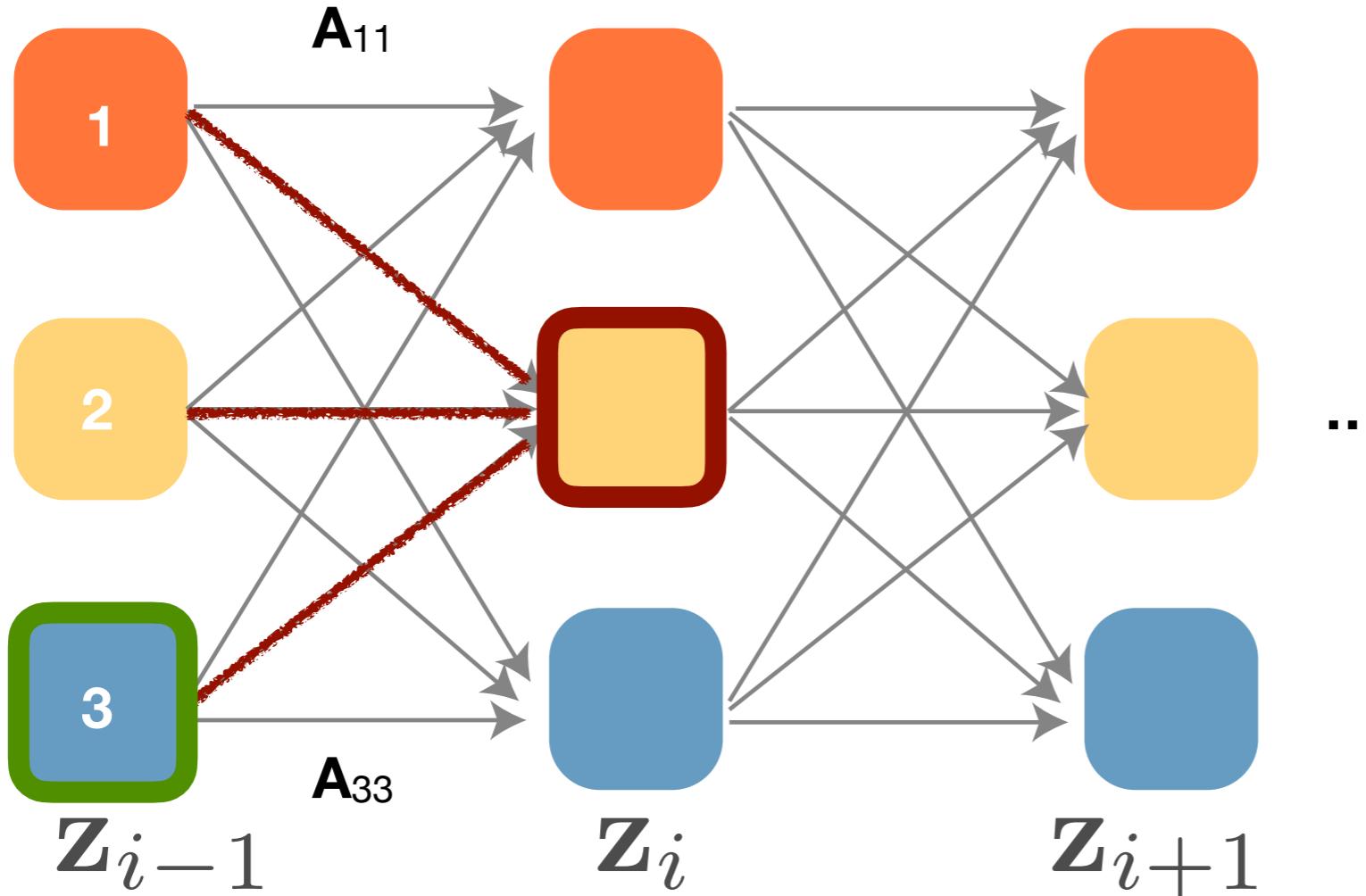


$$w_{i+1} = \log P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) + \max_{\mathbf{z}_i} \{\log P(\mathbf{z}_{i+1} | \mathbf{z}_i) + w_i(\mathbf{z}_i)\}$$

consider **all possible ways** to reach state $\mathbf{z}_{i,k}$ and **pick the best**,
then combine with local evidence

Interpretation: max-product

$$k_i^{\max} = \operatorname{argmax}_{\mathbf{z}_i} \{ \log P(\mathbf{z}_{i+1} | \mathbf{z}_i) + w_i(\mathbf{z}_i) \}$$



+need to remember
how we got there

$$w_{i+1} = \log P(\mathbf{x}_{i+1} | \mathbf{z}_{i+1}) + \max_{\mathbf{z}_i} \{ \log P(\mathbf{z}_{i+1} | \mathbf{z}_i) + w_i(\mathbf{z}_i) \}$$

consider **all possible ways** to reach state $\mathbf{z}_{i,k}$ and **pick the best**,
then combine with local evidence

2. Parameter learning: EM

2. Parameter learning: EM

Goal: find parameters that maximize $P(\mathbf{x}|\theta) = \int_Z P(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z}$

2. Parameter learning: EM

Goal: find parameters that maximize $P(\mathbf{x}|\theta) = \int_Z P(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z}$

Remember the **general recipe**:

2. Parameter learning: EM

Goal: find parameters that maximize $P(\mathbf{x}|\theta) = \int_Z P(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z}$

Remember the **general recipe**:

E-step: $q(\mathbf{z}) = P(\mathbf{z}|\mathbf{x}, \theta^{\text{old}})$ fixed parameters θ^{old}

This we do using alpha-beta

2. Parameter learning: EM

Goal: find parameters that maximize $P(x|\theta) = \int_Z P(x, z|\theta) dz$

Remember the **general recipe**:

E-step: $q(z) = P(z|x, \theta^{\text{old}})$ fixed parameters θ^{old}

This we do using alpha-beta

M-step: $Q(\theta, \theta^{\text{old}}) = \int_Z P(z|x, \theta^{\text{old}}) \log P(x, z|\theta)$
fixed q, optimize parameters

**Separate terms for observation model,
latent state transitions and initial state,
optimize one at a time**

Learning: EM

Log likelihood: $\mathcal{L}(\theta) = \log P(\mathbf{x}_* | \theta)$

$$P(\mathbf{x}_*, \mathbf{z}_* | \theta) = P_\theta(\mathbf{z}_0) \prod_i P_\theta(\mathbf{z}_{i+1} | \mathbf{z}_i) \prod_i P_\theta(\mathbf{x}_i | \mathbf{z}_i)$$

Learning: EM

Log likelihood: $\mathcal{L}(\theta) = \log P(\mathbf{x}_* | \theta)$

$$P(\mathbf{x}_*, \mathbf{z}_* | \theta) = P_\theta(\mathbf{z}_0) \prod_i P_\theta(\mathbf{z}_{i+1} | \mathbf{z}_i) \prod_i P_\theta(\mathbf{x}_i | \mathbf{z}_i)$$

In general:

$$\begin{aligned}\pi_k^{\text{new}} &= \frac{\gamma(z_{1,k})}{\sum_j \gamma(z_{1,j})} \\ A_{jk}^{\text{new}} &= \frac{\sum_i \xi(z_{i,j}, z_{i+1,k})}{\sum_{i,l} \xi(z_{i,j}, z_{i+1,l})}\end{aligned}$$

Learning: EM

Log likelihood: $\mathcal{L}(\theta) = \log P(\mathbf{x}_* | \theta)$

$$P(\mathbf{x}_*, \mathbf{z}_* | \theta) = P_\theta(\mathbf{z}_0) \prod_i P_\theta(\mathbf{z}_{i+1} | \mathbf{z}_i) \prod_i P_\theta(\mathbf{x}_i | \mathbf{z}_i)$$

$$\pi_k^{\text{new}} = \frac{\gamma(z_{1,k})}{\sum_j \gamma(z_{1,j})}$$

In general:

$$A_{jk}^{\text{new}} = \frac{\sum_i \xi(z_{i,j}, z_{i+1,k})}{\sum_{i,l} \xi(z_{i,j}, z_{i+1,l})}$$

For a simple *gaussian* observation model

$$\mu_k^{\text{new}} = \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i$$

$$\Sigma_k^{\text{new}} = \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i \mathbf{x}_i^t - \mu_k \mu_k^t$$

What happens next:

What happens next:

HMMs lab: real data! (supervised setting, so model fit easier, no EM)
but you get to implement Viterbi inference

What happens next:

HMMs lab: real data! (supervised setting, so model fit easier, no EM)
but you get to implement Viterbi inference

next lecture: putting what we've learned in a bigger picture,
variations, what can and can't we do with these models,
where do we go when they are not enough

READING ASSIGNMENT: Roweis & Ghahramani