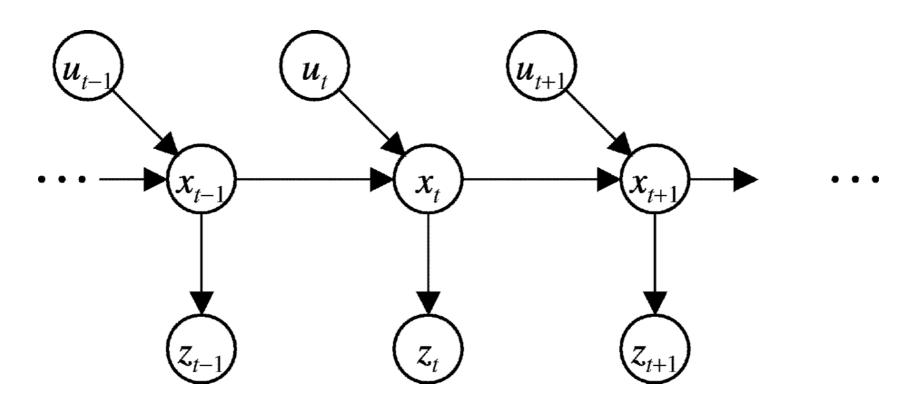


5. Hidden Markov Models

Bayes Filter (Rep.)

We can describe the overall process using a **Dynamic Bayes Network**:



This incorporates the following Markov assumptions:

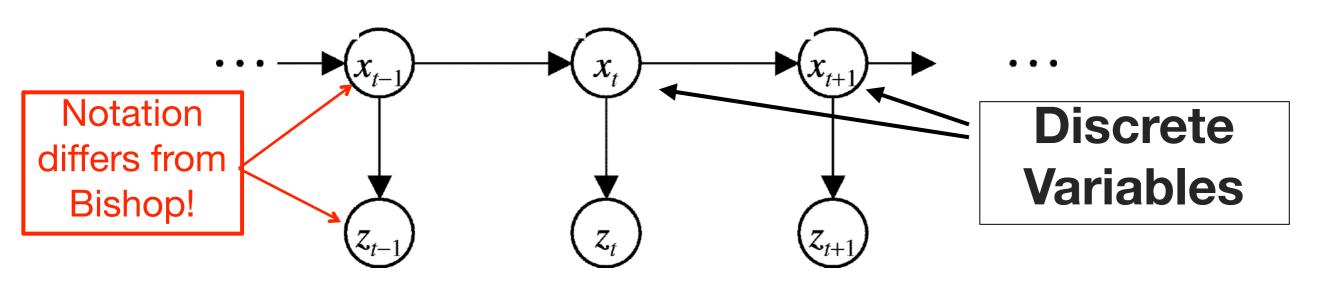
$$p(z_t \mid x_{0:t}, u_{1:t}, z_{1:t}) = p(z_t \mid x_t) \text{ (measurement)}$$

$$p(x_t \mid x_{0:t-1}, u_{1:t}, z_{1:t}) = p(x_t \mid x_{t-1}, u_t) \text{ (state)}$$



Bayes Filter Without Actions

Removing the action variables we obtain:

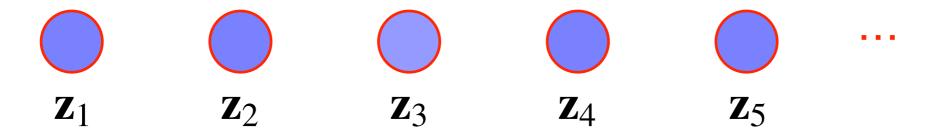


This incorporates the following Markov assumptions:

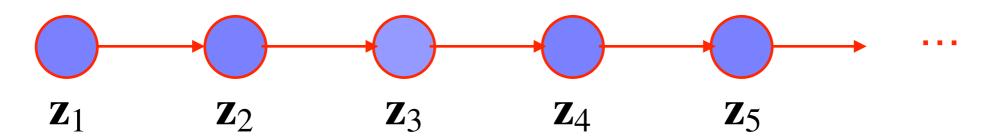
$$p(z_t \mid x_{0:t}, \qquad z_{1:t}) = p(z_t \mid x_t)$$
 (measurement) $p(x_t \mid x_{0:t-1}, \qquad z_{1:t}) = p(x_t \mid x_{t-1})$ (state)

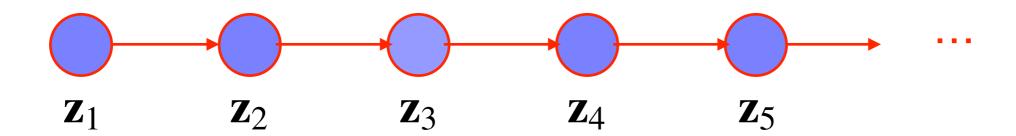


 Observations in sequential data should not be modeled as independent variables such as:



- Examples: weather forecast, speech, handwritten text, etc.
- The observation at time t depends on the observation(s) of (an) earlier time step(s):

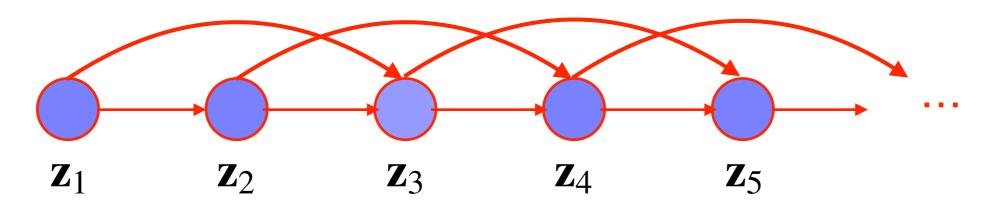


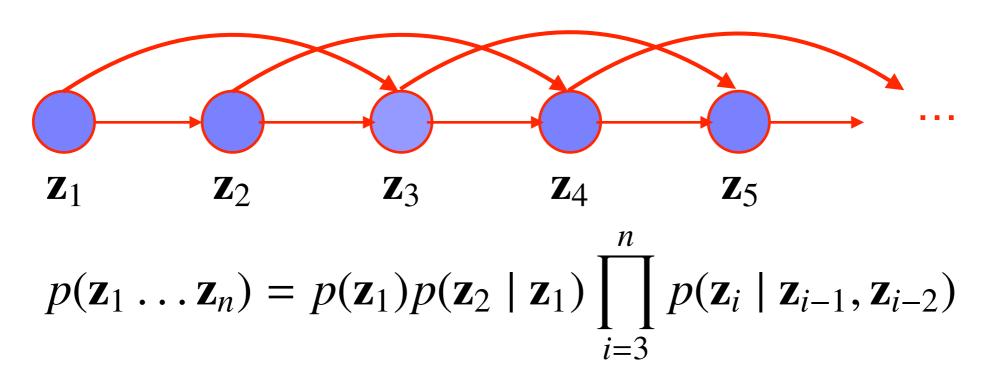


The joint distribution is therefore (d-sep):

$$p(\mathbf{z}_1 \dots \mathbf{z}_n) = p(\mathbf{z}_1) \prod_{i=2}^n p(\mathbf{z}_i \mid \mathbf{z}_{i-1})$$

 However: often data depends on several earlier observations (not just one)

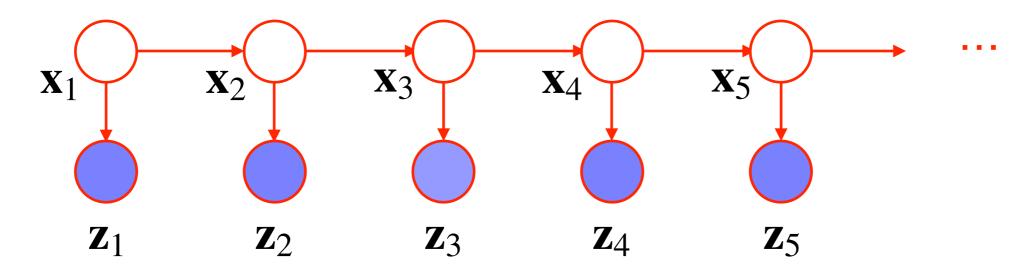




- Problem: number of stored parameters grows exponentially with the order of the Markov chain
- Question: can we model dependency of all previous observations with a limited number of parameters?

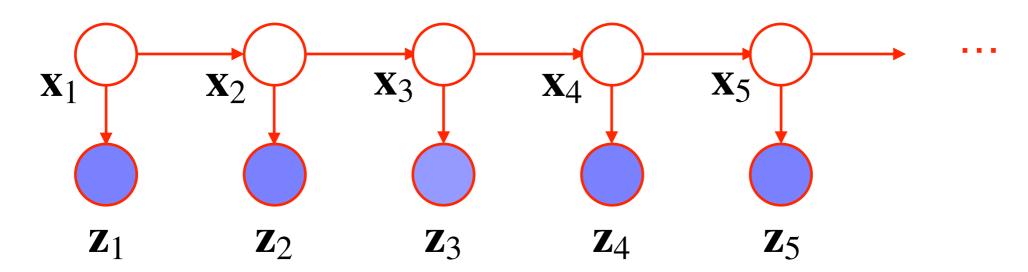


Idea: Introduce hidden (unobserved) variables:





Idea: Introduce hidden (unobserved) variables:



Now we have: $dsep(\mathbf{x}_n, {\{\mathbf{x}_1, ..., \mathbf{x}_{n-2}\}}, \mathbf{x}_{n-1})$

$$\Leftrightarrow p(\mathbf{x}_n \mid \mathbf{x}_1, \dots, \mathbf{x}_{n-2}, \mathbf{x}_{n-1}) = p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$$

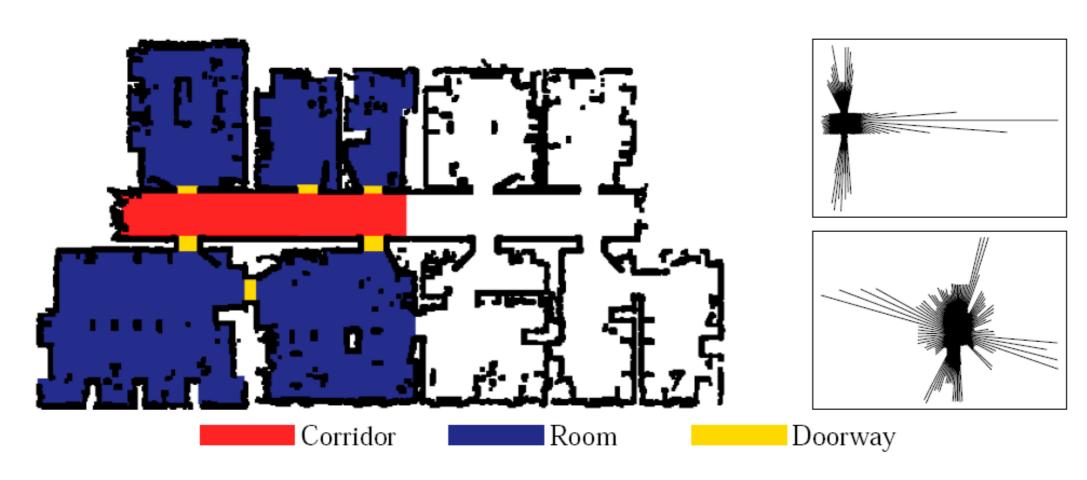
But: $\neg dsep(\mathbf{z}_{n}, \{\mathbf{z}_{1}, \dots, \mathbf{z}_{n-2}\}, \mathbf{z}_{n-1})$

$$\Leftrightarrow p(\mathbf{z}_n \mid \mathbf{z}_1, \dots, \mathbf{z}_{n-2}, \mathbf{z}_{n-1}) \neq p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$$

And: number of parameters is nK(K-1) + const.

Example

- Place recognition for mobile robots
- 3 different states: corridor, room, doorway
- Problem: misclassifications
- Idea: use information from previous time step





General Formulation of an HMM

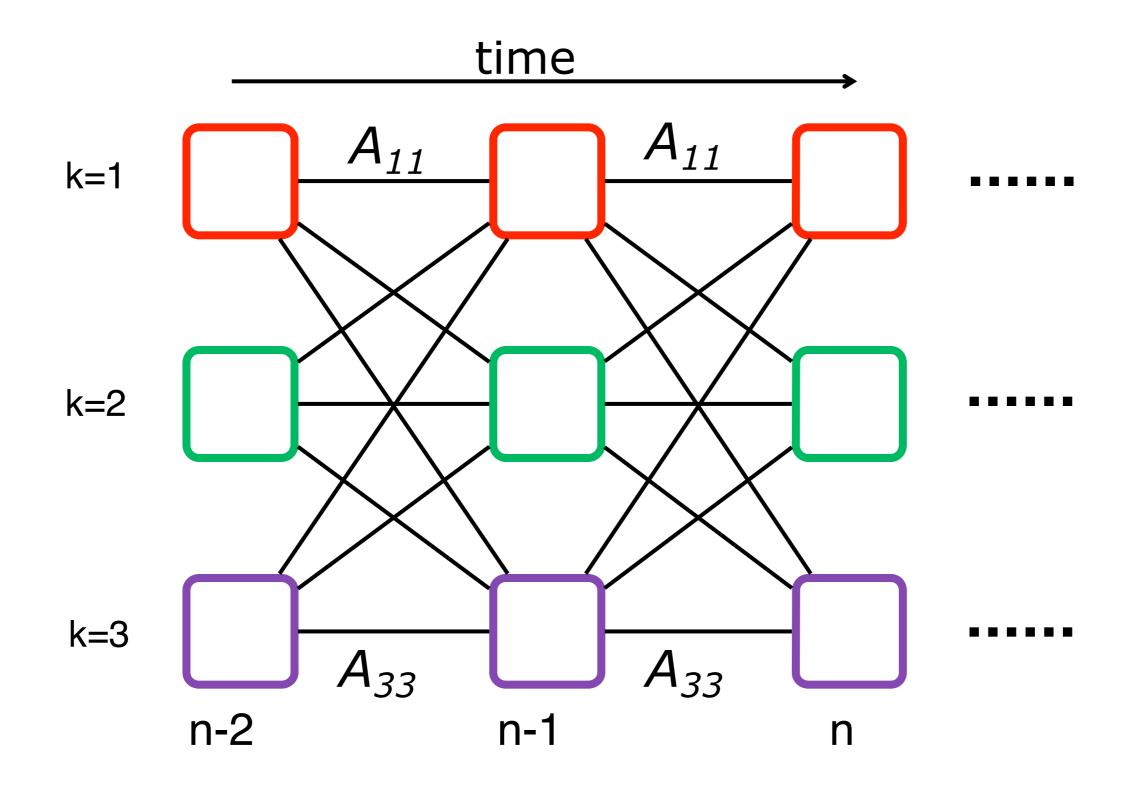
- 1.Discrete random variables
 - Observation variables: $\{z_n\}$, n = 1..N
 - Discrete state variables (unobservable): $\{x_n\}$, n = 1..N
 - Number of states $K: x_n \in \{1...K\}$

Model Parameters θ

- 2. Transition model $p(x_i | x_{i-1})$
 - Markov assumption (x_i only depends on x_i)
 - Represented as a $K \times K$ transition matrix A
 - Initial probability: $p(x_0)$ repr. as π_1, π_2, π_3
- 3. Observation model $p(z_i|x_i)$ with parameter φ
 - Observation only depends on the current state
 - Example: output of a "local" place classifier



The Trellis Representation



Application Example (1)

- Given an observation sequence z₁,z₂,z₃...
- Assume that the model parameters $\theta = (A, \pi, \phi)$ are known
- What is the probability that the given observation sequence is actually observed under this model, i.e. the **data likelihood** $p(Z|\theta)$?
- If we are given several different models, we can choose the one with highest probability
- Expressed as a supervised learning problem, this can be interpreted as the inference step (classification step)

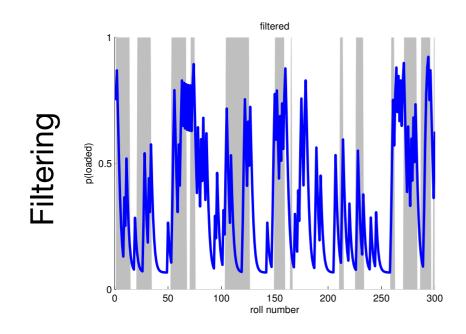


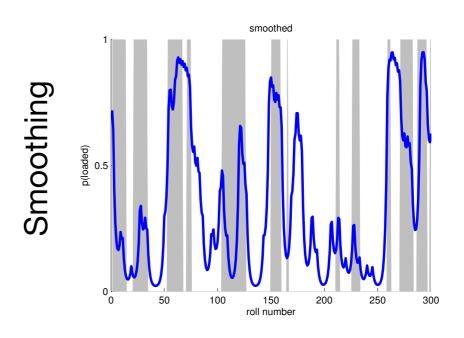


Application Example (2)

Based on the data likelihood we can solve two different kinds of problems:

- Filtering: computes $p(x_t | \mathbf{z}_{1:t})$, i.e. state probability only based on previous observations
- Smoothing: computes $p(x_t | \mathbf{z}_{1:T})$, state probability based on all observations (including those from the future)







Application Example (3)

- Given an observation sequence $z_1, z_2, z_3...$
- Assume that the model parameters $\theta = (A, \pi, \varphi)$ are known
- What is the state sequence $x_1, x_2, x_3...$ that explains best the given observation sequence?
- In the case of place recognition: which is the sequence of truly visited places that explains best the sequence of obtained place labels (classifications)?

Application Example (4)

- Given an observation sequence $z_1, z_2, z_3...$
- What are the optimal model parameters $\theta = (A, \pi, \phi)$?
- This can be interpreted as the training step
- It is in general the most difficult problem



Summary: 4 Operations on HMMs

- 1. Compute data likelihood $p(Z|\theta)$ from a known model
 - Can be computed with the forward algorithm
- 2. Filtering or Smoothing of the state probability
 - Filtering: forward algorithm
 - Smoothing: forward-backward algorithm
- 3. Compute optimal state sequence with a known model
 - Can be computed with the Viterbi-Algorithm
- 4. Learn model parameters for an observation sequence
 - Can be computed using Expectation-Maximization (or Baum-Welch)



The Forward Algorithm

Goal: compute $p(Z|\theta)$ (we drop θ in the following)

$$p(\mathbf{z}_1,\ldots,\mathbf{z}_n)=\sum_{\mathbf{x}_n}p(\mathbf{z}_1,\ldots,\mathbf{z}_n,\mathbf{x}_n)=:\sum_{\mathbf{x}_n}\alpha(\mathbf{x}_n)$$



The Forward Algorithm

Goal: compute $p(Z|\theta)$ (we drop θ in the following)

$$p(\mathbf{z}_1,\ldots,\mathbf{z}_n)=\sum_{\mathbf{x}_n}p(\mathbf{z}_1,\ldots,\mathbf{z}_n,\mathbf{x}_n)=:\sum_{\mathbf{x}_n}\alpha(\mathbf{x}_n)$$

We can calculate α recursively:

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

This is the same recursive formula as we had in the first lecture!



The Forward Algorithm

Goal: compute $p(Z|\theta)$ (we drop θ in the following)

$$p(\mathbf{z}_1,\ldots,\mathbf{z}_n)=\sum_{\mathbf{x}_n}p(\mathbf{z}_1,\ldots,\mathbf{z}_n,\mathbf{x}_n)=:\sum_{\mathbf{x}_n}\alpha(\mathbf{x}_n)$$

We can calculate α recursively:

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

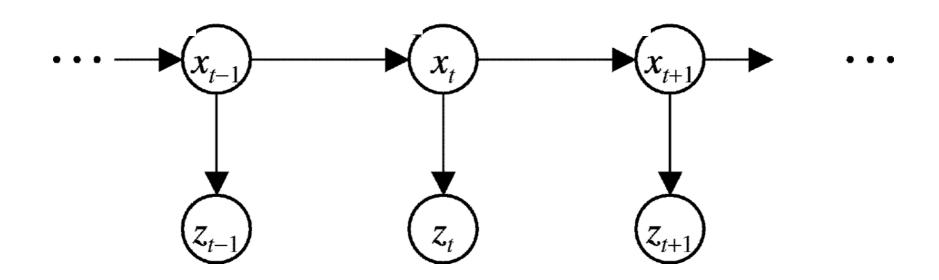
This is the same recursive formula as we had in the first lecture!

Filtering:
$$p(\mathbf{x}_n \mid \mathbf{z}_1, \dots, \mathbf{z}_n) = \frac{p(\mathbf{z}_1, \dots, \mathbf{z}_n, \mathbf{x}_n)}{p(\mathbf{z}_1, \dots, \mathbf{z}_n)} = \frac{\alpha(\mathbf{x}_n)}{\sum_{\mathbf{x}_n} \alpha(\mathbf{x}_n)}$$



The Forward-Backward Algorithm

- As before we set $\alpha(\mathbf{x}_t) = p(\mathbf{z}_1, \dots, \mathbf{z}_t, \mathbf{x}_t)$
- We also define $\beta(\mathbf{x}_t) = p(\mathbf{z}_{t+1}, \dots, \mathbf{z}_n \mid \mathbf{x}_t)$



The Forward-Backward Algorithm

- As before we set $\alpha(\mathbf{x}_t) = p(\mathbf{z}_1, \dots, \mathbf{z}_t, \mathbf{x}_t)$
- We also define $\beta(\mathbf{x}_t) = p(\mathbf{z}_{t+1}, \dots, \mathbf{z}_n \mid \mathbf{x}_t)$
- This can be recursively computed (backwards):

$$\beta(\mathbf{x}_t) = \sum_{\mathbf{x}_{t+1}} \beta(\mathbf{x}_{t+1}) p(\mathbf{z}_{t+1} \mid \mathbf{x}_{t+1}) p(\mathbf{x}_{t+1} \mid \mathbf{x}_t)$$

- This is exactly the same as the message-passing algorithm (sum-product)!
 - forward messages α_t (vector of length K)
 - backward messages β_t (vector of length K)



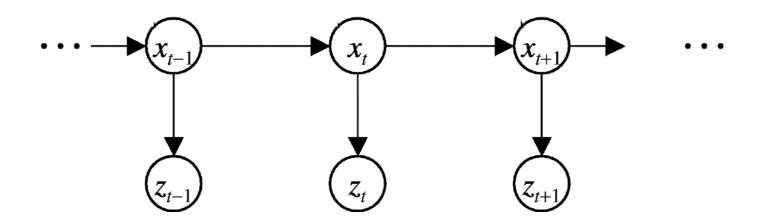


2. Computing the Most Likely States

• Goal: find a state sequence $x_1, x_2, x_3...$ that maximizes the probability $p(X,Z|\theta)$

• Define
$$\delta_t(j) := \max_{x_1, \dots, x_{t-1}} p(\mathbf{x}_{1:t-1}, x_t = j \mid \mathbf{z}_{1:t})$$

This is the probability of state j by taking the most probable path.



2. Computing the Most Likely States

• Goal: find a state sequence $x_1, x_2, x_3...$ that maximizes the probability $p(X,Z|\theta)$

• Define
$$\delta_t(j) := \max_{x_1, \dots, x_{t-1}} p(\mathbf{x}_{1:t-1}, x_t = j \mid \mathbf{z}_{1:t})$$

This can be computed recursively:

$$\delta_t(j) := \max_i \delta_{t-1}(i) p(x_t \mid x_{t-1}) p(z_t \mid x_t)$$

we also have to compute the argmax:

$$a_t(j) := \arg \max_i \delta_{t-1}(i) p(x_t \mid x_{t-1}) p(z_t \mid x_t)$$

The Viterbi algorithm

- Initialize:
 - $\delta(x_0) = p(x_0) p(z_0 \mid x_0)$
 - $\psi(x_0) = 0$
- Compute recursively for n=1...N:
 - $\delta(x_n) = p(z_n|x_n) \max_{x_{n-1}} [\delta(x_{n-1}) p(x_n|x_{n-1})]$
 - $a(x_n) = \underset{x_{n-1}}{\operatorname{argmax}} [\delta(x_{n-1}) p(x_n | x_{n-1})]$
- On termination:
 - $p(Z,X|\theta) = \max \delta(x_N)$
 - $x_N = \operatorname{argmax}^{x_N} \delta(x_N)$
- Backtracking:
 - $x_n = a(x_{n+1})$



3. Learning the Model Parameters

- Given an observation sequence z₁,z₂,z₃...
- Find optimal model parameters $\theta = \pi, A, \varphi$
- We need to maximize the likelihood $p(Z|\theta)$
- Can not be solved in closed form
- Iterative algorithm: Expectation Maximization (EM) or for the case of HMMs: Baum-Welch algorithm



- E-Step (assuming we know π,A,φ, i.e. θold)
- Define the posterior probability of being in state i at step k:
- Define $\gamma(\mathbf{x}_n) = p(\mathbf{x}_n|Z)$



- E-Step (assuming we know π,A,φ, i.e. θold)
- Define the posterior probability of being in state i at step k:
- Define $\gamma(\mathbf{x}_n) = p(\mathbf{x}_n|Z)$
- It follows that $\gamma(\mathbf{x}_n) = \alpha(\mathbf{x}_n) \beta(\mathbf{x}_n) / p(Z)$



- E-Step (assuming we know π,A,φ, i.e. θold)
- Define the posterior probability of being in state i at step k:
- Define $\gamma(x_n) = p(x_n|Z)$
- It follows that $\gamma(x_n) = \alpha(x_n) \beta(x_n) / p(Z)$
- Define $\xi(x_{n-1}, x_n) = p(x_{n-1}, x_n | Z)$
- It follows that

$$\xi(x_{n-1},x_n) = \alpha(x_{n-1})p(z_n|x_n)p(x_n|x_{n-1})\beta(x_n) / p(Z)$$

• We need to compute: $Q(\theta, \theta^{\text{old}}) = \sum_{X} p(X|Z, \theta^{\text{old}}) \log p(Z, X|\theta)$ Expected complete data log-likelihood





- Maximizing Q also maximizes the likelihood: $p(Z|\theta) \ge p(Z|\theta^{old})$
- M-Step:

$$\pi_k = \frac{\sum_{\mathbf{x}} \gamma(\mathbf{x}) x_{1k}}{\sum_{j=1} \sum_{\mathbf{x}} \gamma(\mathbf{x}) x_{1j}}$$

here, we need forward and backward step!

$$A_{jk} = \frac{\sum_{t=2}^{T} \xi(x_{t-1,j}, x_{tk})}{\sum_{l=1}^{K} \sum_{t=2}^{T} \xi(x_{t-1,j}, x_{tl})}$$

- With these new values, Q is recomputed
- This is done until the likelihood does not increase anymore (convergence)





The Baum-Welsh algorithm - summary

- Start with an initial estimate of $\theta = (\pi, A, \phi)$ e.g. uniformly and k-means for ϕ
- Compute Q(θ,θold) (E-Step)
- Maximize Q (M-step)
- Iterate E and M until convergence
- In each iteration one full application of the forward-backward algorithm is performed
- Result gives a local optimum
- For other local optima, the algorithm needs to be started again with new initialization





The Scaling problem

Probability of sequences

$$\prod_{i} p(x_i \mid \ldots) << 1$$

- Probabilities are very small
- The product of the terms soon is very small
- Usually: converting to log-space works
- But: we have sums of products!
- Solution: Rescale/Normalise the probability during the computation, e.g.:

$$\hat{\alpha}(x_n) = \alpha(x_n) / p(z_1, z_2, \dots, z_n)$$





Summary

- HMMs are a way to model sequential data
- They assume discrete states
- Three possible operations can be performed with HMMs:
 - Data likelihood, given a model and an observation
 - Most likely state sequence, given a model and an observation
 - Optimal Model parameters, given an observation
- Appropriate scaling solves numerical problems
- HMMs are widely used, e.g. in speech recognition



