Probabilistic Modelling of Sequences

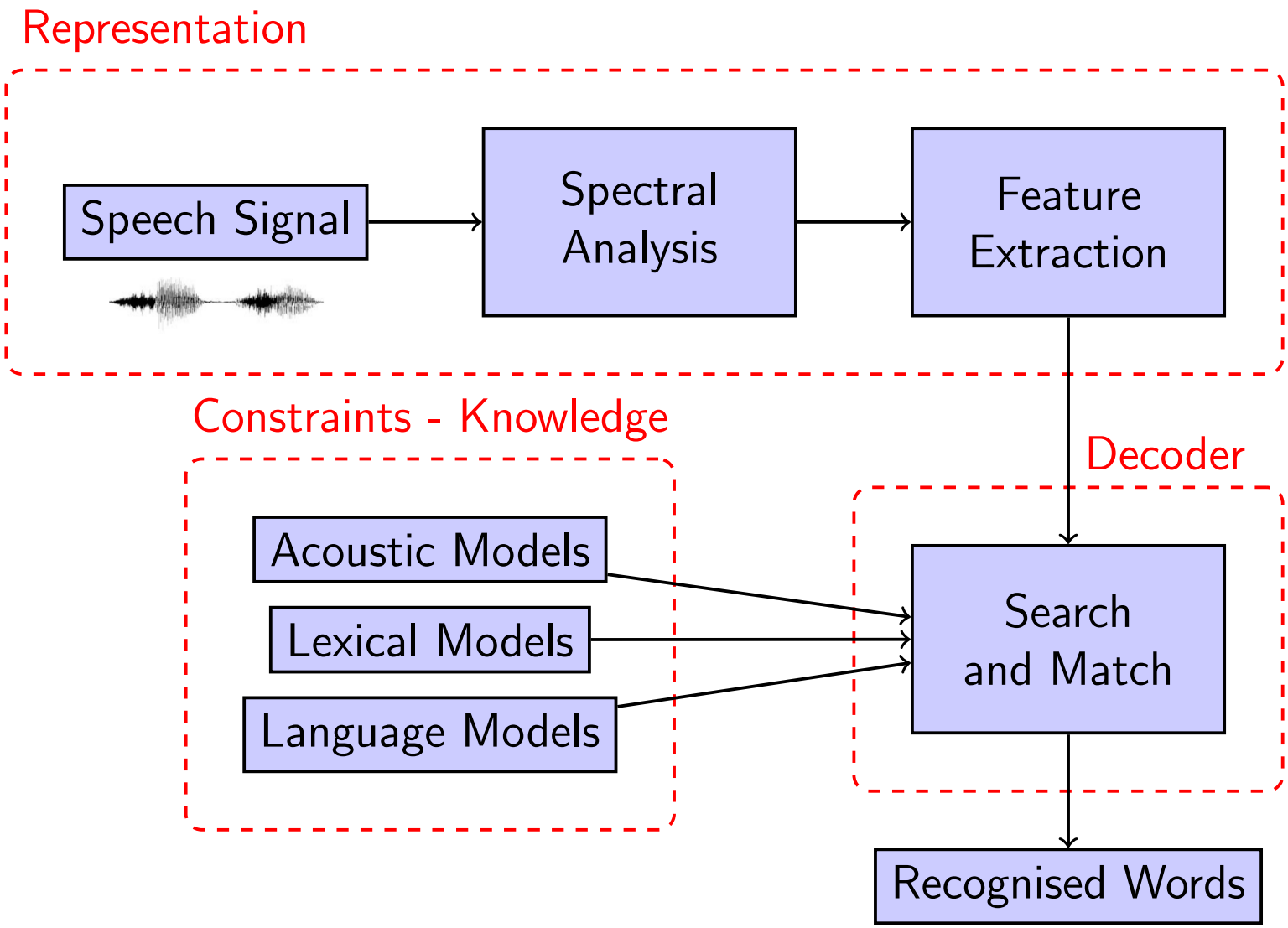
DT2119 Speech and Speaker Recognition

Giampiero Salvi

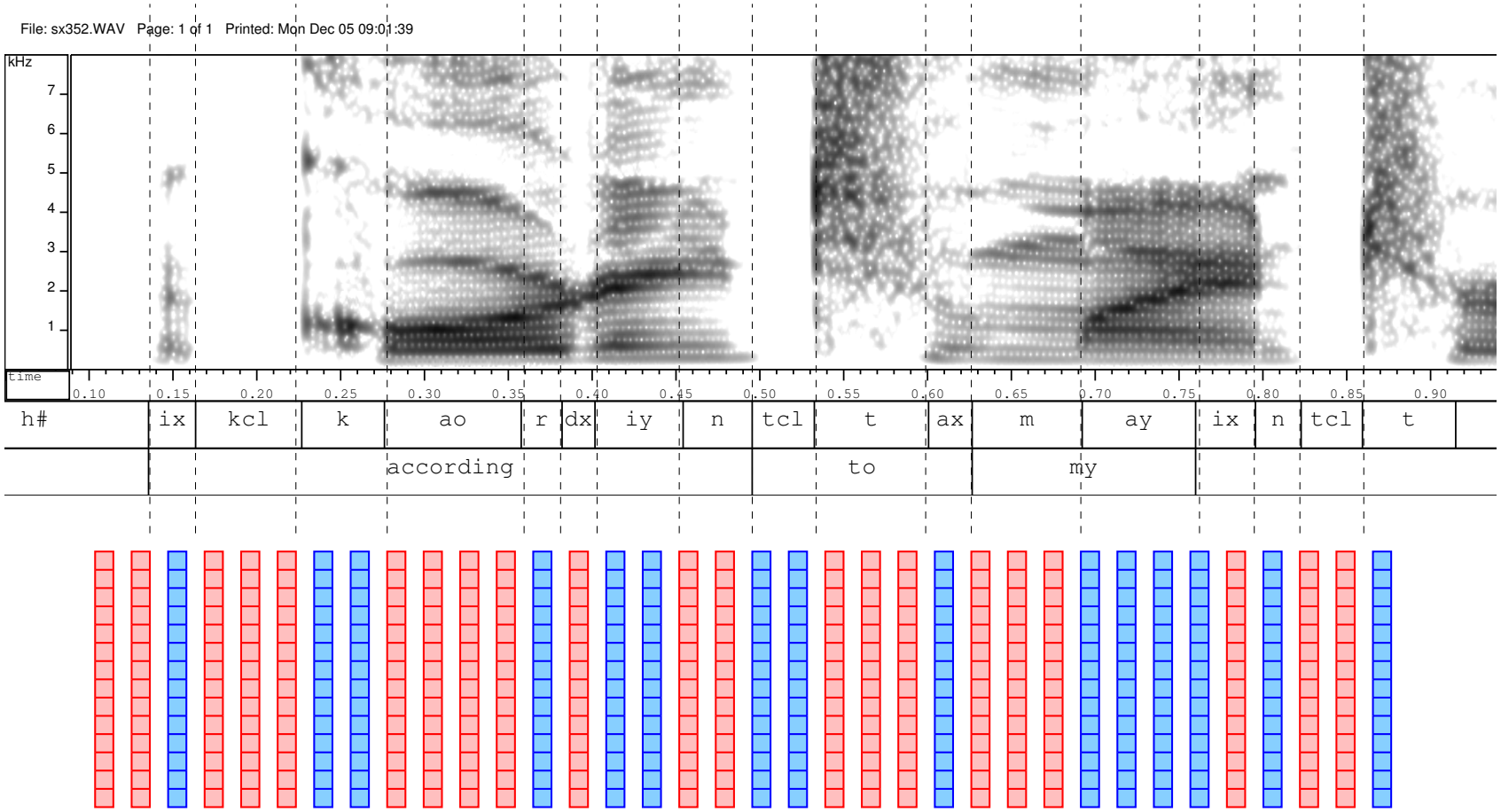
KTH/CSC/TMH giampi@kth.se

VT 2019

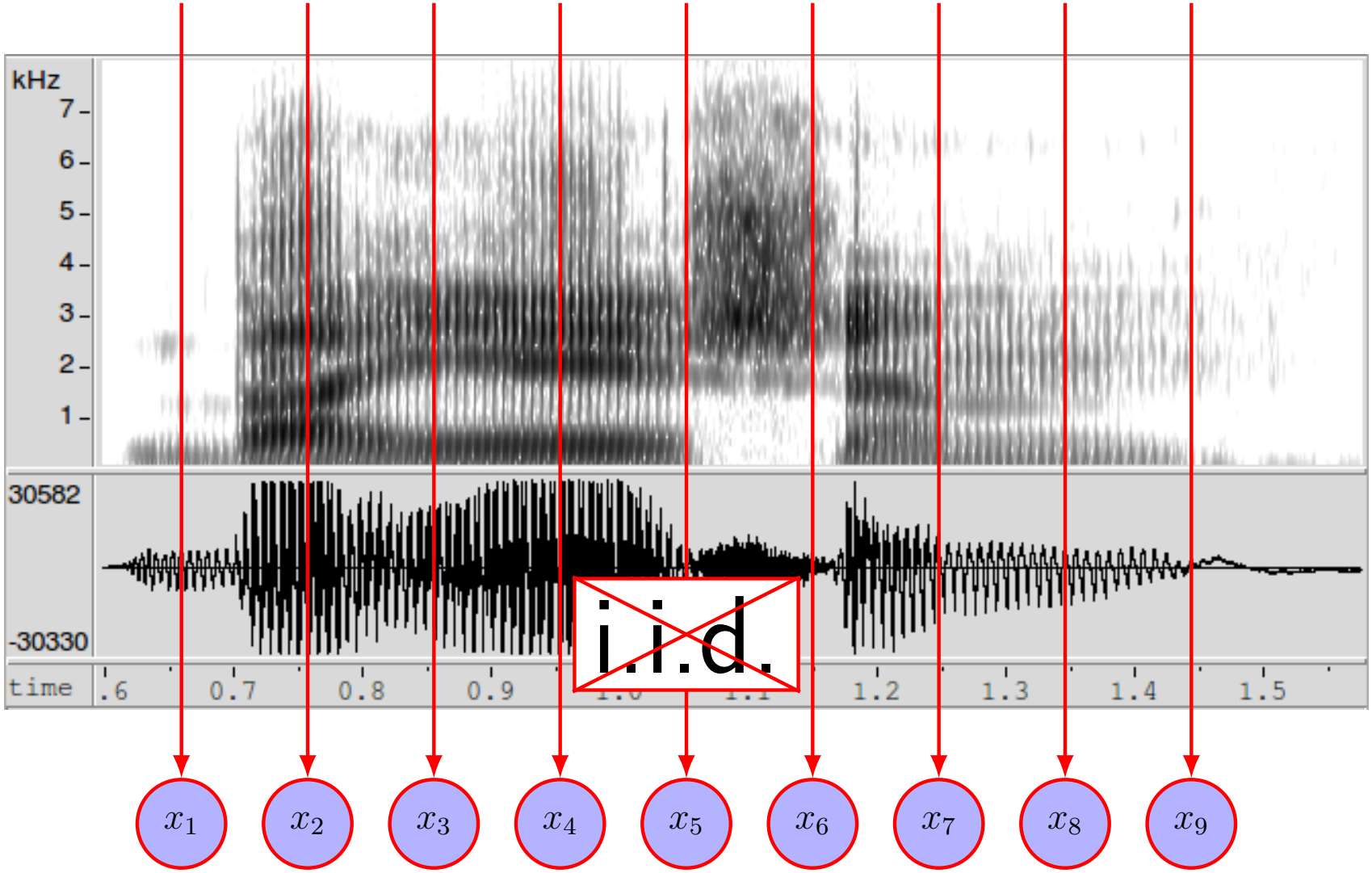
Components of ASR System



Frame-Based Processing

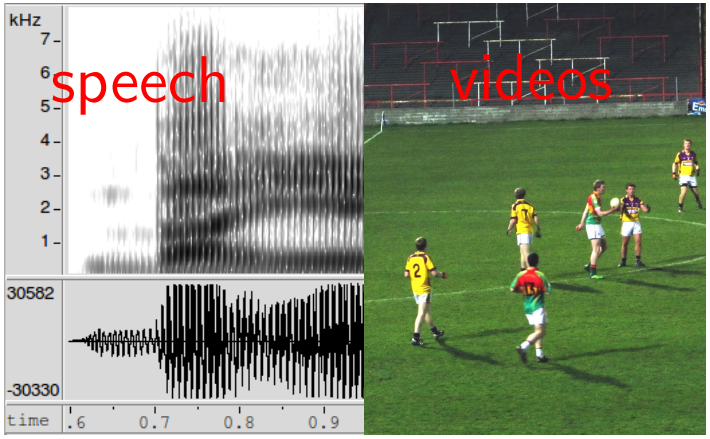


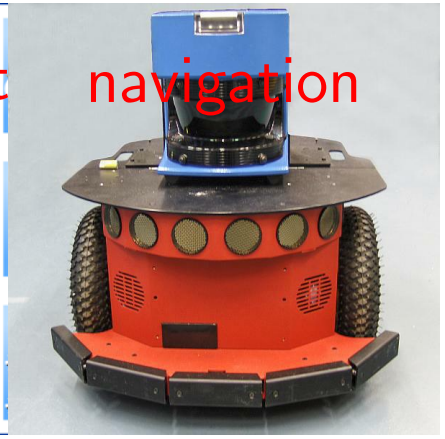
Sequences in Statistical Terms



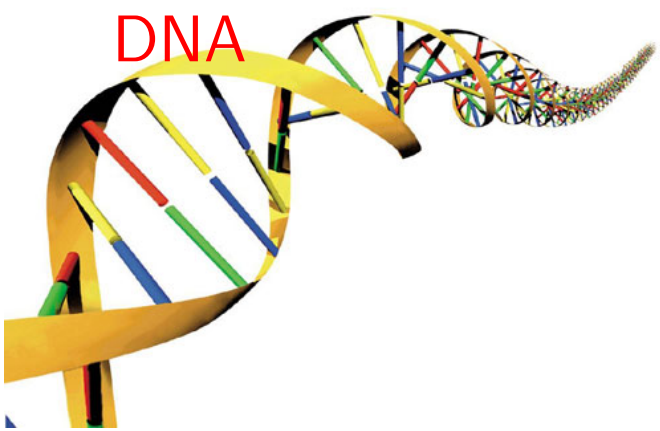
Sequential Data: Not Only Speech

Time sequences





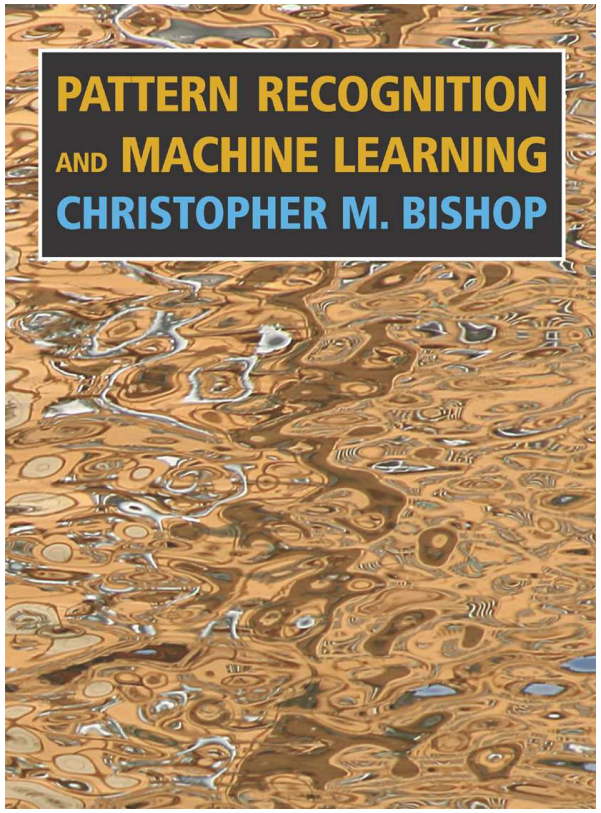
Timeless sequences



text

Nel mezzo del cammin di nostra vita
mi ritrovai per una selva oscura,
ché la diritta via era smarrita.
Ahi quanto a dir qual era è cosa dura
esta selva selvaggia e aspra e forte
che nel pensier rinova la paura!
Tant' è amara che poco è più morte;
ma per trattar del ben ch'i' vi trovai,
dirò de l'altre cose ch'i' v'ho scorte.
Io non so ben ridir com' i' v'intrai,
tant' era pien di sonno a quel punto
che la verace via abbandonai.

Extra Literature (Optional)



C. M. Bishop. *Pattern Recognition and Machine Learning*.
Springer, 2006

Historical Perspective

- Hidden Markov Models first studied in the '60s[1,2]
- applied to ASR in the mid '70s[3]
- later seen as special case of Bayesian Networks[4]

---

[1] R. Stratonovich. "Conditional Markov Processes". In: *Theory of Probability and its Applications* 5.2 (1960), pp. 156–178.

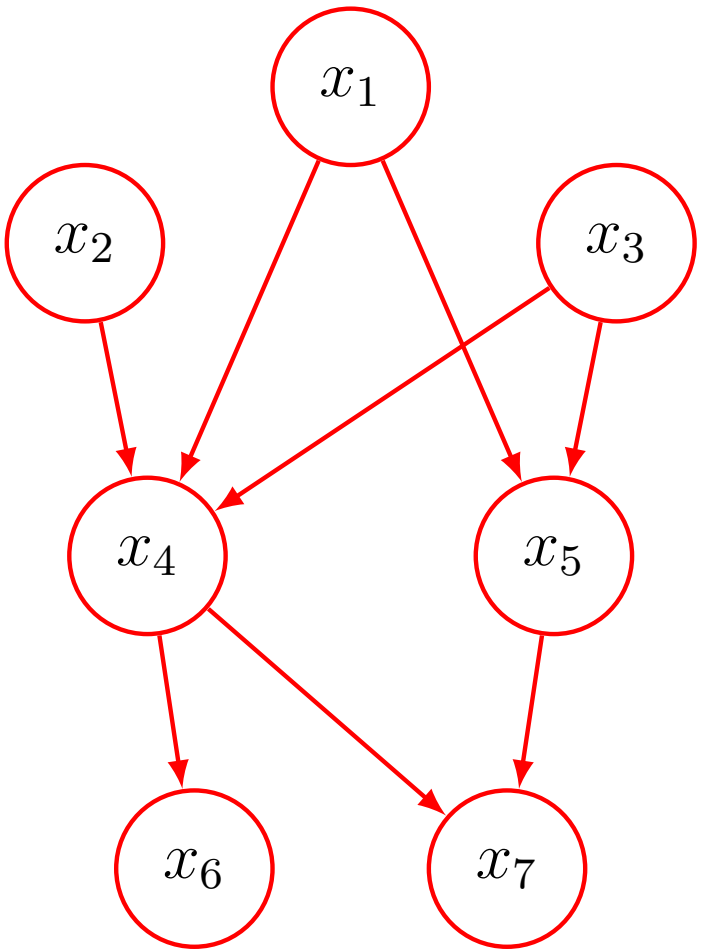
[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

[3] J. Baker. "The DRAGON system—An overview". In: 23 (1975), pp. 24–29.

[4] J. Pearl. "Bayesian networks: a model of self-activated memory for evidential reasoning". In: *Proceedings of the 7th Conference of the Cognitive Science Society*. University of California, Irvine, 1985, pp. 329–334.

Bayesian Networks (reminder)

$$
\begin{aligned}
p(x_1, \ldots, x_7) = \\
& p(x_1) \\
& p(x_2) \\
& p(x_3) \\
& p(x_4 | x_1, x_2, x_3) \\
& p(x_5 | x_1, x_3) \\
& p(x_6 | x_4) \\
& p(x_7 | x_4, x_5)
\end{aligned}
$$



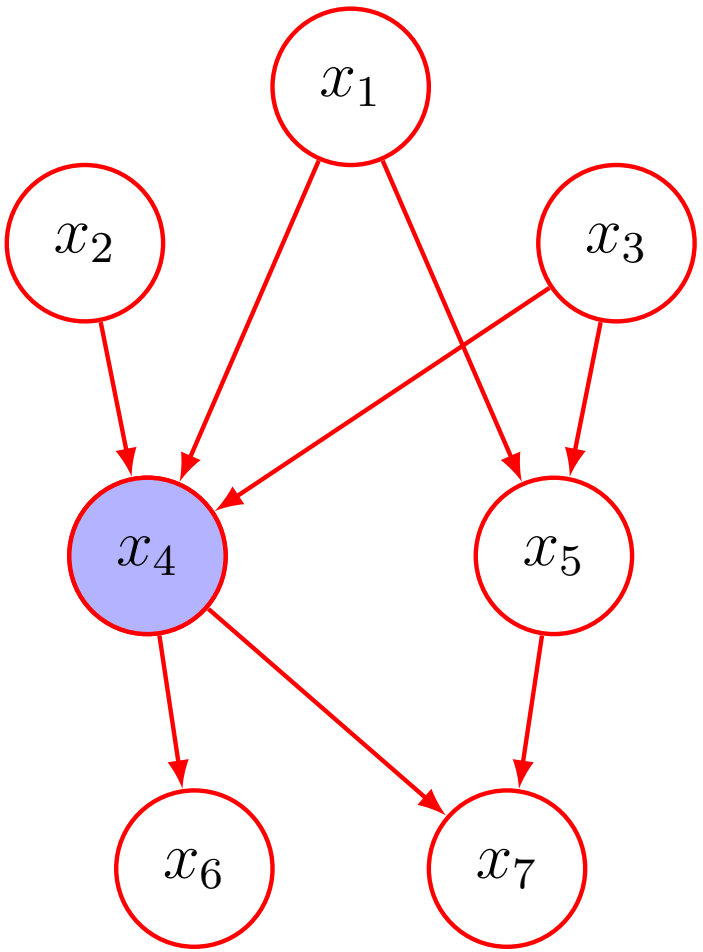
Bayesian Networks (reminder)

If we observe x_4...
d-separation:

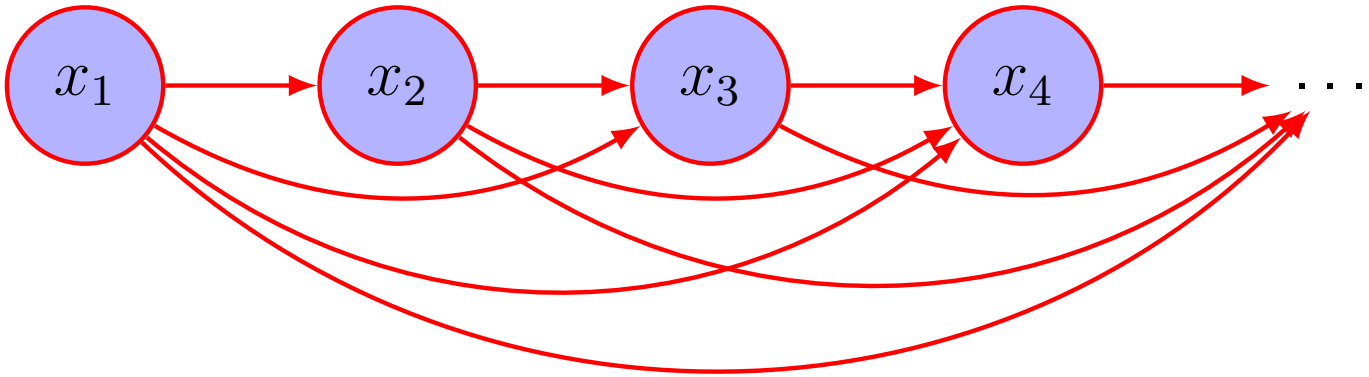
Head-to-tail:
x_2 and x_6 conditionally independent

Tail-to-tail:
x_6 and x_7 conditionally independent

Head-to-head:
x_1, x_2 and x_3 dependent (explaining away)



(Dynamic) Bayesian Networks

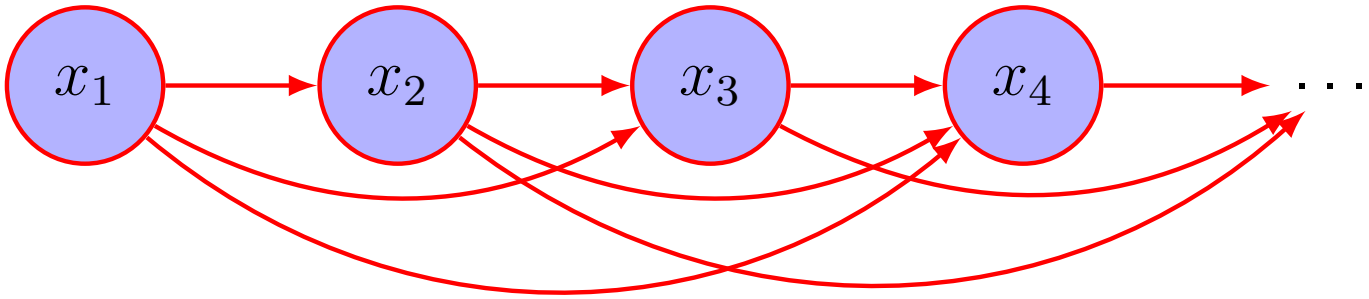


independence assumption (e.g. i.i.d) not satisfactory Most general case, applying product rule recursively ($p(a, b) = p(a)p(b|a)$)

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots \\ \cdots p(x_N|x_1, \ldots, x_{N-1})$$

Grows quadratically with sequence length (N)!!!

Markov assumption



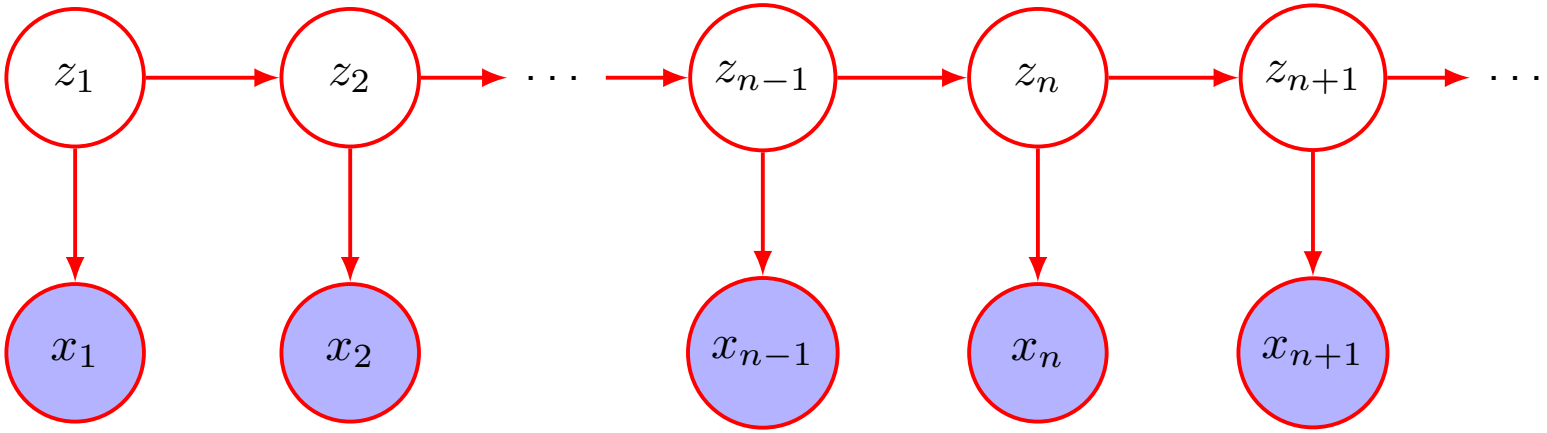
Third order Markov assumption:

$$p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \prod_{n=4}^{N} p(x_n|x_{n-3}, x_{n-2}, x_{n-1})$$

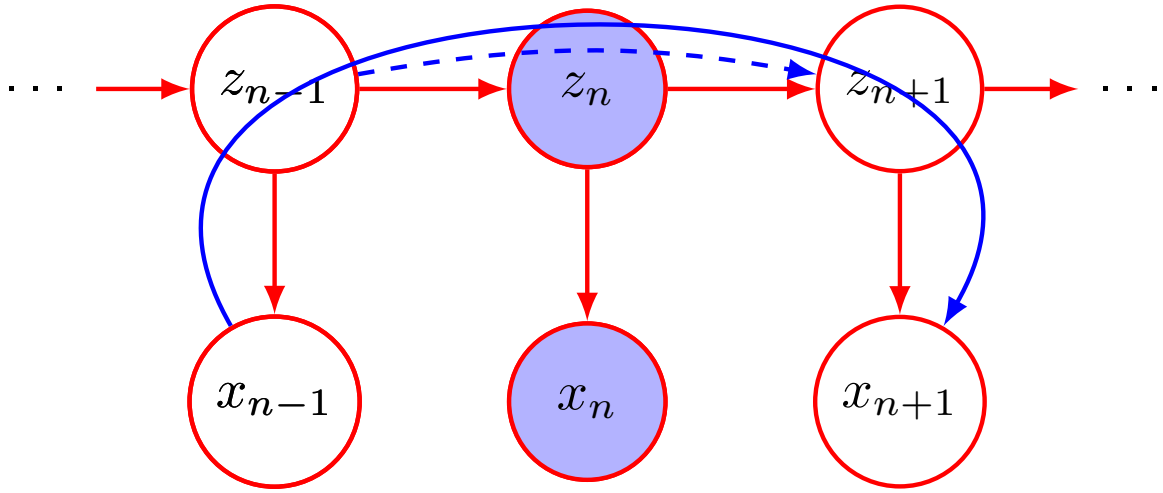
Grows quadratically with order!!!

State Space Models

Adding latent variables z_n



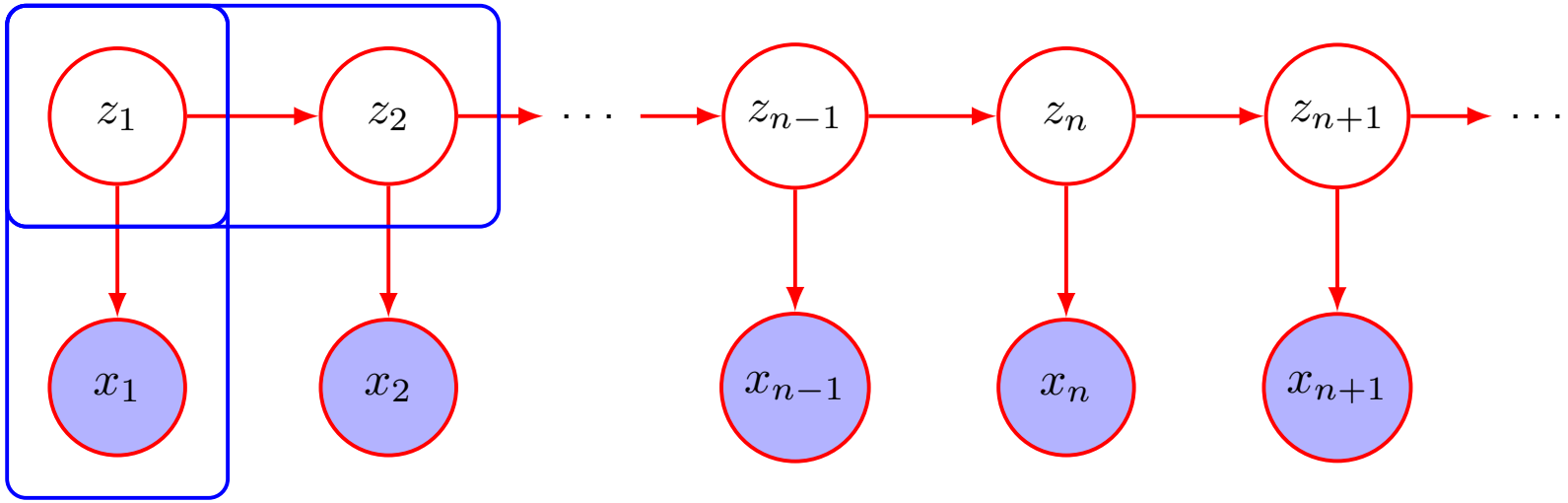
State Space Models: Properties



- given z_n, z_{n+1} is independent of $z_1, \ldots, z_{n-1}$
  $p(z_{n+1}|z_1, \ldots, z_n) = p(z_{n+1}|z_n)$
- $p(x_{n+1}|x_1, \ldots, x_n)$ does not simplify

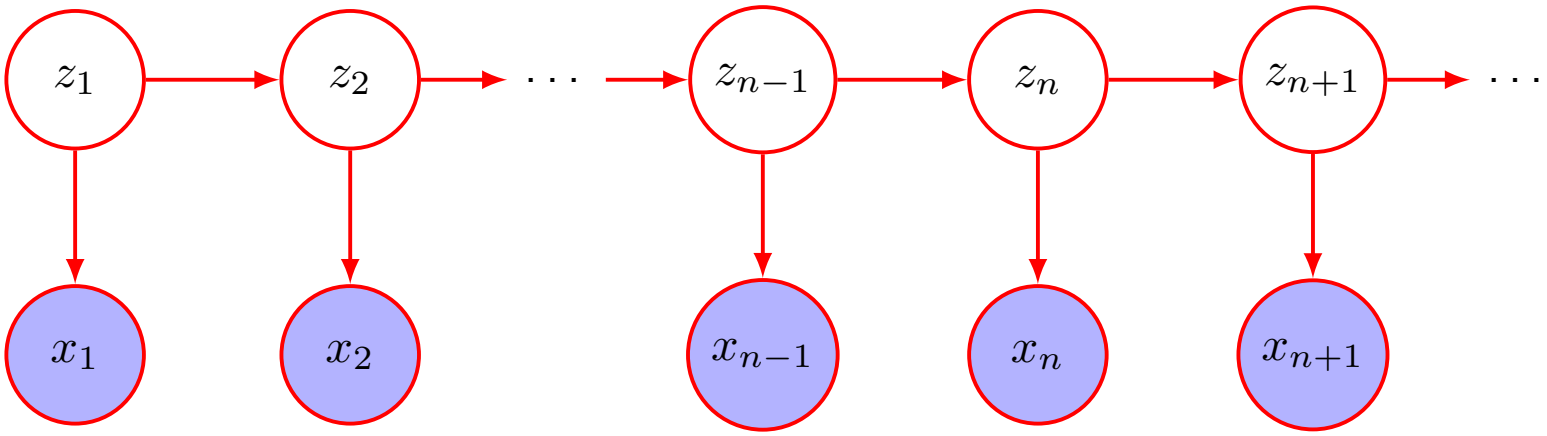
We have modelled indefinitely long dependencies with a limited set of parameters!

Stationary State Space Models



- Emission: $p(x_n|z_n)$
- Transition: $p(z_n|z_{n-1})$
- Initial: $p(z_1)$

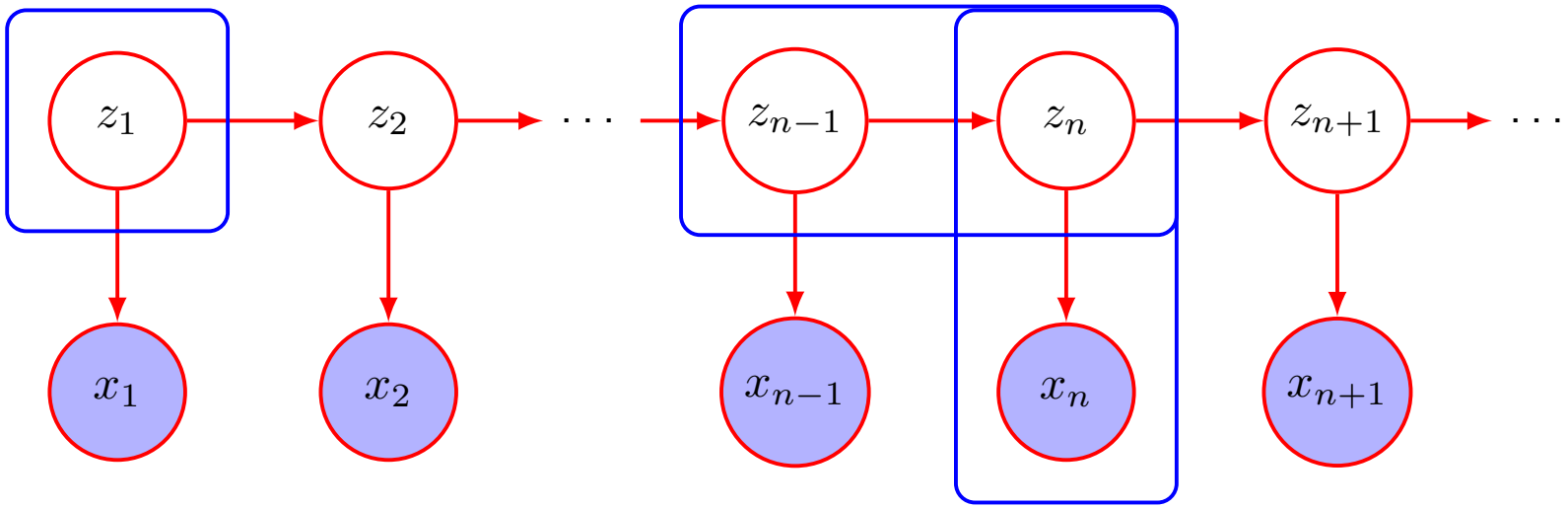
State Space Models Instances



- if z_n are discrete: Hidden Markov Models
- if z_n are continuous: Linear Dynamical Systems

Hidden Markov Models

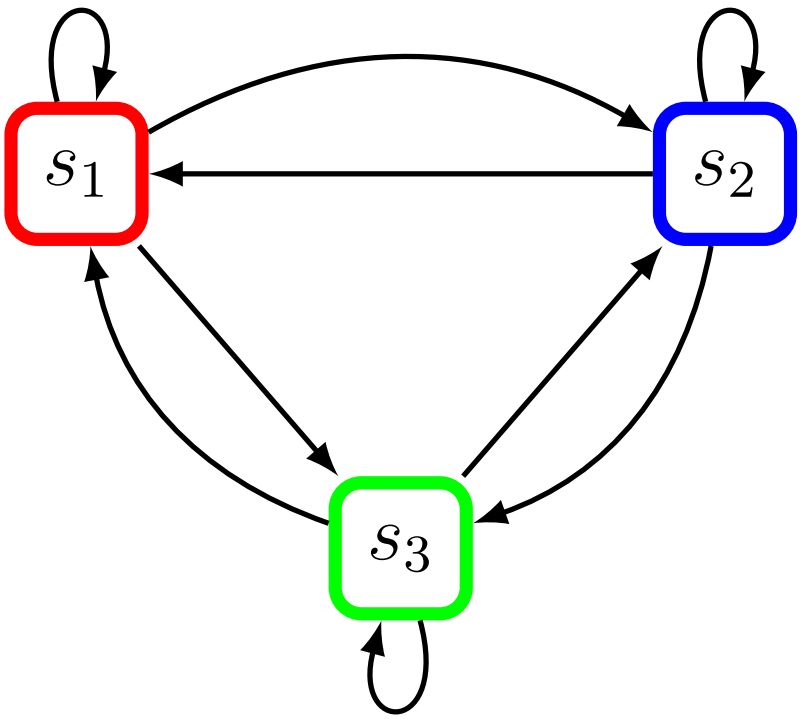
State space models with discrete z_n



- Emission: $p(x_n|z_n) = p(x_n|z_n, \phi)$
  equivalent to Mixture Model
- Transition: $p(z_n|z_{n-1}) = p(z_n|z_{n-1}, A)$
- Initial: $p(z_1) = p(z_1|\pi)$

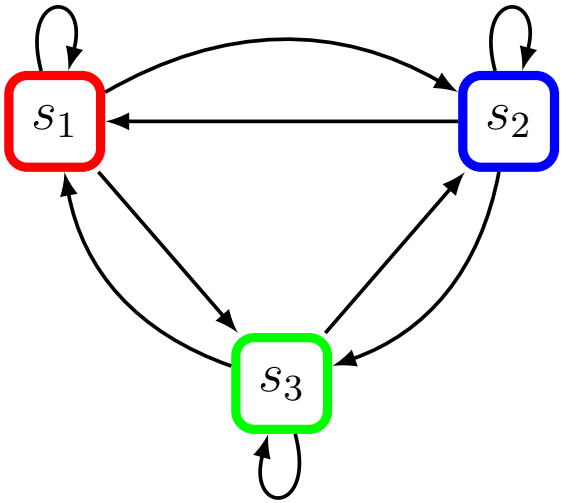
Hidden Markov Models (HMMs)

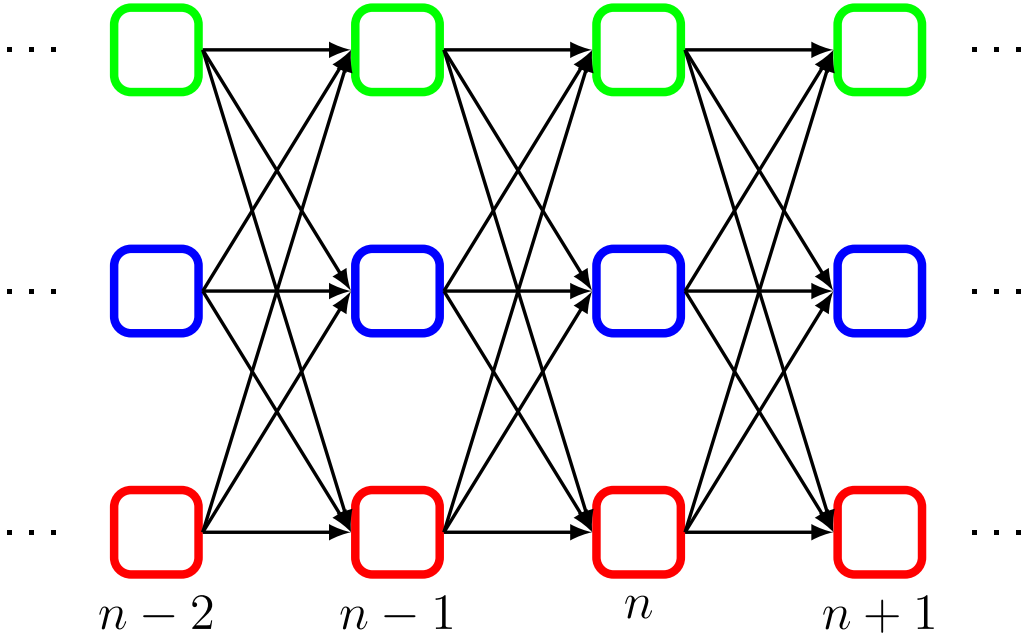
Ergodic HMM



Elements:	
set of states:	$S = \{s_1, s_2, s_3\}$
transition probabilities:	$A(s_a, s_b) = P(s_b, t \mid s_a, t-1)$
prior probabilities:	$\pi(s_a) = P(s_a, t_0)$
state to observation probs:	$\phi(o, s_a) = P(o \mid s_a)$

HMMs: Trellis (Lattice)





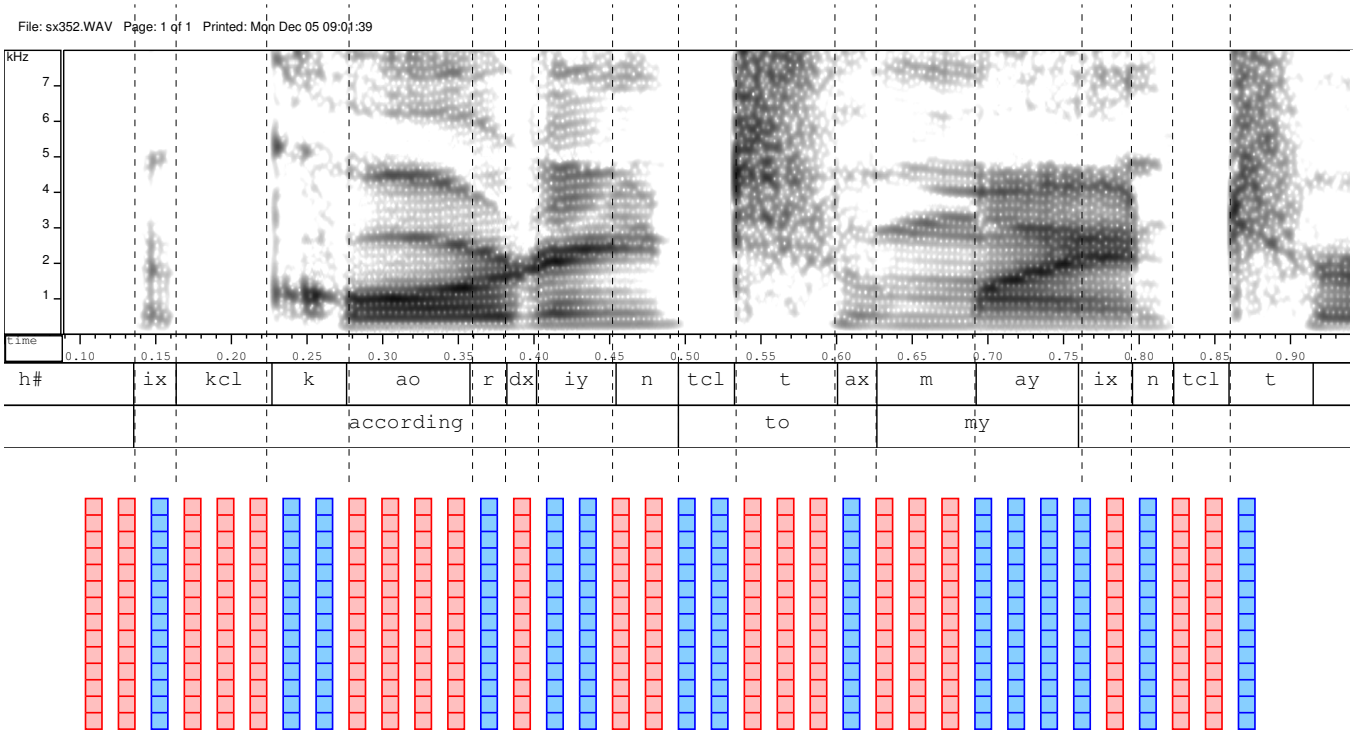
A probabilistic perspective: Bayes' rule

$$P(\text{words}|\text{sounds}) = \frac{P(\text{sounds}|\text{words})P(\text{words})}{P(\text{sounds})}$$

- $P(\text{sounds}|\text{words})$ can be estimated from training data and transcriptions
- $P(\text{words})$: *a priori* probability of the words (Language Model)
- $P(\text{sounds})$: *a priori* probability of the sounds (constant, can be ignored)

Probabilistic Modelling

Problem: How do we model $P(\text{sounds}|\text{words})$?

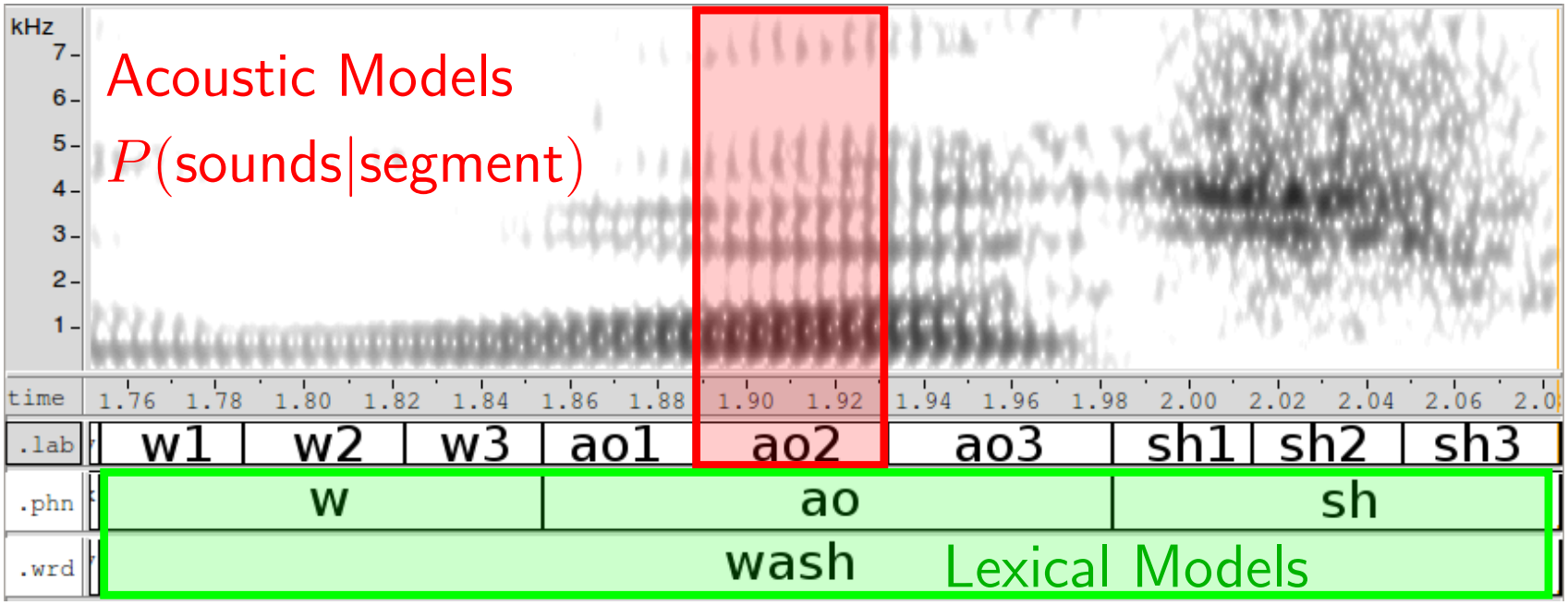


Every feature vector (observation at time t) is a continuous stochastic variable (e.g. MFCC)

Stationarity

Problem: speech is not stationary

- we need to model short segments independently
- the fundamental unit can not be the word, but must be shorter
- usually we model three segments for each phoneme



Local probabilities (frame-wise)

If segment sufficiently short

$$P(\text{sounds}|\text{segment})$$

can be modelled with standard probability distributions

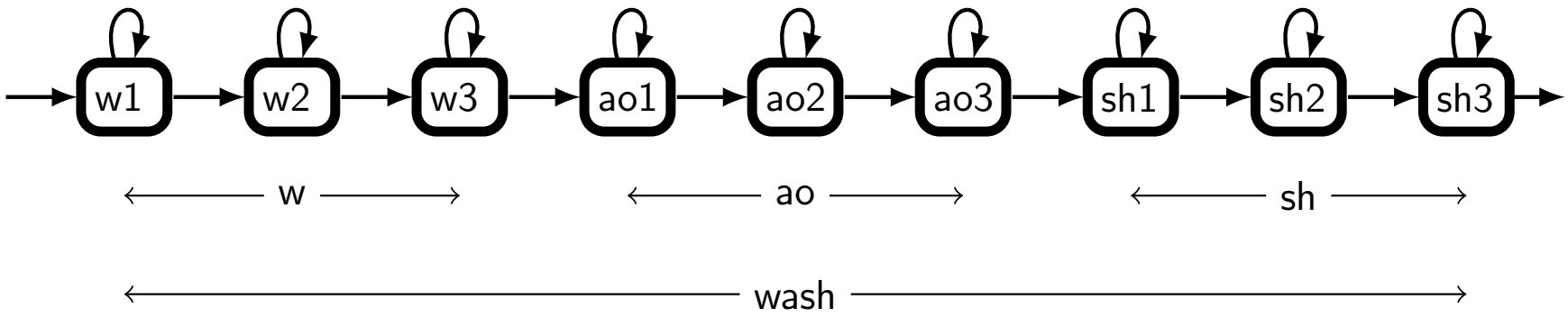
$$\phi_{s_a}(x) = P(x|s_a)$$

Usually Gaussian or Gaussian Mixture

Global Probabilities (utterance)

Problem: How do we combine the different $P(\text{sounds}|\text{segment})$ to form $P(\text{sounds}|\text{words})$?

Answer: Hidden Markov Model (HMM)



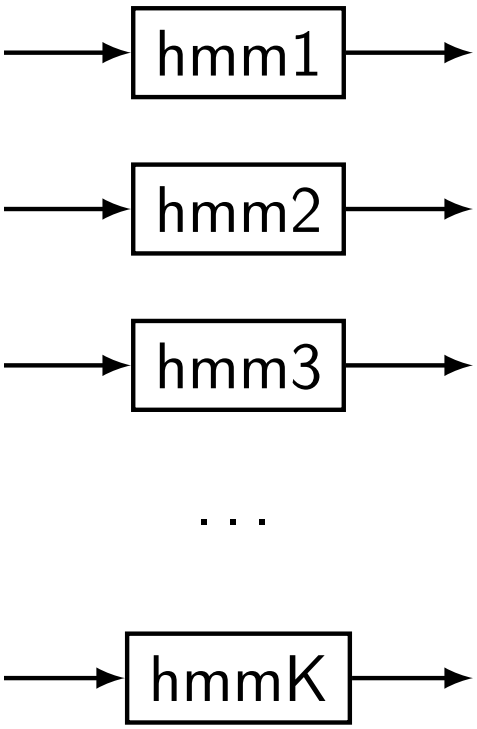
HMM-questions (Inference)

1. what is the probability that the model has generated the sequence of observations? (isolated word recognition)
   forward algorithm
2. what is the most likely state sequence given the observation sequence? (continuous speech recognition)
   Viterbi algorithm[5]
3. how can the model parameters be estimated from examples? (training)
   Baum-Welch[6]

[5] A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymtotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (1967), pp. 260–269.

[6] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

Isolated Words Recognition

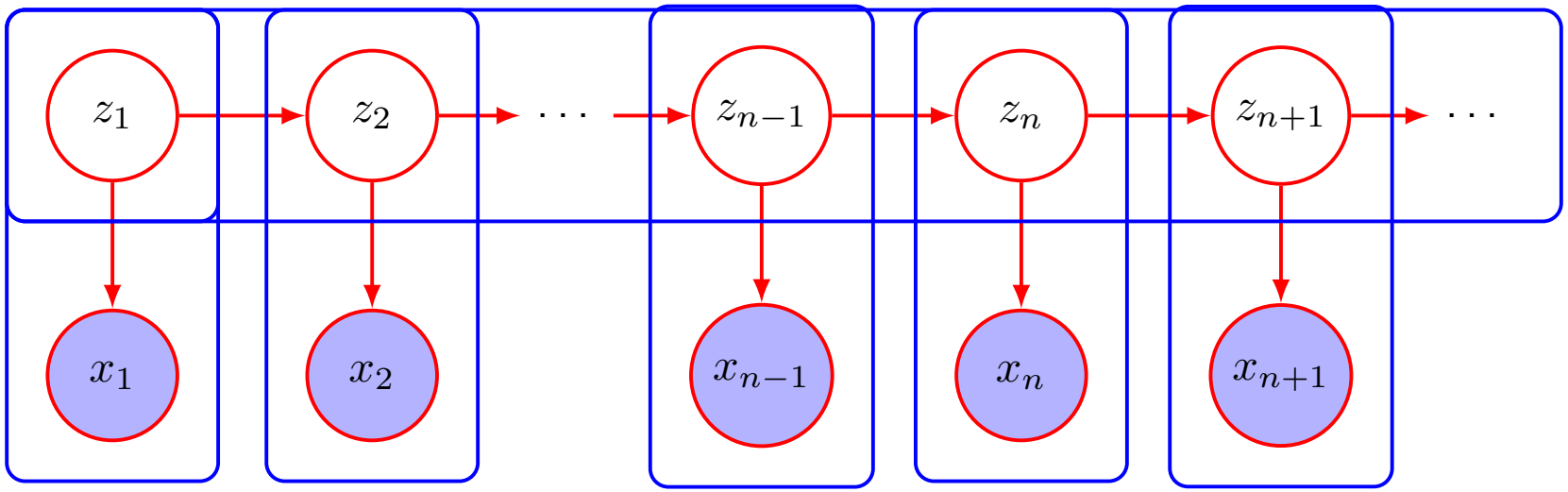


Compare Likelihoods (forward algorithm)

HMM Inference: Joint Distribution

$$X = \{x_1, \ldots, x_N\}$$
$$Z = \{z_1, \ldots, z_N\}$$

$$P(X, Z|\theta) = p(z_1|\pi) \left[\prod_{n=2}^{N} p(z_n|z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m|z_m, \phi)$$

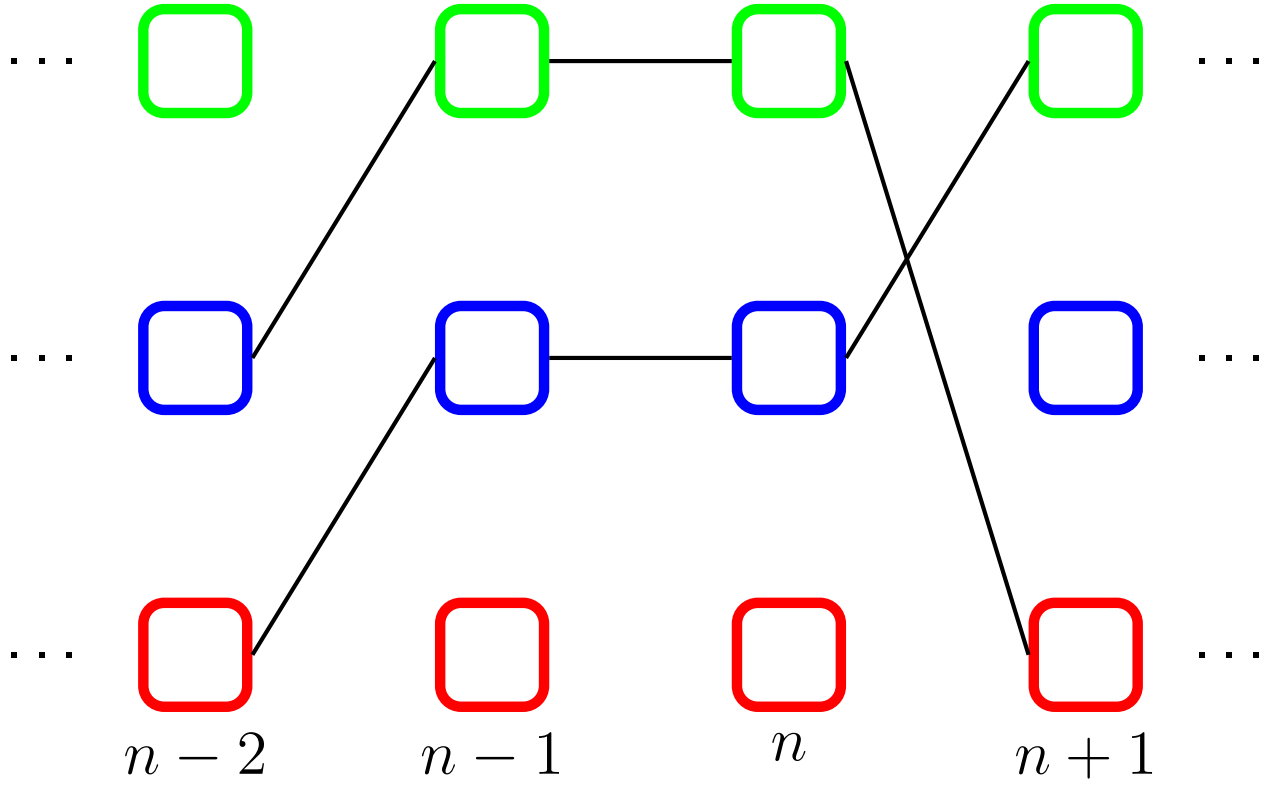


HMM Inference: Likelihood Function

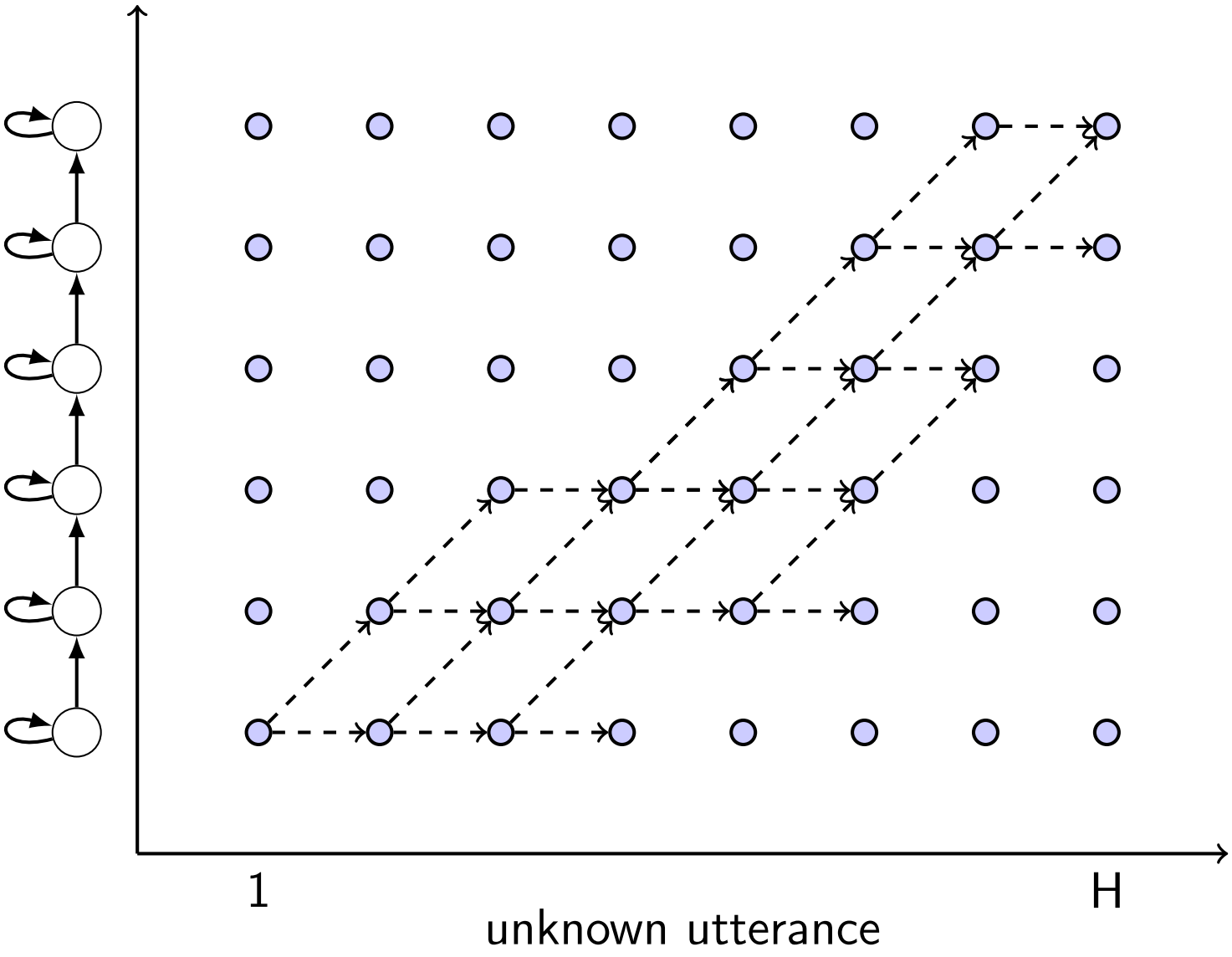
marginalise joint distribution over Z:

$$P(X|\theta) = \sum_Z p(X, Z|\theta)$$

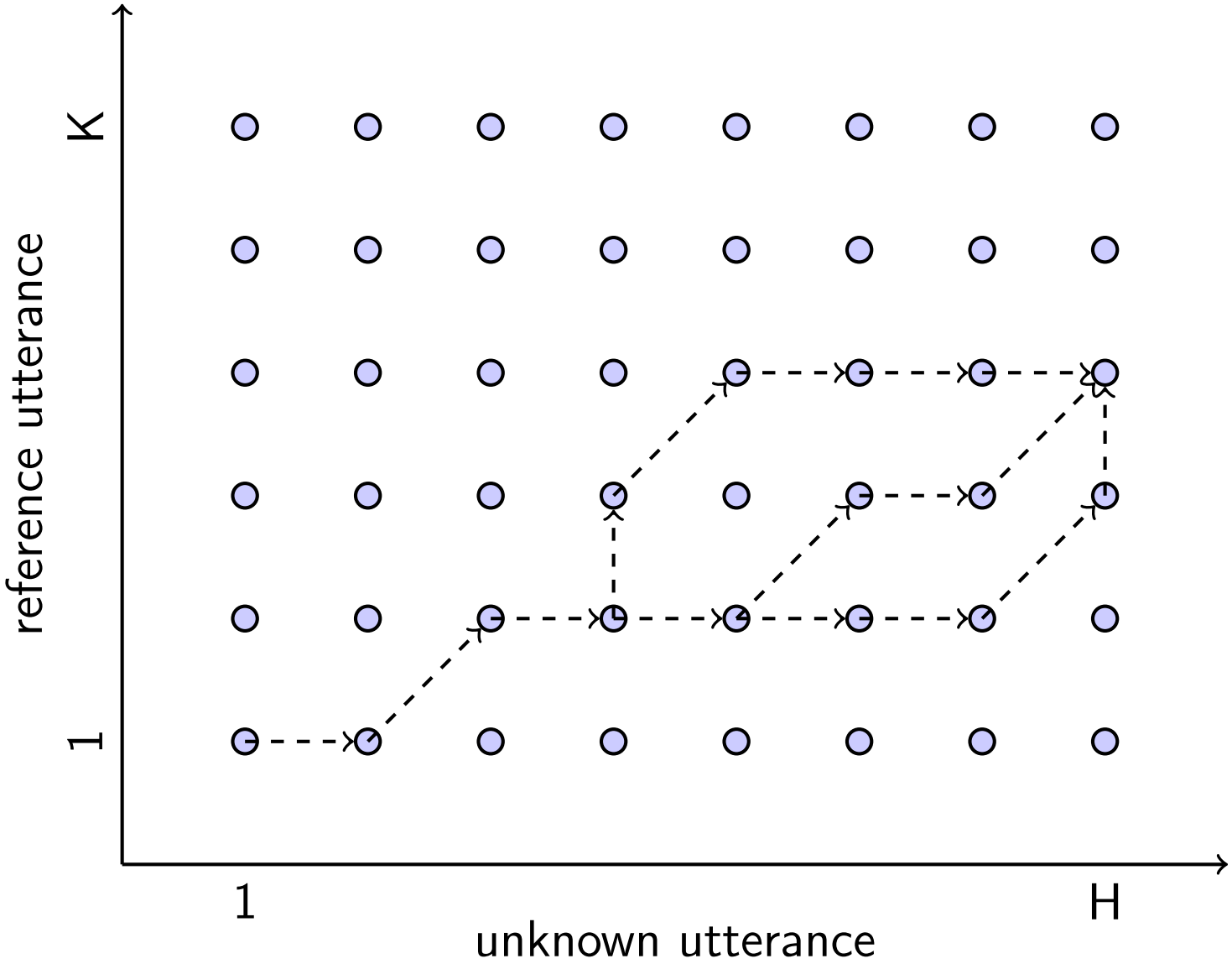
Problem: there are K^N possible sequences for Z



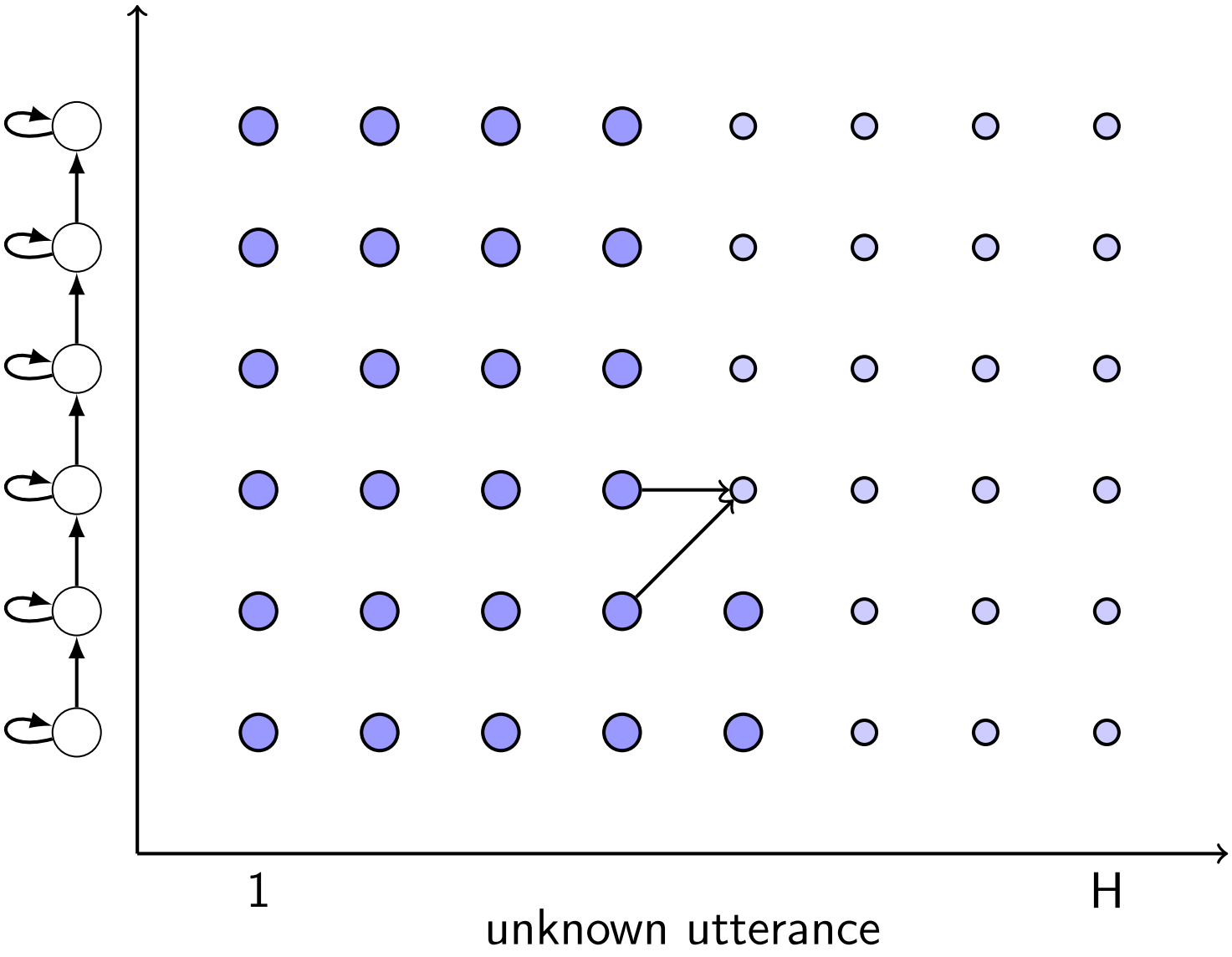
HMM Likelihood



Very Similar to Template Matching



Same Solution: Dynamic Programming



Solution: Forward algorithm

Instead of AccD[h,k] (Template Matching)

$$\alpha_n(j) \equiv p(x_1, \ldots, x_n, z_n = s_j | \theta)$$

At the end, instead of AccD[H,K]:

$$P(X|\theta) = \sum_{i=1}^{M} \alpha_N(i)$$

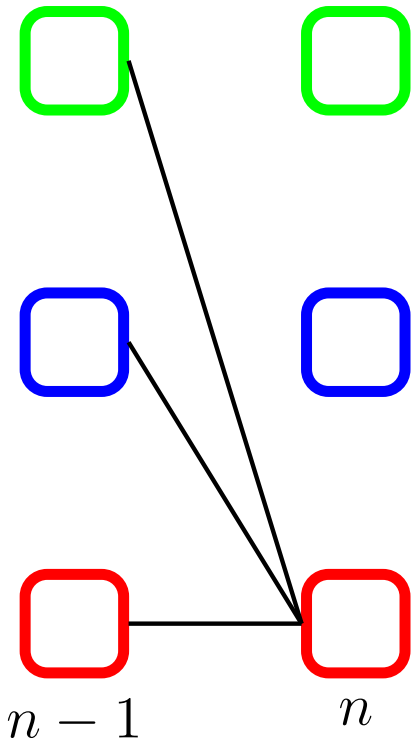
Forward Probability

Initialization:

$$\alpha_1(j) = \pi_j \phi_j(x_1)$$

Recursion:

$$\alpha_n(j) = \left[\sum_{i=1}^{M} \alpha_{n-1}(i) a_{ij} \right] \phi_j(x_n)$$



equivalent to sum-product in Bayesian Networks

Backward probability

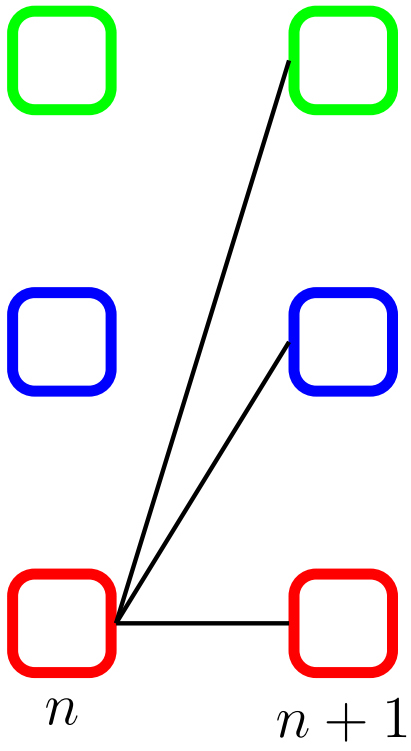
$$\beta_n(i) \equiv p(x_{n+1}, \ldots, x_N | z_n = s_i)$$

Initialization:

$$\beta_N(i) \equiv p(?|z_n = s_i) \equiv 1$$

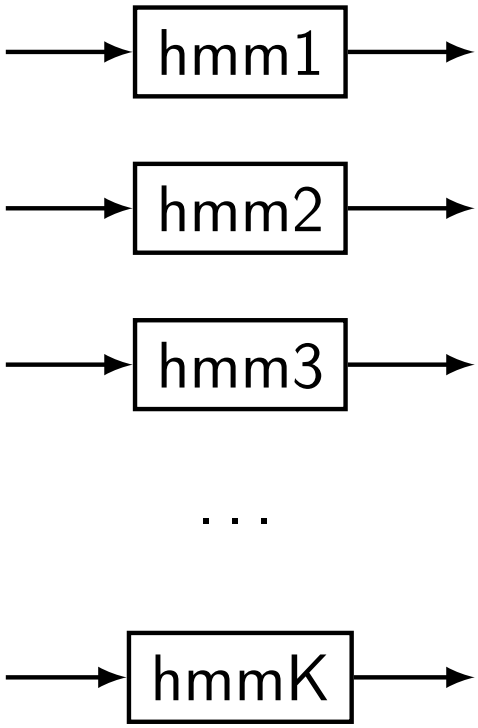
Recursion:

$$\beta_n(i) = \left[\sum_{j=1}^{M} a_{ij} \phi_j(x_{n+1}) \beta_{n+1}(j) \right]$$



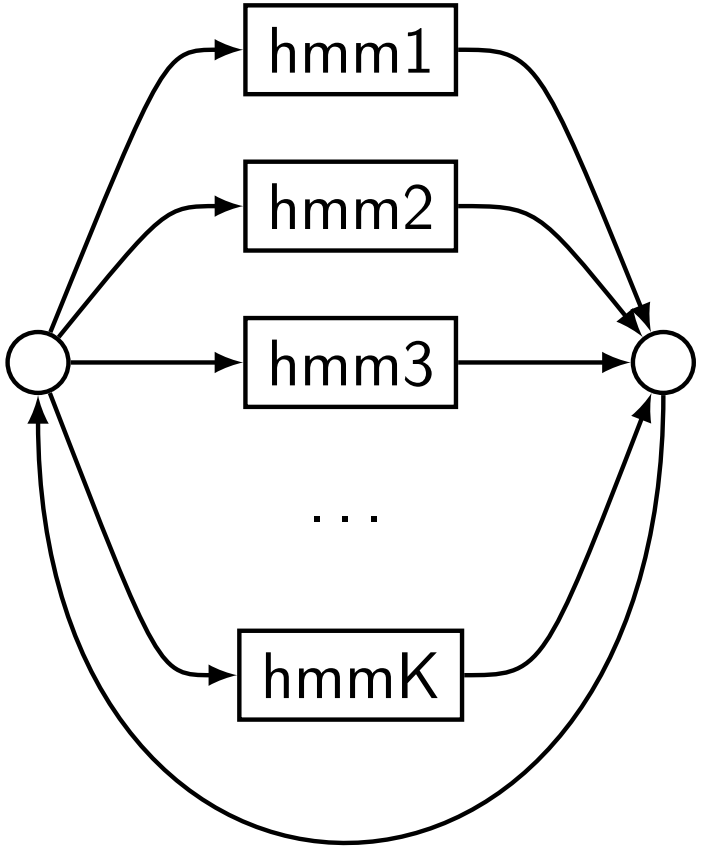
Find best sequence of states: why?

Isolated Words



(Likelihood)

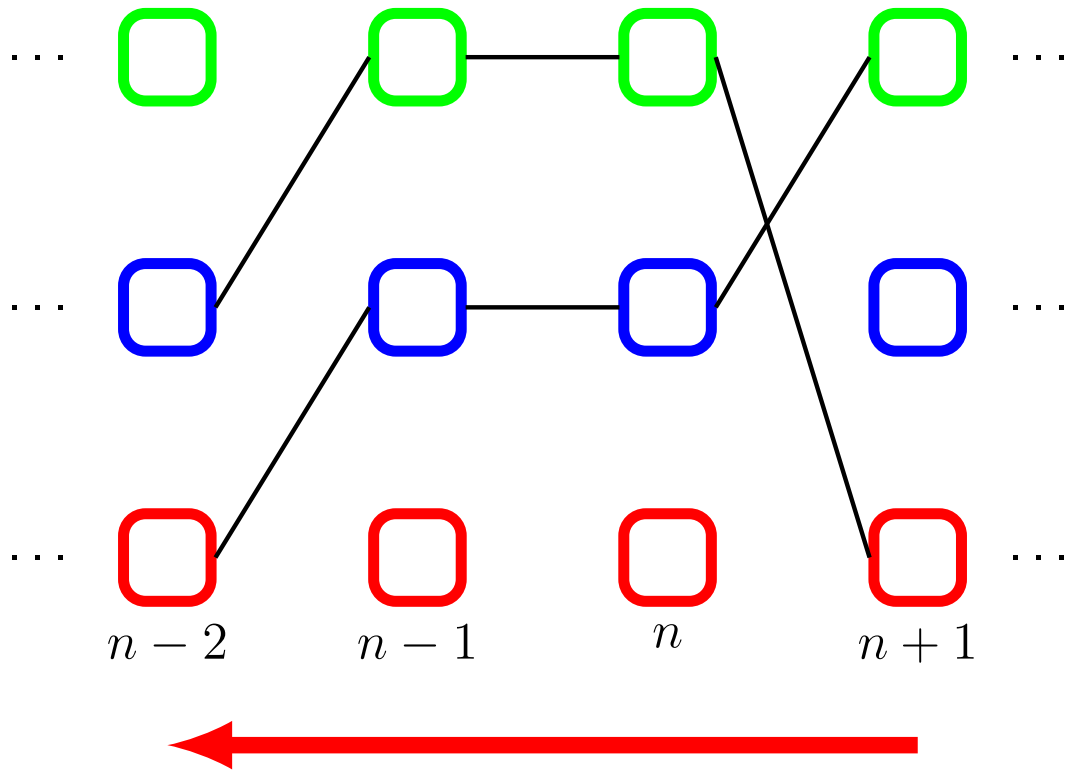
Continuous Speech



(best path)

Find best sequence of states: how?

- Viterbi algorithm[7]
- equivalent to max-sum in Bayesian Networks



[7] A. J. Viterbi. "Error Bounds for Convolutional Codes and an Asymtotically optimum decoding algorithm". In: *IEEE Trans. Inf. Theory* IT-13 (1967), pp. 260–269.

Summary: update rules

Forward algorithm (sum-product):

$$\alpha_n(j) = \left[\sum_{i=1}^{M} \alpha_{n-1}(i) a_{ij}\right] \phi_j(x_n)$$

Viterbi algorithm (max-sum):

$$V_n(j) = \max_{i=1}^{M} \left[V_{n-1}(i) a_{ij}\right] \phi_j(x_n)$$

$$B_n(j) = \arg\max_{i=1}^{M} \left[V_{n-1}(i) a_{ij}\right]$$

Practical Issues

Product of many probabilities: numerical problems
Solution: work in log domain

$$
\begin{aligned}
\alpha'_1(j) &= \pi'_j + \phi'_j(x_1) \\
\alpha'_n(j) &= \log \sum_{i=1}^{M} e^{\left(\alpha'_{n-1}(i) + a'_{ij}\right)} + \phi'_j(x_n) \\
\beta'_N(i) &= 0 \\
\beta'_n(i) &= \log \sum_{j=1}^{M} e^{\left(a'_{ij} + \phi'_j(x_{n+1}) + \beta'_{n+1}(j)\right)}
\end{aligned}
$$

In Lab2 `logsumexp()`.

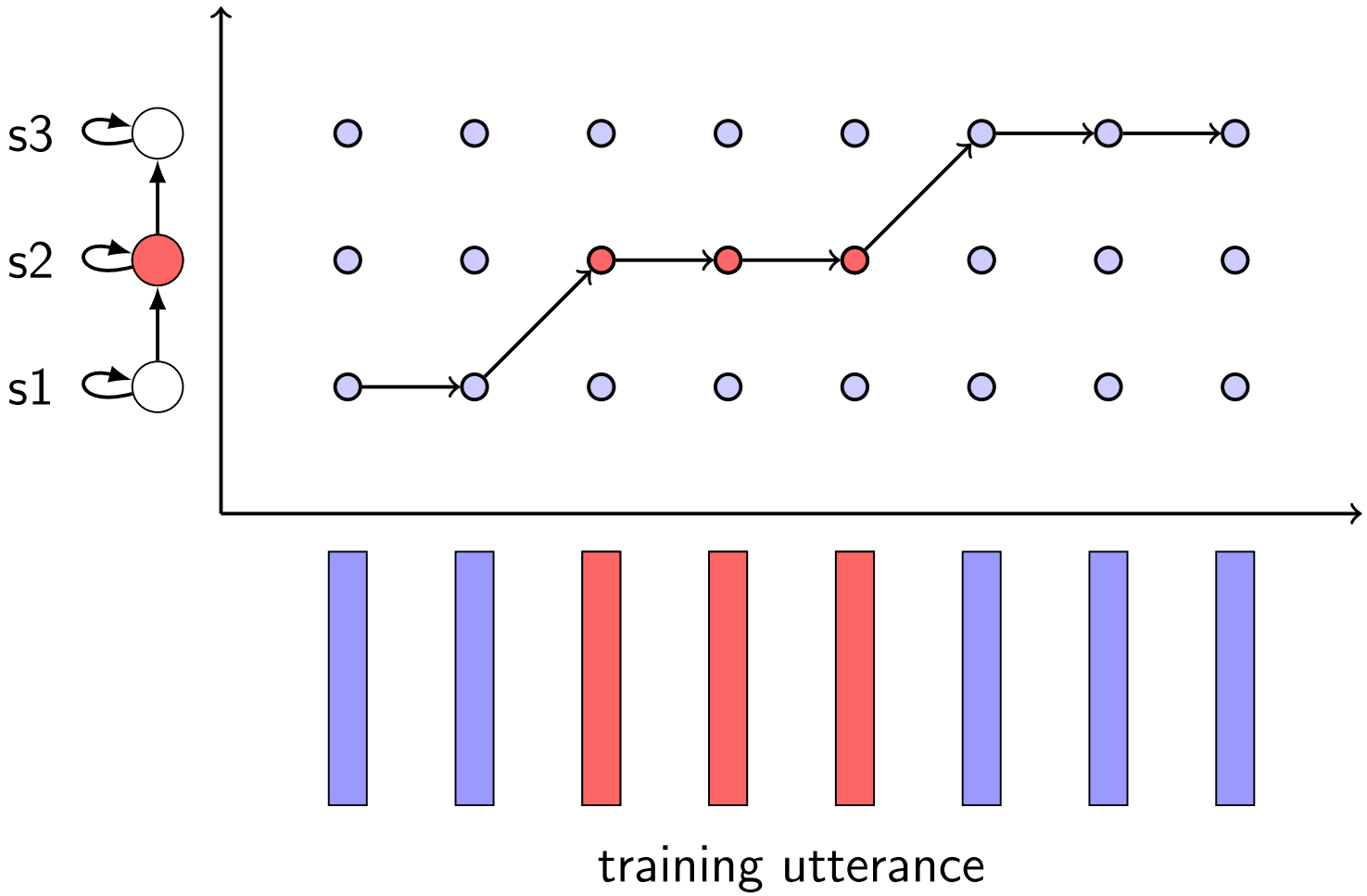
HMM Inference: Learning

- Given observations X update model parameters

$$\theta = \{\pi, A, \phi\}$$

- to maximise either:
  - model fit to data (e.g. likelihood, posterior)
  - classification performance (discriminative training)
- Incomplete data: state sequence Z

Viterbi training (simple approach)



problem: sensitive to misalignments

but still used for ANN/DNN training (Lab3)

HMM Inference: Learning

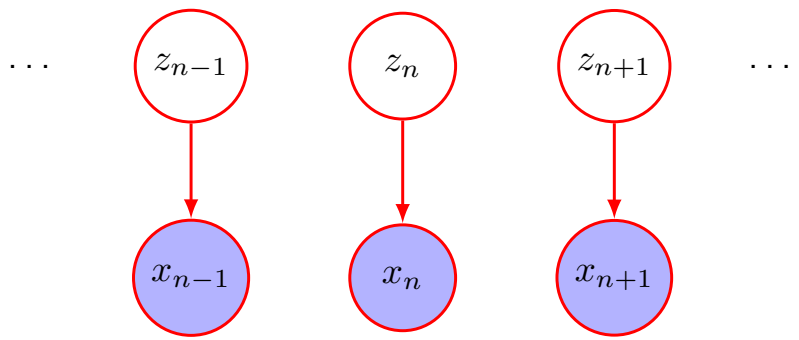
Latent variables $\rightarrow$ Expectation Maximisation

- locally maximise the likelihood of the complete data X, Z
- close form and efficient solution with froward-backward or Baum-Welch algorithm[8]
- general idea: sum over all possible paths weighted by probability of the path
- also: every observation vector contributes to all parameter updates

[8] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *Ann. Math. Statist.* 41.1 (1970), pp. 164–171.

Expectation Maximization for Mixture Models

$$P(x|\theta) = \sum_{k=1}^{K} \pi_k P(x|\theta_k),$$



$$\theta = \{\pi_1, \ldots, \pi_k, \theta_1, \ldots, \theta_K\},$$

$$\sum_{k=1}^{K} \pi_k = 1$$

- augment the data with the latent variables: $z_i \in \{1, \ldots, K\}$ assignment of each data point x_i to a component of the mixture
- interpret the mixture as marginal of the joint

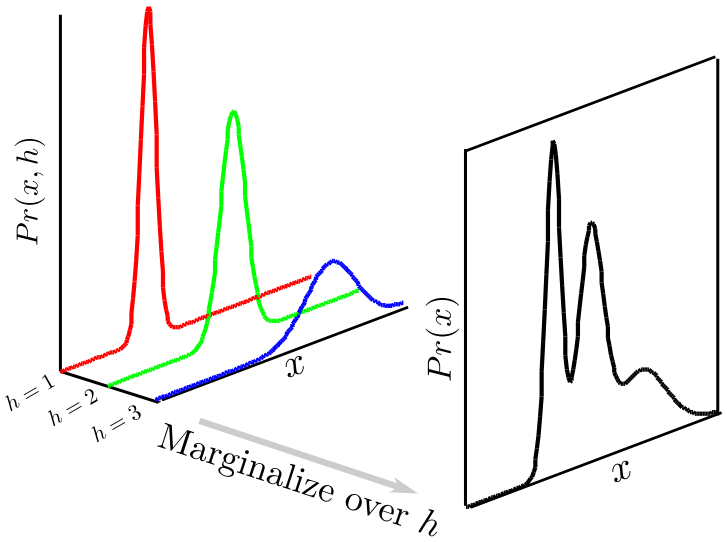
$$P(x|\theta) = \sum_{z} P(x, z|\theta)$$

Example: Mixture of Gaussians

Using the discrete hidden/latent variable z and $P(x, z)$:

$$p(x|\theta) = \sum_{k=1}^{K} p(x, z = k|\theta) = \sum_{k=1}^{K} p(x \mid z = k) P(z = k)$$

$$= \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$



$\leftarrow$ **mixture density**

Figures taken from **Computer Vision: models, learning and inference** by Simon Prince.

Expectation Maximization: Idea

Ideally we would like to maximize:

$$\log p(X|\theta) = \log \left\{ \sum_Z p(X, Z|\theta) \right\}$$

with $X = \{x_1, \ldots, x_N\}, \quad Z = \{z_1, \ldots, z_N\}$
...but log of sum hard to optimize
Instead optimize likelihood of complete data:

$$\log p(X, Z|\theta)$$

Z not known, but we can compute posterior given current model $p(Z|X, \theta^{\text{old}})$
Optimize the expected value of the likelihood:

$$\mathcal{Q}(\theta, \theta^{\text{old}}) = \sum_Z p(Z|X, \theta^{\text{old}}) \log p(X, Z|\theta)$$

Expectation Maximization in Practice

1. Initialization: choose initial value of θ^{old}
2. Expectation step: evaluate posterior $p(Z|X, \theta^{\text{old}})$
3. Maximization step: evaluate θ^{new} with:

$$\begin{aligned}\theta^{\text{new}} &= \arg\max_{\theta} \mathcal{Q}(\theta, \theta^{\text{old}}) \\ &= \arg\max_{\theta} \sum_{Z} p(Z|X, \theta^{\text{old}}) \log p(X, Z|\theta)\end{aligned}$$

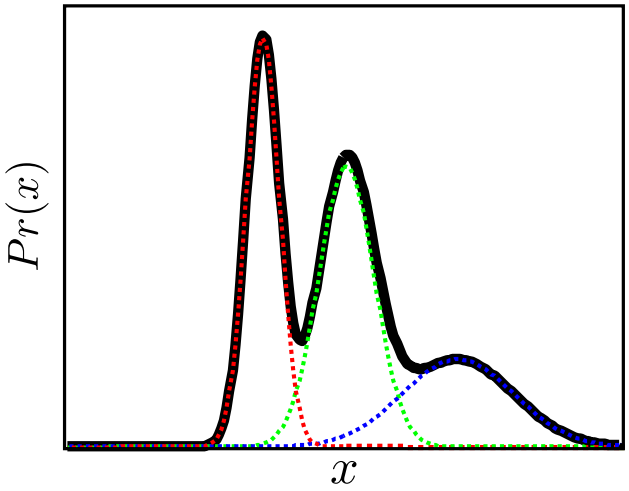
4. Check for convergence, otherwise $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ and go to step 2.

Mixture of Gaussians

This distribution is a weighted sum of K Gaussian distributions

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x; \mu_k, \sigma_k^2)$$

$$\theta = \{\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \sigma_1^2, \ldots, \sigma_K^2\}$$



This model can describe **complex multi-modal** probability distributions by combining simpler distributions.

EM for two Gaussians

Assume: We know the pdf of x has this form:

$$P(x) = \pi_1 \mathcal{N}(x; \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x; \mu_2, \sigma_2^2)$$

where $\pi_1 + \pi_2 = 1$ and $\pi_k > 0$ for components $k = 1, 2$.

Unknown: Values of the parameters

$$\theta = \{\pi_1, \mu_1, \sigma_1, \mu_2, \sigma_2\}.$$

Have: Observed N samples $x_1, \ldots, x_N$ drawn from $P(x)$.

Want to: Estimate θ from $x_1, \ldots, x_N$.

EM for two Gaussians

For each sample x_n introduce a *hidden variable* z_n

$$z_n = \begin{cases} 1 & \text{if sample } x_n \text{ was drawn from } \mathcal{N}(x; \mu_1, \sigma_1^2) \\ 2 & \text{if sample } x_n \text{ was drawn from } \mathcal{N}(x; \mu_2, \sigma_2^2) \end{cases}$$

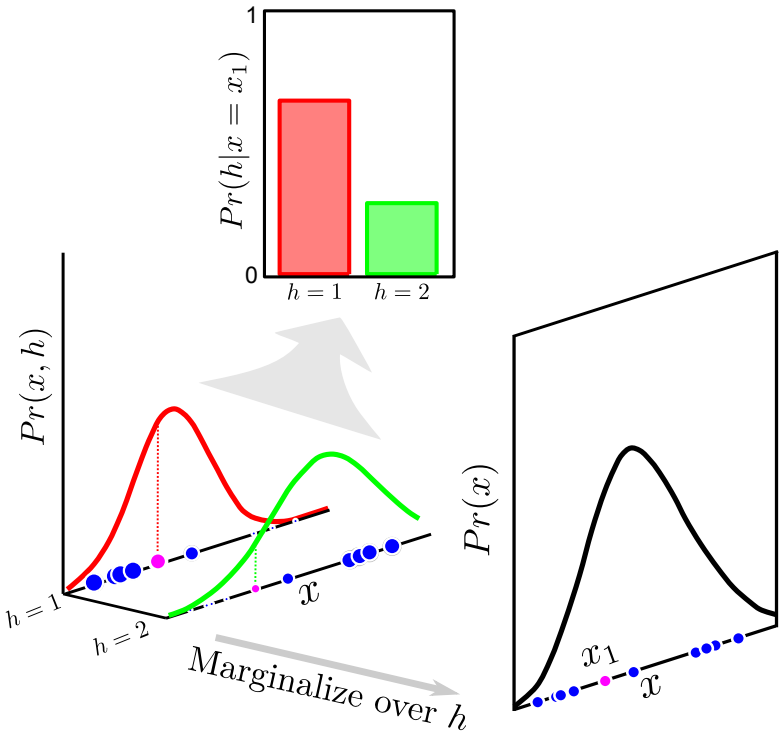
and come up with initial values

$$\theta^{(0)} = \{\pi_1^{(0)}, \mu_1^{(0)}, \sigma_1^{(0)}, \mu_2^{(0)}, \sigma_2^{(0)}\}$$

for each of the parameters.

EM for two Gaussians: E-step

The responsibility of k-th Gaussian for each sample x (indicated by the size of the projected data point)



Look at each sample x along hidden variable h in the E-step

Figure from **Computer Vision: models, learning and inference** by Simon Prince.

EM for two Gaussians: E-step (cont.)

E-step: Compute the *"posterior probability"* that x_n was generated by component k given the current estimate of the parameters $\theta^{(t)}$. (responsibilities)

for $n = 1, \ldots N$

for $k = 1, 2$

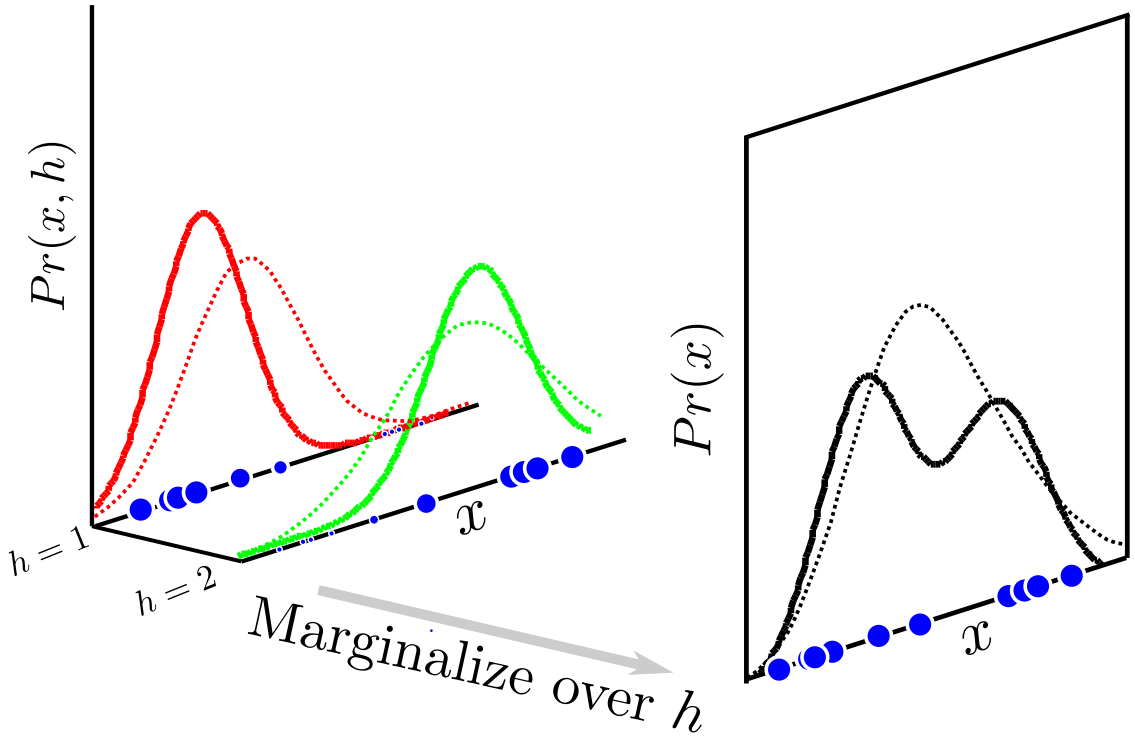
$$\gamma_n(k) = P(z_n = k \,|\, x_n, \theta^{(t)})$$

$$= \frac{\pi_k^{(t)} \mathcal{N}(x_n; \mu_k^{(t)}, \sigma_k^{(t)})}{\pi_1^{(t)} \mathcal{N}(x_n; \mu_1^{(t)}, \sigma_1^{(t)}) + \pi_2^{(t)} \mathcal{N}(x_n; \mu_2^{(t)}, \sigma_2^{(t)})}$$

Note: $\gamma_n(1) + \gamma_n(2) = 1$ and $\pi_1 + \pi_2 = 1$

EM for two Gaussians: M-step

Fitting the Gaussian model for each of k-th constituent.
Sample x_i contributes according to the responsibility $\gamma_i(k)$.



(dashed and solid lines for fit before and after update)

Look along samples x for each z in the M-step

Figure from **Computer Vision: models, learning and inference** by Simon Prince.

EM for two Gaussians: M-step (cont.)

M-step: Compute the *Maximum Likelihood* of the parameters of the mixture model given out data's membership distribution, the γ_n's:

for $k = 1, 2$

$$\mu_k^{(t+1)} = \frac{\sum_{n=1}^{N} \gamma_n(k) x_n}{\sum_{n=1}^{N} \gamma_n(k)},$$

$$\left(\sigma_k^{(t+1)}\right)^2 = \frac{\sum_{n=1}^{N} \gamma_n(k)(x_n - \mu_k^{(t+1)})^2}{\sum_{n=1}^{N} \gamma_n(k)},$$

$$\pi_k^{(t+1)} = \frac{\sum_{n=1}^{N} \gamma_n(k)}{N}.$$

Example of Expectation Maximization

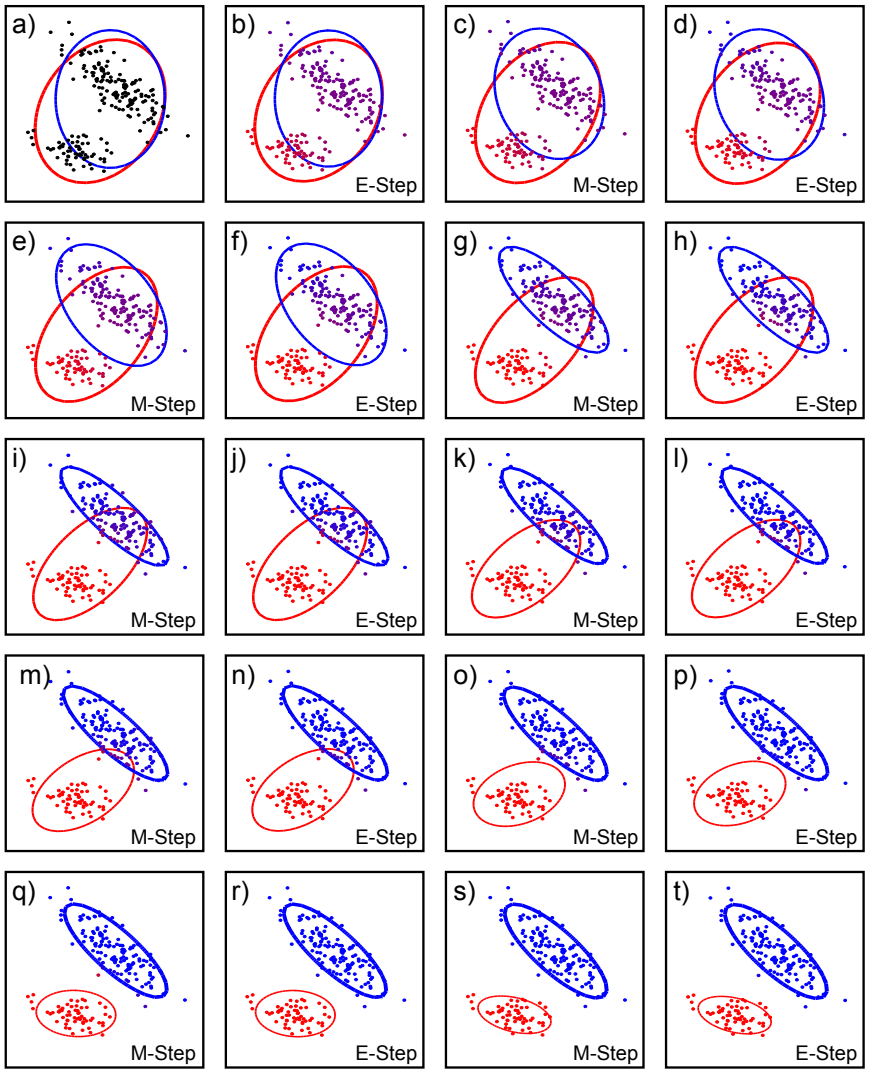


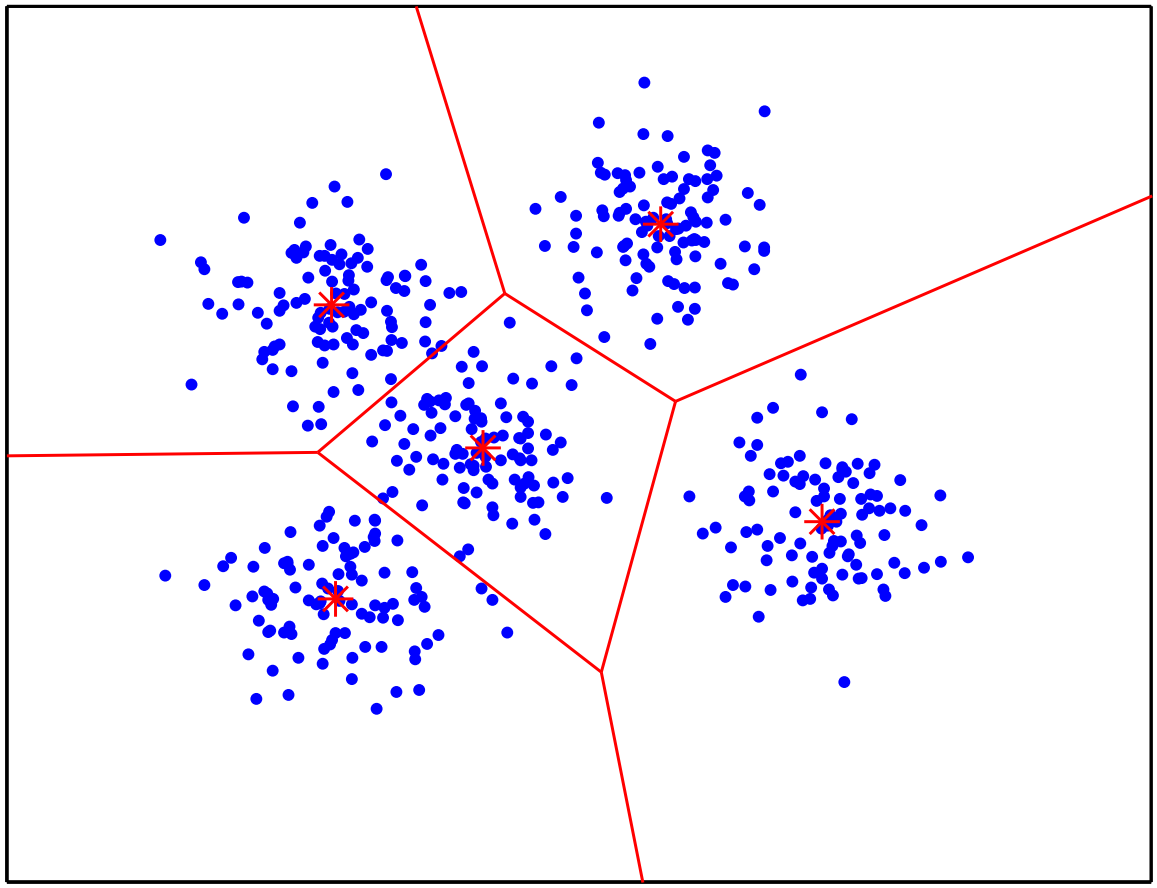
Figure from **Computer Vision: models, learning and inference** by Simon Prince.

Parallel with K-means

Data: k (number of desired clusters), n data points $\mathbf{x}_i$
Result: k clusters
initialization: assign initial value to k centroids $\mathbf{c}_i$;
repeat
  assign each point $\mathbf{x}_i$ to closest centroid $\mathbf{c}_j$;
  compute new centroids as mean of each group of points;
until *centroids do not change*;
return k clusters;

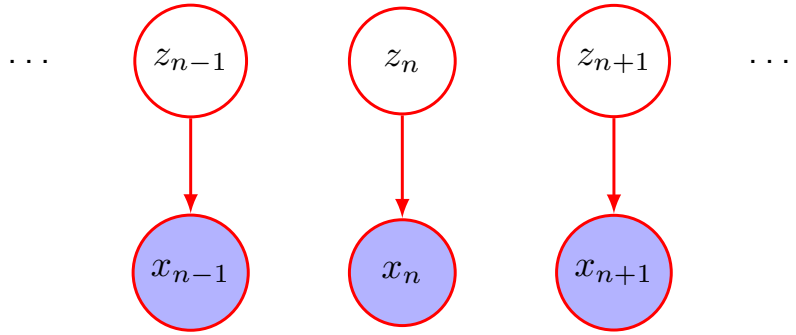
K-means: example

iteration 20, update clusters



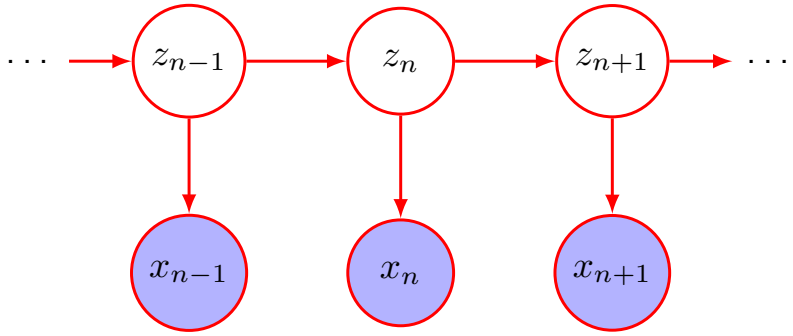
Mixture Models vs Hidden Markov Models

Mixture Model



EM Algorithm

Hidden Markov Model



Baum-Welch Algorithm (EM)

$$\gamma_n(k) = P(z_n = k \,|\, X, \theta^{(t)}) \neq P(z_n = k \,|\, x_n, \theta^{(t)})$$

Baum-Welch: Goal

Estimating parameters:

$$\theta = \{A, \pi, \phi\}$$

- transition prob. $A : a_{ij} = P(z_{n+1} = s_j | z_n = s_i)$
- prior prob. $\pi_i = P(z_1 = s_i)$
- emission prob. $\phi_i(x_n) = p(x_n | z_n = s_i)$
  For example, for
  - Discrete HMMS, $\phi_i(x_n) = \mathsf{Cat}(x_n | \lambda_{i1}, \ldots \lambda_{id})$
  - Gaussian HMMs, $\phi_i(x_n) = \mathcal{N}(x_n | \mu_i, \Sigma_i)$
  - GMM-HMMs, $\phi_i(x_n) = \sum_{k=1}^{K} w_k \mathcal{N}(x_n | \mu_{ik}, \Sigma_{ik})$

Baum-Welch E-Step

Estimate posterior $P(Z|X, \theta^{\text{old}})$ or, at least the sufficient statistics given:

- current model parameters
- full sequence of observations X

1. Posterior of latent variable z_n

$$\gamma_n(i) = P(z_n = s_i|X, \theta)$$

2. Posterior of two subsequent latent variables

$$\xi_n(i, j) = P(z_{n-1} = s_i, z_n = s_j|X, \theta)$$

Note: Huang, Acero and Hon call this $\gamma_n(i, j)$

Note: posteriors conditioned to full sequence X not only x_n

Baum-Welch M-Step

Weighted Maximum Likelihood estimates:
Emission probabilities:
Same as mixture model given $\gamma_n(i) = P(z_n = s_j | X, \theta)$
Transition probabilities

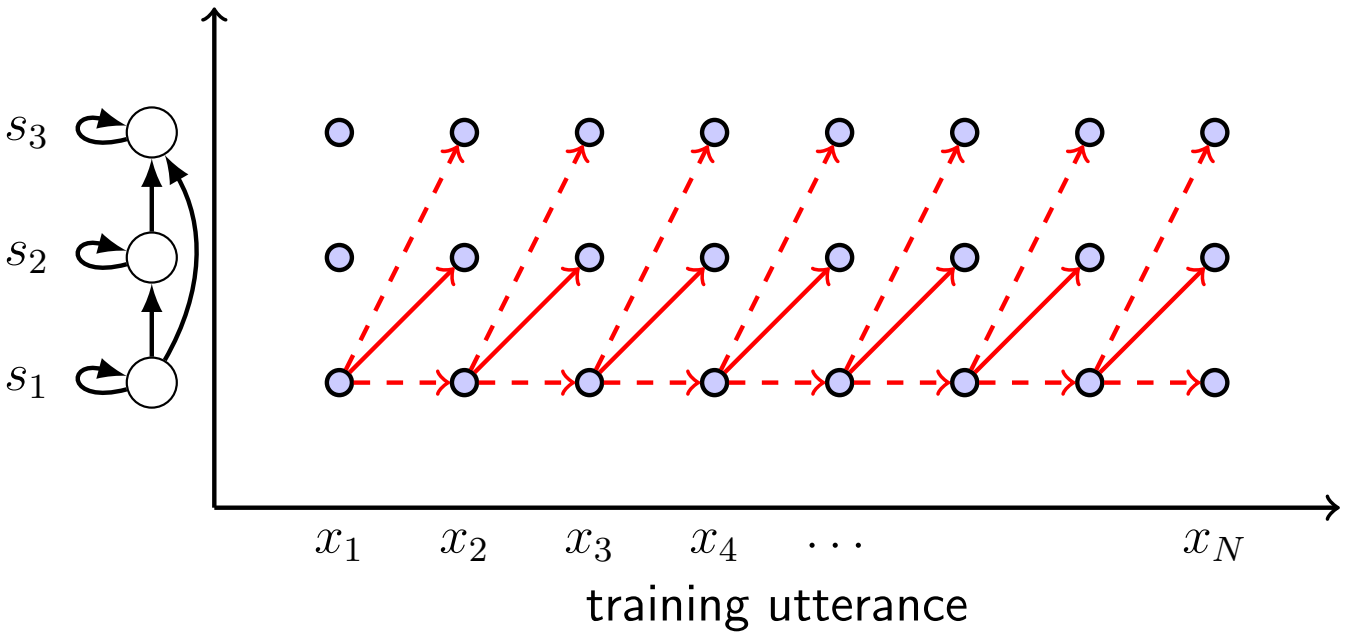
$$a_{ij}^{\text{new}} = \frac{E\left[s_i \to s_j | X, \theta^{\text{old}}\right]}{E\left[s_i \to s_{\text{any}} | X, \theta^{\text{old}}\right]} = \frac{\sum_{n=2}^{N} \xi_n(i,j)}{\sum_{n=2}^{N} \sum_{k=1}^{M} \xi_n(i,k)}$$

or, equivalently,

$$= \frac{E\left[s_i \to s_j | X, \theta^{\text{old}}\right]}{E\left[s_i | X, \theta^{\text{old}}\right]} = \frac{\sum_{n=2}^{N} \xi_n(i,j)}{\sum_{n=1}^{N-1} \gamma_n(i)}$$

Expectations are over the posteriors $P(Z | X, \theta^{\text{old}})$.

Example: Transition Probability



$$
\begin{aligned}
a_{12}^{\text{new}} &= \frac{E\left[s_1 \to s_2 | X, \theta^{\text{old}}\right]}{E\left[s_1 \to s_{\text{any}} | X, \theta^{\text{old}}\right]} = \frac{\sum_{n=2}^{N} \xi_n(1,2)}{\sum_{n=2}^{N} \sum_{k=1}^{3} \xi_n(1,k)} \\
&= \frac{E\left[s_1 \to s_2 | X, \theta^{\text{old}}\right]}{E\left[s_1 | X, \theta^{\text{old}}\right]} = \frac{\sum_{n=2}^{N} \xi_n(1,2)}{\sum_{n=1}^{N-1} \gamma_n(1)}
\end{aligned}
$$

Example: Transition Probability

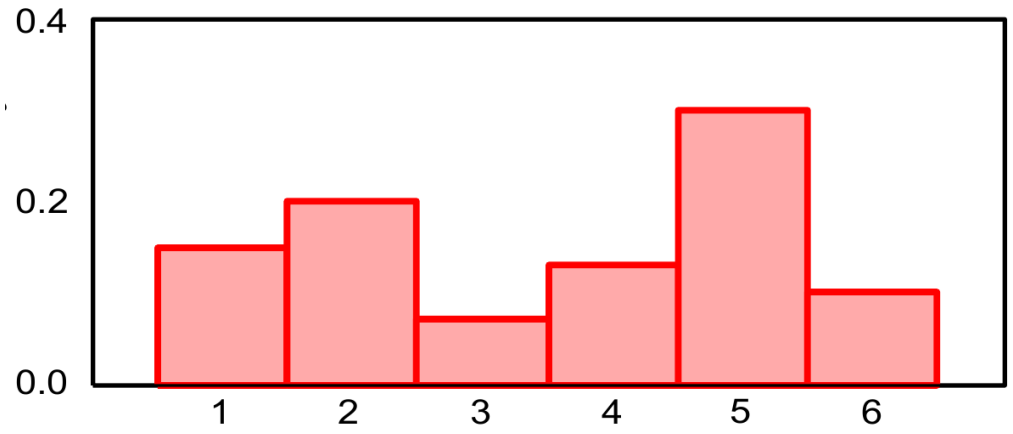
- $\sum_{n=2}^{N} \xi_n(i,j)$ is the expected number of transitions between state s_i and s_j (given X and θ^{old})
- $\sum_{n=1}^{N-1} \gamma_n(i)$ is the expected number of times we are in state s_i (given X and θ^{old})
- we never take a hard decision on when the transition happened

Emission probabilities

- Discrete HMMs (DHMMs)
  - vector quantisation
- Continuous HMMs
  - Single Gaussian $\phi_j(x_n) = N(x_n|\mu_j, \Sigma_j)$
  - Gaussian Mixture
- Semi-continuous HMMs (SCHMMs)

Discrete HMMs

- quantise feature vectors
- observation: sequence of discrete symbols
- $\phi_j(x_n)$ simple discrete probability distribution
- problem: quantisation error



Discrete HMMs: learn $\phi_j(x_n)$

Remember that

$$\gamma_n(j) = P(z_n = s_j | X, \theta) = \sum_{i=1}^{M} \xi_n(i, j)$$

are the posteriors of the latent variable
Update rule:

$$\phi_j(x_n = k) = \frac{E[x_n = k, z_n = s_j]}{E[z_n = s_j]} = \frac{\sum_{n:(x_n=k)} \gamma_n(j)}{\sum_{n=1}^{N} \gamma_n(j)}$$

HMMs with Gaussian Emission Probability

$$\phi_j(x_n) = N(x_n|\mu_j, \Sigma_j)$$

Update rules:

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma_n(j) x_n}{\sum_{n=1}^{N} \gamma_n(j)}$$

$$\Sigma_j = \frac{\sum_{n=1}^{N} \gamma_n(j) \left(x_n - \mu_j\right) \left(x_n - \mu_j\right)^T}{\sum_{n=1}^{N} \gamma_n(j)}$$

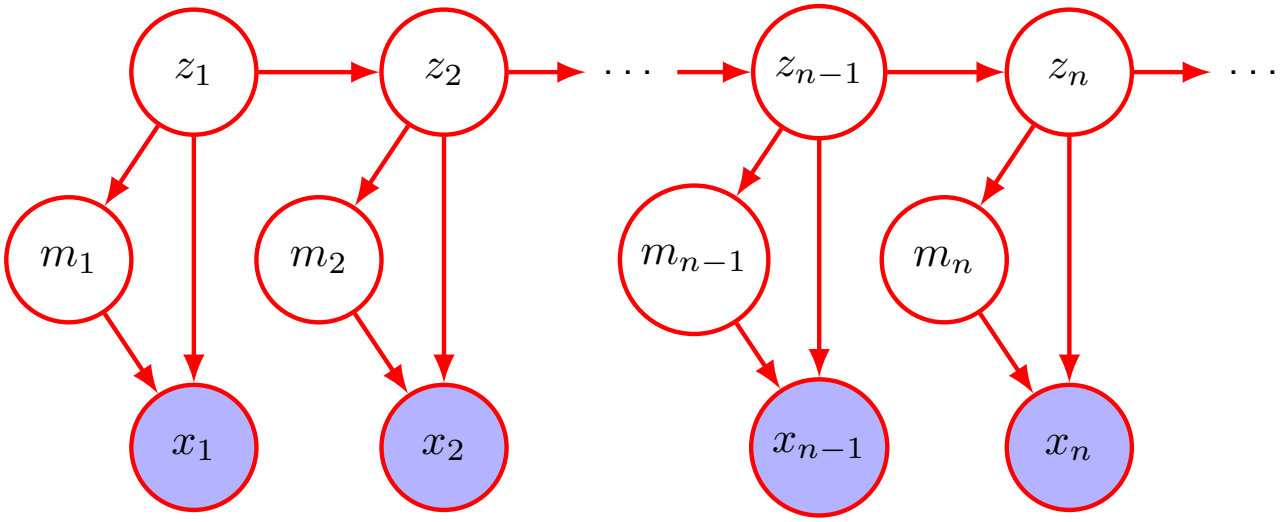
HMMs with Mixture Emission Probability

Often the Emission probability is modelled as a Mixture of Gaussians

$$\phi_j(x_n) = \sum_{k=1}^{K} w_{jk} N(x_n | \mu_{jk}, \Sigma_{jk})$$

$$\sum_{k=1}^{M} w_{jk} = 1$$

HMMs with Mixture Emission Probability



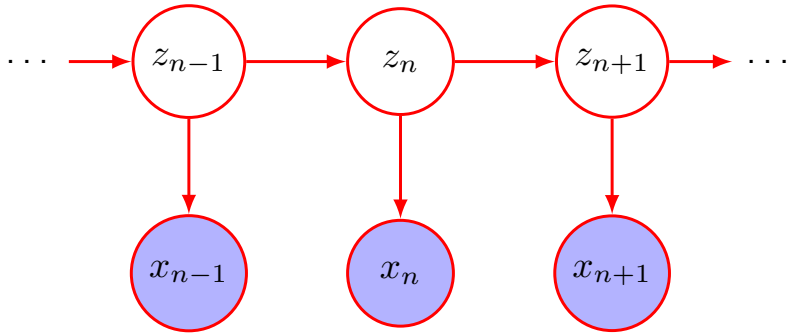
Emission:

$$
\begin{aligned}
p(x_n|z_n, m_n) &= \mathcal{N}(x_n; \mu_{z_n,m_n}, \Sigma_{z_n,m_n}) \\
p(m_n|z_n) &= W(m_n, z_n)
\end{aligned}
$$

Semi-Continuous HMMs

- All Gaussian distributions in a pool of pdfs
- each $\phi_j(x_n)$ is a discrete probability distribution over the pool of Gaussians
- similar to quantisation, but probabilistic
- used for sharing parameters

Calculate sufficient statistics



$$\begin{aligned}\gamma_n(i) &= P(z_n = s_i | X, \theta) \\ \xi_n(i,j) &= P(z_{n-1} = s_i, z_n = s_j | X, \theta)\end{aligned}$$

We can do this with the help of the forward and backward variables:

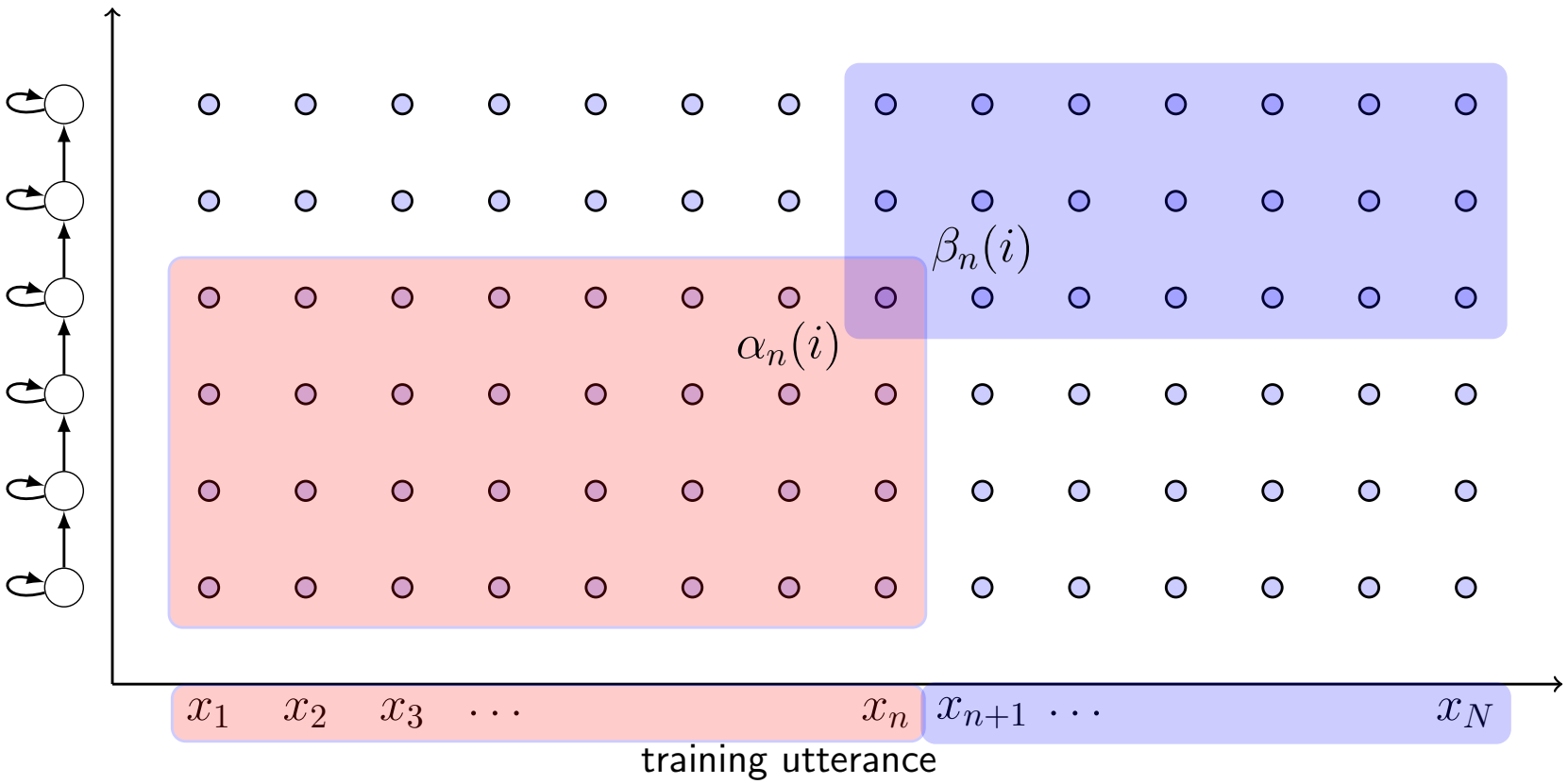
$$\begin{aligned}\alpha_n(i) &= P(x_1, \ldots, x_n, z_n = s_i | \theta) \\ \beta_n(i) &= P(x_{n+1}, \ldots, x_N | z_n = s_i, \theta)\end{aligned}$$

Calculate γ (forward-backward)

$$
\begin{aligned}
\gamma_n(i) &= P(z_n = s_i | X, \theta) \\
&= \frac{p(X, z_n = s_j | \theta)}{p(X|\theta)} \\
&= \frac{p(x_1, \ldots, x_n, x_{n+1}, \ldots, x_N, z_n = s_j | \theta)}{p(X|\theta)} \\
&= \frac{p(x_1, \ldots, x_n, x_{n+1}, \ldots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X|\theta)} \\
&= \frac{p(x_1, \ldots, x_n | z_n = s_j, \theta) p(x_{n+1}, \ldots, x_N | z_n = s_j, \theta) P(z_n = s_j | \theta)}{p(X|\theta)} \\
&= \frac{p(x_1, \ldots, x_n, z_n = s_j | \theta) p(x_{n+1}, \ldots, x_N | z_n = s_j, \theta)}{p(X|\theta)} \\
&= \frac{\alpha_n(i) \beta_n(i)}{\sum_i \alpha_N(i)}
\end{aligned}
$$

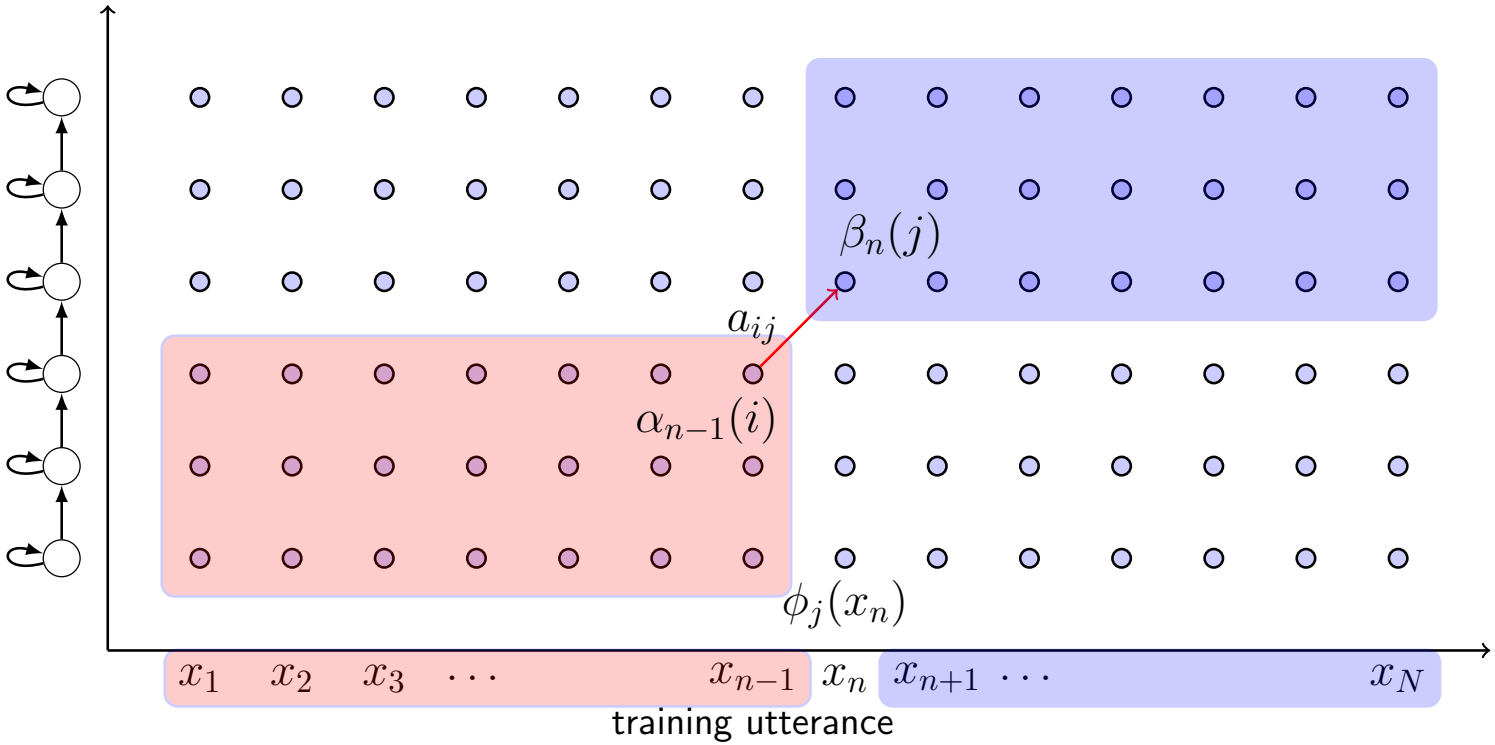
Calculate γ: Illustration

$$\gamma_n(i) = P(z_n = s_i | X, \theta) = \frac{\alpha_n(i)\beta_n(i)}{\sum_i \alpha_N(i)}$$



Calculate ξ (forward-backward)

$$
\begin{aligned}
\xi_n(i,j) &= P(z_n = s_j, z_{n-1} = s_i | X, \theta) \\
&= \frac{P(z_n = s_j, z_{n-1} = s_i, X | \theta)}{P(X|\theta)} \\
\ldots &= \frac{\alpha_{n-1}(i)\ a_{ij}\ \phi_j(x_n)\ \beta_n(j)}{\sum_{k=1}^{M} \alpha_N(k)}
\end{aligned}
$$



Baum-Welch: Properties

instance of Expectation Maximisation:

- iterative procedure
- guaranteed to convert to local maximum of the likelihood $P(X|\theta^{\text{new}})$
- sensitive to initialisation
- update formulae for emission probability model $\phi_j(x_n)$ same as for mixture models (with new version of posteriors)

Practical issues

Set of utterances $X^1, X^2, \ldots, X^U$

- cannot concatenate them: need to calculate α, β, γ and ξ each time

Each utterance corresponds to several models

- reuse model states (sentence $\rightarrow$ words $\rightarrow$ phonemes)