

Exact Inference for Hidden Markov Models

Michael U. Gutmann

Probabilistic Machine Learning (INFR11298)
School of Informatics, The University of Edinburgh

Autumn Semester 2026

Recap

- ▶ Assuming a factorisation / set of statistical independencies allowed us to efficiently represent the pdf or pmf of random variables
- ▶ Factorisation can be exploited for inference
 - ▶ by using the distributive law
 - ▶ by re-using already computed quantities
- ▶ Inference for general factor graphs (variable elimination)
- ▶ Inference for factor trees
- ▶ Sum-product and max-sum message passing

Program

1. Markov models
2. Inference by message passing

Program

1. Markov models

- Markov chains
- Transition distribution
- Hidden Markov models (HMMs)
- Emission distribution
- Important instances of HMMs

2. Inference by message passing

Applications of (hidden) Markov models

Markov and hidden Markov models are widely used, e.g. for

- ▶ speech modelling (speech recognition)
- ▶ text modelling (natural language processing)
- ▶ gene sequence modelling (bioinformatics)
- ▶ spike train modelling (neuroscience)
- ▶ object tracking (robotics)
- ▶ stock price prediction (finance)
- ▶ navigation systems (aerospace)
- ▶ ⋮

Recent applications

They also play a role in modern generative AI, e.g.

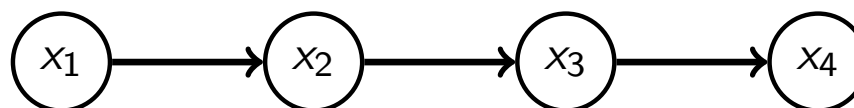
- ▶ to help control text generation in LLMs
(<https://openreview.net/forum?id=58X9v92zRd>)
- ▶ to speed up text generation in LLMs
(<https://openreview.net/forum?id=6kCEyw9god>)
- ▶ for text-to-speech generation
(<https://arxiv.org/pdf/2108.13320>)
- ▶ to help understand representations learned by transformers
(<https://proceedings.mlr.press/v267/piotrowski25a.html>)
- ▶ ...

Markov chains

- ▶ First-order Markov chain: models that factorise as

$$p(x_1, \dots, x_d) = \prod_{i=1}^d p(x_i | x_{i-1})$$

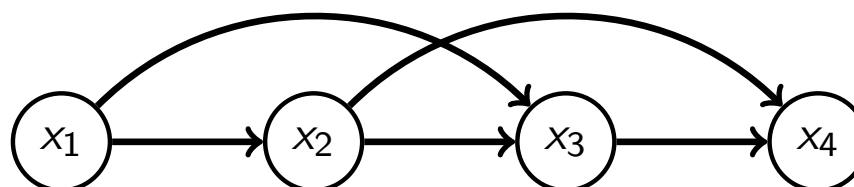
- ▶ DAG (for $d=4$)



- ▶ L -th order Markov chain: models that factorise as

$$p(x_1, \dots, x_d) = \prod_{i=1}^d p(x_i | x_{i-L}, \dots, x_{i-1})$$

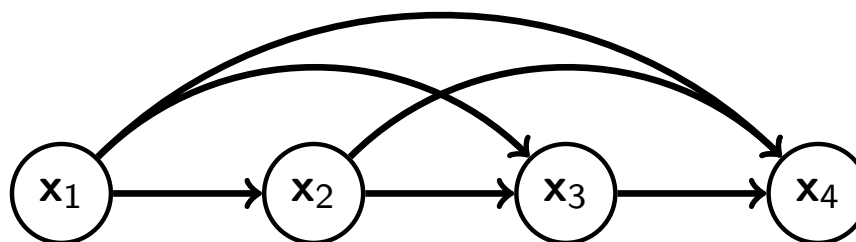
- ▶ DAG ($L=2, d=4$)



Vector-valued Markov chains

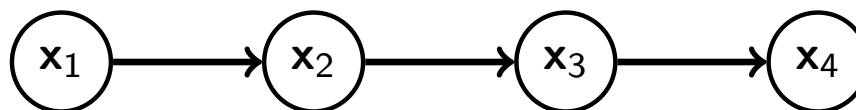
- ▶ While not explicitly discussed, the graphical models extend to vector-valued variables.
- ▶ Chain rule with ordering $\mathbf{x}_1, \dots, \mathbf{x}_d$

$$p(\mathbf{x}_1, \dots, \mathbf{x}_d) = \prod_{i=1}^d p(\mathbf{x}_i | \mathbf{x}_1, \dots, \mathbf{x}_{i-1})$$



- ▶ 1st order Markov chain:

$$p(\mathbf{x}_1, \dots, \mathbf{x}_d) = \prod_{i=1}^d p(\mathbf{x}_i | \mathbf{x}_{i-1})$$



Modelling time series

- ▶ Index i may refer to time t
- ▶ For example, 1st order Markov chain of length T :

$$p(x_1, \dots, x_T) = \prod_{t=1}^T p(x_t | x_{t-1})$$

- ▶ Only the last time point x_{t-1} is relevant for x_t .

Transition distribution

(Consider 1st order Markov chain.)

- ▶ $p(x_i|x_{i-1})$ is called the transition distribution
- ▶ For discrete random variables, $p(x_i|x_{i-1})$ is defined by a transition matrix $\mathbf{A}^{(i)}$

$$p(x_i = k|x_{i-1} = k') = A_{k,k'}^{(i)} \quad (A_{k',k}^{(i)} \text{ convention is also used})$$

- ▶ For continuous random variables, $p(x_i|x_{i-1})$ is a conditional pdf, e.g.

$$p(x_i|x_{i-1}) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x_i - f_i(x_{i-1}))^2}{2\sigma_i^2}\right)$$

for some (typically parameterised) function f_i

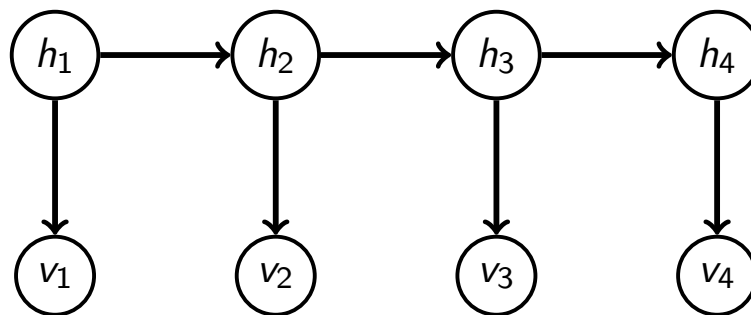
- ▶ Homogeneous Markov chain: $p(x_i|x_{i-1})$ does not depend on i , e.g.

$$\mathbf{A}^{(i)} = \mathbf{A} \quad \text{or} \quad \sigma_i = \sigma, \quad f_i = f$$

- ▶ Inhomogeneous Markov chain: $p(x_i|x_{i-1})$ does depend on i

Hidden Markov models (HMMs)

DAG:



- ▶ 1st order Markov chain on hidden (latent) variables h_i .
- ▶ Each visible (observed) variable v_i only depends on the corresponding hidden variable h_i
- ▶ Factorisation

$$p(h_{1:d}, v_{1:d}) = p(v_1|h_1)p(h_1) \prod_{i=2}^d p(v_i|h_i)p(h_i|h_{i-1})$$

- ▶ The visibles are d-connected if hiddens are not observed
- ▶ Visibles are d-separated (independent) given the hiddens
- ▶ The h_i s model/explain all dependencies between the v_i s

Emission distribution

- ▶ $p(v_i|h_i)$ is called the emission distribution
- ▶ Discrete-valued v_i and h_i :
 $p(v_i|h_i)$ can be represented as a matrix
- ▶ Discrete-valued v_i and continuous-valued h_i :
 $p(v_i|h_i)$ is a conditional pmf.
- ▶ Continuous-valued v_i : $p(v_i|h_i)$ is a density
- ▶ As for the transition distribution, the emission distribution $p(v_i|h_i)$ may depend on i or not.
- ▶ If neither the transition nor the emission distribution depend on i , we have a stationary (or homogeneous) hidden Markov model.

Gaussian emission model with discrete-valued latents

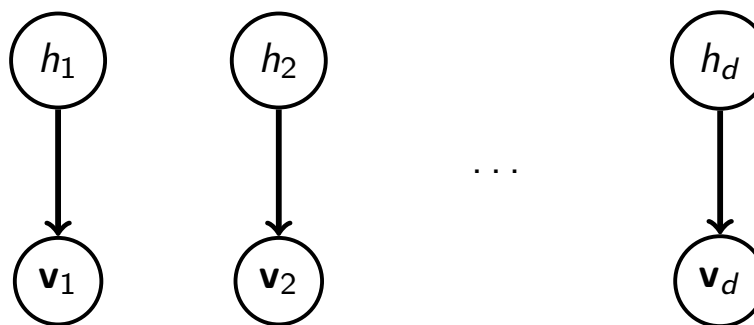
- Special case: $h_i \perp\!\!\!\perp h_{i-1}$, and $\mathbf{v}_i \in \mathbb{R}^m$, $h_i \in \{1, \dots, K\}$

$$p(h = k) = p_k$$

$$p(\mathbf{v} | h = k) = \frac{1}{|\det 2\pi \Sigma_k|^{1/2}} \exp \left(-\frac{1}{2} (\mathbf{v} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{v} - \boldsymbol{\mu}_k) \right)$$

for all h_i and $\mathbf{v}_i$.

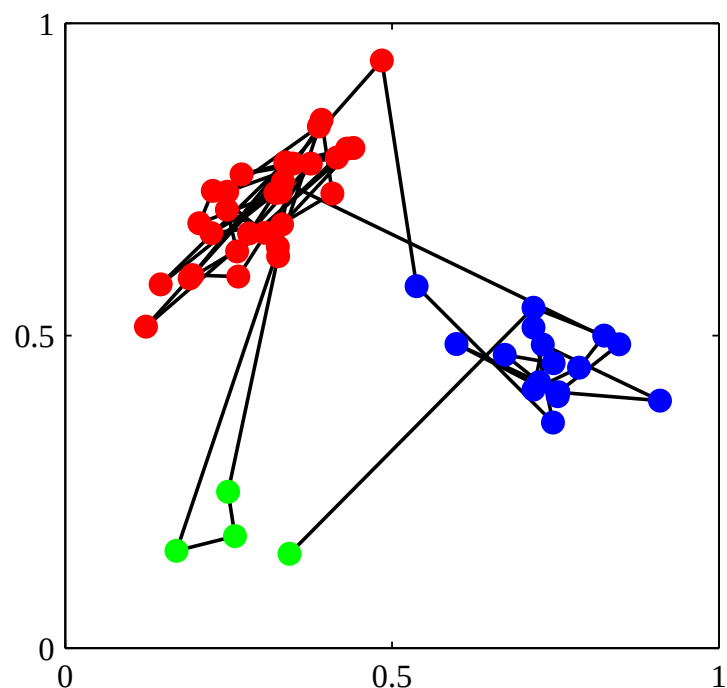
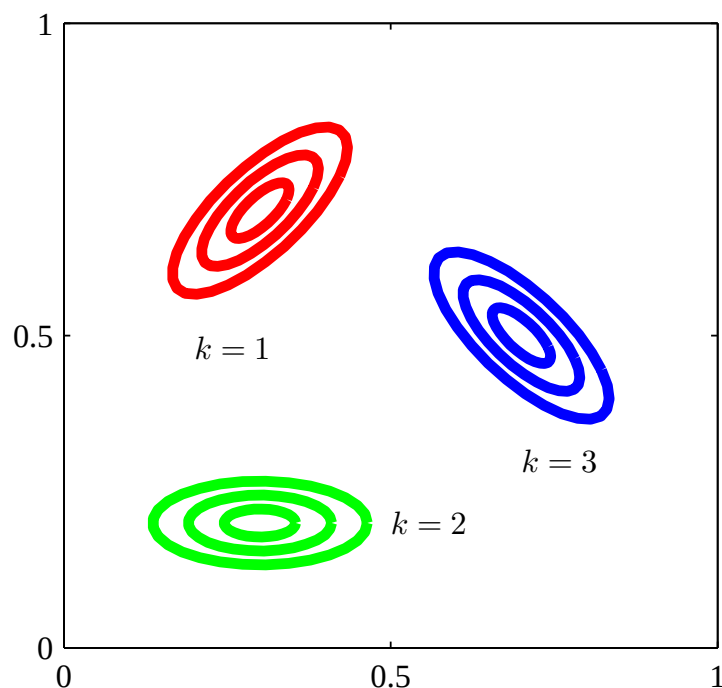
- DAG



- Corresponds to d iid draws from a Gaussian mixture model with K mixture components
 - Mean $\mathbb{E}[\mathbf{v} | h = k] = \boldsymbol{\mu}_k$
 - Covariance matrix $\mathbb{V}[\mathbf{v} | h = k] = \Sigma_k$

Gaussian emission model with discrete-valued latents

The HMM is a generalisation of the Gaussian mixture model where cluster membership at “time” i (the value of h_i) generally depends on cluster membership at “time” $i - 1$ (the value of h_{i-1}).



Example for $\mathbf{v}_i \in \mathbb{R}^2$, $h_i \in \{1, 2, 3\}$. Left: $p(\mathbf{v}|h = k)$. Right: samples

(Bishop, Figure 13.8)

Gaussian emission model with Gaussian latents

- ▶ HMMs with Gaussian emission and transition distributions correspond to linear dynamical systems.

- ▶ Transition model:

$$p(\mathbf{h}_i | \mathbf{h}_{i-1}) = \mathcal{N}(\mathbf{h}_i; \mathbf{A}\mathbf{h}_{i-1}, \mathbf{\Sigma}^h) \quad (1)$$

$$\mathbf{h}_i = \mathbf{A}\mathbf{h}_{i-1} + \mathbf{n}_i^h, \quad \mathbf{n}_i^h \sim \mathcal{N}(\mathbf{n}_i^h; 0, \mathbf{\Sigma}^h) \quad (2)$$

- ▶ Emission model:

$$p(\mathbf{v}_i | \mathbf{h}_{i-1}) = \mathcal{N}(\mathbf{v}_i; \mathbf{C}\mathbf{h}_{i-1}, \mathbf{\Sigma}^v) \quad (3)$$

$$\mathbf{v}_i = \mathbf{C}\mathbf{h}_{i-1} + \mathbf{n}_i^v, \quad \mathbf{n}_i^v \sim \mathcal{N}(\mathbf{n}_i^v; 0, \mathbf{\Sigma}^v) \quad (4)$$

- ▶ If $p(\mathbf{h}_1)$ is Gaussian, the whole model is jointly Gaussian
- ▶ Computation of $p(\mathbf{h}_t | \mathbf{v}_{1:t})$ is the filtering problem: for the model above, this was solved by Kalman (1960) and is now called Kalman filtering.
- ▶ Very widely used, e.g. in location and navigation systems.

Program

1. Markov models

- Markov chains
- Transition distribution
- Hidden Markov models (HMMs)
- Emission distribution
- Important instances of HMMs

2. Inference by message passing

Program

1. Markov models

2. Inference by message passing

- Inference: filtering, prediction, smoothing, Viterbi
- Filtering: Sum-product message passing yields the α -recursion
- Smoothing: Sum-product message passing yields the α - β recursion

The classical inference problems

(Considering the index i to refer to time t)

Filtering	(Inferring the present)	$p(h_t v_{1:t})$
Smoothing	(Inferring the past)	$p(h_t v_{1:u}) \quad t < u$
Prediction	(Inferring the future)	$p(h_t v_{1:u}) \quad t > u$ $p(v_t v_{1:u}) \quad t > u$
Most likely hidden path	(Viterbi algorithm)	$\operatorname{argmax}_{h_{1:t}} p(h_{1:t} v_{1:t})$
Posterior sampling	(Forward filtering backward sampling)	$h_{1:t} \sim p(h_{1:t} v_{1:t})$

For the HMM, all tasks can be solved via message passing (sum-product or max-sum algorithm).

The classical inference problems

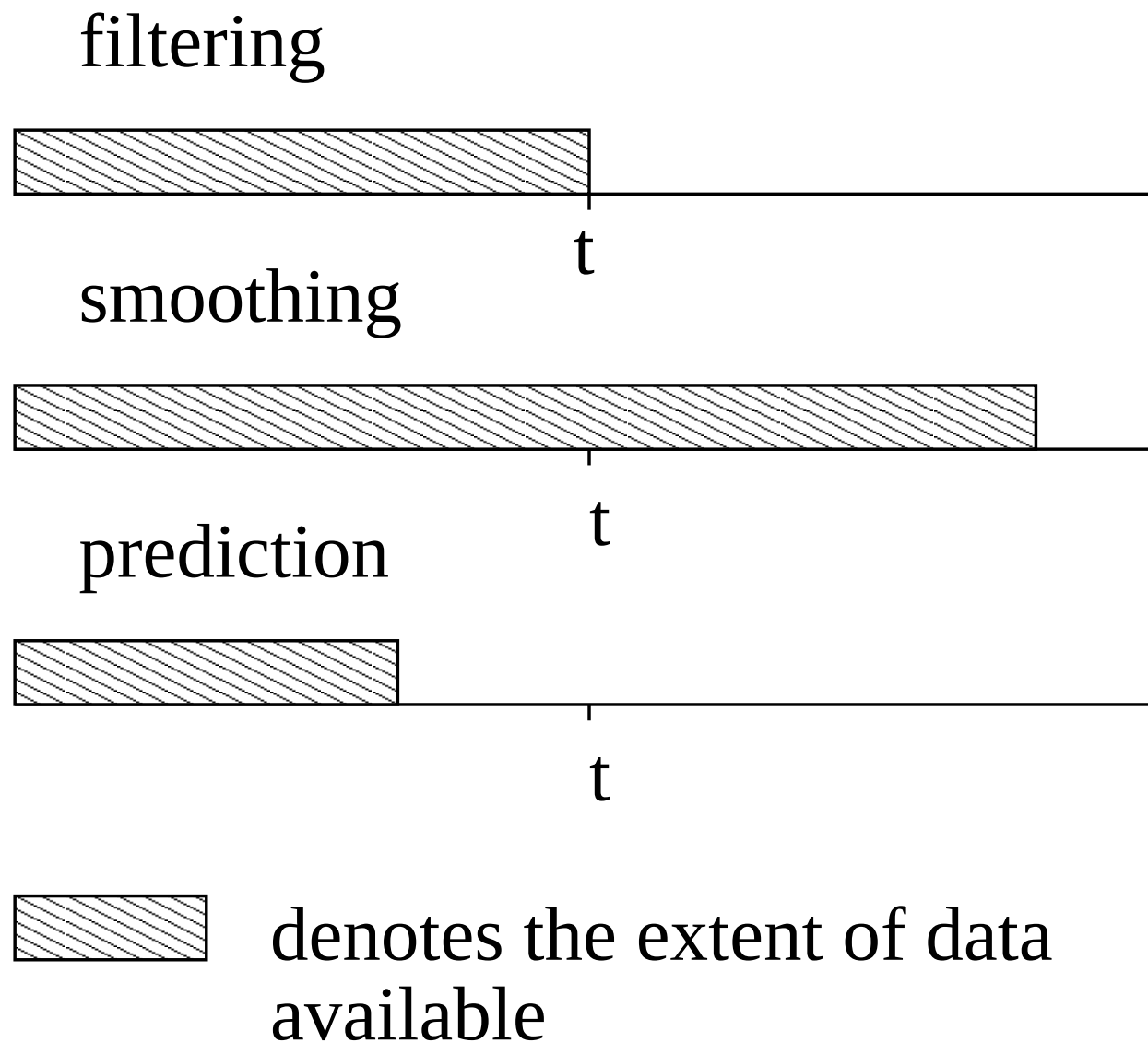
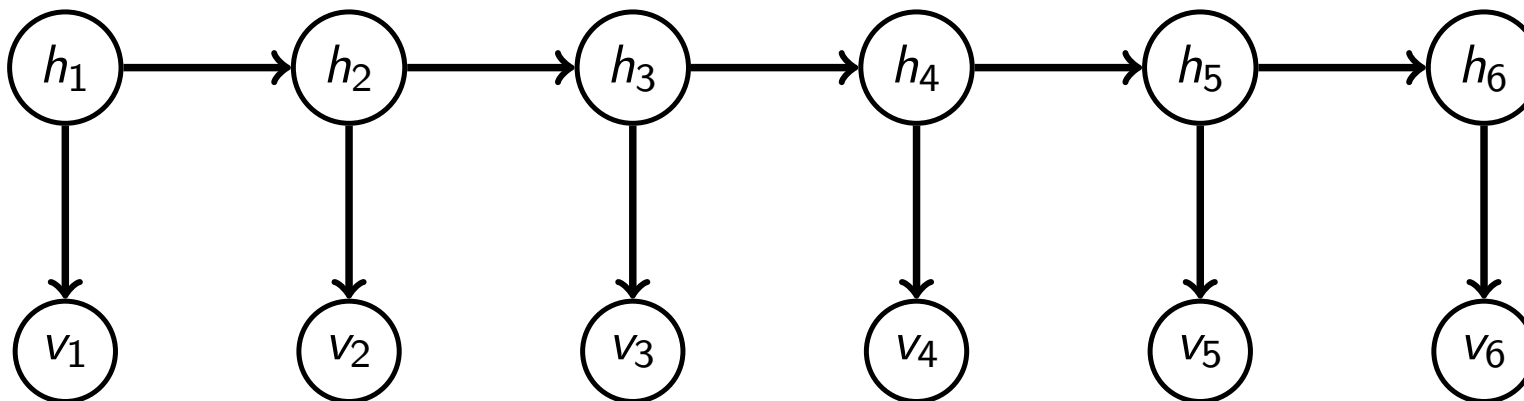


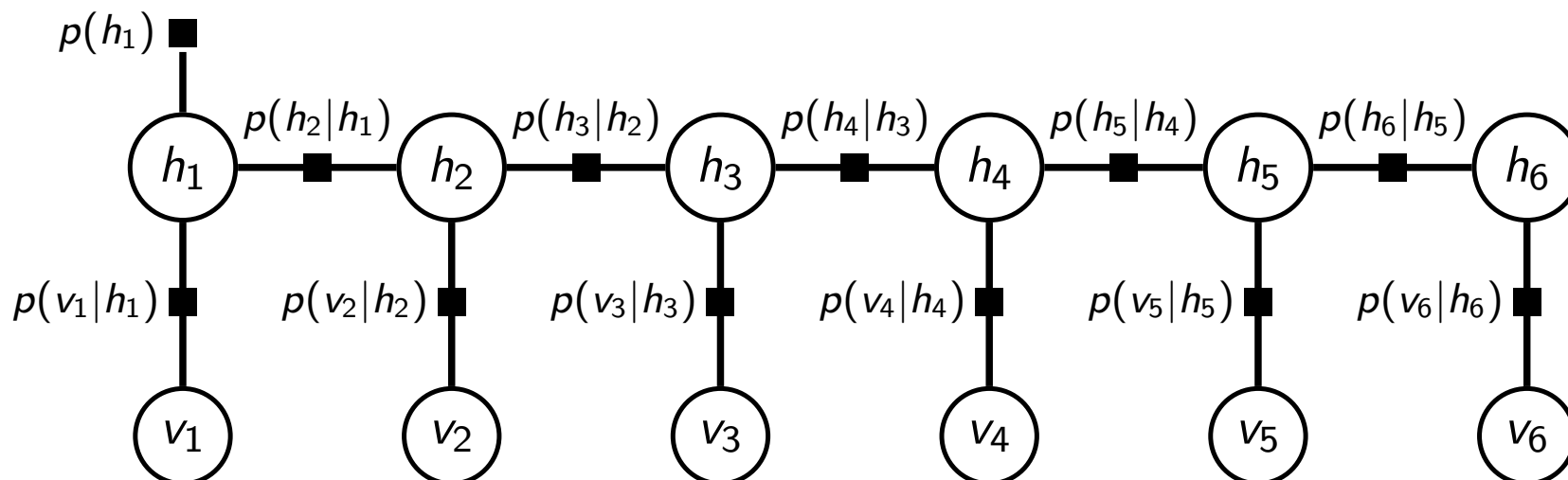
Figure based on Fig. 1.0-1 of Gelb et al (1974)

Factor graph for hidden Markov model

DAG:



Factor graph:

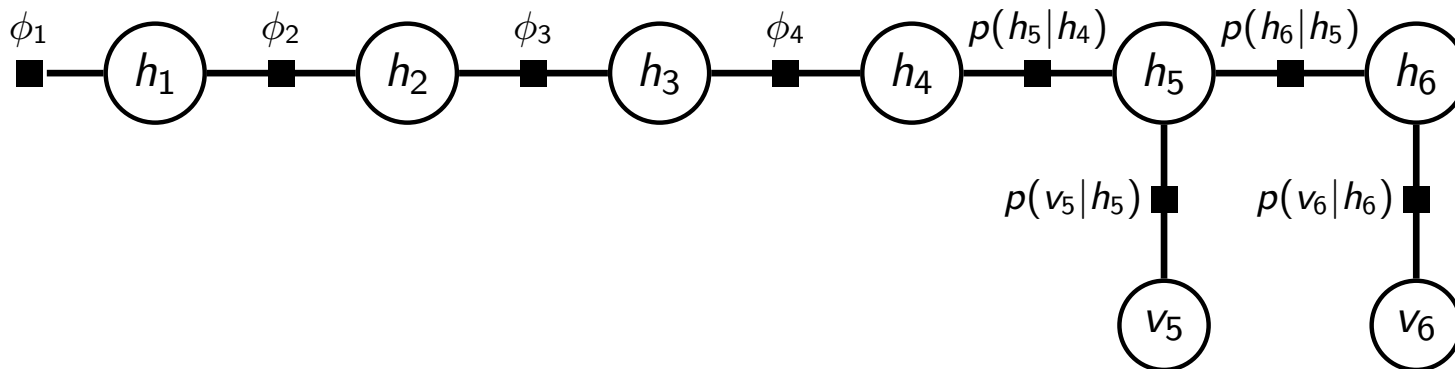


Filtering $p(h_t|v_{1:t})$: factor graph

- ▶ When computing $p(h_t|v_{1:t})$, the $v_{1:t} = (v_1, \dots, v_t)$ are assumed known and are kept fixed (e.g. $t = 4$)
- ▶ For $s = 1, \dots, t$, the factors $p(v_s|h_s)$ depend only on h_s . Combine them with $p(h_s|h_{s-1})$ and form new factors ϕ_s

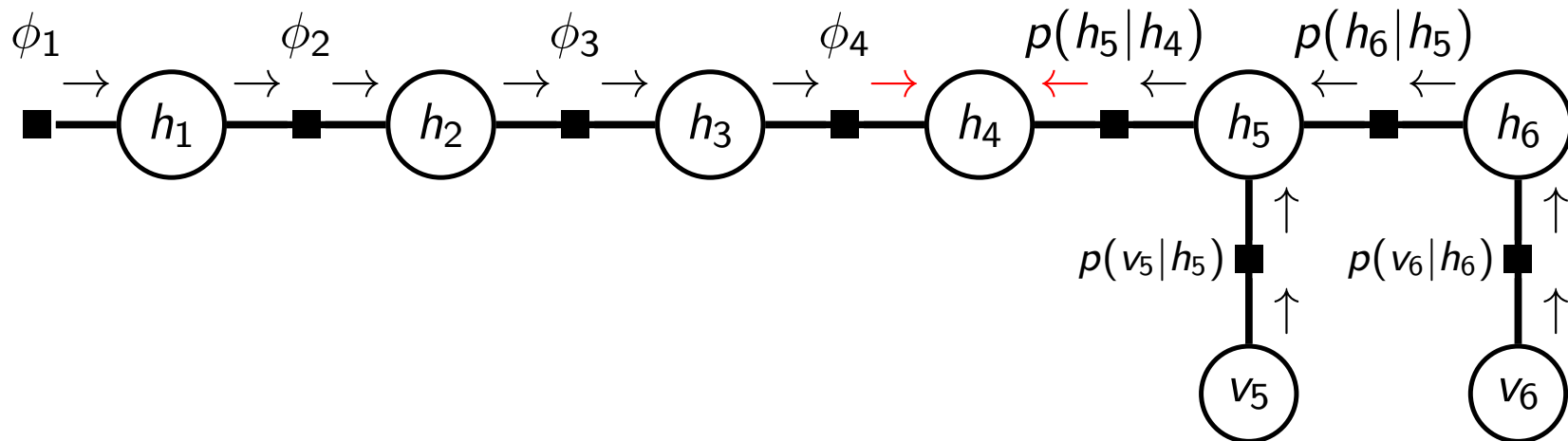
$$\phi_1(h_1) = p(v_1|h_1)p(h_1), \quad \phi_s(h_{s-1}, h_s) = p(v_s|h_s)p(h_s|h_{s-1})$$

- ▶ Factor graph



Filtering $p(h_t|v_{1:t})$: messages

Messages needed to compute $p(h_4|v_{1:4})$: ($t = 4$)



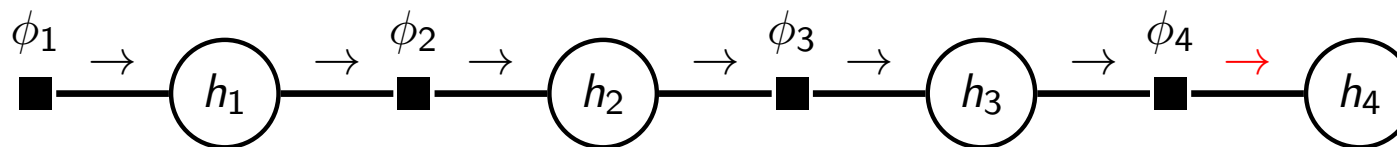
There is a simplification:

- ▶ The message from $p(h_5|h_4)$ to h_4 equals 1!
- ▶ Follows from message passing starting at leaves v_5 and v_6 since the factors $p(.|.)$ are conditionals and sum to one, e.g.

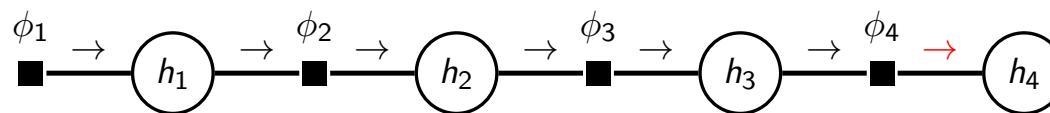
$$\sum_{v_6} p(v_6|h_6) = 1 \quad \sum_{h_6} p(h_6|h_5) = 1$$

Filtering $p(h_t | v_{1:t})$: reduce to inference on chain

- ▶ A message is an effective factor obtained by summing out all variables downstream from where the message is coming from.
- ▶ This means that we can replace the factor sub-graph to the right of the last observed variable v_t and latent h_t (here v_4 and h_4) with the effective factor.
- ▶ Effective factor is 1, so that we can just remove the sub-graph.
- ▶ Also can be seen by “marginalising out” the unobserved future
- ▶ Reduces problem to message passing on a chain.



Filtering $p(h_t|v_{1:t})$: message passing on the chain



- ▶ Initialisation: $\mu_{\phi_1 \rightarrow h_1}(h_1) = \phi_1(h_1)$
- ▶ Variable node h_1 copies the message:

$$\mu_{h_1 \rightarrow \phi_2}(h_1) = \mu_{\phi_1 \rightarrow h_1}(h_1)$$

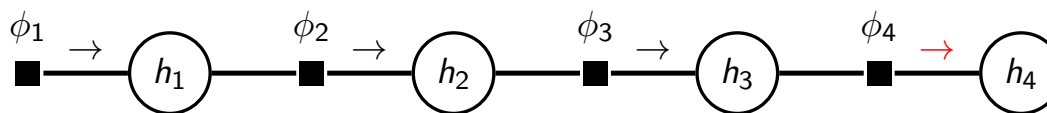
- ▶ Same for other variable nodes. Let us write the algorithm in terms of $\mu_{\phi_i \rightarrow h_i}(h_i)$ messages only.
- ▶ Message from ϕ_2 to h_2 :

$$\mu_{\phi_2 \rightarrow h_2}(h_2) = \sum_{h_1} \phi_2(h_1, h_2) \mu_{\phi_1 \rightarrow h_1}(h_1)$$

- ▶ Message from ϕ_s to h_s , for $s = 2, \dots, t$:

$$\mu_{\phi_s \rightarrow h_s}(h_s) = \sum_{h_{s-1}} \phi_s(h_{s-1}, h_s) \mu_{\phi_{s-1} \rightarrow h_{s-1}}(h_{s-1})$$

Filtering $p(h_t|v_{1:t})$: message passing on the chain



- ▶ The messages $\mu_{\phi_s \rightarrow h_s}(h_s)$ are traditionally denoted by $\alpha(h_s)$.
- ▶ Message passing for filtering becomes:
 - ▶ Init: $\alpha(h_1) = \phi_1(h_1) = p(v_1|h_1)p(h_1)$
 - ▶ Update rule for $s = 2, \dots, t$:

$$\begin{aligned}\alpha(h_s) &= \sum_{h_{s-1}} \phi_s(h_{s-1}, h_s) \alpha(h_{s-1}) \\ &= p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1}) \alpha(h_{s-1})\end{aligned}$$

- ▶ Algorithm known as “alpha-recursion”.
- ▶ Desired probability:

$$p(h_t|v_{1:t}) = \frac{1}{Z_t} \alpha(h_t) \qquad Z_t = \sum_{h_t} \alpha(h_t)$$

Filtering $p(h_t|v_{1:t})$: likelihood

- ▶ Joint model for $h_{1:t}$ and $v_{1:t}$

$$p(h_{1:t}, v_{1:t}) = p(v_1|h_1)p(h_1) \prod_{i=2}^t p(v_i|h_i)p(h_i|h_{i-1})$$

- ▶ We built the factor graph for the conditional $p(h_{1:t}|v_{1:t})$

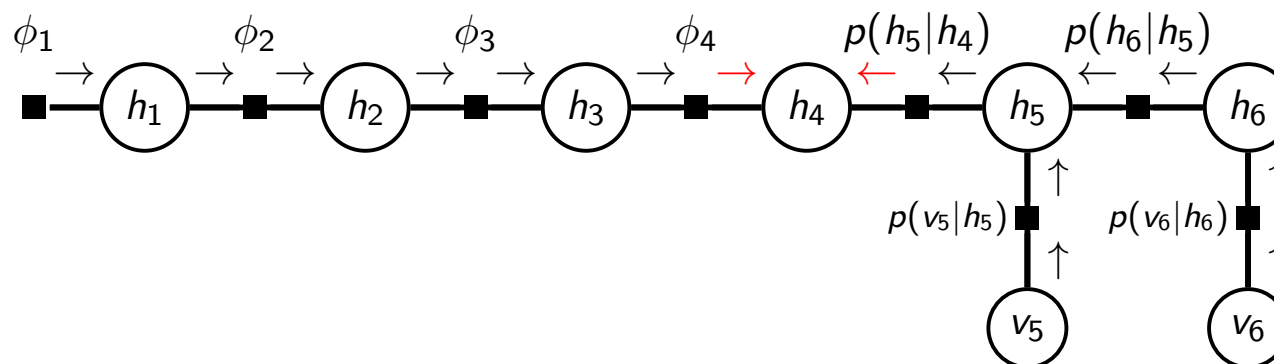
$$p(h_{1:t}|v_{1:t}) \propto p(v_1|h_1)p(h_1) \prod_{i=2}^t p(v_i|h_i)p(h_i|h_{i-1})$$

- ▶ Normalising constant for the equation above is the likelihood/marginal $p(v_{1:t})$
- ▶ Since $p(h_t|v_{1:t})$ has the same normalising constant, we have:

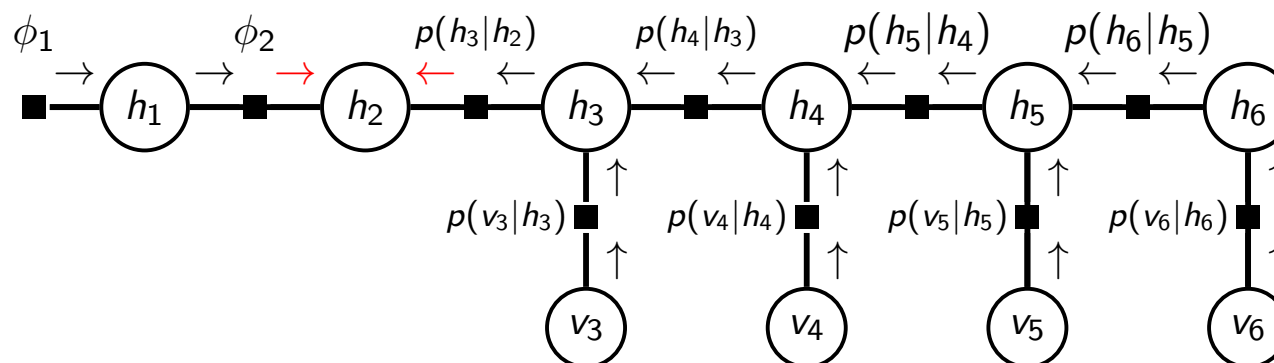
$$Z_t = \sum_{h_t} \alpha(h_t) = p(v_{1:t})$$

Filtering $p(h_t|v_{1:t})$: interpretation

- We have seen that $p(h_t|v_{1:t}) \propto \alpha(h_t)$.



- Consider $p(h_s|v_{1:s})$ with $s < t$ (e.g. $s = 2$ and $t = 4$)



- Messages to the left of h_s are the same as for $p(h_t|v_{1:t})$.
- Messages to the right of h_s are all equal to one.

Filtering $p(h_t|v_{1:t})$: interpretation

- ▶ This means that the intermediate $\alpha(h_s)$ that we compute when computing $p(h_t|v_{1:t})$ are unnormalised posteriors themselves:

$$\alpha(h_s) \propto p(h_s|v_{1:s})$$

Note that we condition on $v_{1:s}$ and not $v_{1:t}$.

- ▶ Moreover $p(v_{1:s}) = \sum_{h(s)} \alpha(h_s)$.
- ▶ Hence, the alpha-recursion gives us posteriors $p(h_s|v_{1:s})$ and likelihoods $p(v_{1:s})$ for $s = 1, \dots, t$.

Filtering $p(h_t|v_{1:t})$: interpretation

- The update rule is a prediction-correction algorithm:

$$\begin{aligned}\alpha(h_s) &= p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1})\alpha(h_{s-1}) \\ &\propto p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1})p(h_{s-1}|v_{1:s-1}) \\ &\stackrel{(\text{Markov prop})}{=} p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1}, v_{1:s-1})p(h_{s-1}|v_{1:s-1}) \\ &\stackrel{(\text{product rule})}{=} p(v_s|h_s) \sum_{h_{s-1}} p(h_s, h_{s-1}|v_{1:s-1}) \\ &\stackrel{(\text{sum rule})}{=} \underbrace{p(v_s|h_s)}_{\text{correction}} \underbrace{p(h_s|v_{1:s-1})}_{\text{prediction}}\end{aligned}$$

- The correction term updates the predictive distribution $p(h_s|v_{1:s-1})$ to include the new data v_s .

Filtering $p(h_t|v_{1:t})$: summary

- ▶ Conditioning reduces the factor graph for the HMM to a chain.
- ▶ Message passing for filtering:
 - ▶ Init: $\alpha(h_1) = p(v_1|h_1)p(h_1)$
 - ▶ Update rule for $s = 2, \dots, t$:

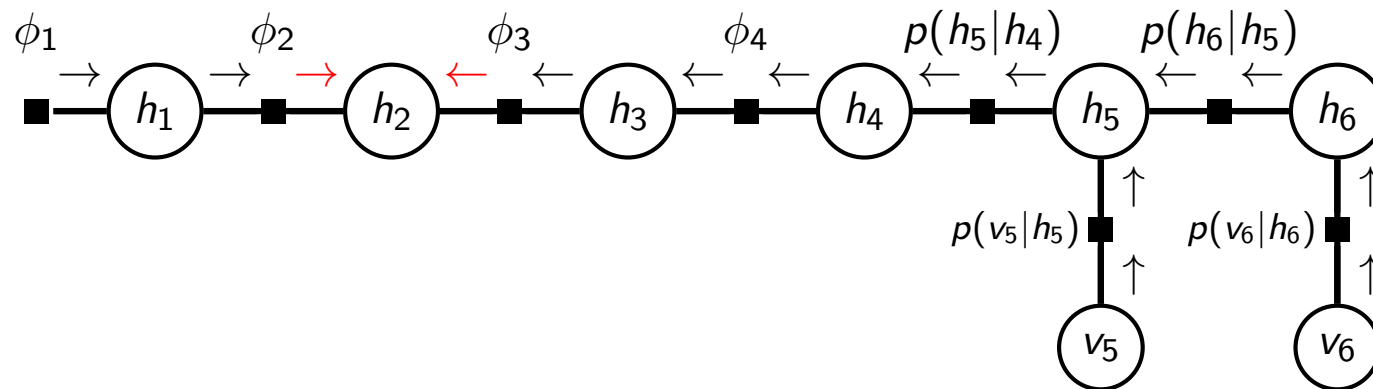
$$\alpha(h_s) = p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1})\alpha(h_{s-1})$$

which involves the predictive distribution of h_s given $v_{1:s-1}$ and a correction using the new datum v_s .

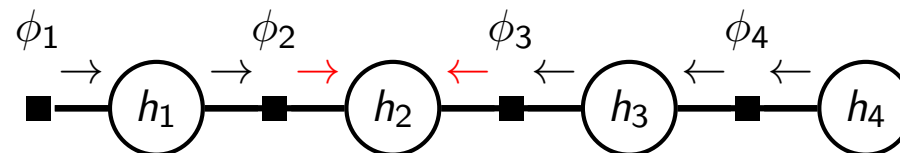
- ▶ $p(h_s|v_{1:s}) \propto \alpha(h_s)$ and $\sum_{h_s} \alpha(h_s) = p(v_{1:s})$, for $s = 1, \dots, t$

Smoothing $p(h_t|v_{1:u})$, $t < u$: reduce to inference on chain

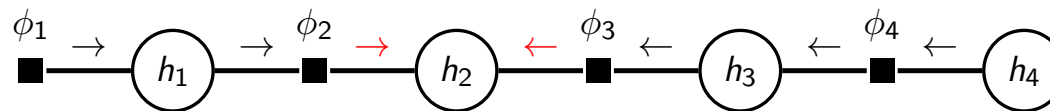
- ▶ Unlike in filtering where we predict h_t from data up to time t , in smoothing we have observations from later time points.
- ▶ Messages needed to compute $p(h_t|v_{1:u})$ (e.g. $t = 2$, $u = 4$)



- ▶ As in filtering, we can simplify to a chain



Smoothing $p(h_t|v_{1:u})$, $t < u$: message passing on chain

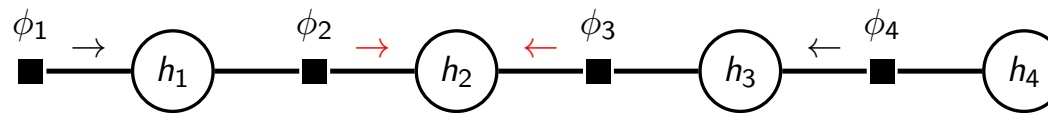


- ▶ Messages $\rightarrow$ from factor leaf ϕ_1 to h_t same as in filtering.
- ▶ Messages $\leftarrow$ from variable leaf h_u to h_t via message passing.
- ▶ Init: $\mu_{h_u \rightarrow \phi_u}(h_u) = 1$
- ▶ Next message $\mu_{\phi_u \rightarrow h_{u-1}}(h_{u-1}) = \sum_{h_u} \phi_u(h_{u-1}, h_u)$
- ▶ Variable nodes just copy the incoming message. Write the algorithm in terms of $\beta(h_s) = \mu_{\phi_{s+1} \rightarrow h_s}(h_s)$ only:

$$\begin{aligned}\beta(h_{s-1}) &= \sum_{h_s} \phi_s(h_{s-1}, h_s) \beta(h_s) \\ &= \sum_{h_s} p(v_s | h_s) p(h_s | h_{s-1}) \beta(h_s)\end{aligned}$$

- ▶ Gives “alpha-beta recursion” for smoothing.

Smoothing $p(h_t|v_{1:u})$, $t < u$: message passing on chain



► $\rightarrow$ Forwards via alpha-recursion

► Init: $\alpha(h_1) = p(v_1|h_1)p(h_1)$

► Update rule for $s = 2, \dots, t$:

$$\alpha(h_s) = p(v_s|h_s) \sum_{h_{s-1}} p(h_s|h_{s-1})\alpha(h_{s-1})$$

► $\leftarrow$ Backwards via beta-recursion

► Init: $\beta(h_u) = 1$

► Update rule for $s = u, \dots, t + 1$:

$$\beta(h_{s-1}) = \sum_{h_s} p(v_s|h_s)p(h_s|h_{s-1})\beta(h_s)$$

► Desired probability:

$$p(h_t|v_{1:u}) = \frac{1}{Z_t^u} \alpha(h_t)\beta(h_t) \quad Z_t^u = \sum_{h_t} \alpha(h_t)\beta(h_t)$$

Doing more with the $\alpha(h_s), \beta(h_s)$

- ▶ Due to link to message passing: Knowing all $\alpha(h_s), \beta(h_s) \implies$ knowing all marginals *and* all joints of neighbouring latents given the observed data, which will be needed when estimating the parameters of HMMs (see later).
- ▶ We can use the $\alpha(h_s)$ for predictions (see exercises).
- ▶ We can use the $\alpha(h_s)$ for sampling posterior trajectories, i.e. to sample from $p(h_1, \dots, h_t | v_1, \dots, v_t)$ (see exercises).
- ▶ Algorithms extend to the case of continuous random variables: replace sums with integrals.

Program recap

1. Markov models

- Markov chains
- Transition distribution
- Hidden Markov models (HMMs)
- Emission distribution
- Important instances of HMMs

2. Inference by message passing

- Inference: filtering, prediction, smoothing, Viterbi
- Filtering: Sum-product message passing yields the α -recursion
- Smoothing: Sum-product message passing yields the α - β recursion