LTAT.02.004 Machine Learning II

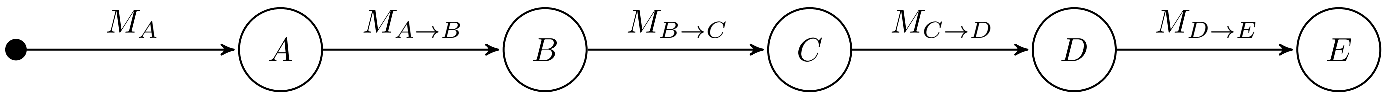
Sequence models

Sven Laur
University of Tartu

Motivating examples

Supervised learning

Higher-order Markov chains



Time-series models

- We assume that x_{i+1} depends only on the values of $x_i, \ldots, x_{i-\ell}$
- A linear model assumes $x_{i+1} = w_0 + w_1 x_i + \cdots + w_{\ell+1} x_{i-\ell} + \varepsilon_i$.
- All error terms ε_i are assumed to be independent.
- All error terms ε_i are drawn from a normal distribution $\mathcal{N}(0, \sigma)$.

Linear time-series model

- ▷ Fix a set of initial inputs $x_{-\ell}, \ldots, x_0 \in \mathbb{R}$. Denote them by $\boldsymbol{x}_\circ$.
- ▷ Think of $x_1, x_2, \ldots, x_n$ as observations. Denote them by $\boldsymbol{x}$.
- ▷ A probabilistic model for state transitions is defined as follows

$$x_{i+1} = \underbrace{w_0 + w_1 x_i + \ldots w_{\ell+1} x_{i-\ell}}_{\hat{x}_{i+1}} + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma)$$

- ▷ Consequently

$$p[\boldsymbol{x}|\boldsymbol{x}_\circ, \boldsymbol{w}, \sigma] = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(x_i - \hat{x}_i)^2}{2\sigma^2}\right)$$

Maximum likelihood estimate

As usual we can find $\boldsymbol{w} \in \mathbb{R}^{\ell+2}$ and $\sigma \in \mathbb{R}$ that maximise the log-likelihood

$$\log p[\boldsymbol{x}|\boldsymbol{x}_\circ, \boldsymbol{\beta}, \sigma] = const - n \log \sigma - \sum_{i=1}^{n} \frac{(x_i - \hat{x}_i)^2}{2\sigma^2}$$

and thus we can find $\boldsymbol{w}$ by minimising

$$\mathsf{MSE} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - w_0 - w_1 x_{i-1} - \ldots - w_{\ell+1} x_{i-1-\ell})^2 \enspace .$$

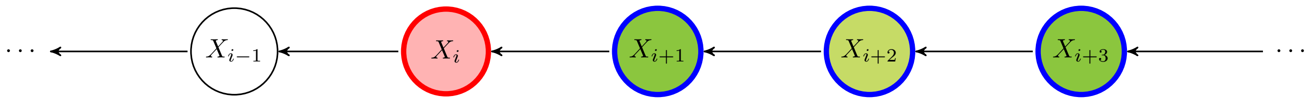
The latter is the standard multivariate linear regression setup. The variance of the model σ^2 can be found by the same formula as for linear regression.

Prediction intervals for time-series

After we have fitted the linear regrssion model to timeseries data we might want to compute prediction intervals for iterative stepwise predictions.

- ▷ Let $\boldsymbol{x}_0$ be the known initial state and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ iterative predictions.
- ▷ We need priors $\pi[\boldsymbol{x}_i] = p[\boldsymbol{x}_i|\boldsymbol{x}_0]$ to compute confidence intervals.
- ▷ It turns out that all priors $p[\boldsymbol{x}_i]$ are normal distributions.
- ▷ Moment matching allows us to learn the parameters of the distributions.

Smoothing and reverse Markov chain



Sometimes we have to interpolate observations in the time series. This can be stated as amoothing task where we know $\boldsymbol{x}_0$ and $\boldsymbol{x}_n$.

- We need likelihoods $\lambda[\boldsymbol{x}_i] = p[\boldsymbol{x}_n|\boldsymbol{x}_i]$ for the smoothing.
- Likelihood propagation formula is analogous to the prior propagation.
- We can define a reverse Markov chain such that the prior $\pi^*[\boldsymbol{x}_i] \propto \lambda[\boldsymbol{x}_i]$.
- The resulting chain has reversed dynamics.
- It turns out that all likelihoods $\lambda[\boldsymbol{x}_i]$ are normal distributions.
- The posterior as product $\pi[\boldsymbol{x}_i] \cdot \lambda[\boldsymbol{x}_i]$ is also a normal distribution.

How to write a good touchscreen keyboard?

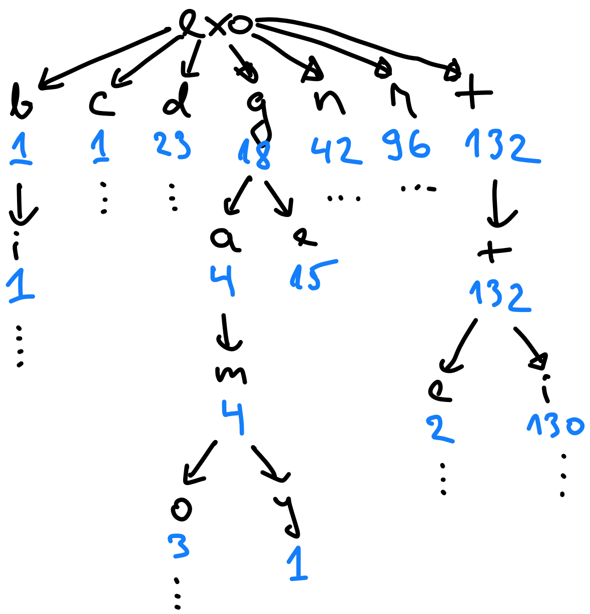
Possibilities

exobiology
exocarp
exodus
exogamous
exogamy
exogenous
exonerate
exoneration
exorbitant
exorbitance
exorcism
exorcist
exorcize
exordium
exoteric
exotic

Likelihoods

exobiology	1
exocarp	1
exodus	23
exogamous	3
exogamy	1
exogenous	15
exonerate	30
exoneration	12
exorbitant	43
exorbitance	5
exorcism	16
exorcist	19
exorcize	12
exordium	1
exoteric	2
exotic	130

Trie of possibilities



Discrete random variables

- ▷ A *random variable* X with possible *outcomes* $x \in \mathrm{supp}(X)$
- ▷ Compact notation for probabilities

$$\Pr\left[x_1\right] := \Pr\left[\xi \leftarrow X_1 : \xi = x_1\right]$$

$$\Pr\left[x_1 \wedge x_2\right] := \Pr\left[\xi_1 \leftarrow X_1, \xi_2 \leftarrow X_2 : \xi_1 = x_1 \wedge \xi_2 = x_2\right]$$

- ▷ Bayes formula

$$\Pr\left[a|b\right] = \frac{\Pr\left[a \wedge b\right]}{\Pr\left[b\right]} = \frac{\Pr\left[b|a\right]\Pr\left[a\right]}{\Pr\left[b\right]}$$

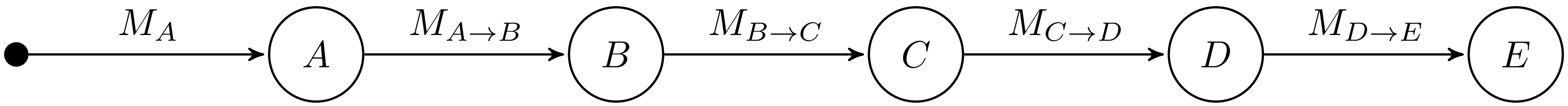
- ▷ Independence of random variables $X_1 \ldots X_m \perp Y_1, \ldots Y_n$:

$$\Pr\left[x_1 \wedge \ldots \wedge x_m \wedge y_1 \wedge \ldots \wedge y_n\right] = \Pr\left[x_1 \wedge \ldots \wedge x_m\right] \cdot \Pr\left[y_1 \wedge \ldots \wedge y_n\right]$$

- ▷ Marginalisation over variables $Y_1, \ldots, Y_n$:

$$\Pr\left[x_1 \wedge \ldots \wedge x_m\right] = \sum_{y_1,\ldots,y_n} \Pr\left[x_1 \wedge \ldots \wedge x_m \wedge y_1 \wedge \ldots \wedge y_n\right]$$

Markov chain



Definition. Let $X_1, X_2, \ldots$ be correlated random variables such that the probability of the observation x_{i+1} depends only on the observation x_i. Then the entire process is known as Markov chain.

Parametrisation. Markov chain is determined by specifying

- state spaces $\mathcal{S}_1 \ldots, \mathcal{S}_n$
- initial probabilities $\Pr\left[x_1\right]$ given as vectors
- state transition probabilities $\Pr\left[x_{i+1}|x_i\right]$ given as matrices

What questions can we ask?

Sampling: What are typical outcomes of the chain?
$\triangleright$ Synthesis of time-series, textures, sounds, games movements.

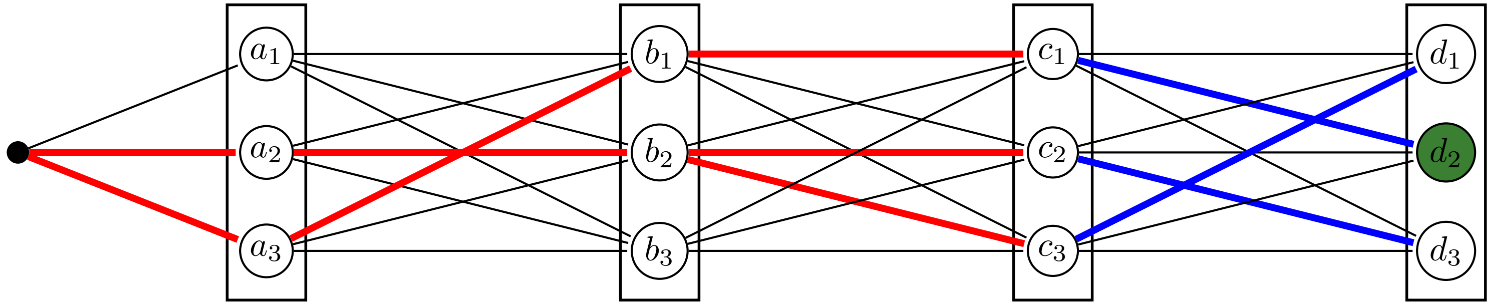
Stationary distribution: What happens if we run the chain infinitely long?
$\triangleright$ Getting samples from an unnormalised posterior, optimisation tasks.

Likelihood estimation: What is a probability of an observation $x_1, \ldots, x_n$?
$\triangleright$ Reasoning about probabilities and clustering sequences.

Decoding: What is the most probable outcome $x_1, \ldots, x_n$?
$\triangleright$ Imputing missing values. Rudimentary logical reasoning.

Parameter estimation: What is are the model parameters?
$\triangleright$ Machine learning – finding parameters based on observations.

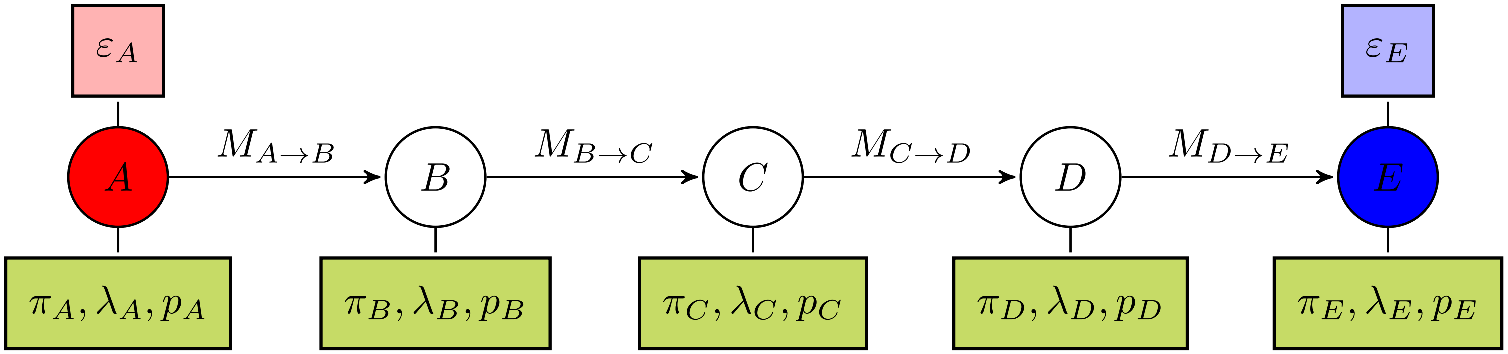
Posterior maximisation in a chain



Inference goal. Given evidence at the ends of the chain find the sequence of states $\boldsymbol{x}$ that maximise the posterior probability $\Pr[\boldsymbol{x}|\text{evidence}]$.

- The log-posterior $\log\Pr[\boldsymbol{x}|\text{evidence}]$ decomposes into a sum.
- We must find a sequence with maximal weight.
- The task can be split into subtask as all subpaths of the path with maximal weight must have maximal weight.
- The corresponding iterative algorithm is known as Viterbi algorithm.

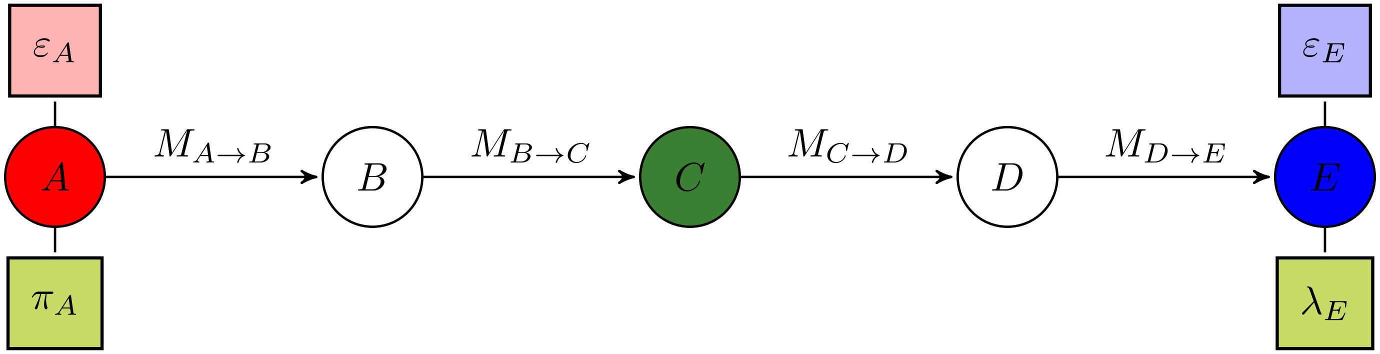
Belief propagation in a chain



Inference goal. Given evidence at the ends of the chain find marginal posterior probabilities for each node in the chain.

- Evidence ε_V is an observational data associated with the node V.
- Upstream evidence$^+$ is the evidence at the beginning of chain.
- Downstream evidence$^-$ is the evidence at the end of chain.
- Attributes π_V, λ_V, p_V are needed to compute marginal distributions.

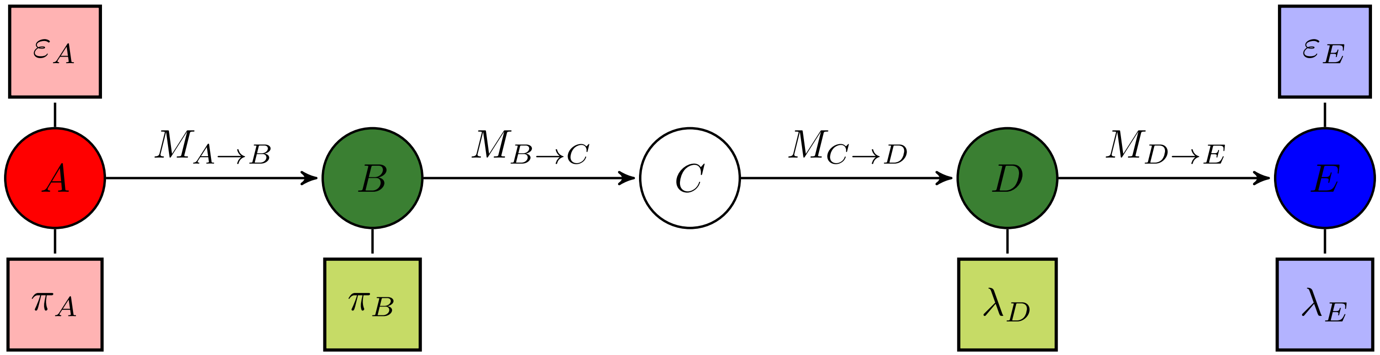
Initialisation



- ▷ Direct evidence ε_V determines the value of V.
- ▷ Indirect evidence ε_V determines the value distribution for V.
- ▷ We can assign the prior for the first and likelihood for the last node

$$\pi_A(a) = \Pr\left[A = a|\text{evidence}^{+}\right] = \Pr\left[A = a|\varepsilon_A\right]$$
$$\lambda_E(e) = \Pr\left[\text{evidence}^{-}|E = e\right] = \Pr\left[\varepsilon_E|E = e\right]$$

Belief propagation



Inference goal

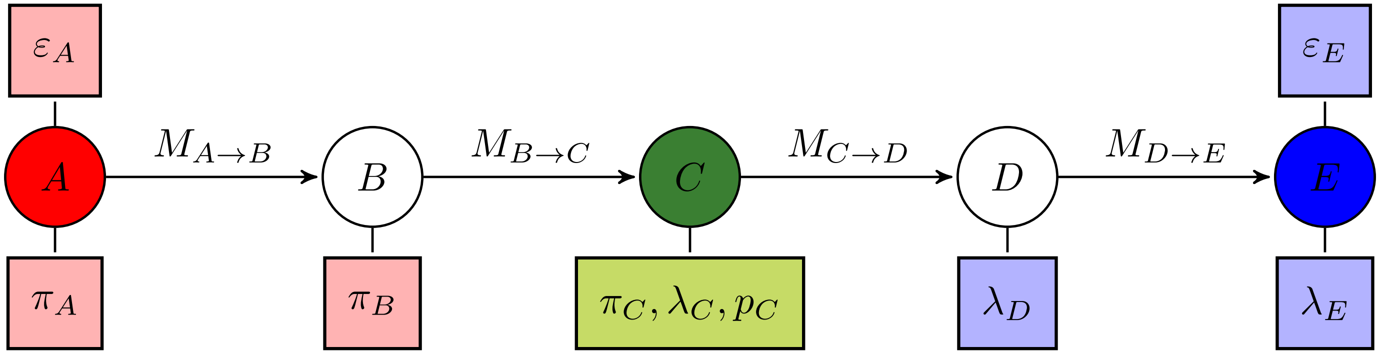
$$\pi_B(b) = \Pr\left[b|\text{evidence}^+\right]$$

$$\lambda_D(d) = \Pr\left[\text{evidence}^-|d\right]$$

Iterative propagation rules

- ▷ Marginalisation gives an update rule $\lambda_D = M_{D\to E}\lambda_E$.
- ▷ Marginalisation gives an update rule $\pi_B \propto \pi_A M_{A\to B}$.

Belief propagation



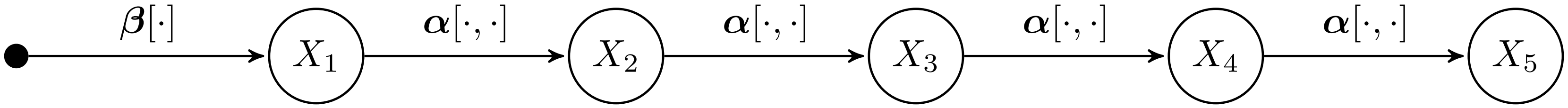
Inference goal

$$p_C(c) = \Pr\left[c|\text{evidence}^+, \text{evidence}^-\right]$$

Iterative update rule

- Bayes formula gives $p_C \propto \pi_C \otimes \lambda_C$.

Parameter inference for homogenous case


$\boldsymbol{\beta}[\cdot]$
X_1
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_2
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_3
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_4
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_5

For a sequence of observations $\boldsymbol{x} = (x_1, \ldots, x_n)$ the log-likelihood is

$$\begin{aligned}\ell[\boldsymbol{x}] &= \log \underbrace{\Pr\left[x_1\right]}_{\beta[x_1]} + \sum_{i=1}^{n-1} \log \underbrace{\Pr\left[x_{i+1}|x_i\right]}_{\alpha[x_i, x_{i+1}]} \\ &= \log \beta[x_1] + \sum_{u_1, u_2} k(u_1, u_2) \log \alpha[u_1, u_2]\end{aligned}$$

where $k(u_1, u_2)$ is the count of bigrams u_1, u_2 in the sequence $\boldsymbol{x}$.

Posterior decomposition

As a result the log-likelihood of unnormalised posterior decomposes into the sum of independent terms

$$\begin{aligned}\log p[\boldsymbol{\alpha},\boldsymbol{\beta}|\boldsymbol{x}] =& \sum_{u_1} k(u_1)\log\beta[u_1] + \log p(\boldsymbol{\beta})\\ &+ \sum_{u_1,u_2} k(u_1,u_2)\log\alpha[u_1,u_2] + \sum_{u_1}\log p(\boldsymbol{\alpha}[u_1,\cdot])\end{aligned}$$

where

- ▷ $k(u_1)$ is the count u_1 at the beginning of the observed sequences
- ▷ $k(u_1,u_2)$ is the count of bigrams u_1,u_2 in the observed sequences.
- ▷ $p(\boldsymbol{\beta})$ is the prior for an entire vector of initial probabilities
- ▷ $p(\boldsymbol{\alpha}[u_1,\cdot])$ is the prior for the transition probabilities from u_1

Reduction to the dice throwing experiment

Posterior decomposition leads to many independent optimisation tasks

$$\sum_{u_1} k(u_1)\log\beta[u_1] + \log p(\boldsymbol{\beta}) \to \max$$

$$\sum_{u_2} k(u_1,u_2)\log\alpha[u_1,u_2] + \log p(\boldsymbol{\alpha}[u_1,\cdot]) \to \max$$

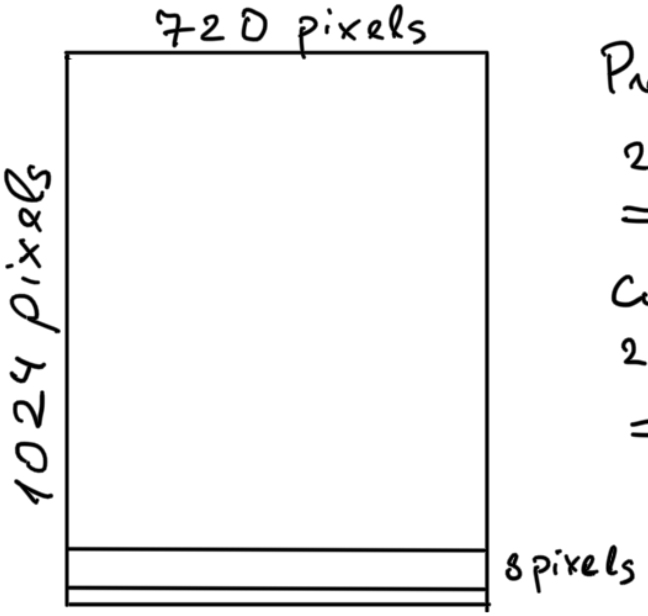
where each of these is equivalent to optimisation of dice throwing posterior. Thus Maximum Aposteriori estimates for parameters are

$$\beta[u_1] = \frac{k(u_1)+c}{k(*)+mc} \qquad\qquad \alpha[u_1,u_2] = \frac{k(u_1,u_2)+c}{k(u_1,*)+mc}$$

where

- ▷ $*$ is a wildcard symbol in the count queries
- ▷ m is the number of states and c is a constant for Laplacian smoothing.

Why discrete Markov chains fail in practice?



Previous state

256 colors × 720 pixels × 8 pixels

$\Rightarrow 256^{720\times 8}$

Current state

256 colors × 720 pixels

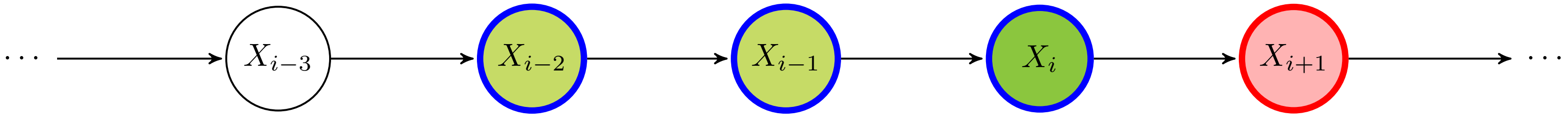
$\Rightarrow 256^{720}$

The number of possible observation is to big already for 8×8 patch:

$$256^{8\times8} \times 256^{8} \times 2^{10} = 2^{8\times8\times8+8\times8+10} = 2^{586}$$

8×9 pathces are needed to estimate probabilities within ± 3 percent points.

Higher-order Markov chains



Time-series models

- We assume that x_{i+1} depends only on the values of $x_i, \ldots, x_{i-\ell}$
- A linear model assumes $x_{i+1} = w_0 + w_1 x_i + \cdots + w_{\ell+1} x_{i-\ell} + \varepsilon_i$.
- All error terms ε_i are assumed to be independent.
- All error terms ε_i are drawn from a normal distribution $\mathcal{N}(0, \sigma)$.

Linear time-series model

▷ Fix a set of initial inputs $x_{-\ell}, \ldots, x_0 \in \mathbb{R}$. Denote them by $\boldsymbol{x}_\circ$.

▷ Think of $x_1, x_2, \ldots, x_n$ as observations. Denote them by $\boldsymbol{x}$.

▷ A probabilistic model for state transitions is defined as follows

$$x_{i+1} = \underbrace{w_0 + w_1 x_i + \ldots w_{\ell+1} x_{i-\ell}}_{\hat{x}_{i+1}} + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma)$$

▷ Consequently

$$p[\boldsymbol{x}|\boldsymbol{x}_\circ, \boldsymbol{w}, \sigma] = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(x_i - \hat{x}_i)^2}{2\sigma^2}\right)$$

Maximum likelihood estimate

As usual we can find $\boldsymbol{w} \in \mathbb{R}^{\ell+2}$ and $\sigma \in \mathbb{R}$ that maximise the log-likelihood

$$\log p[\boldsymbol{x}|\boldsymbol{x}_\circ, \boldsymbol{\beta}, \sigma] = const - n \log \sigma - \sum_{i=1}^{n} \frac{(x_i - \hat{x}_i)^2}{2\sigma^2}$$

and thus we can find $\boldsymbol{w}$ by minimising

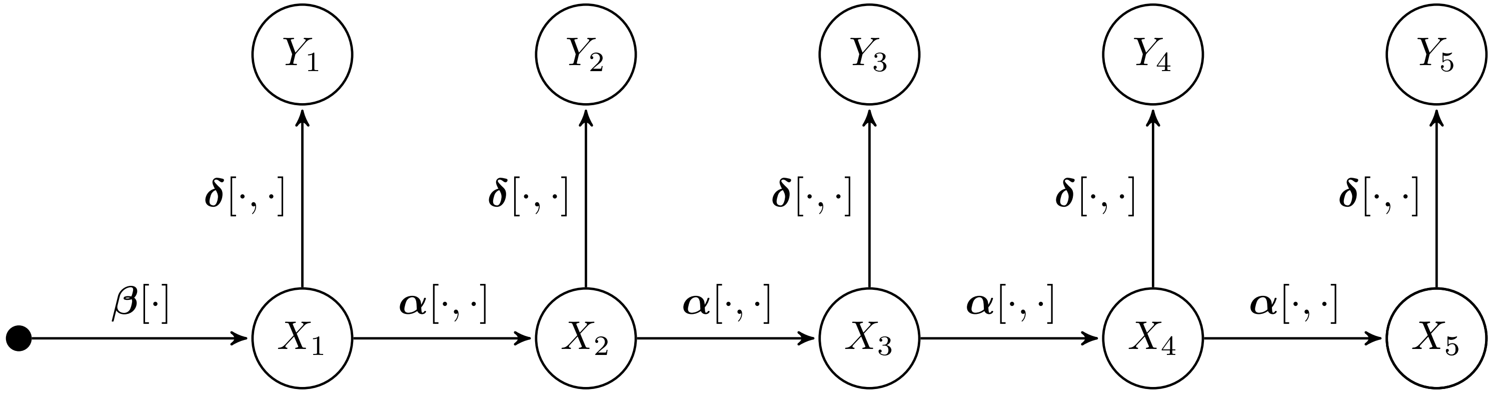
$$\mathsf{MSE} = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - w_0 - w_1 x_{i-1} - \ldots - w_{\ell+1} x_{i-1-\ell})^2 \enspace .$$

The latter is the standard multivariate linear regression setup. The variance of the model σ^2 can be found by the same formula as for linear regression.

Two ways to build continious Markov chains

- ▷ Replace a list of discrete states with continous variable.
  - ◇ We get 8×8 input features and 8 output features.
  - ◇ We need 8 functions of type $f_i : \mathbb{R}^{64} \rightarrow \mathbb{R}$ to fix expectation.
  - ◇ We need 8 functions of type $g_i : \mathbb{R}^{64} \rightarrow \mathbb{R}$ to fix variance.
  - ◇ If we use linear functions then we need $8 \times 65 \times 2$ parameters.
- ▷ Embed discrete states into lower-dimensional feature space.
  - ◇ Ideally, these features are have semantical meaning.
  - ◇ In practice, features are fixed up to affine transformations.
  - ◇ Thus, features do not have clear interpretation.

Hidden Markov Model


Y_1
Y_2
Y_3
Y_4
Y_5
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\beta}[\cdot]$
$\boldsymbol{\alpha}[\cdot,\cdot]$
$\boldsymbol{\alpha}[\cdot,\cdot]$
$\boldsymbol{\alpha}[\cdot,\cdot]$
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_1
X_2
X_3
X_4
X_5

Definition. Let $X_1, X_2, \ldots$ be hidden states that form a Markov chain and let $Y_1, Y_2, \ldots$ be observations that the probability of y_i depends only on the state x_i. Then the entire process is known as Hidden Markov Model.

Common tasks

▷ parameter estimation

▷ filtering, smoothing, prediction

Applications

Modelling and prediction

▷ stock prices

▷ linear control algorithms

Sequence annotation

▷ fraud detection

▷ change detection

▷ functional motifs of DNA sequences

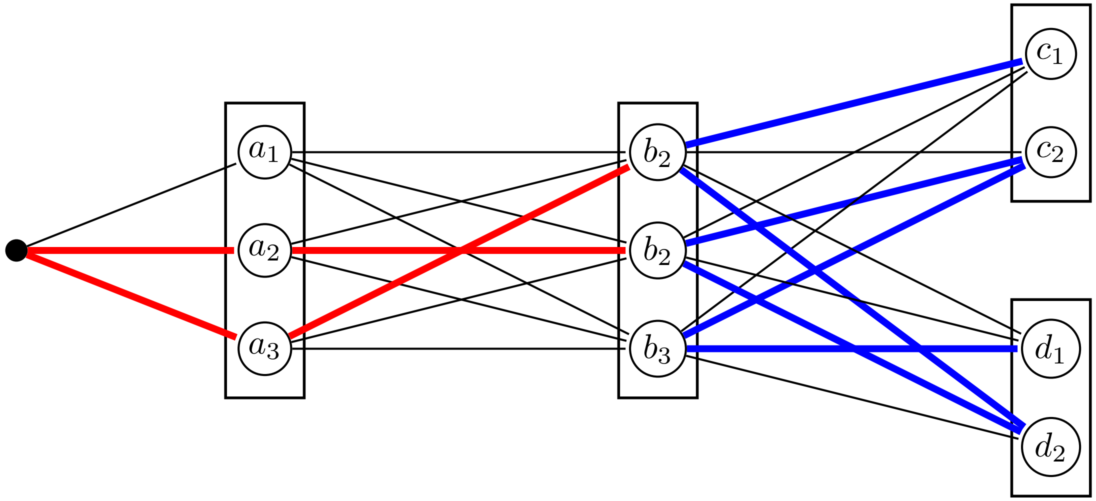
Decoding

▷ speech recognition

▷ communication over a nosy channels

▷ object tracking and data fusion

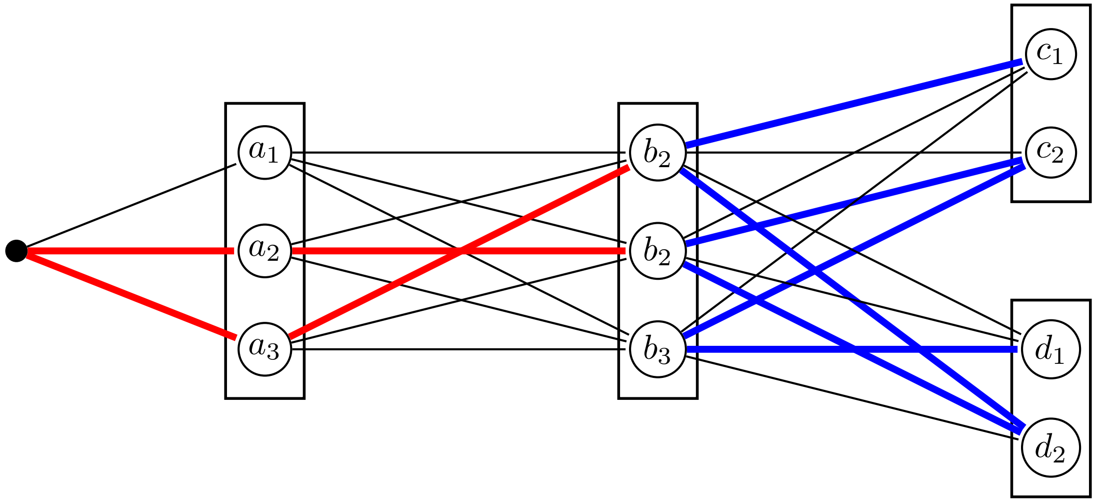
Posterior maximisation in a tree



Inference goal. Given evidence at the ends of the chain find the sequence of states $\boldsymbol{x}$ that maximise the posterior probability $\Pr\left[\boldsymbol{x}|\text{evidence}\right]$.

- ▷ The log-posterior $\log\Pr\left[\boldsymbol{x}|\text{evidence}\right]$ decomposes into a sum.
- ▷ We must find a tree with maximal weight.

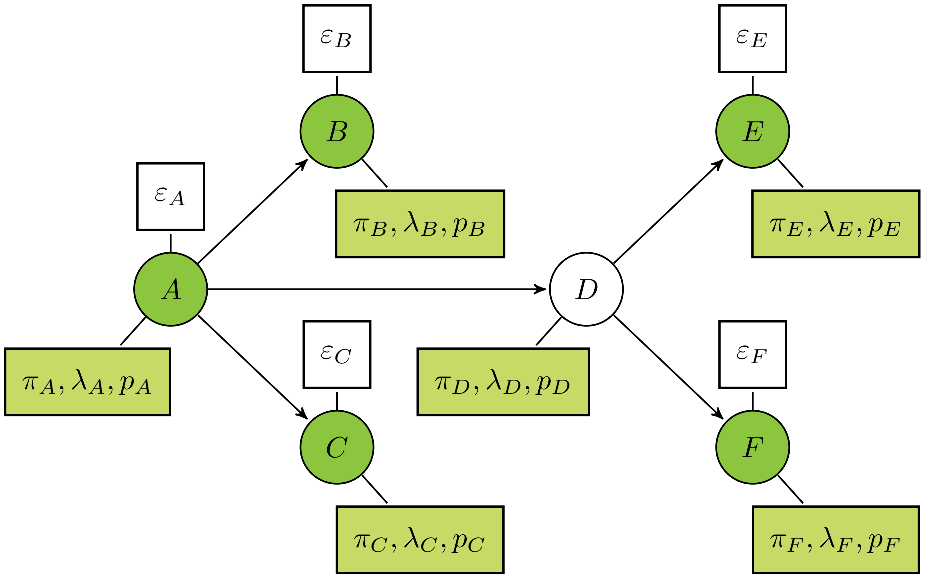
Decomposition into subtasks



All subtrees of the tree with maximal weight must have maximal weight.

- ▷ We can build chains with maximum weight form leafs
- ▷ We can merge subtrees with maximum weight to maximise the weight.
- ▷ The algorithm works from leafs to the root node.
- ▷ The corresponding iterative algorithm is known as Viterbi algorithm.

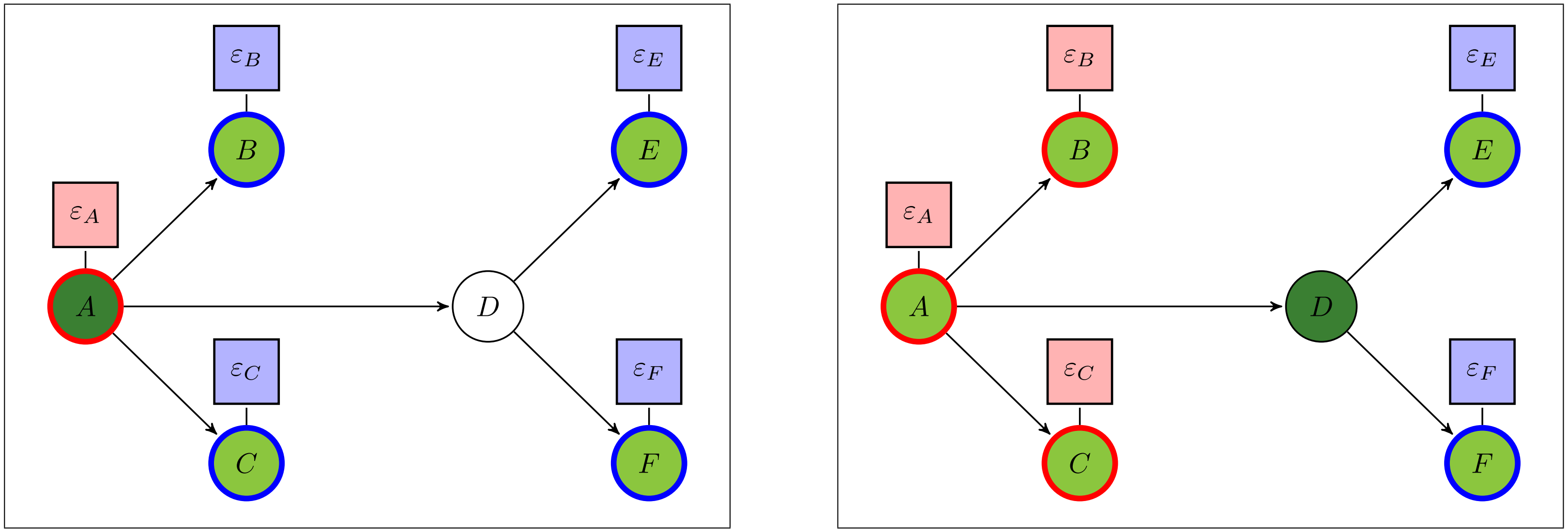
Belief propagation in a tree



Inference goal. Given evidence at the ends of the leafs and the root of tree find marginal posterior probabilities for each node in the tree.

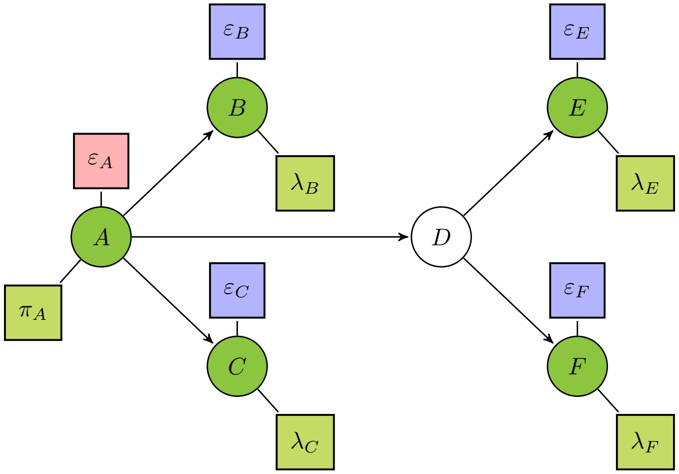
- Evidence ε_V is an observational data associated with the node V.
- Attributes π_V, λ_V, p_V are needed to compute marginal distributions.

Evidence decomposition



- ▷ Evidence decomposes into up- and downstream evidence
- ▷ Downstream evidence$^{-}(V)$ is reachable through child nodes.
- ▷ Upstream evidence$^{+}(V)$ is reachable through the predessesor node.
- ▷ Different nodes have totally different decompositions.

Initialisation



Goal. Assign prior to the root node and likelihood to the leaf nodes.

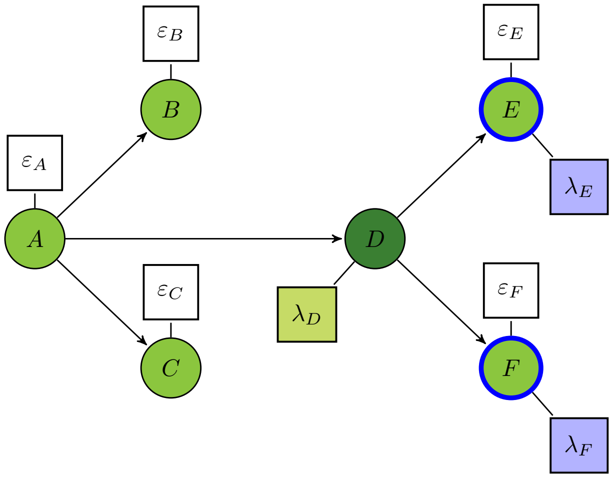
$$\pi_A(a) = \Pr\left[A = a | \text{evidence}^{+}(A)\right] = \Pr\left[A = a | \varepsilon_A\right]$$

$$\lambda_B(b) = \Pr\left[\text{evidence}^{-}(B) | F = f\right] = \Pr\left[\varepsilon_B | B = b\right]$$

$$\ldots$$

$$\lambda_F(f) = \Pr\left[\text{evidence}^{-}(F) | F = f\right] = \Pr\left[\varepsilon_F | F = f\right]$$

Likelihood propagation



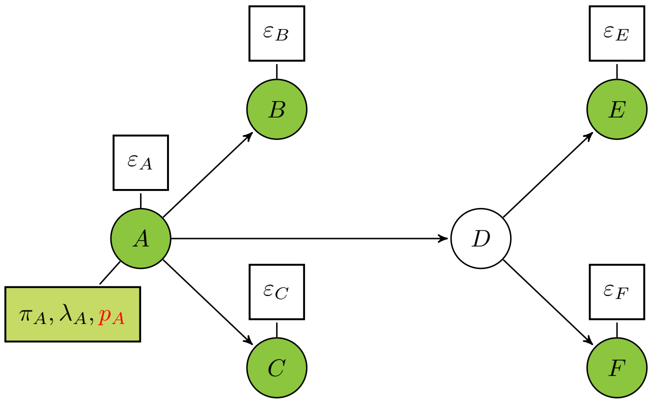
Inference goal

$$\lambda_D(d) = \Pr\left[\text{evidence}^{-}(D)|D = d\right]$$

Iterative propagation rules

- ▷ Independence gives a pooling rule $\lambda_D = \lambda_1 \otimes \lambda_2$
- ▷ Marginalisation gives rules $\lambda_1 = M_{D\to E}\lambda_E$ and $\lambda_2 = M_{D\to F}\lambda_F$.

Posterior propagation



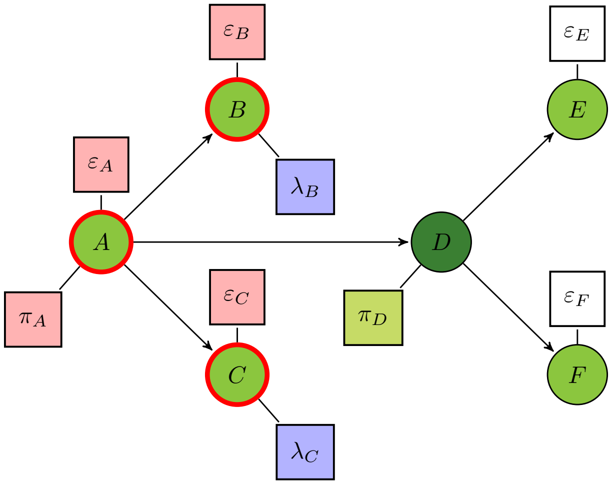
Inference goal

$$p_A(a) = \Pr\left[A = a \middle| \text{evidence}^+(A), \text{evidence}^-(A)\right]$$

Iterative propagation rule

▷ Marginal conditional probability $p_A \propto \pi_A \otimes \lambda_A$

Prior propagation



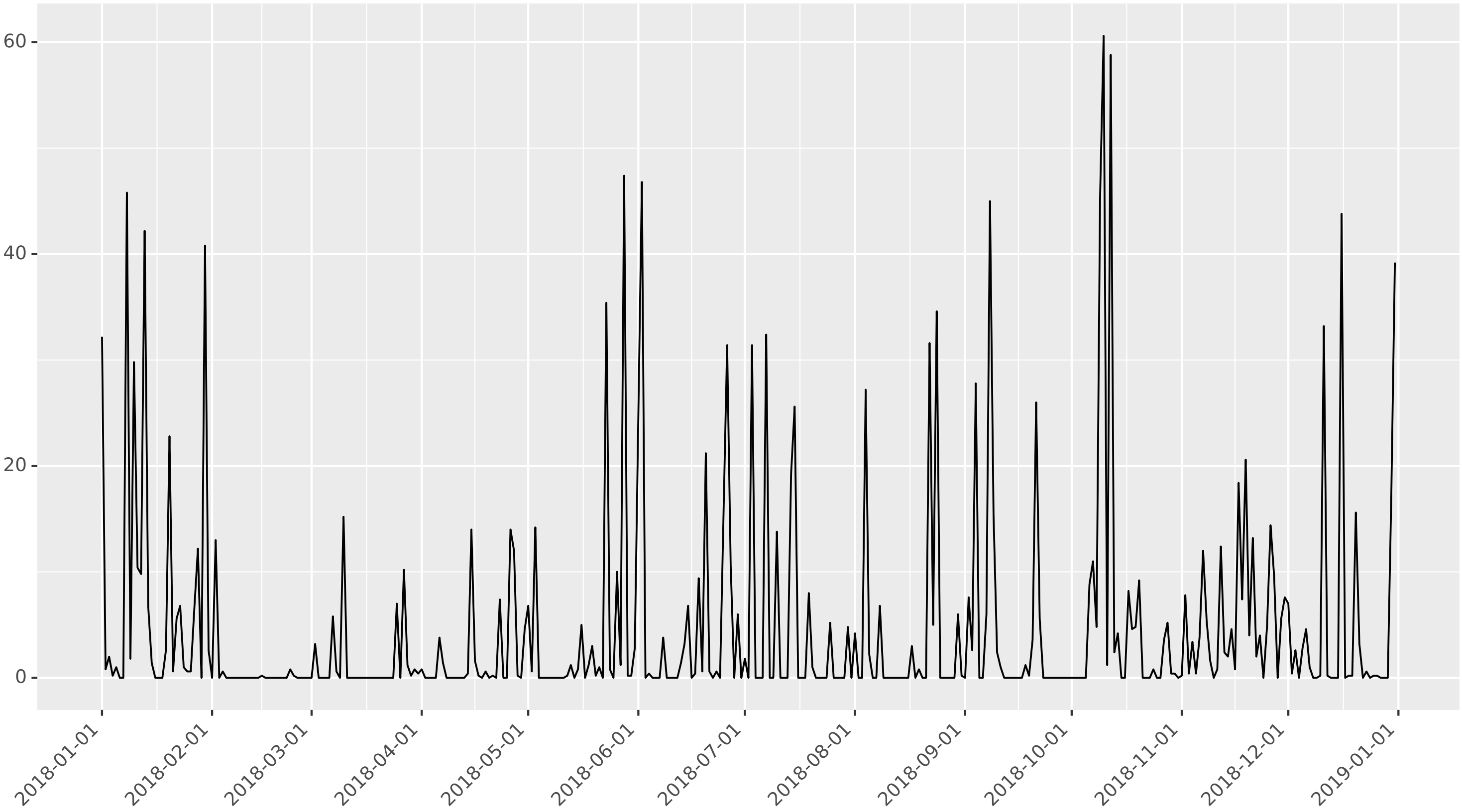
Inference goal

$$\begin{aligned}\pi_D(d) &= \Pr\left[D = d | \text{evidence}^+(D)\right]\\ &= \Pr\left[D = d | \text{evidence}^+(A), \text{evidence}^-(B), \text{evidence}^-(C)\right]\end{aligned}$$

Iterative propagation rule

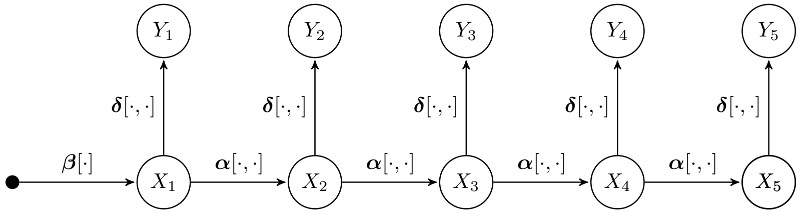
▷ Prior can be computed as $\pi_D \propto \pi_A M_{A\to D} \otimes M_{A\to B}\lambda_B \otimes M_{A\to C}\lambda_C$.

Application on rainfall data



There are two monsoon seasons in Singapore: dry and wet phase.

Modelling with Hidden Markov Model


Y_1
Y_2
Y_3
Y_4
Y_5
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\delta}[\cdot,\cdot]$
$\boldsymbol{\beta}[\cdot]$
X_1
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_2
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_3
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_4
$\boldsymbol{\alpha}[\cdot,\cdot]$
X_5

Markov chain with states $\mathcal{S} = \{0, 1\}$ and parameters

$$\boldsymbol{\beta} = (0.5, 0.5)$$

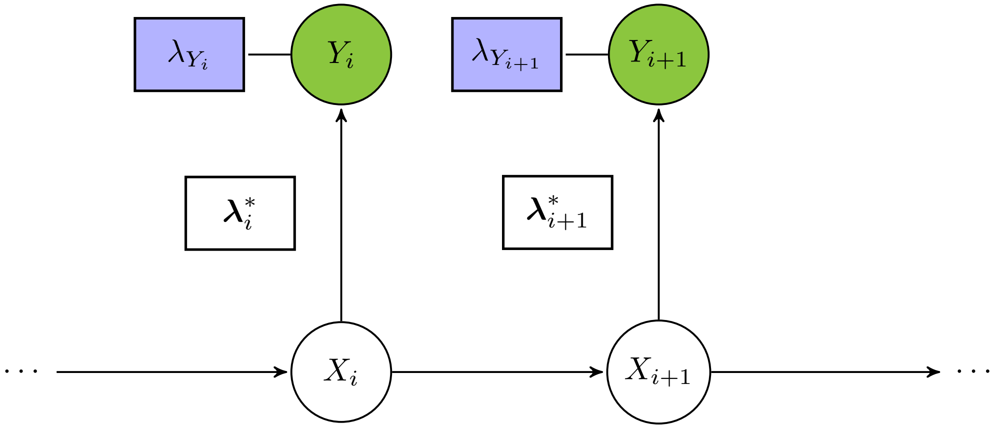
$$\boldsymbol{\alpha} = \begin{pmatrix} 0.95 & 0.05 \\ 0.05 & 0.95 \end{pmatrix}$$

Emission distributions

$$Y_i | X_i = 0 \sim \mathcal{N}(\mu_0, \sigma_0)$$

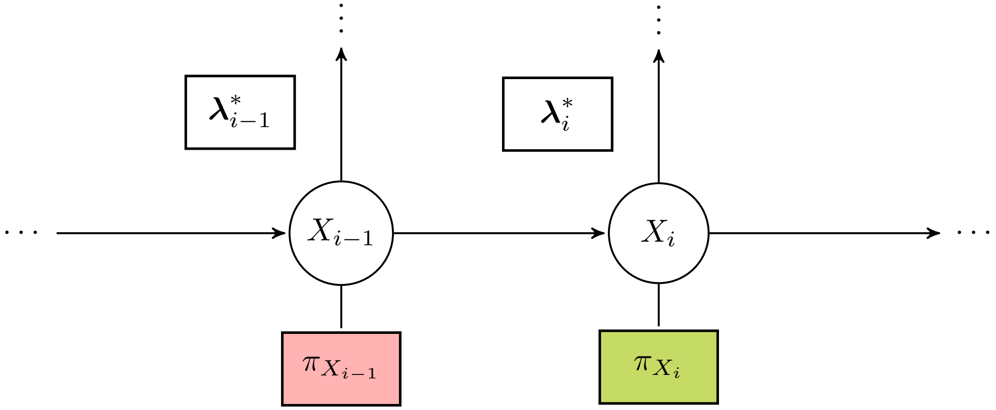
$$Y_i | X_i = 1 \sim \mathcal{N}(\mu_1, \sigma_1)$$

Belief propagation. Initialisation



- ▷ We have a direct evidence $Y_i = y_i$ for each node Y_i.
- ▷ The likelihood vector is infinite and captured by $\lambda_{Y_i} = \delta_{y_i}$.
- ▷ The local likelihood $\lambda_i^*(x_i) = \Pr\left[Y_i = y_i | x_i\right]$ is a finite vector.

Prior propagation. Filtering



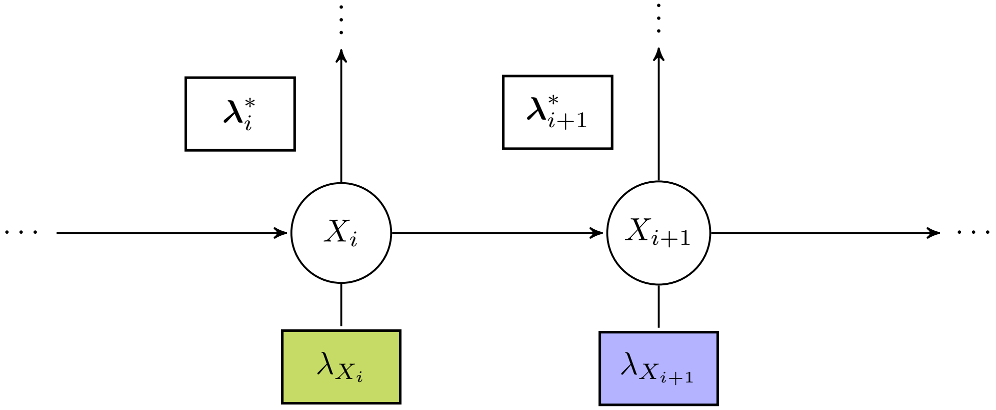
Prior propagation rule yields

$$\pi_{X_i}(x_i) \propto \sum_{x_{i-1}\in\mathcal{S}} \alpha[x_{i-1}, x_i] \cdot \lambda^*_{i-1}(x_{i-1}) \cdot \pi_{X_{i-1}}(x_{i-1})$$

Now we can do filtering

$$\Pr\left[x_i|y_1,\ldots,y_i\right] \propto \pi_{X_i}(x_i) \cdot \lambda^*_i(x_i)$$

Likelihood propagation. Smoothing



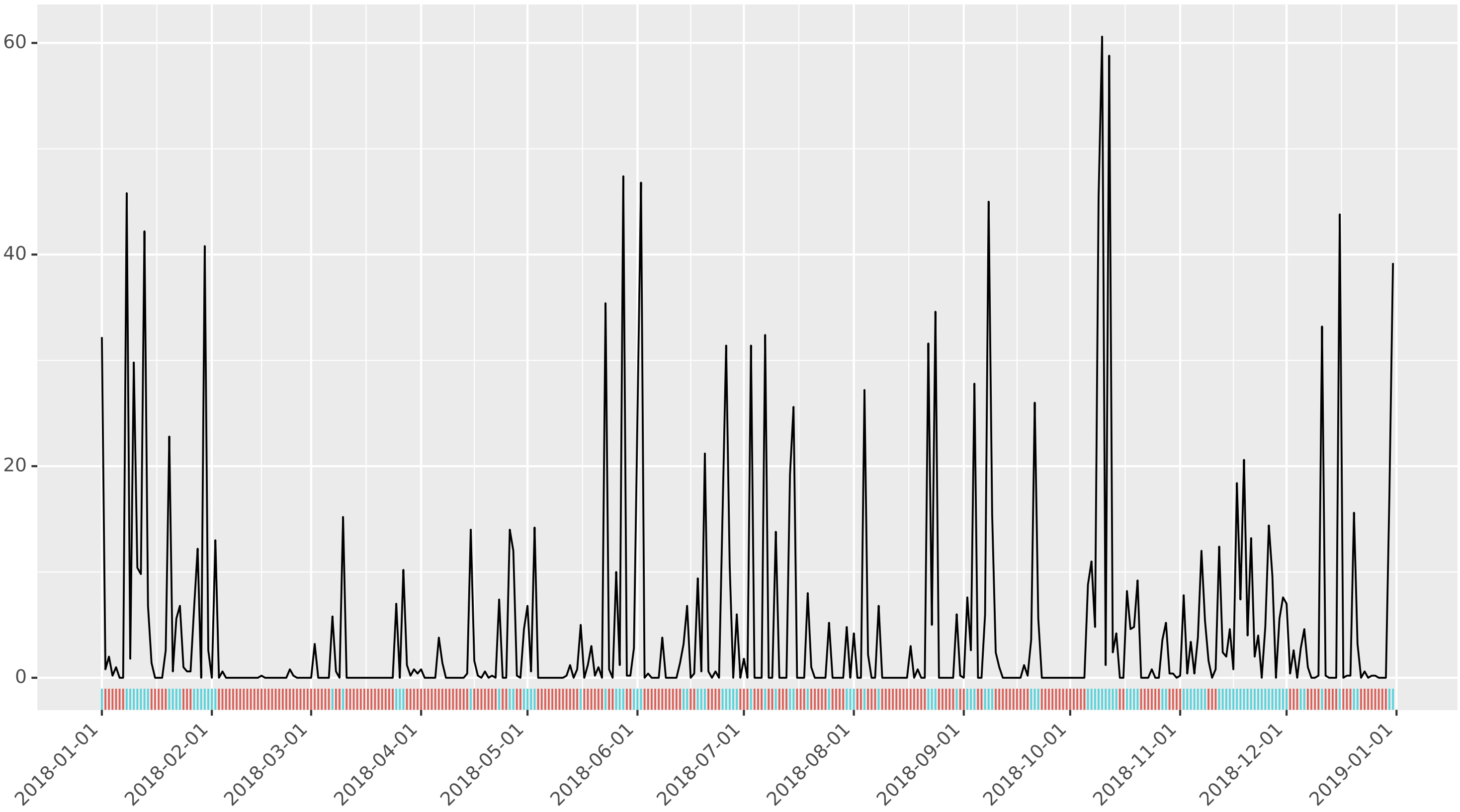
Likelihood propagation rule yields

$$\lambda_{X_i}(x_i) \propto \sum_{x_{i+1}\in\mathcal{S}} \alpha[x_i, x_{i+1}] \cdot \lambda_{X_{i+1}}(x_{i+1}) \cdot \lambda_i^*(x_i)$$

Now we can do smoothing

$$\Pr\left[x_i | y_1, \ldots, y_n\right] \propto \pi_{X_i}(x_i) \cdot \lambda_{X_i}(x_i)$$

Annotated rainfall data


60
40
20
0
2018-01-01
2018-02-01
2018-03-01
2018-04-01
2018-05-01
2018-06-01
2018-07-01
2018-08-01
2018-09-01
2018-10-01
2018-11-01
2018-12-01
2019-01-01