

# LTAT.02.004 MACHINE LEARNING II

## **Sequence models**

Sven Laur  
University of Tartu

# How to write a good touchscreen keyboard?

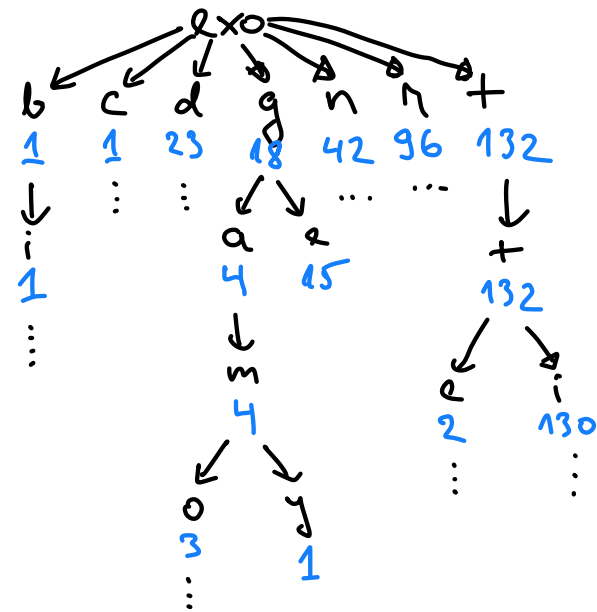
## Possibilities

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

Tree of possibilities



## Discrete random variables

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

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

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

- ▷ Bayes formula

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

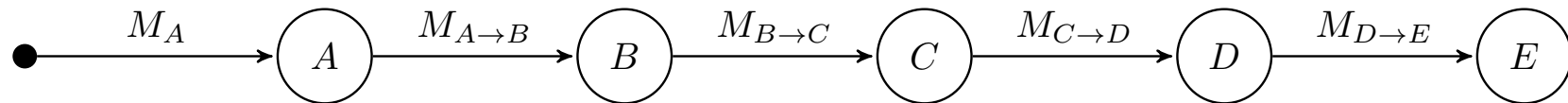
- ▷ Independence of random variables  $X_1 \dots X_m \perp Y_1, \dots Y_n$ :

$$\Pr[x_1 \wedge \dots \wedge x_m \wedge y_1 \wedge \dots \wedge y_n] = \Pr[x_1 \wedge \dots \wedge x_m] \cdot \Pr[y_1 \wedge \dots \wedge y_n]$$

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

$$\Pr[x_1 \wedge \dots \wedge x_m] = \sum_{y_1, \dots, y_n} \Pr[x_1 \wedge \dots \wedge x_m \wedge y_1 \wedge \dots \wedge y_n]$$

# Markov chain



**Definition.** Let  $X_1, X_2, \dots$  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  $\mathcal{S}_1 \dots, \mathcal{S}_n$
- ▷ initial probabilities  $\Pr[x_1]$  given as vectors
- ▷ state transition probabilities  $\Pr[x_{i+1}|x_i]$  given as matrices

## What questions can we ask?

**Sampling:** What are typical outcomes of the chain?

- ▷ Synthesis of time-series, textures, sounds, games movements.

**Stationary distribution:** What happens if we run the chain infinitely long?

- ▷ Getting samples from an unnormalised posterior, optimisation tasks.

**Likelihood estimation:** What is a probability of an observation  $x_1, \dots, x_n$ ?

- ▷ Reasoning about probabilities and clustering sequences.

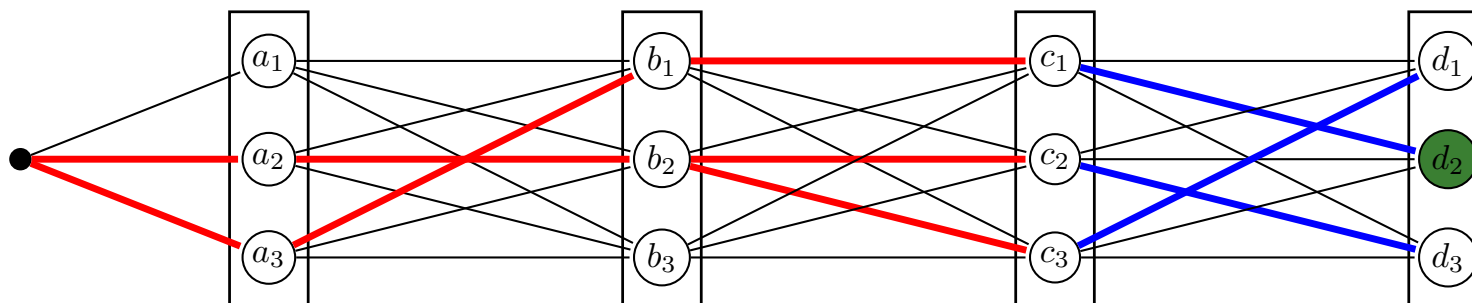
**Decoding:** What is the most probable outcome  $x_1, \dots, x_n$ ?

- ▷ Imputing missing values. Rudimentary logical reasoning.

**Parameter estimation:** What are the model parameters?

- ▷ Machine learning – finding parameters based on observations.

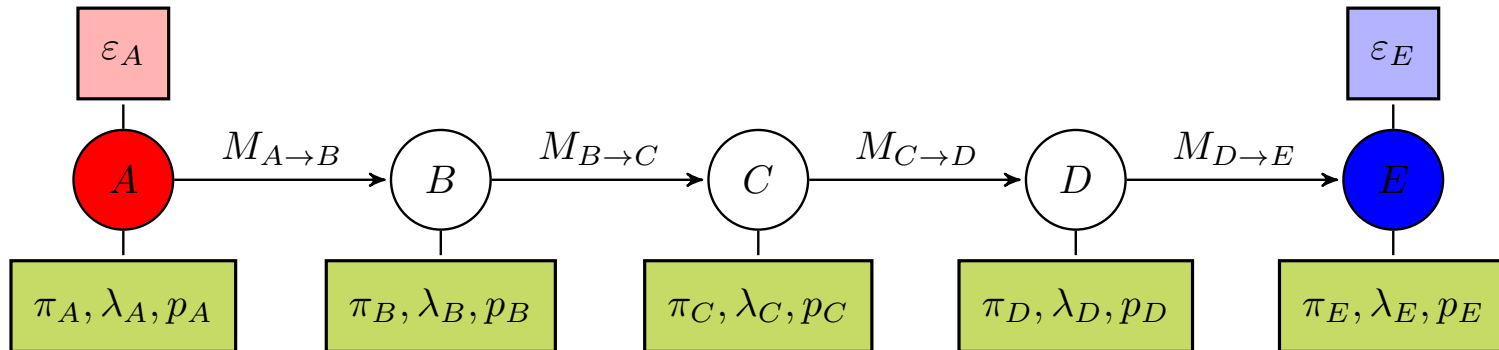
## Posterior maximisation in a chain



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

- ▷ The log-posterior  $\log \Pr[x|\text{evidence}]$  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.

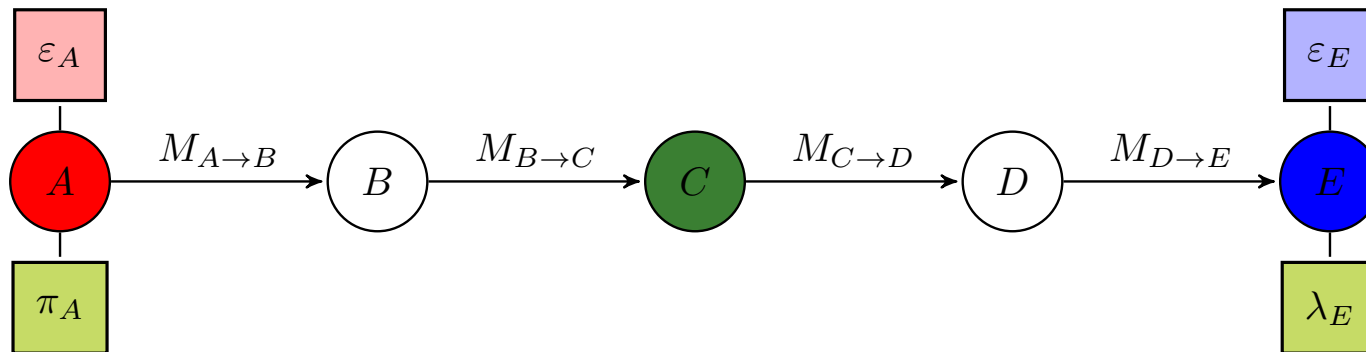
## Belief propagation in a chain



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

- ▷ Evidence  $\varepsilon_V$  is an observational data associated with the node  $V$ .
- ▷ Upstream **evidence<sup>+</sup>** is the evidence at the beginning of chain.
- ▷ Downstream **evidence<sup>-</sup>** is the evidence at the end of chain.
- ▷ Attributes  $\pi_V, \lambda_V, p_V$  are needed to compute marginal distributions.

## Initialisation



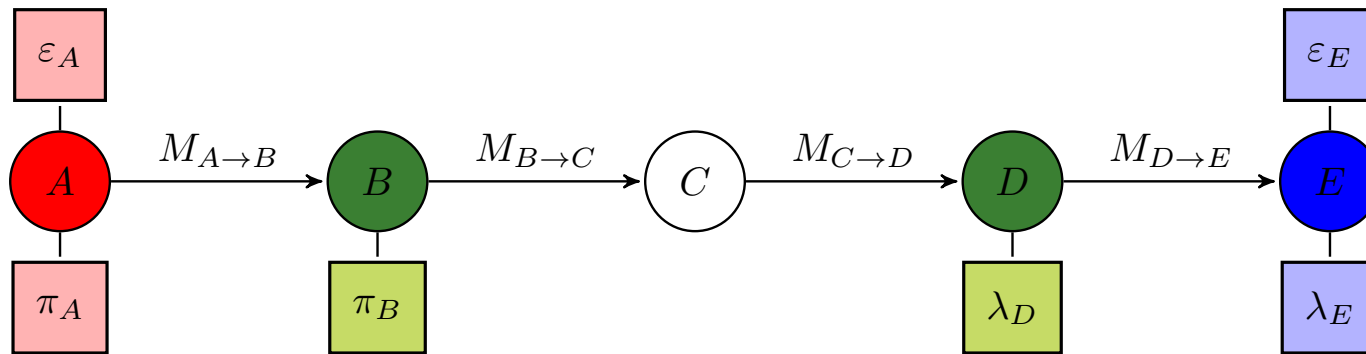
- ▷ Direct evidence  $\varepsilon_V$  determines the value of  $V$ .
- ▷ Indirect evidence  $\varepsilon_V$  determines the value distribution for  $V$ .
- ▷ We can assign the prior for the first and likelihood for the last node

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

$$\lambda_E(e) = \Pr [\text{evidence}^- | E = e] = \Pr [\varepsilon_E | E = e]$$



## Belief propagation



### Inference goal

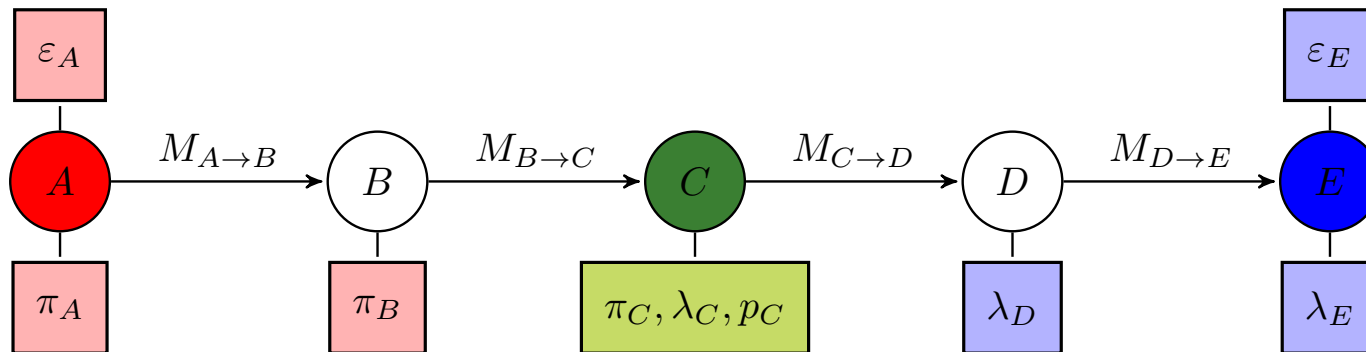
$$\pi_B(b) = \Pr [b | \text{evidence}^+]$$

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

### Iterative propagation rules

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

## Belief propagation



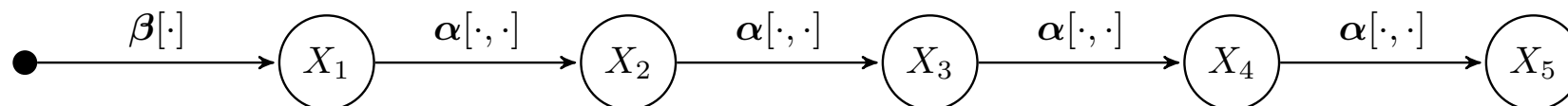
### Inference goal

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

### Iterative update rule

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

## Parameter inference for homogenous case



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

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

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

## 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

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

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

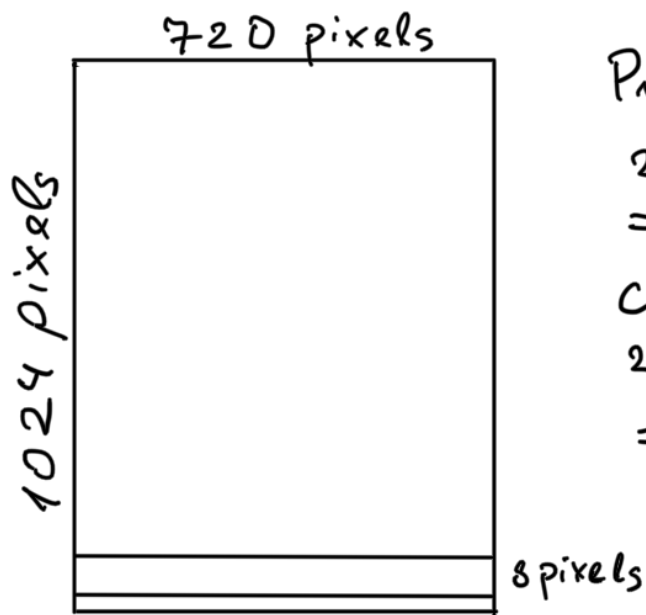
where each of these is equivalent to optimisation of dice throwing posterior. Thus Maximum A posteriori estimates for parameters are

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

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

$$256 \text{ colors} \times 720 \text{ pixels} \times 8 \text{ pixels} \\ \Rightarrow 256^{720 \times 8}$$

Current state

$$256 \text{ colors} \times 720 \text{ pixels} \\ \Rightarrow 256^{720}$$

The number of possible observation is too big already for  $8 \times 8$  patch:

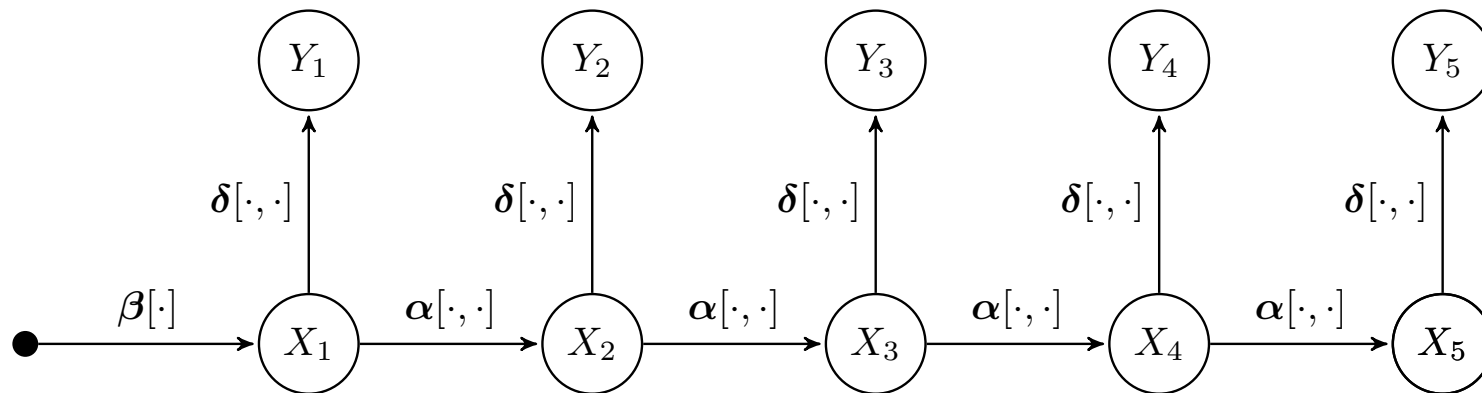
$$256^{8 \times 8} \times 256^8 \times 2^{10} = 2^{8 \times 8 \times 8 + 8 \times 8 + 10} = 2^{586}$$

$8 \times 9$  patches are needed to estimate probabilities within  $\pm 3$  percent points.

## Two ways to build continuous Markov chains

- ▷ Replace a list of discrete states with continuous variable.
  - ◇ We get  $8 \times 8$  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



**Definition.** Let  $X_1, X_2, \dots$  be hidden states that form a Markov chain and let  $Y_1, Y_2, \dots$  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

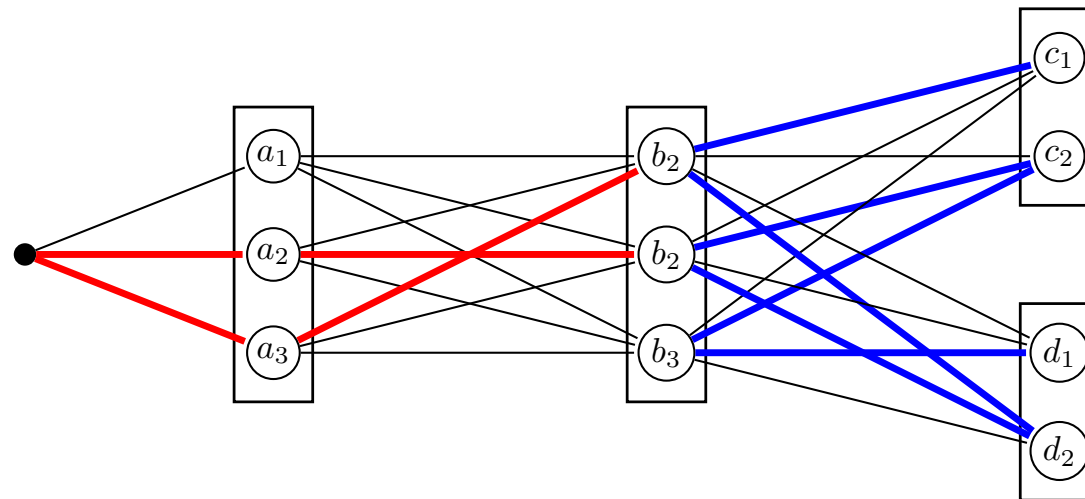
## Sequence annotation

- ▷ fraud detection
- ▷ change detection
- ▷ functional motifs of DNA sequences

## Decoding

- ▷ speech recognition
- ▷ communication over a noisy channels
- ▷ object tracking and data fusion

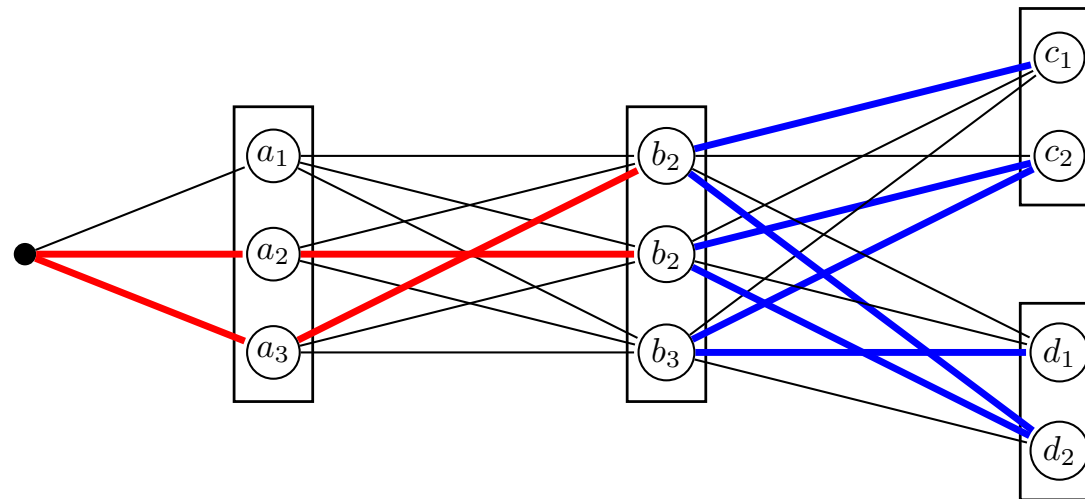
## Posterior maximisation in a tree



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

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

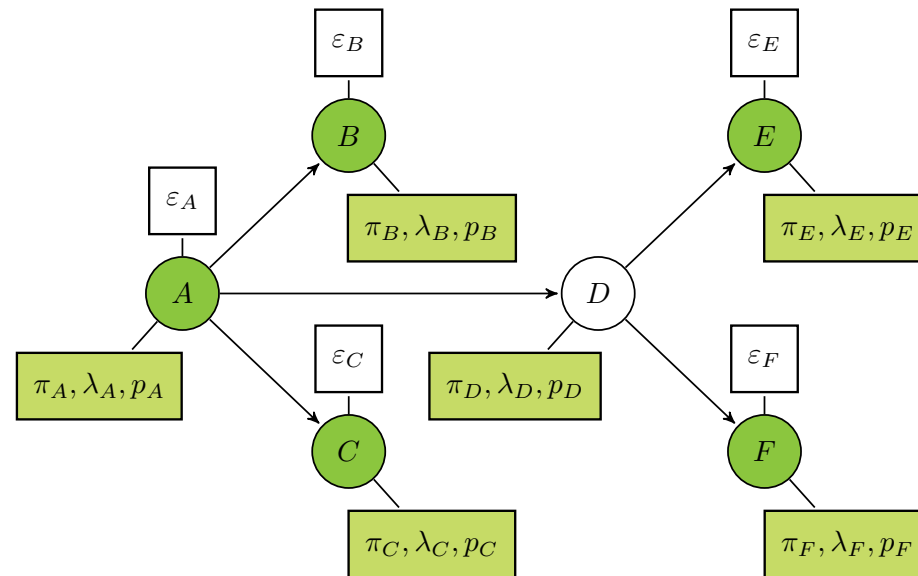
## Decomposition into subtasks



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

- ▷ We can build chains with maximum weight from 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.

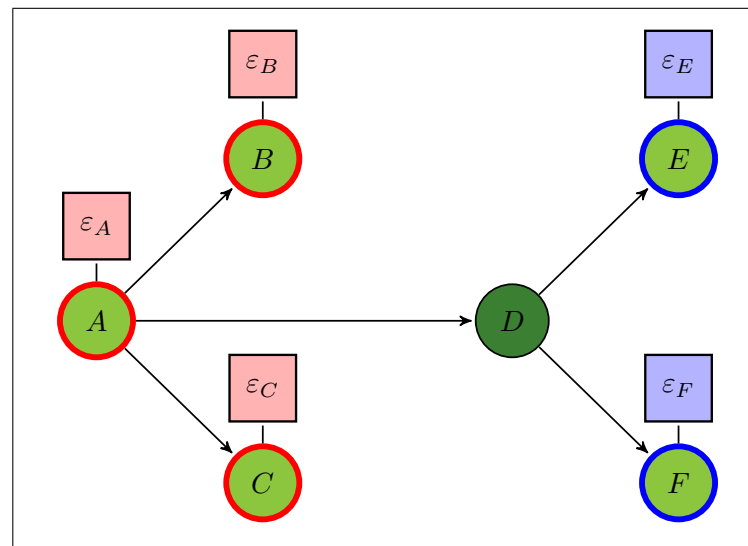
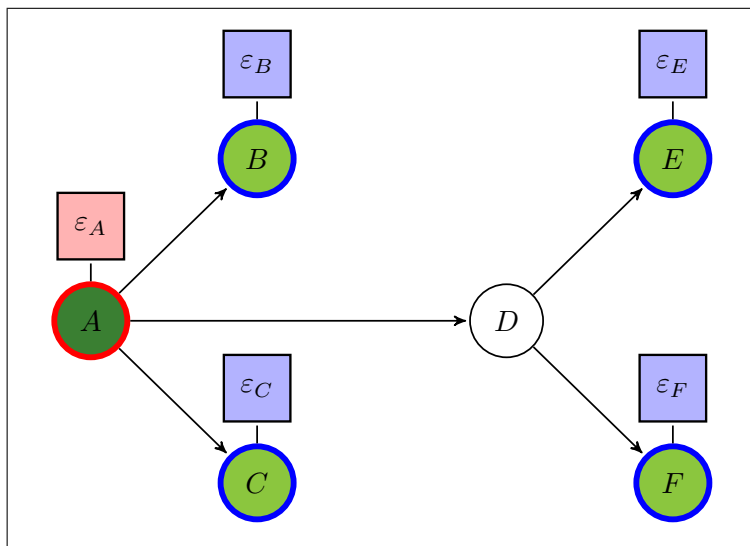
## Belief propagation in a tree



**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.

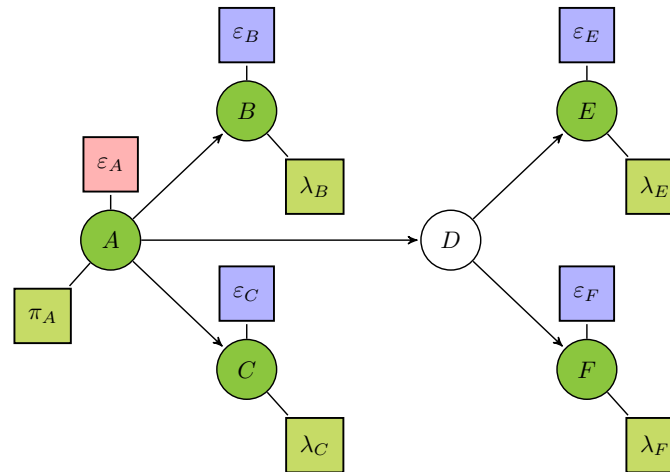
- ▷ Evidence  $\varepsilon_V$  is an observational data associated with the node  $V$ .
- ▷ Attributes  $\pi_V, \lambda_V, p_V$  are needed to compute marginal distributions.

## Evidence decomposition



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

# Initialisation



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

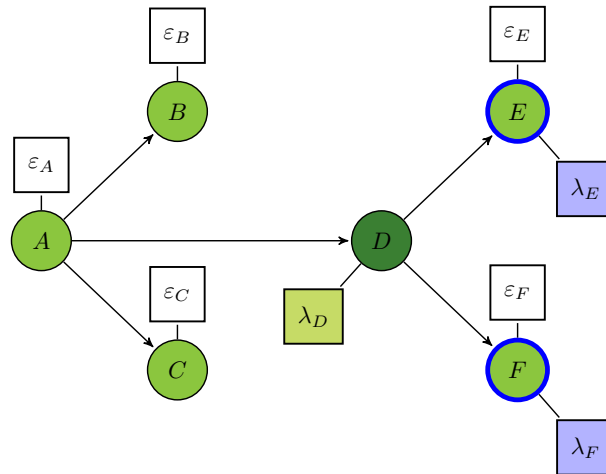
$$\pi_A(a) = \Pr [A = a | \text{evidence}^+(A)] = \Pr [A = a | \varepsilon_A]$$

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

...

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

# Likelihood propagation



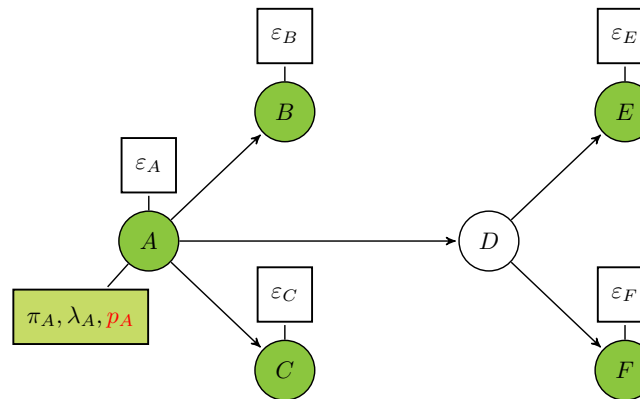
## Inference goal

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

## Iterative propagation rules

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

# Posterior propagation



## Inference goal

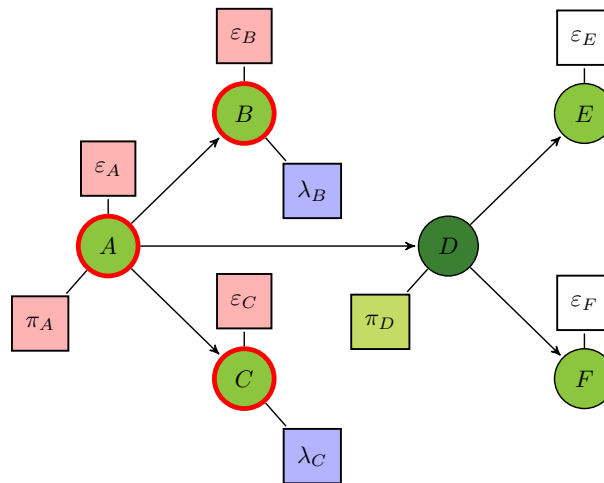
$$p_A(a) = \Pr [A = a | \text{evidence}^+(A), \text{evidence}^-(A)]$$

## Iterative propagation rule

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



## Prior propagation



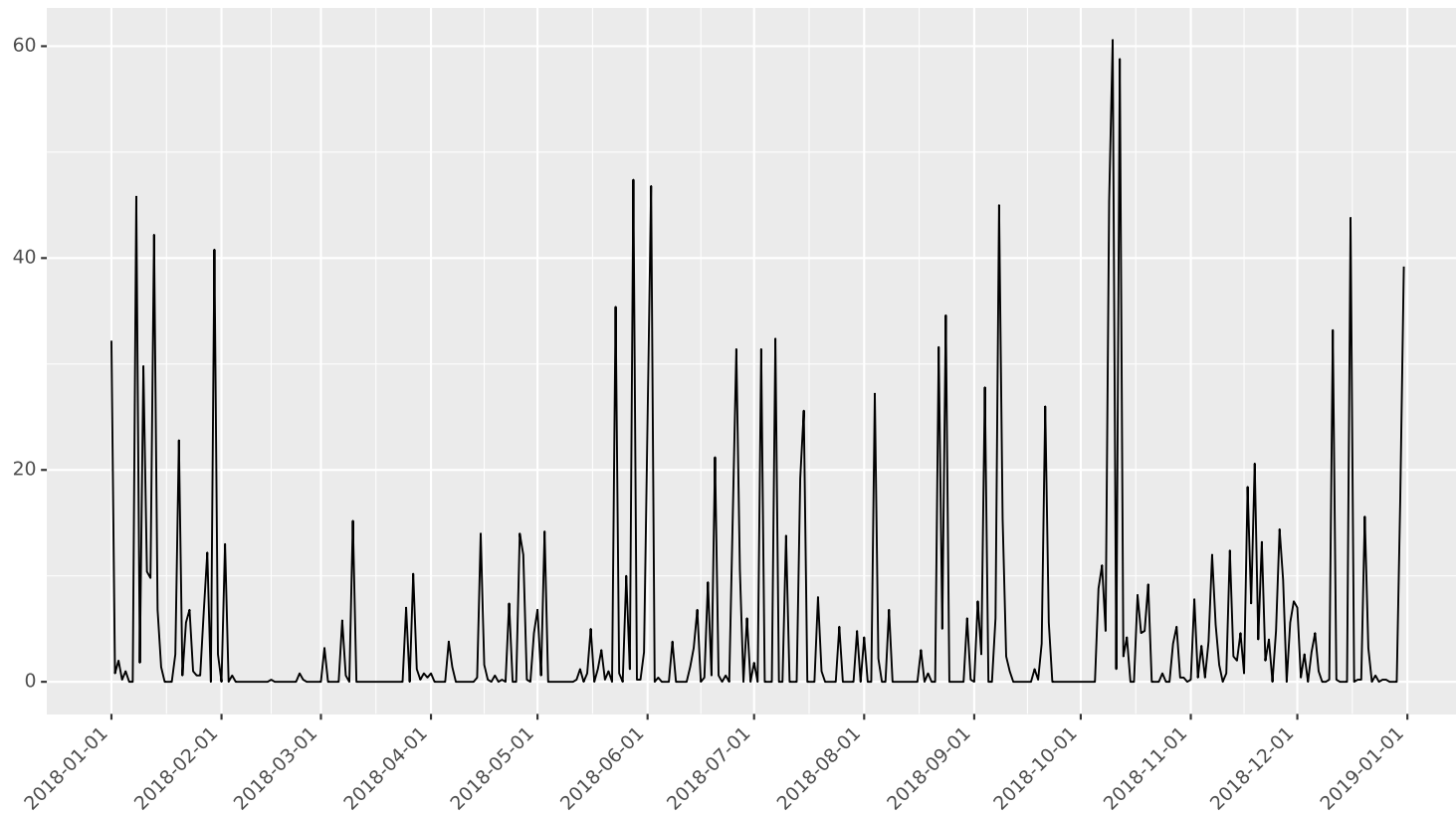
### Inference goal

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

### Iterative propagation rule

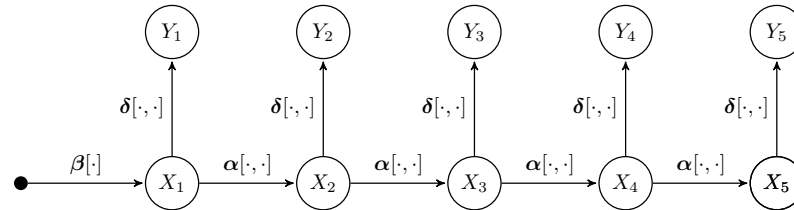
▷ Prior can be computed as  $\pi_D \propto \pi_A M_{A \rightarrow D} \otimes M_{A \rightarrow B} \lambda_B \otimes M_{A \rightarrow C} \lambda_C$  .

## Application on rainfall data



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

# Modelling with Hidden Markov Model



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

$$\beta = (0.5, 0.5)$$

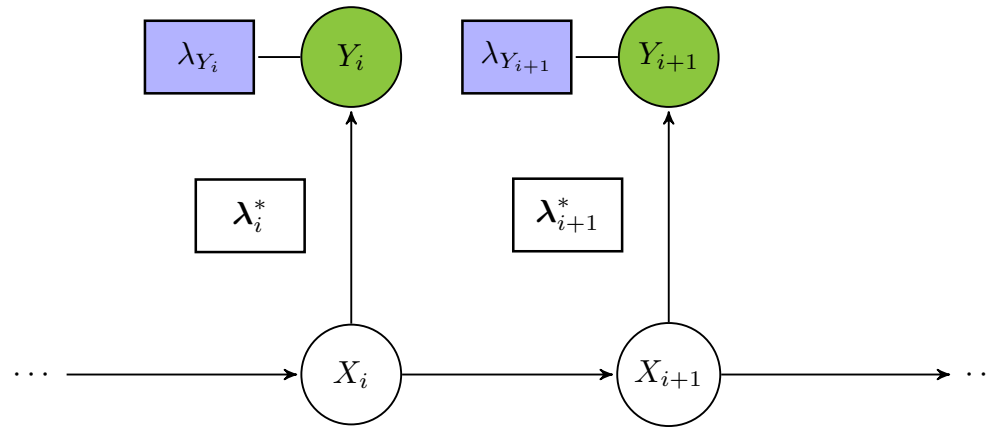
$$\alpha = \begin{pmatrix} 0.95 & 0.05 \\ 0.05 & 0.95 \end{pmatrix}$$

Emission distributions

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

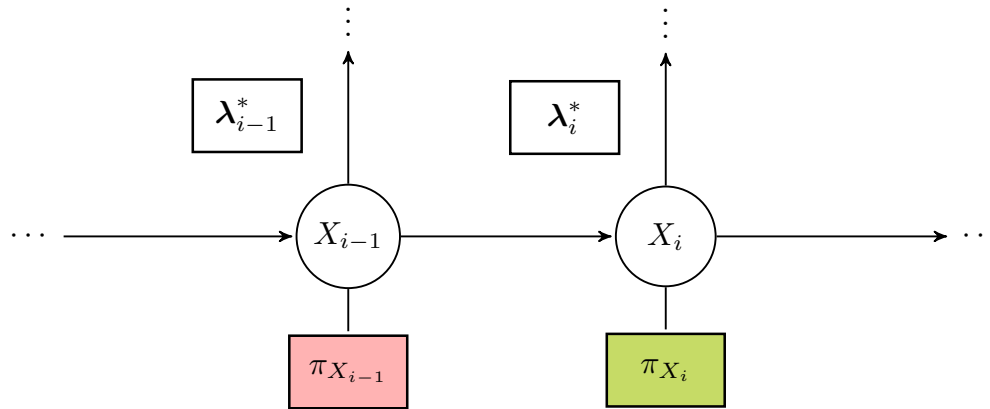
$$Y_i | X_i = 1 \sim \mathcal{N}(\mu_1, \sigma_1)$$

## 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  $\lambda_i^*(x_i) = \Pr[Y_i = y_i | x_i]$  is a finite vector.

## Prior propagation. Filtering



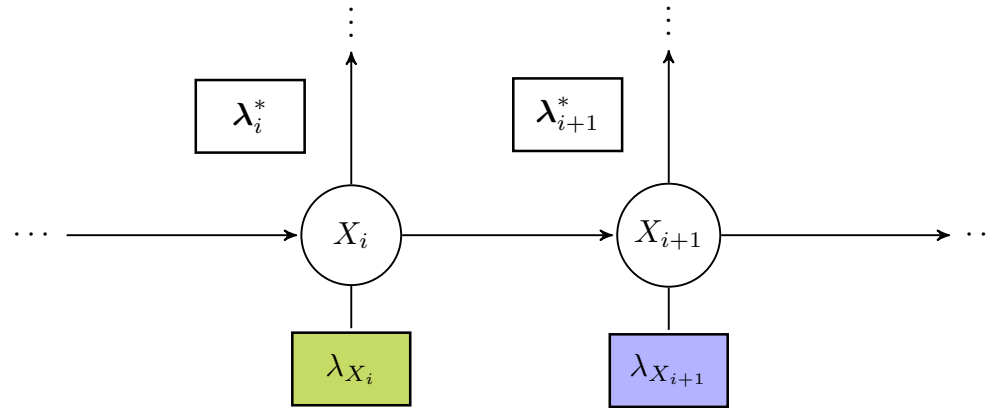
Prior propagation rule yields

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

Now we can do filtering

$$\Pr[x_i | y_1, \dots, y_i] \propto \pi_{X_i}(x_i) \cdot \lambda_i^*(x_i)$$

## 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

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

# Annotated rainfall data

