Data analysis and stochastic modeling

Lecture 6 – Observable and hidden Markov processes

Guillaume Gravier

guillaume.gravier@irisa.fr

What are we gonna talk about?

1. A gentle introduction to probability
2. Data analysis
3. Cluster analysis
4. Estimation theory and practice
5. Mixture models
6. Random processes
   - basic notions of process and fields
   - Markov chains
   - hidden Markov models
7. Queing systems
8. Hypothesis testing

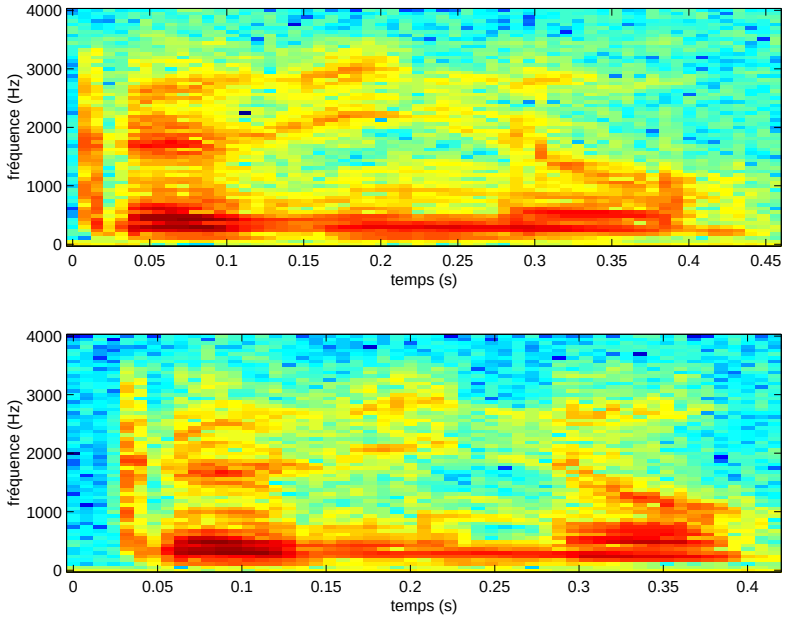
What are we gonna talk about today?

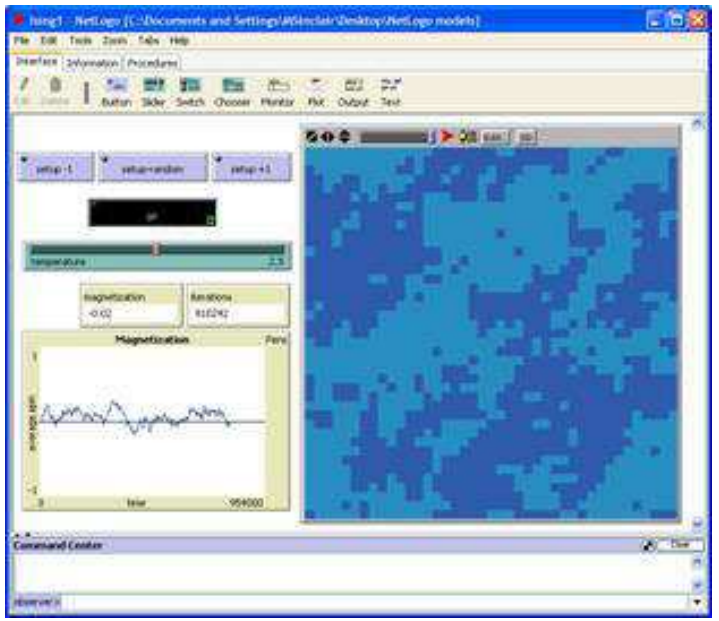
1. Basic notions of random processes
2. Observable discrete Markov chains
3. Hidden Markov models
   - Dynamic programming
     - Viterbi algorithm
     - Dynamic time warping
     - Edit distance (Levenstein)
   - HMM parameter estimation
     - Segmental k-means
     - EM solution: Baum-Welch algorithm
     - Forward-Backward algorithm

What about structure?

Temporal structure

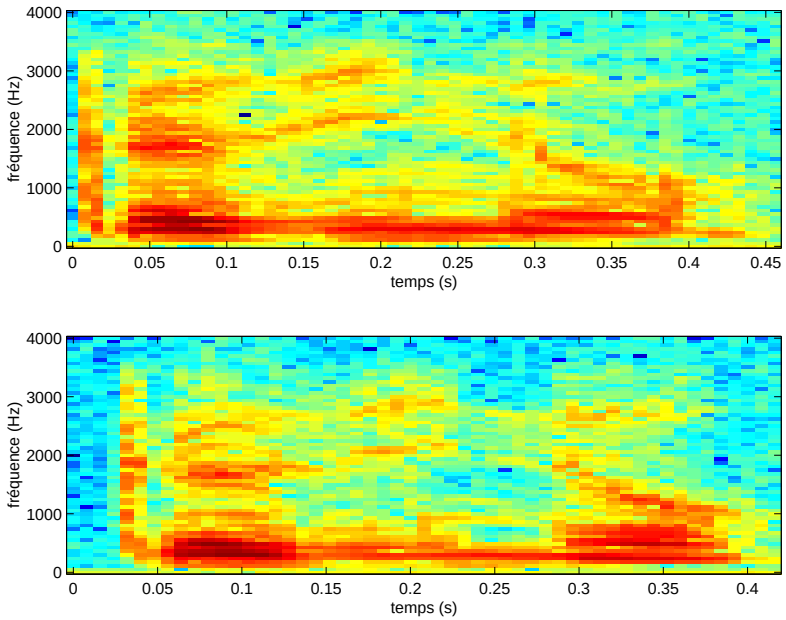
Spatial structure

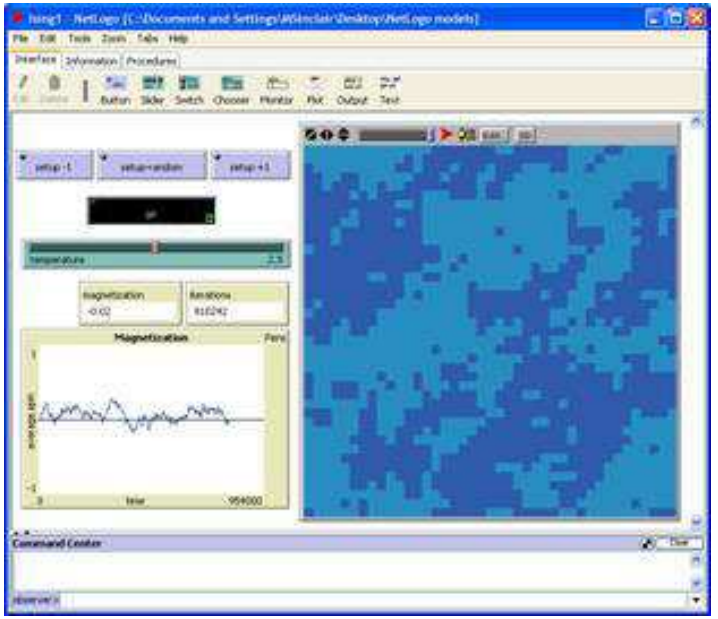




We need for models to take into account structural information!

Processes and fields





Random process

Random field

$$X = \{X_1, \ldots, X_T\}$$

$$X = \{X_s \quad \forall s \in S\}$$

Processes and fields

- **process = a collection of random variables indexed over time**
  - need for models to take into account the temporal evolution
    $\Rightarrow$ Markov models and hidden Markov models
    $\Rightarrow$ death and birth, Poisson and queuing systems

- **field = a collection of random variables indexed over a grid**
  - need for models to take into account the spatial interactions
    $\Rightarrow$ Markov random fields

Note: a temporal relation is a form of spatial relation!

State and index spaces

A process is a collection of random variables $\{X(t)|t \in T\}$ over an index set T where $X(t)$ take values in a state space I.

		T is	
		discrete	continuous
I is	discrete	discrete-time chain	continuous-time chain
	continuous	discrete-time continuous-state process	continuous-time continuous-state process

Note: Formally, one should write $X(t, s)$ where $s \in S$ is an event.

Density functions for processes

- $X(t)$ describes the “state” of the process at time t
- first order distribution function

$$F(x;t) = P[X(t) \leq x]$$

- second order distribution function

$$F(x,x';t,t') = P[X(t) \leq x, X(t') \leq x']$$

- n-th order

$$F(\mathbf{x};\mathbf{t}) = P[X(\mathbf{t}_1) \leq \mathbf{x}_1, \ldots, X(\mathbf{t}_n) \leq \mathbf{x}_n]$$

Definitions

- A stochastic process is **independent** if

$$F(\mathbf{x};\mathbf{t}) = \prod F(\mathbf{x}_i, \mathbf{t}_i)$$

- A stochastic process is **strictly stationary** if

$$F(\mathbf{x};\mathbf{t}) = F(\mathbf{x};\mathbf{t}+\tau) \quad \forall \mathbf{x}, \mathbf{t}, \tau, n \geq 1$$

- A stochastic is **wide sense stationary** if
  1. $\mu(t) = E[X(t)]$ is independent of t
  2. $E[X(t_1)X(t_2)] = R(t_1, t_2) = R(0, t_2 - t_1) = R(\tau)$
  3. $R(0) < \infty$

Markov property

A stochastic process $\{X(t)|t \in T\}$ is a **Markov process** if, for any $t_0 < t_1 < \ldots < t_n < t$, the conditional distribution of $X(t)$ given $X(t_0), \ldots, X(t_n)$ depends only on $X(t_n)$, i.e.

Andreï A. Markov

1856–1922

$$P[X(t) \leq x | X(t_n) \leq x_n, \ldots, X(t_0) \leq t_0] = P[X(t) \leq x | X(t_n) \leq x_n]$$

A Markov chain is said to be homogeneous if

$$P[X(t) \leq x | X(t_n) \leq x_n] = P[X(t - t_n) \leq x | X(0) \leq x_n] \ .$$

DISCRETE TIME

MARKOV CHAINS

Markov random processes

Markov chain of order N

$$P[X_t|X_{t-1},\ldots,X_1] = P[X_t|X_{t-1},\ldots,X_{t-N}]$$

- Process: $X_t \quad t = 1,\ldots,T$
- Joint probability

$$P[X_1,\ldots,X_T] = P[X_1]P[X_2|X_1]P[X_3|X_2,X_1]\ldots P[X_T|X_{T-1},\ldots,X_1]$$

- Markov process of order N

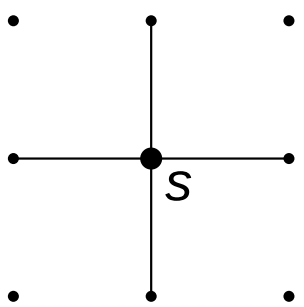
$$P[X_1,\ldots,X_T] = P[X_1]P[X_2|X_1]P[X_3|X_2,X_1]\ldots P[X_T|X_{T-1},\ldots,X_{T-N}]$$

Markov random fields

Markov property in random fields

$$P[X_s = x_s | X_{S\setminus s} = x_{S\setminus s}] = P[X_s = x_s | X_{V_s} = x_{V_s}]$$

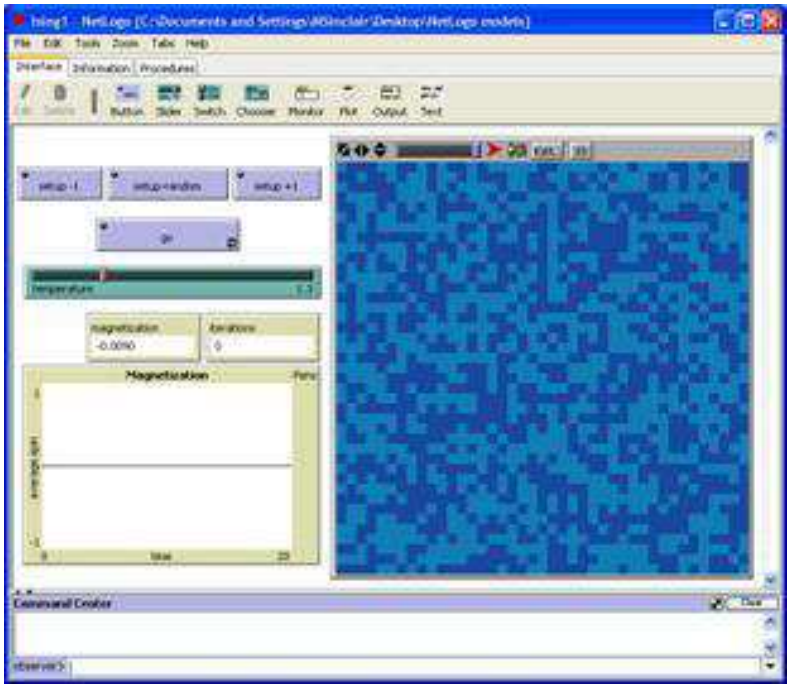
- $\forall s \in S \;\; x_s \in E$
- sample space: $\Omega = E^{|S|}$
- neighborhood system: V
- set of cliques : $c \in \mathcal{C}$

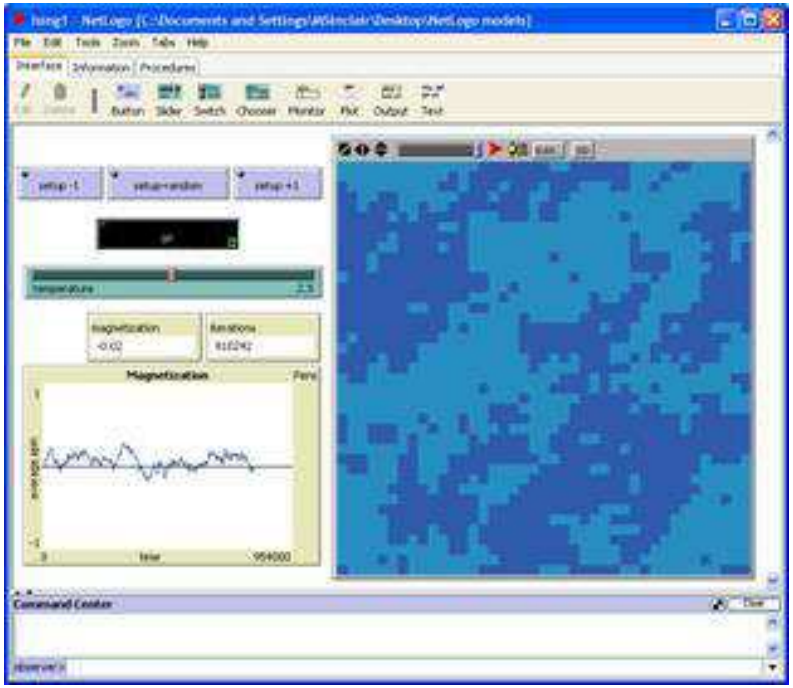


$$P[X = x] = \frac{1}{Z} \exp - \underbrace{\sum_{c \in \mathcal{C}} U_c(x)}_{U(x)} \qquad \text{with } Z = \sum_{x \in \Omega} \exp - U(x)$$

Markov random fields: an example

$$P[X_s|X_{V_s}] = \frac{\exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}{1 + \exp\left(\alpha x_s + \sum_{r\in V_s} \beta x_s x_r\right)}$$





Markov chains

- A Markov chain is a Markov random process where the variable X_t is discrete.
- The discrete variable X_t indicates the state of the process.
- The set of possible values for X_t ($[1, N]$) is known as the state space
- For a stationary Markov chain (of order 1), the parameters of the model can be summarized in a transition matrix A where

$$P[X_t = j | X_{t-1} = i] = a_{ij} \qquad \forall t$$

with $\sum_j a_{ij} = 1$.

Markov chains: a trivial example

- Three states: rainy, cloudy, sunny
- Tomorrow's weather only depends on today's, according to

$$A = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$

- Initially, all three states are equiprobable, *i.e.*

$$\pi = \{1/3, 1/3, 1/3\}$$

Markov chains: a trivial example (cont'd)

What is the probability of the sequence "sun sun sun rain rain cloudy sun"?

$$\mathbf{x} = \{3, 3, 3, 1, 1, 2, 3\}$$

$$\begin{aligned} P[\mathbf{x}] &= P[3]P[3|3]^2 P[1|3]P[1|1]P[2|1]P[3|2] \\ &= \pi_3 a_{33}^2 a_{31} a_{11} a_{12} a_{23} \\ &= 0.0004608 \end{aligned}$$

Markov chains: a trivial example (cont'd)

Given that today is sunny, what is the probability that the weather remain sunny for d days?

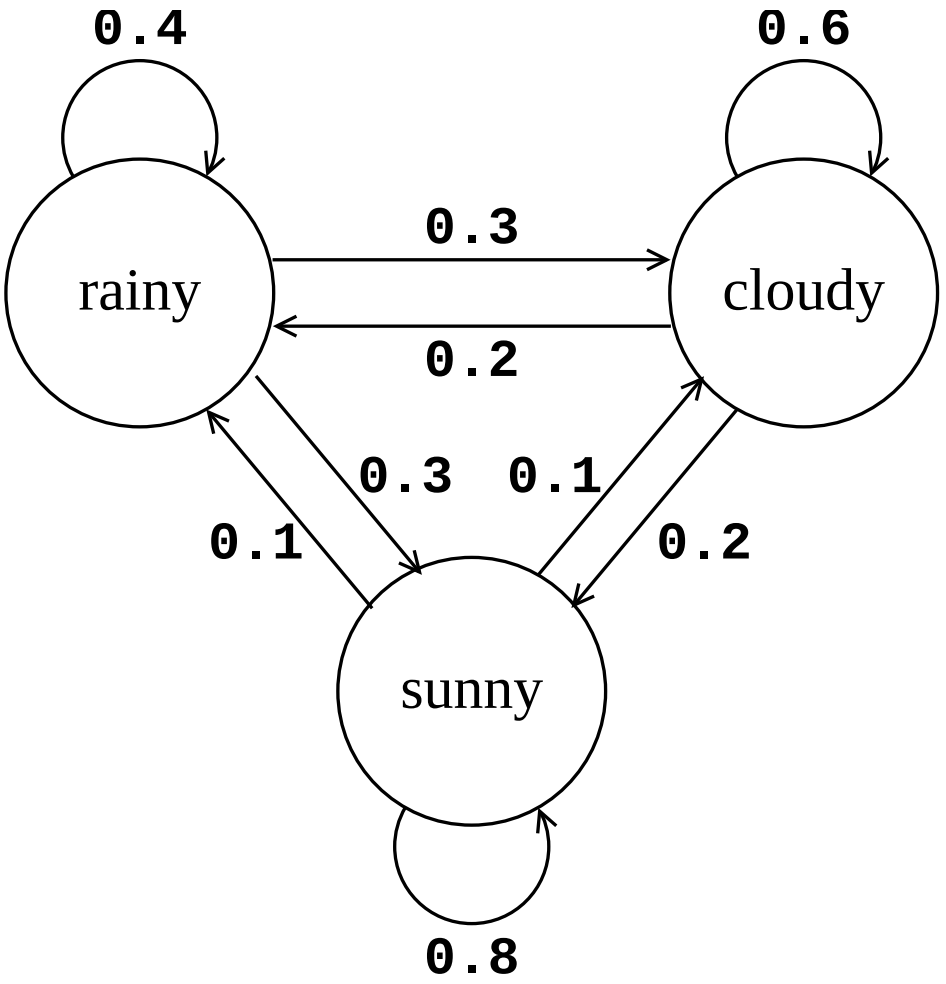
$$\mathbf{x} = \{\underbrace{i, \ldots, i}_{d \text{ times}}, j \neq i\}$$

$$\begin{aligned} P[\mathbf{x}|X_1 = i] &= P[i|i]^{d-1}(\sum_{j \neq i} P[j|i]) \\ &= a_{ii}^{d-1}(1 - a_{ii}) \end{aligned}$$

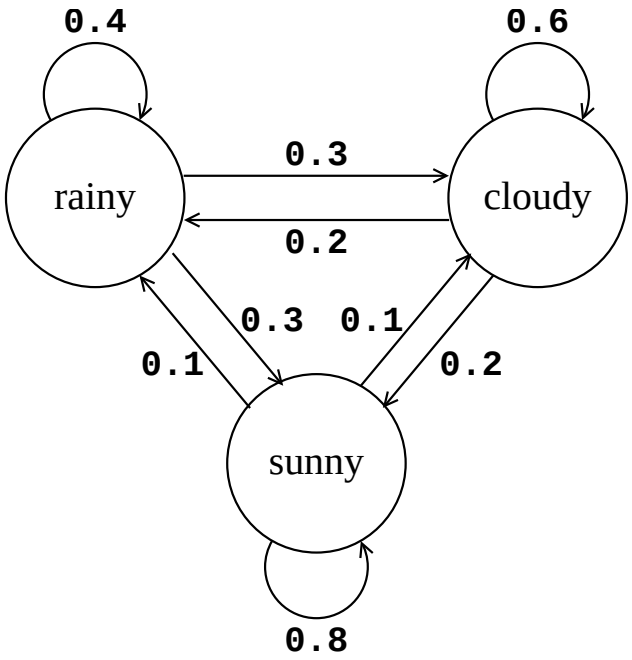
$\Rightarrow$ An implicit exponential state duration model!

Markov chains and automaton

$$A = \begin{pmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{pmatrix}$$



Markov chains from the generative viewpoint



1. pick an initial state according to π
2. pick next state according to the distribution for the current state
3. repeat step 3 until happy or bored

Markov approximation of words

- **Order 0**: XFOML RXKXRJFFUJ ZLPWCFWKCYJ FFJEYVKCQSGHYD QPAAMKBZAACIBZLHJQD
- Order 1: OCRO HLI RGWR NWIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH RBL
- **Order 2**: ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO TIZIN ANDY TOBE SEACE CITSBE
- **Order 3**: IN NO IST LAT WHEY CRATICT FROURE BIRS GROCID PONDENOME OF DEMONSTURES OF THE REPTABIN IS REGOACTIONA OF CRE
- **Markov random field** with 1000 «features,» no underlying «machine» (Della Pietra et. Al, 1997): WAS REASER IN THERE TO WILL WAS BY HOMES THING BE RELOVERATED THER WHICH CONISTS AT RORES ANDITING WITH PROVERAL THE CHESTRAING FOR HAVE TO INTRALLY OF QUT DIVERAL THIS OFFECT INATEVER THIFER CONSTRANDED STATER VILL MENTTERING AND OF IN VERATE OF TO

[courtesy of F. Coste]

Markov approximation of language

- **Order 1**: REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE T
- **Order 2**: THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPETED

«It would be interesting if further approximations could be constructed, but the labor involved becomes enormous at the next stage.» Shannon.

[courtesy of F. Coste]

Parameter estimation in Markov chains

Transition probabilities estimated using empirical frequency estimators (also ML) on a training corpus. For a model of order 3:

$$P[w_k|w_i\ w_j] = \frac{C(w_i, w_j, w_k)}{C(w_i, w_j)}$$ $\Rightarrow$ lots of null probabilities if many events!!!!

$\longrightarrow$ interpolation: interpolate models at different orders

$$P[w_k|w_i\ w_j] = \lambda_1 P[w_k|w_i\ w_j] + \lambda_2 P[w_k|w_j] + \lambda_3 P[w_k]$$

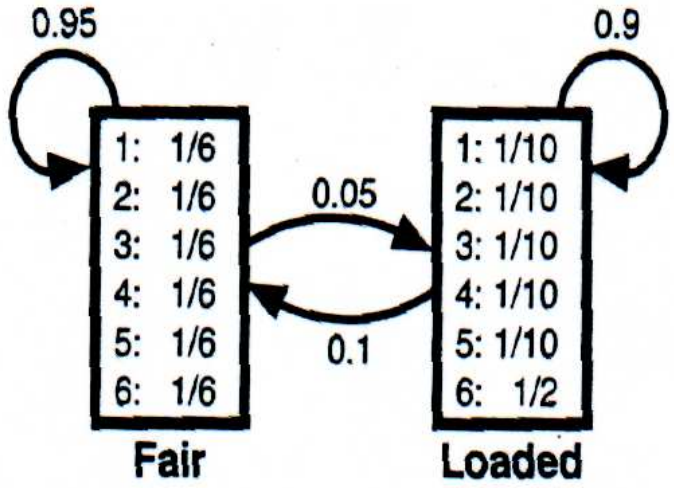
$\longrightarrow$ discounting and back-off: remove probability mass to frequent events and redistribute over unobserved ones based on lower order models.

$$P[w_k|w_i\ w_j] = \begin{cases} \frac{C^*(w_i,w_j,w_k)}{C(w_i,w_j)} & \text{si} \quad C(w_i, w_j, w_k) > K \\ \lambda(w_i, w_j) P[w_k|w_j] & \text{sinon} \end{cases}$$

HIDDEN MARKOV MODELS

Hidden Markov models

- The (state of the) Markov process is not observable
- The observation is a stochastic function depending on the (hidden) state of the underlying Markov process
- A simple example:
  - Throwing two dices where one is loaded and one is not
  - One doesn't know which dice is used but only has access to the result



hidden state sequence –	F	F	F	F	L	L	L	L	L	F	$F\ldots$
observation sequence –	1	6	4	3	6	3	6	6	2	4	$2\ldots$

Hidden Markov model: the Urn-and-Ball model

Urn 1	Urn 2	...	Urn N
P[red] = $b_1(1)$	P[red] = $b_2(1)$	...	P[red] = $b_N(1)$
P[blue] = $b_1(2)$	P[blue] = $b_2(2)$	...	P[blue] = $b_N(2)$
P[green] = $b_1(3)$	P[blue] = $b_2(2)$	...	P[blue] = $b_N(2)$
P[black] = $b_1(4)$	P[blue] = $b_2(4)$	...	P[blue] = $b_N(4)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
P[orange] = $b_1(M)$	P[orange] = $b_2(M)$	...	P[orange] = $b_N(M)$

Balls are selected from an urn i and then, a new urn j is selected based on some (transition) probabilities depending on i. Finally, one only observes the balls $\mathbf{o} = \{\text{green}, \text{green}, \text{blue}, \text{yellow}, \ldots, \text{red}\}$ but does not know the urns the balls are coming from.

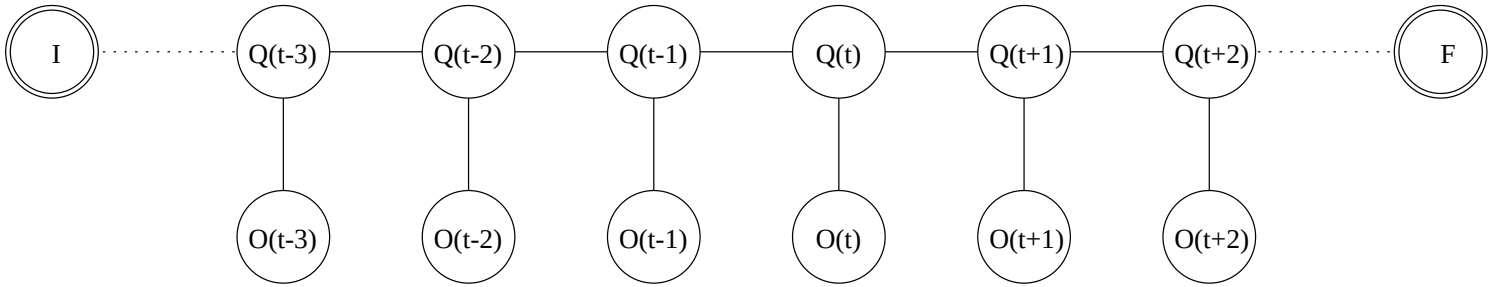
Hidden Markov model: definition

The hidden Markov model represents two random processes:

- the hidden process $S = S_1, \ldots, S_T$ representing the sequence of states in a Markov chain, where $S_t \in [1, N]$.
- the observation process $O = O_1, \ldots, O_T$ where the probability of an observation depends on the state sequence

  Note: the observation can be either discrete or continuous!

- The observations are supposed independent given the hidden state sequence



Hidden Markov model: the maths

- Joint probability of the two sequences:

$$P[O = o_1, \ldots, _T, S = s_1, \ldots, s_T] = P[O|S]\ P[S]$$

- State sequence probability given by a Markov chain (or order 1)

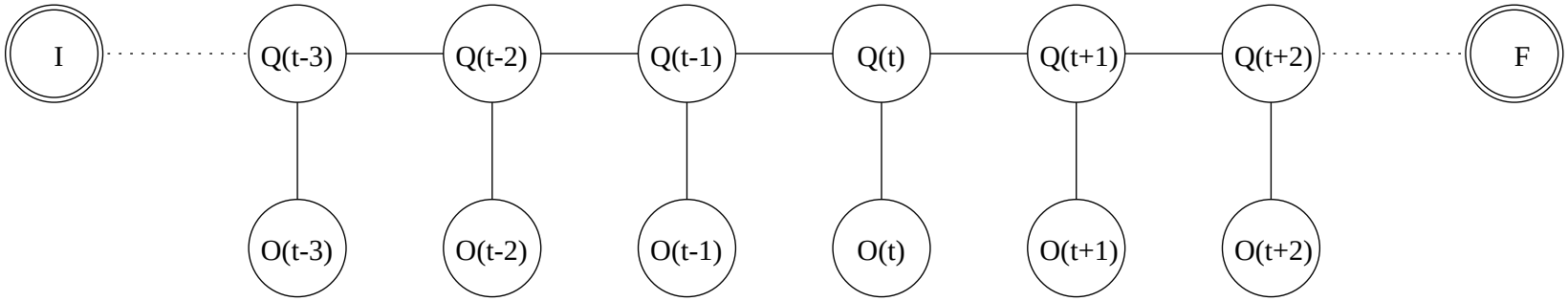
$$P[S = s_1, \ldots, s_T] = \pi_{s_1} \prod_{t=2}^{T} P[s_t|s_{t-1}]$$

- Conditional probability of the observation sequence using the conditional independence assumption

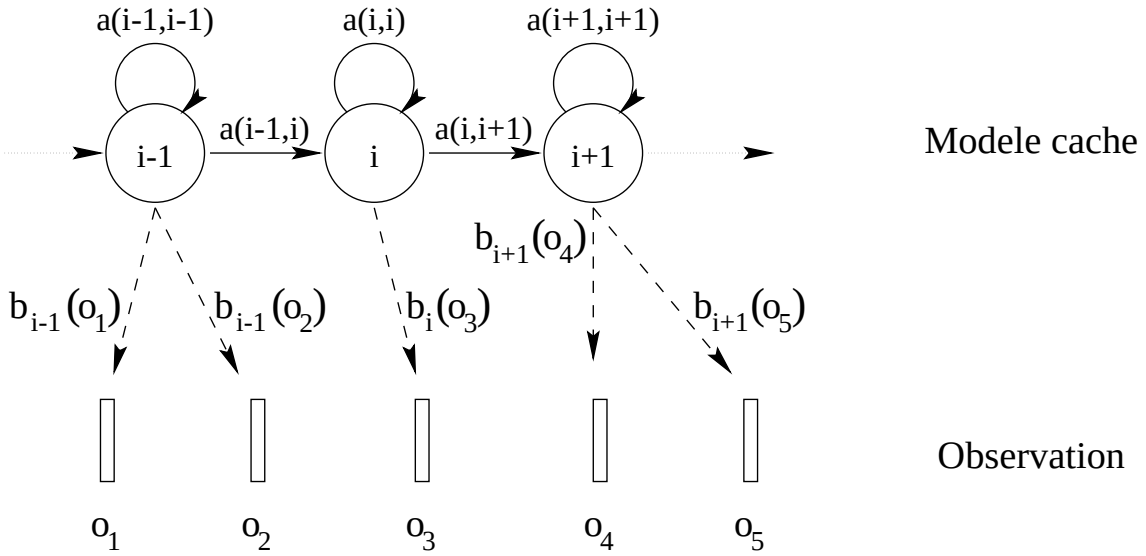
$$P[O = o_1, \ldots, _T|S = s_1, \ldots, s_T] = \prod_{t=1}^{T} P[o_t|s_t]$$

Hidden Markov model and automaton

- Graphical representation of a HMM



- Automaton representation of a HMM



Elements of a HMM

Model = $\lambda_N(\pi, A, B)$

- Number of states in the model: N
- Initial state distribution: π
- Transition probability matrix: A where $a_{ij} = P[S_t = j | S_{t-1} = i]$
- State conditional densities
  - Discrete HMM

    $\longrightarrow$ matrix of CPD B where $b_{ik} = P[O_t = k | S_t = i]$
  - Continuous density HMM

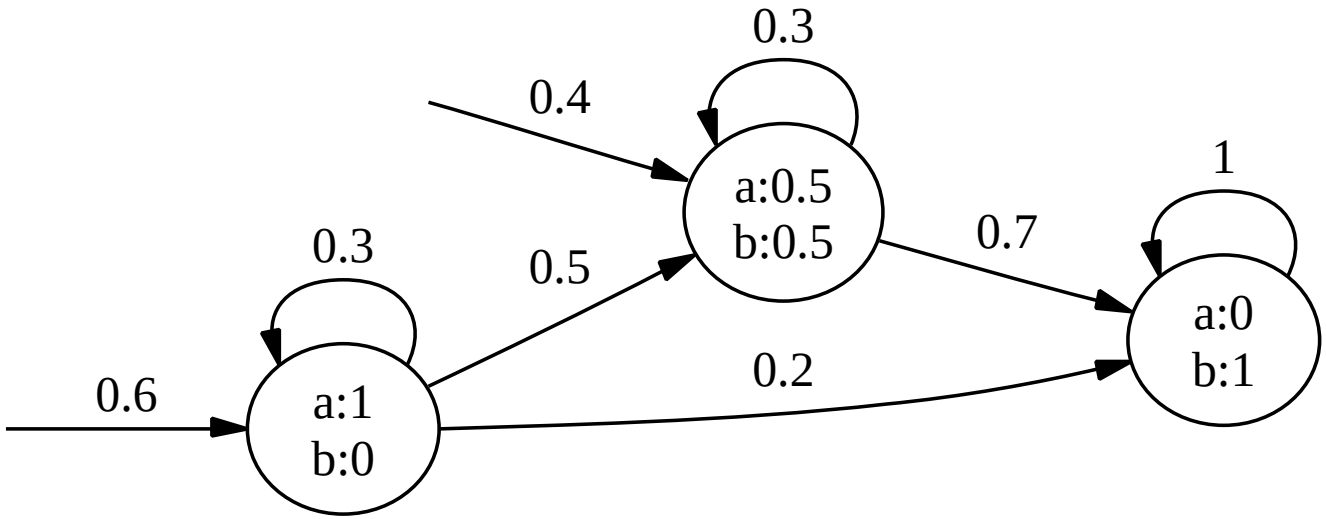
    $\longrightarrow$ collection of conditional parametric densities $b_i()$

    $\longrightarrow$ in practice, $b_i()$ is often a Gaussian mixture

Hidden Markov model: example

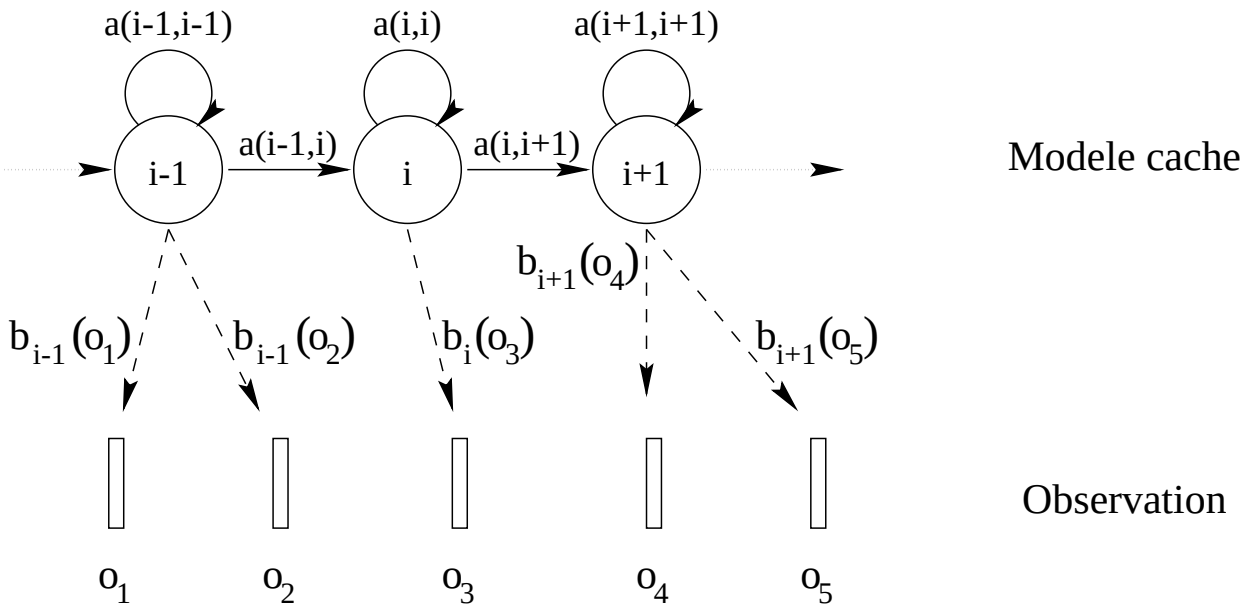
Soit le modèle $\lambda = (A, B, \pi)$ comprenant trois états 1,2,3 et permettant d'observer deux symboles a et b.

$$A = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0 & 0.3 & 0.7 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.6 \\ 0.4 \\ 0 \end{pmatrix}$$

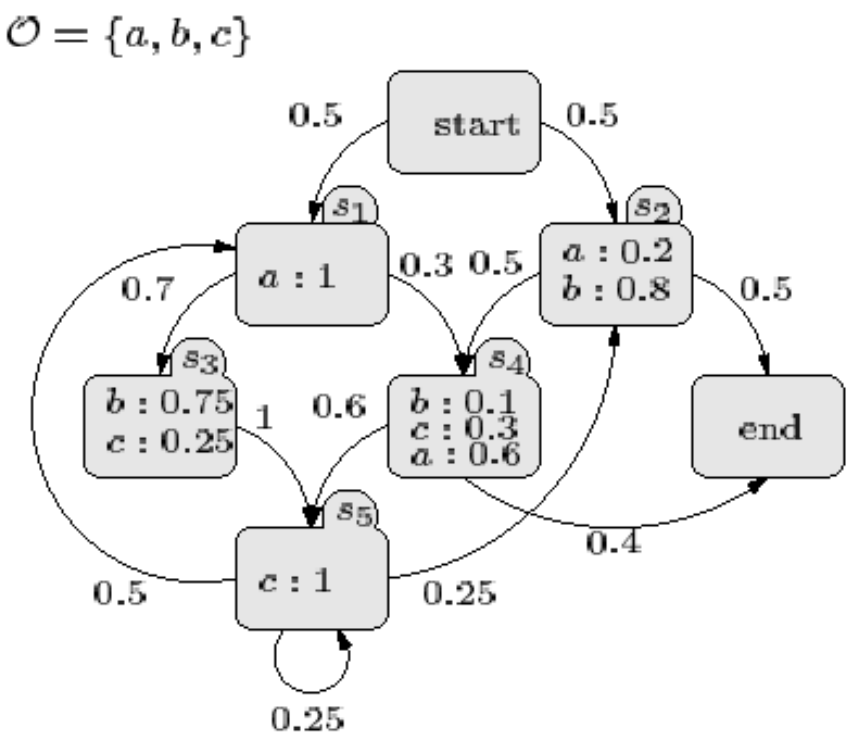


HMM from the generative viewpoint

1. choose initial state s_1 according to the initial distribution π
2. draw an observation sample according to the law $b_{s_1}()$
3. foreach $t = 1, \ldots, T$
   (a) choose next state s_{t+1} according to the transition probabilities
   $$P[S_{t+1} = j | S_t = s_t] = a_{s_t,j}$$
   (b) draw an observation sample according to the law $b_{s_1}()$
4. end



HMM from the generative viewpoint



start		start		start	
s_1	a	s_1	a	s_2	a
s_3	b	s_4	b	s_4	b
s_5	c	s_5	c	s_5	c
s_5	c	s_5	c	s_5	c
s_5	c	s_5	c	s_5	c
s_2	b	s_2	b	s_2	b
end		end		end	
$\approx 4 \cdot 10^{-4}$		$\approx 4.8 \cdot 10^{-5}$		$\approx 8 \cdot 10^{-6}$	

Source: Laurent Bréhélin

The three problems [Rabiner 89]

1. **Find out the most likely state sequence**

   Given a model λ_N, how to efficiently compute the state sequence $\mathbf{s} = s_1, \ldots, s_T$ for which the probability of a given observation sequence $\mathbf{o} = o_1, \ldots, o_T$ is maximum.

2. **Evaluate the probability of an observation sequence**

   Given a model λ_N, how to efficiently compute the probability of a given observation sequence $\mathbf{o} = o_1, \ldots, o_T$?

3. **Parameter estimation**

   - Given a set of training sequences, how to efficiently estimate the parameters of a model λ_N according to the maximum likelihood criterion.

DYNAMIC PROGRAMMING

The Viterbi algorithm

Problem 1: Most likely state sequence

Find out the most likely state sequence

Given a model λ_N, how to efficiently compute the state sequence $\mathbf{s} = s_1, \ldots, s_T$ for which the probability of a given observation sequence $\mathbf{o} = o_1, \ldots, o_T$ is maximum, *i.e.*

$$\widehat{\mathbf{s}} = \arg \max_{s_1,\ldots,s_T} P[o_1, \ldots, o_T | s_1, \ldots, s_T; \lambda_N] P[s_1, \ldots, s_T; \lambda_N] \ .$$

Problem 1: Most likely state sequence

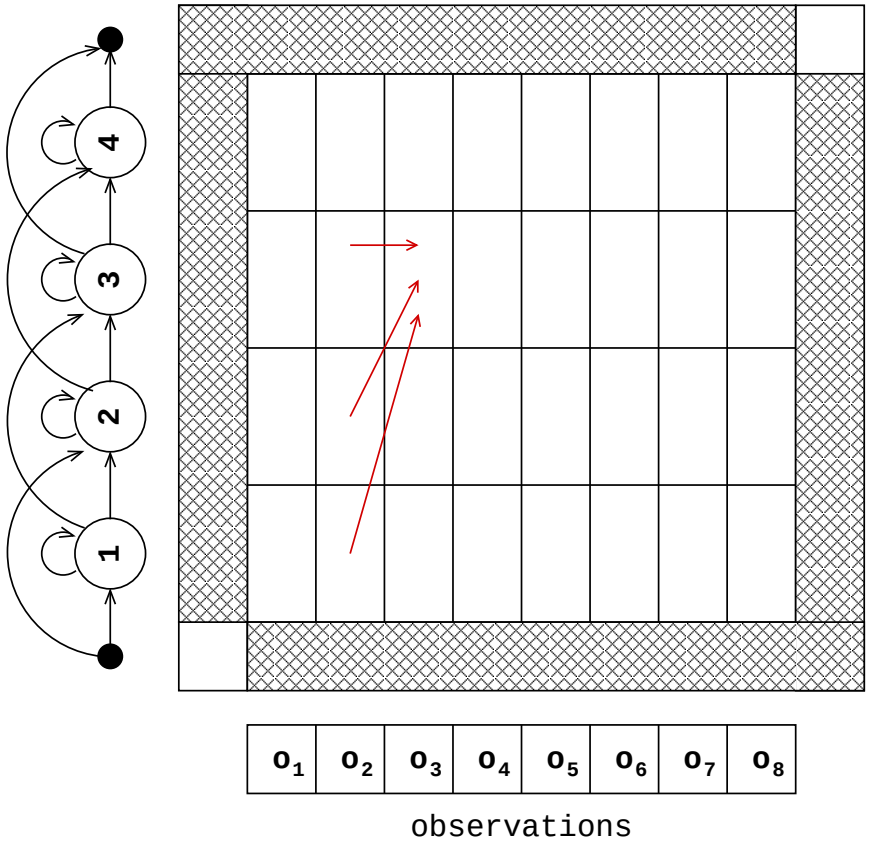
In the log-domain, we seek to maximize the following joint log-probability of the observation and state sequences

$$\ln P[\mathbf{o}, \mathbf{s}] = \ln(\pi_{s_1}) + \ln(b_{s_1}(o_1)) + \sum_{t=2}^{T} \left(\ln(a_{s_{t-1}s_t}) + \ln(b_{s_t}(o_t))\right)$$

$$\Downarrow$$

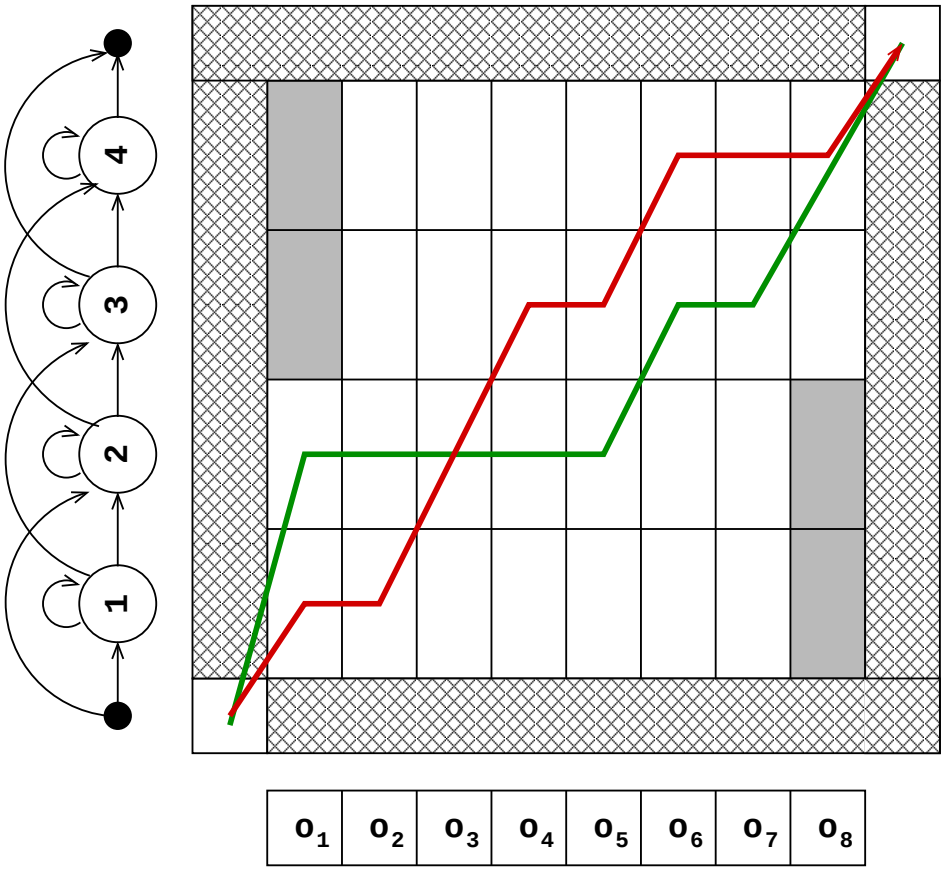
Efficiently solved using a dynamic programming algorithm to find the best path in a treillis.

Treillis structure



- A treillis can be seen as a matrix (or a graph) where the elements are (state,observation) pairs.
- Each element of the treillis has a set of pre-decessors as defined by the topology of the Markov chain.

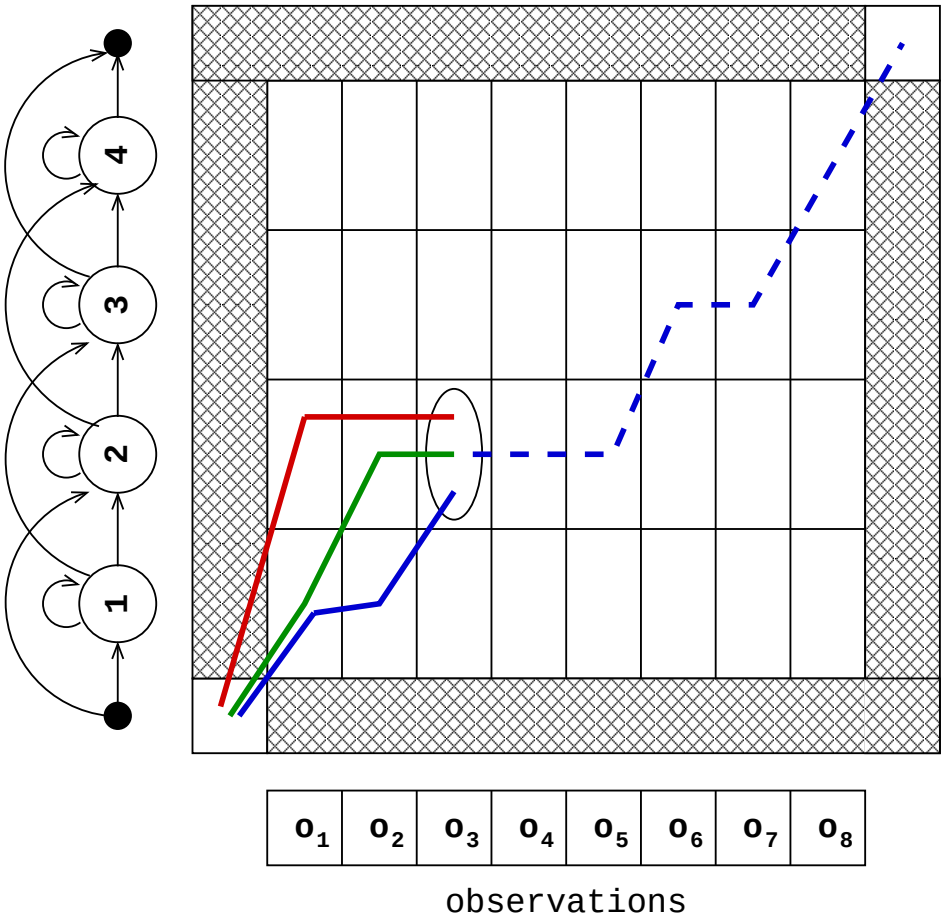
Paths in a treillis



State sequence: $S_1 = 1, S_2 = 1, S_3 = 2, S_4 = 3, S_5 = 3, S_6 = 4, S_7 = 4, S_8 = 4$

State sequence: $S_1 = 2, S_2 = 2, S_3 = 2, S_4 = 2, S_5 = 2, S_6 = 3, S_7 = 3, S_8 = 4$

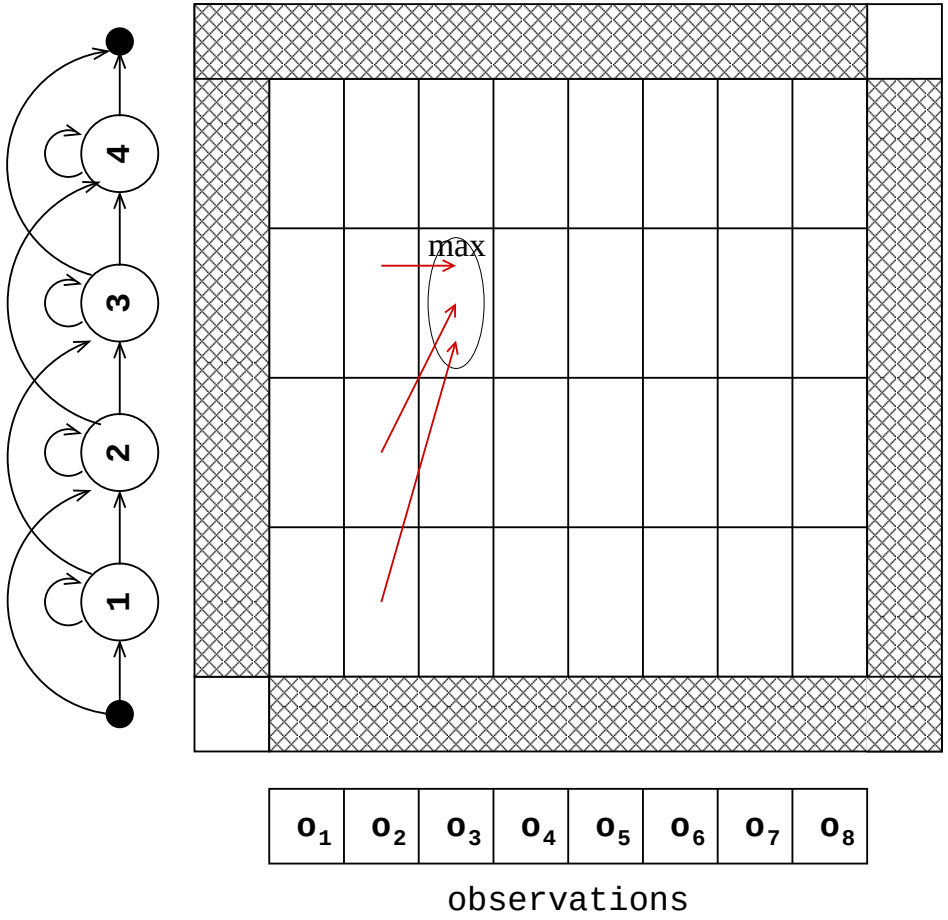
Bellman's principle



The best partial path to state (i, t) is necessarily a part of the best path which goes through (i, t).

Principle of the Viterbi algorithm

Principle: iteratively construct the best partial paths until treillis has been completely explored.



Viterbi algorithm

Let's define the two following quantities:

- $H(i,t)$: score of the best partial path up to (i,t)
- $B(i,t)$: best predecessor at node (i,t)

To compute the best path up to (i,t), one has to solve

$$H(i,t) = \max_j H(j,t-1) + \ln a_{ji} + \ln b_i(o_t)$$

[Andrew J. Viterbi. *Error bounds for convolutional codes and asymptotically optimal decoding algorithm.* In IEEE Information Theory, 13:260–269, April, 1967]

Viterbi algorithm

- Initialization

$$H(i,1) = \ln \pi_i + \ln b_i(o_1)$$
$$B(i,1) = 0$$

- Propagation:

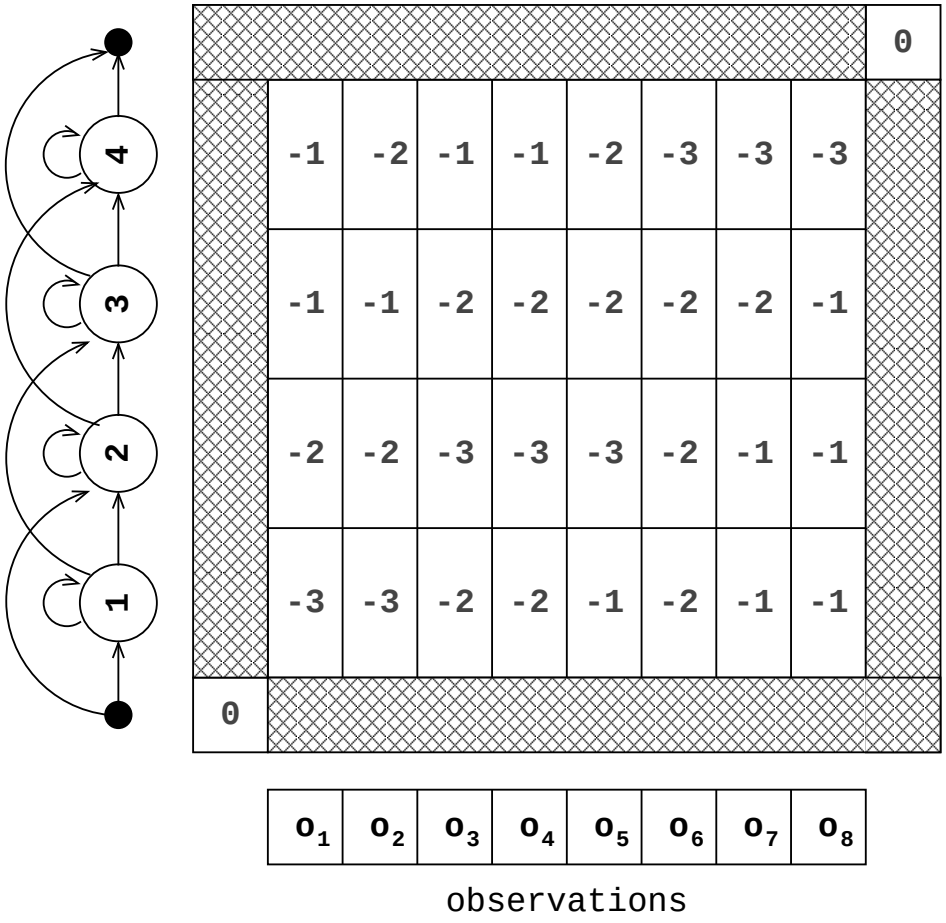
For $i = 1, \ldots, N$ *and* $t = 1, \ldots, T$

$$H(i,t) = \max_j H(j,t-1) + \ln a_{ji} + \ln b_i(o_t)$$
$$B(i,t) = \arg\max_j H(j,t-1) + \ln a_{ji} + \ln b_i(o_t)$$

- Backtracking:

$$H^* = \max_i H(i,T)$$
$$\widehat{s}_T = \arg\max_i H(i,T)$$
$$\widehat{s}_t = B(\widehat{s}_{t+1}, t+1) \quad \text{for } t = T-1, \ldots, 1$$

Viterbi at work

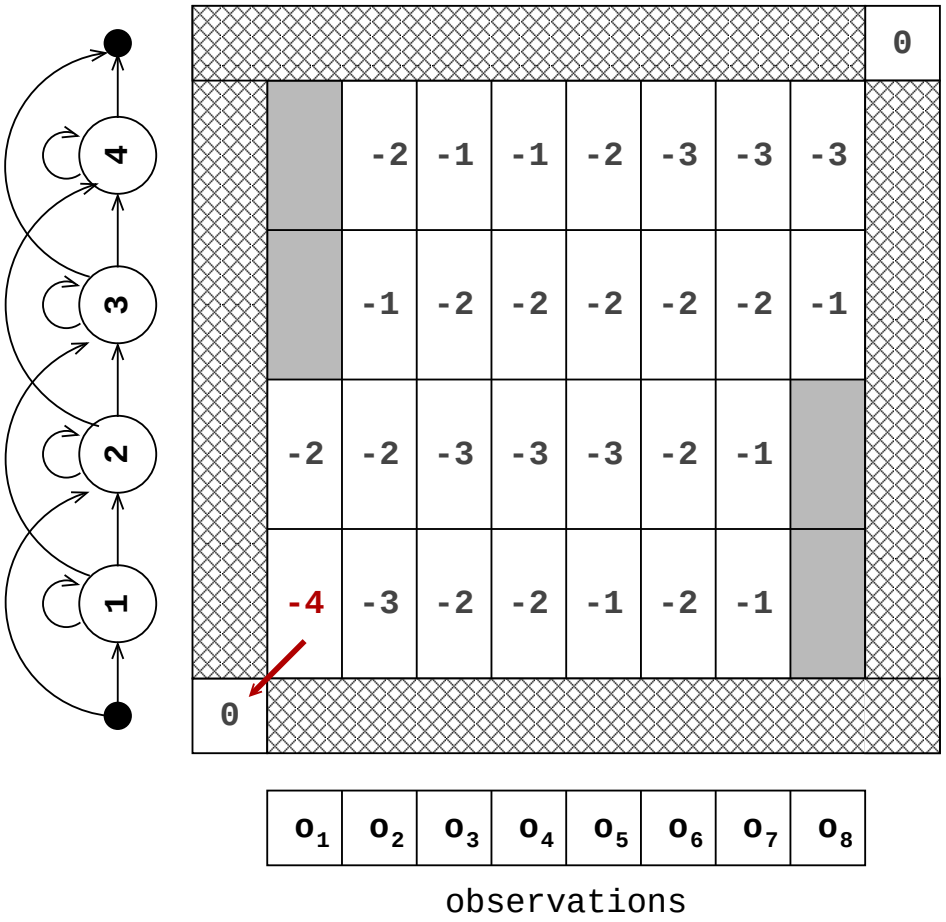


$$H(i,t) = \ln(b_i(o_t))$$

For sake of simplicity, assume

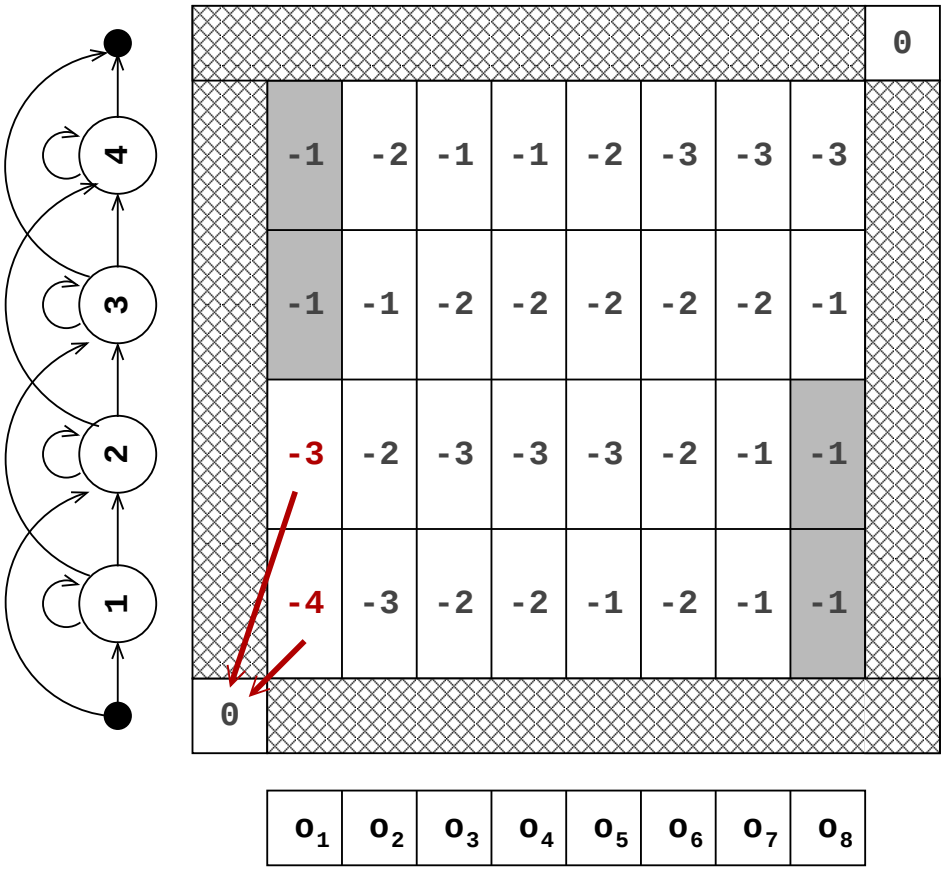
- $\ln(\pi_i) = -1 \; \forall i$
- $\ln(a_{ij}) = -1 \; \forall i, j$

Viterbi at work – step 1



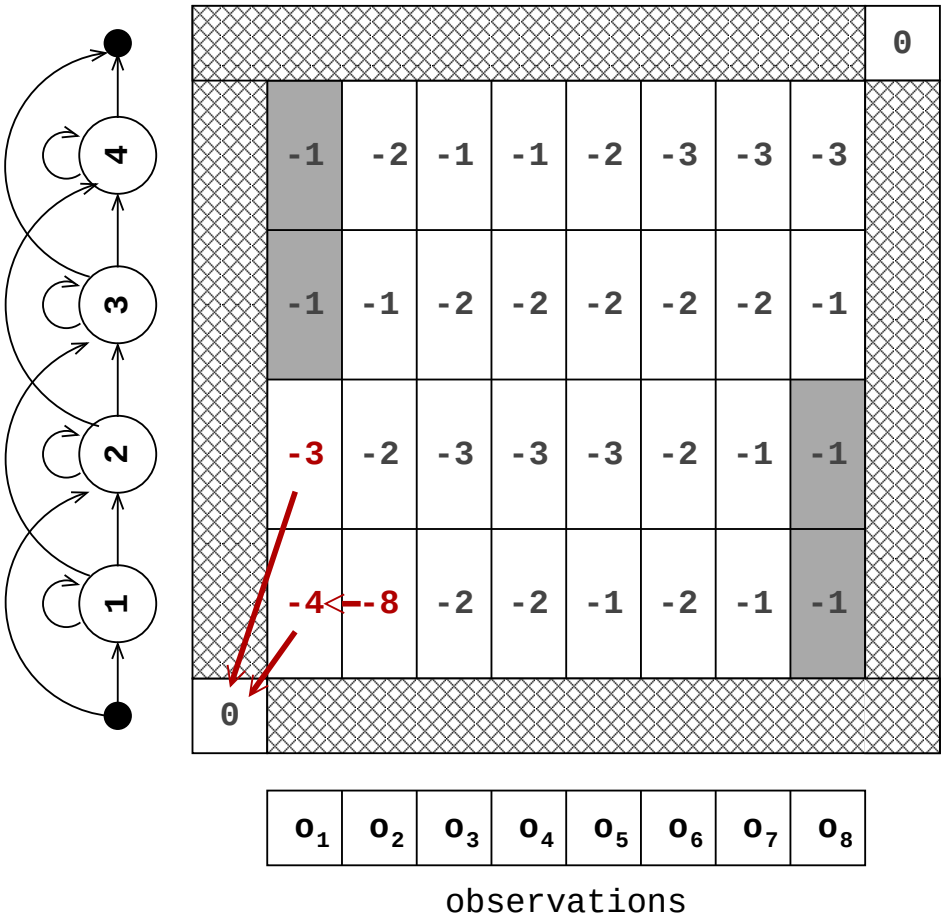
$$
\begin{aligned}
H(1,1) &= \ln b_1(o_1) + \ln(\pi_1) \\
&= -(3+1)
\end{aligned}
$$

Viterbi at work – step 2



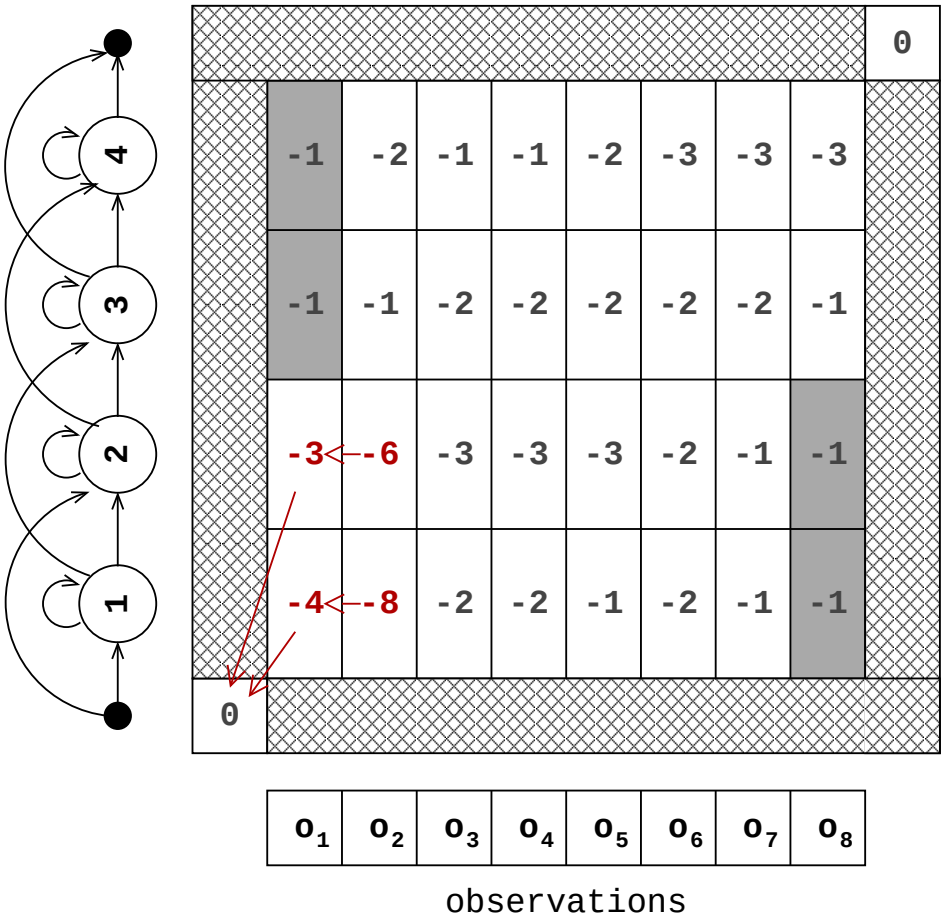
$$
\begin{aligned}
H(2,1) &= \ln b_2(o_1) + \ln(\pi_2) \\
&= -(2+1)
\end{aligned}
$$

Viterbi at work – step 3



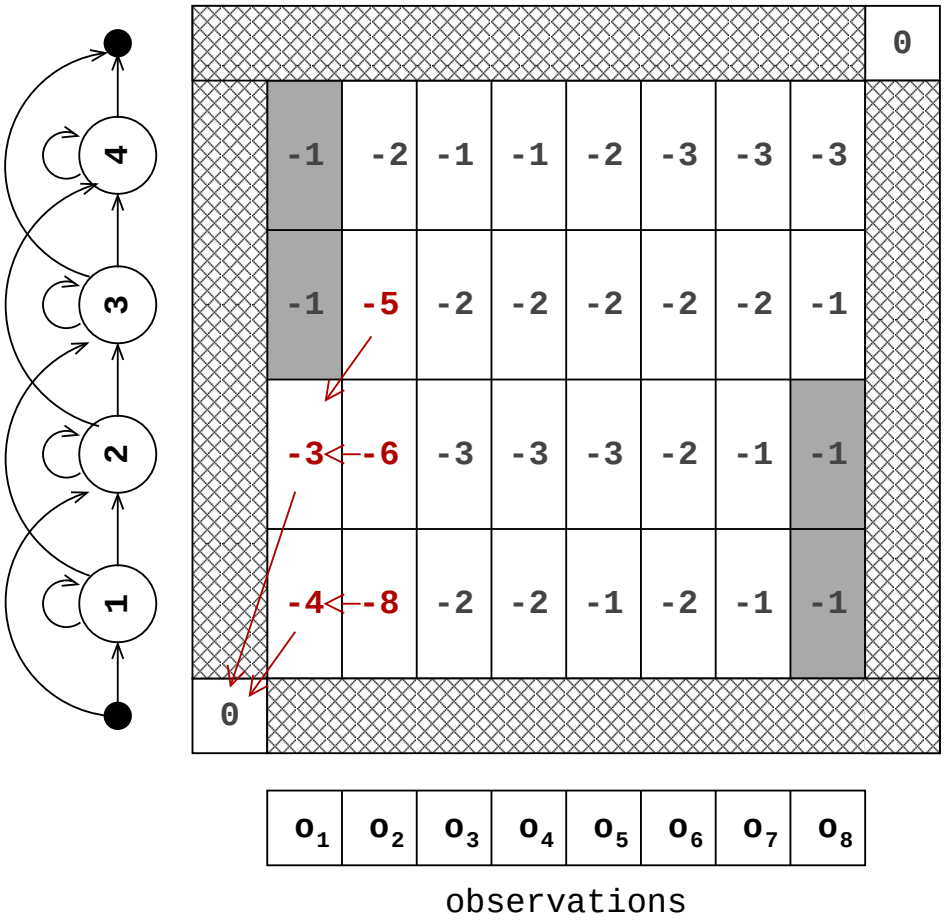
$$
\begin{aligned}
H(1,2) &= \ln b_1(o_2) + \ln(a_{11}) + H(1,1) \\
&= -(3+1+4)
\end{aligned}
$$

Viterbi at work – step 4



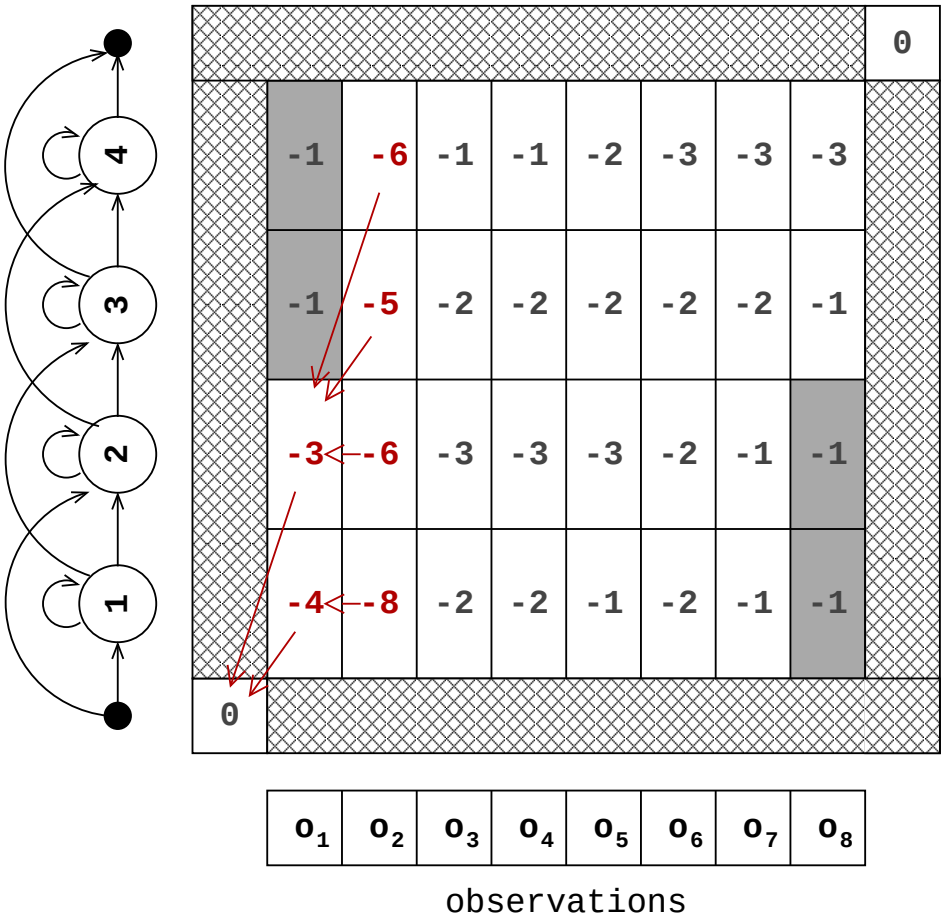
$$H(2,2) = \max \left\{ \begin{array}{l} \ln b_2(o_2) + \ln(a_{22}) + H(2,1) \\ \ln b_2(o_2) + \ln(a_{12}) + H(1,1) \end{array} \right.$$

Viterbi at work – step 5



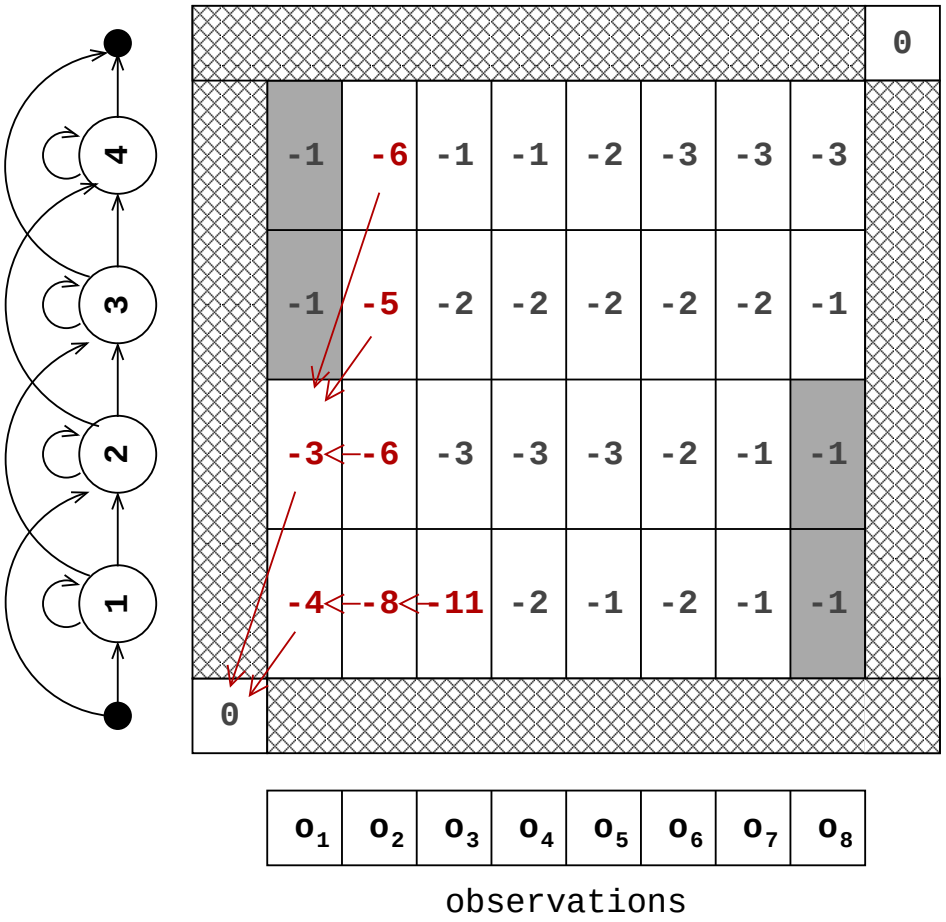
$$H(3,2) = \max \left\{ \begin{array}{l} \ln b_3(o_2) + \ln(a_{13}) + H(1,1) \\ \ln b_3(o_2) + \ln(a_{23}) + H(2,1) \end{array} \right.$$

Viterbi at work – step 6



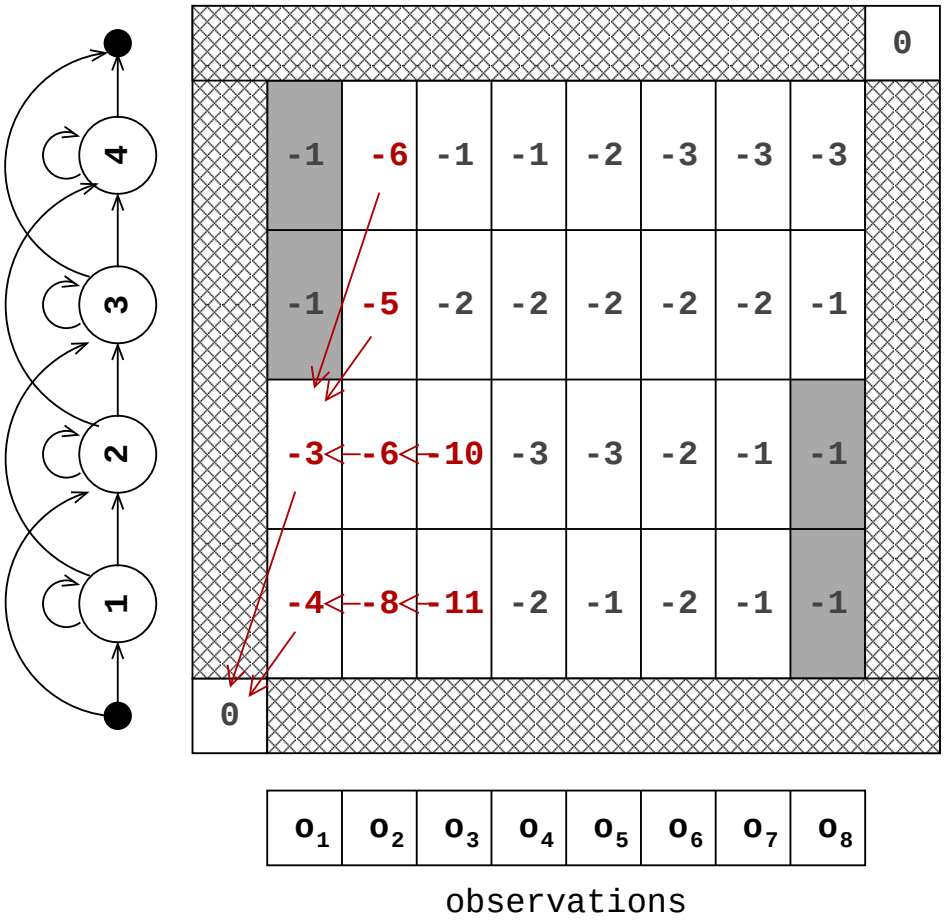
$$H(4,2) = \ln b_4(o_2) + \ln(a_{24}) + H(2,1)$$

Viterbi at work – step 7



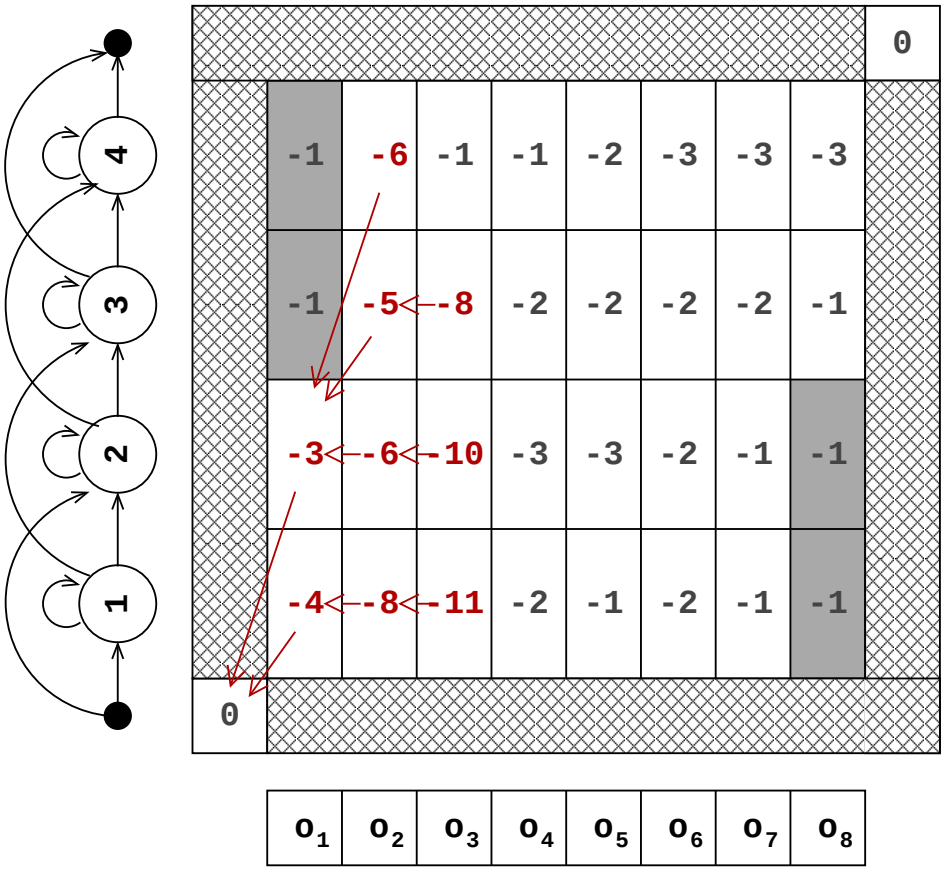
$$H(1,3) = \ln b_1(o_3) + \ln(a_{11}) + H(1,2)$$

Viterbi at work – step 8



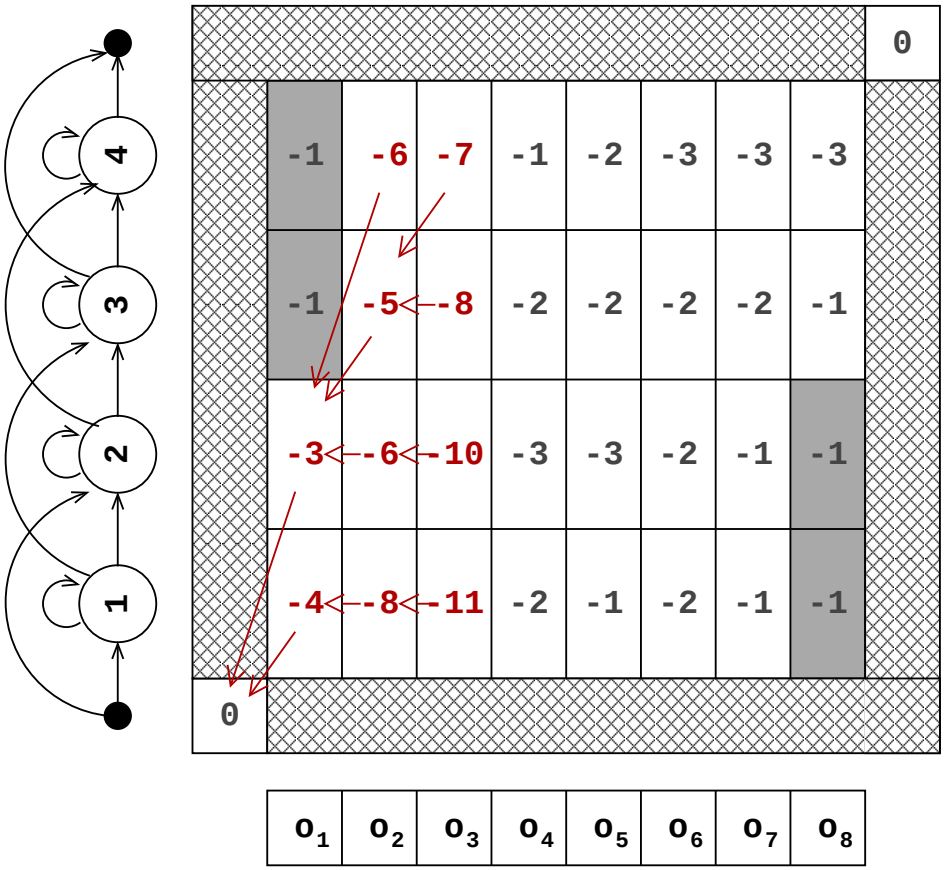
$$H(2,3) = \max \begin{cases} \ln b_2(o_3) + \ln(a_{22}) + H(2,2) \\ \ln b_2(o_3) + \ln(a_{12}) + H(1,2) \end{cases}$$

Viterbi at work – step 9



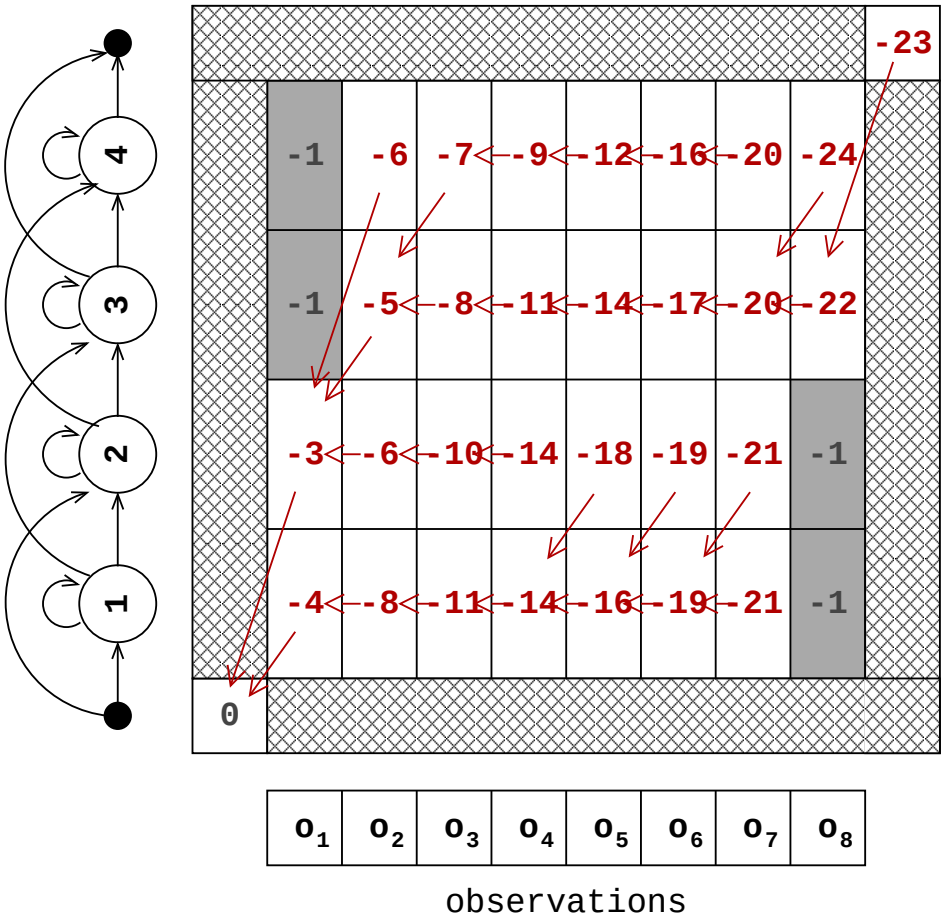
$$H(3,3) = \max \begin{cases} \ln b_3(o_3) + \ln(a_{33}) + H(3,2) \\ \ln b_3(o_3) + \ln(a_{23}) + H(2,2) \\ \ln b_3(o_3) + \ln(a_{13}) + H(1,2) \end{cases}$$

Viterbi at work – step 10



$$H(4,3) = \max \begin{cases} \ln b_4(o_3) + \ln(a_{44}) + H(4,2) \\ \ln b_4(o_3) + \ln(a_{34}) + H(3,2) \\ \ln b_4(o_3) + \ln(a_{24}) + H(2,2) \end{cases}$$

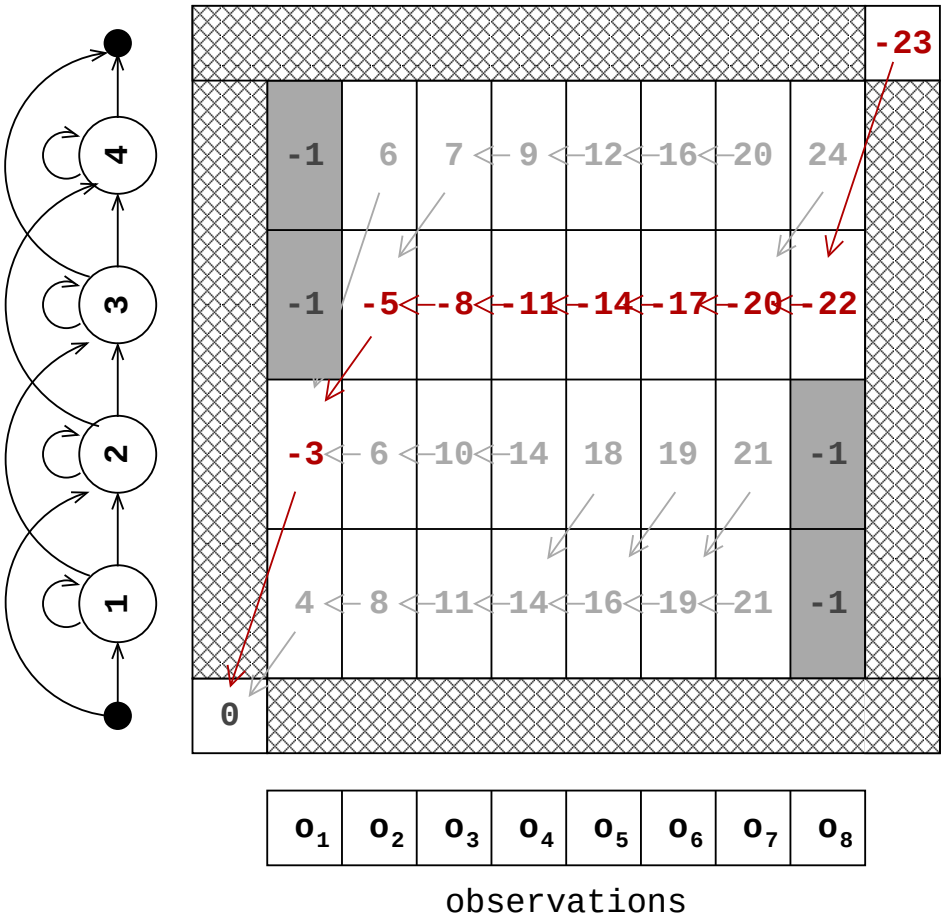
Viterbi at work – finally



$$H^* = -23$$

$$\widehat{s}_8 = 3$$

Viterbi at work – backtracking



Optimal state sequence:

$$S_1 = 2, S_2 = 3, S_3 = 3, S_4 = 3, S_5 = 3, S_6 = 3, S_7 = 3, S_8 = 3$$

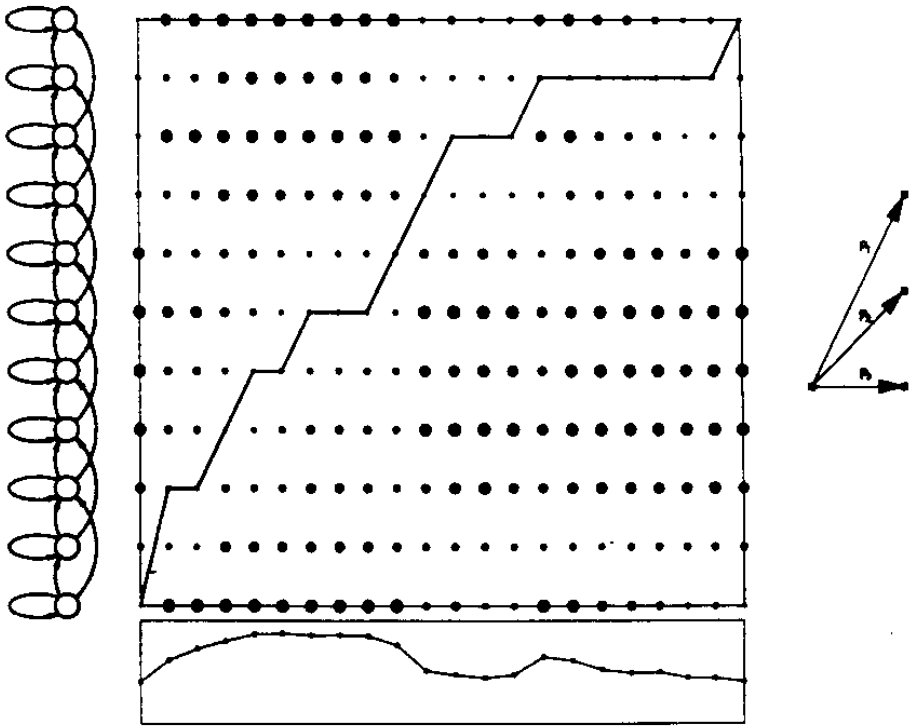
Other application of dynamic programming

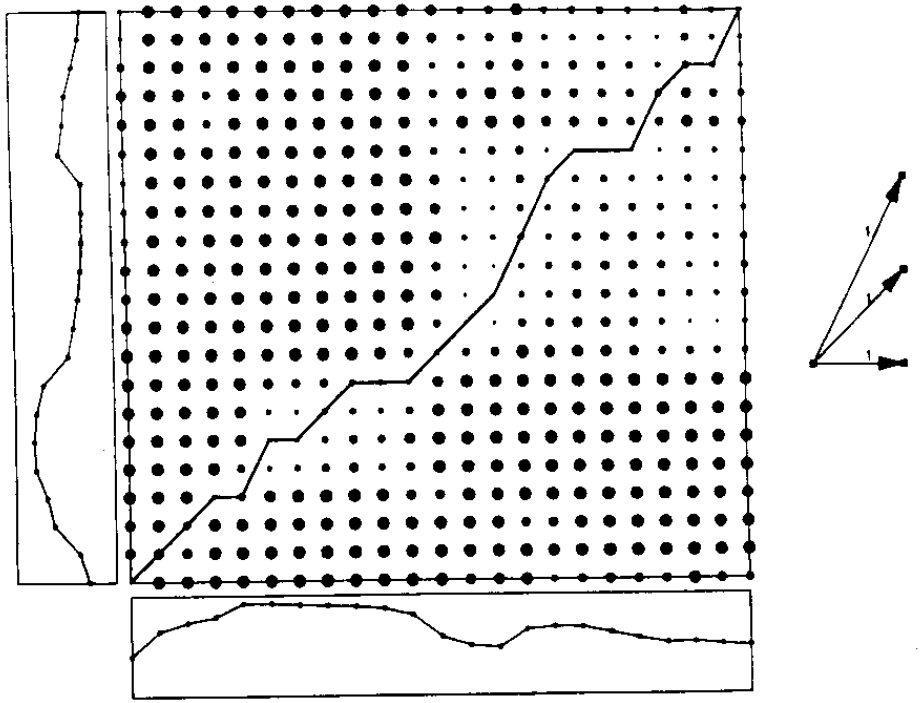
Dynamic programming is a general principle that can be applied for other problems than for finding out the best state sequence.

Examples of problems involving dynamic programming:

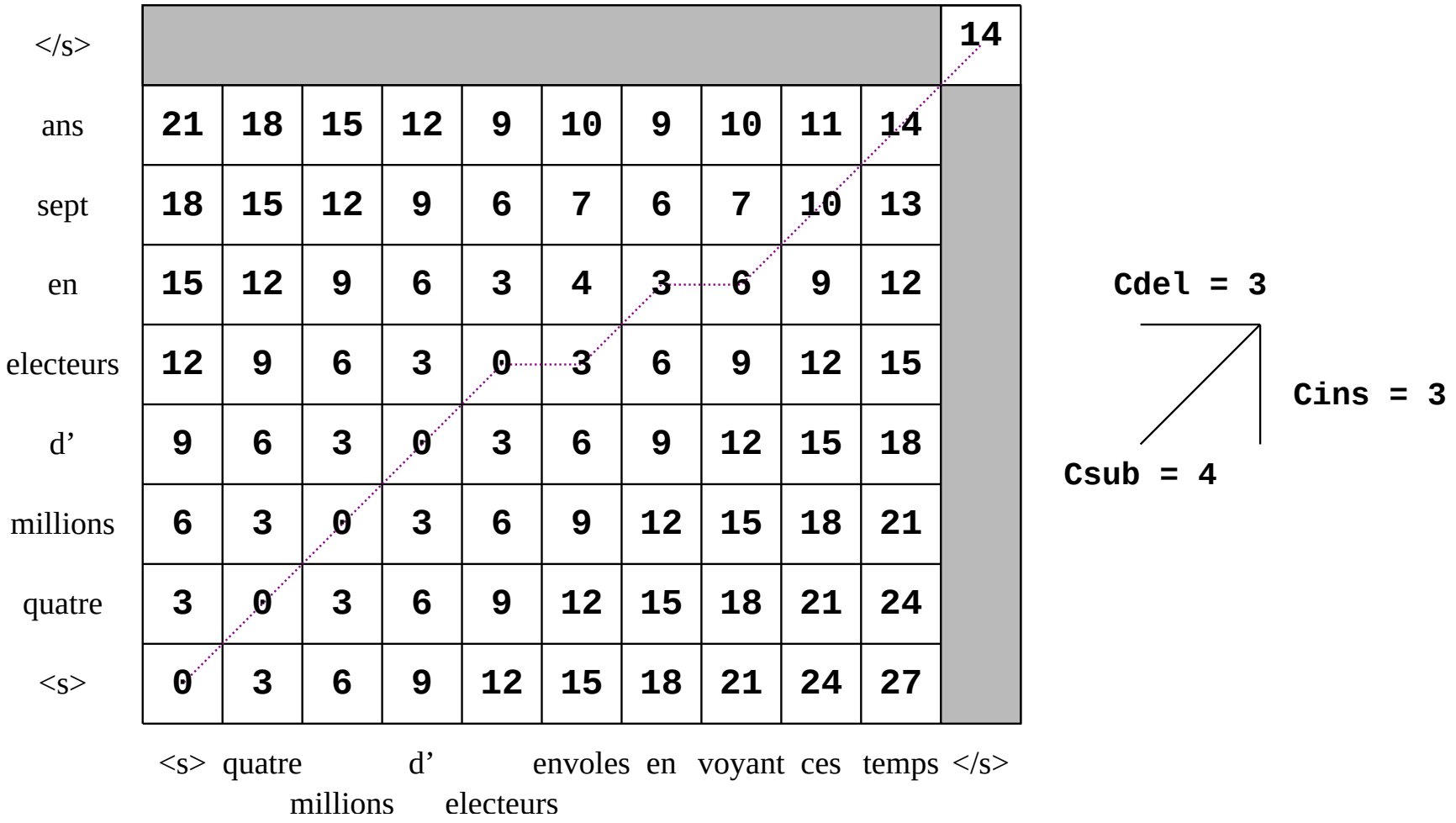
- finding out the **minimum cost path in weighted directed graphs**
  - scan nodes in topological order
  - store the minimum cost partial path up to each node
  - get the minimum over all predecessors
- **Dynamic Time Warping** (DTW)
  - find the optimal mapping between two time series
  - accounts for time stretching and dilatation
- **Edit distance** between strings of words
  - find the optimal mapping between two sentences
  - accounts for deletions, insertions and substitutions

Dynamic time warping



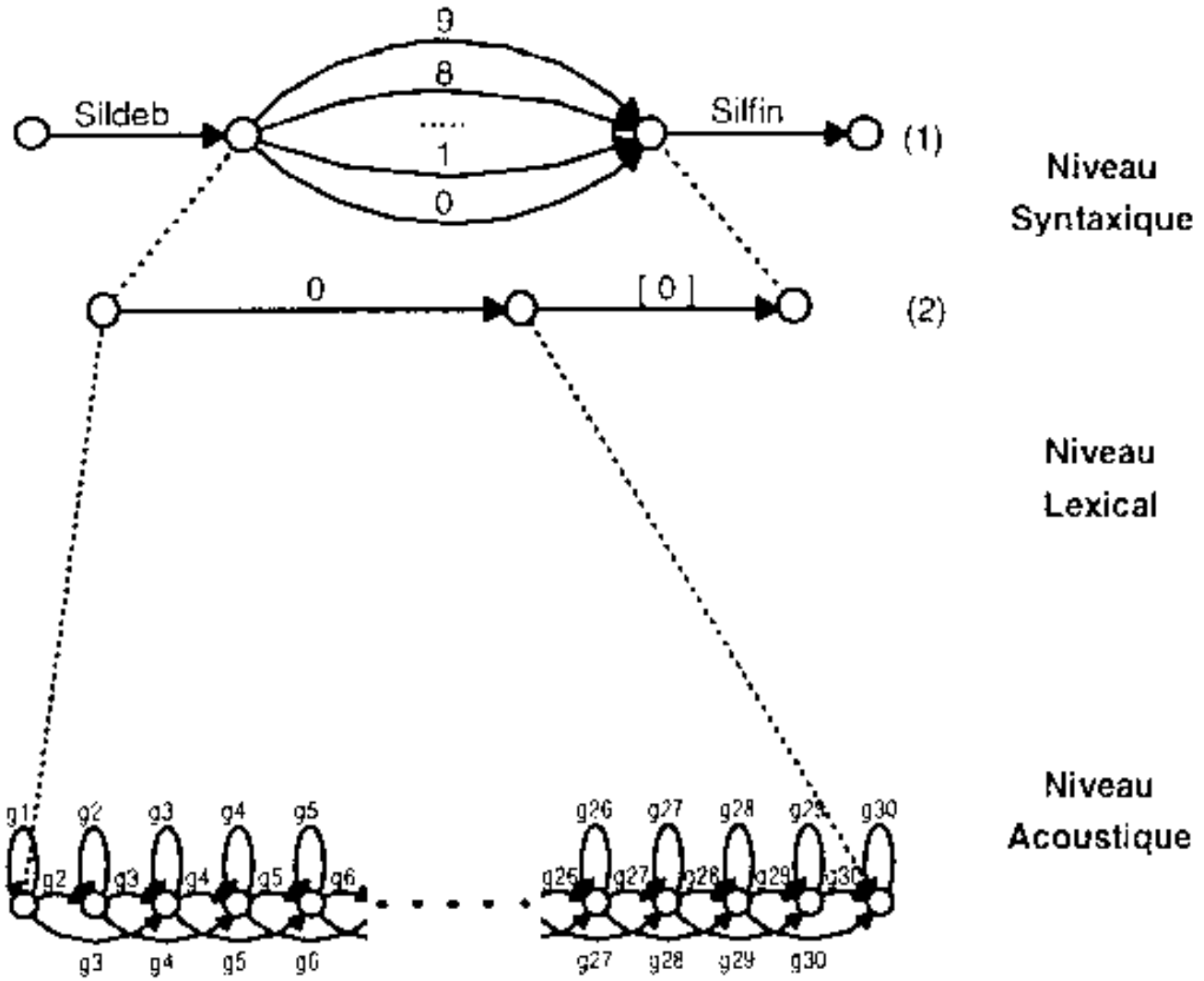


Edit distance (Levenshtein distance)



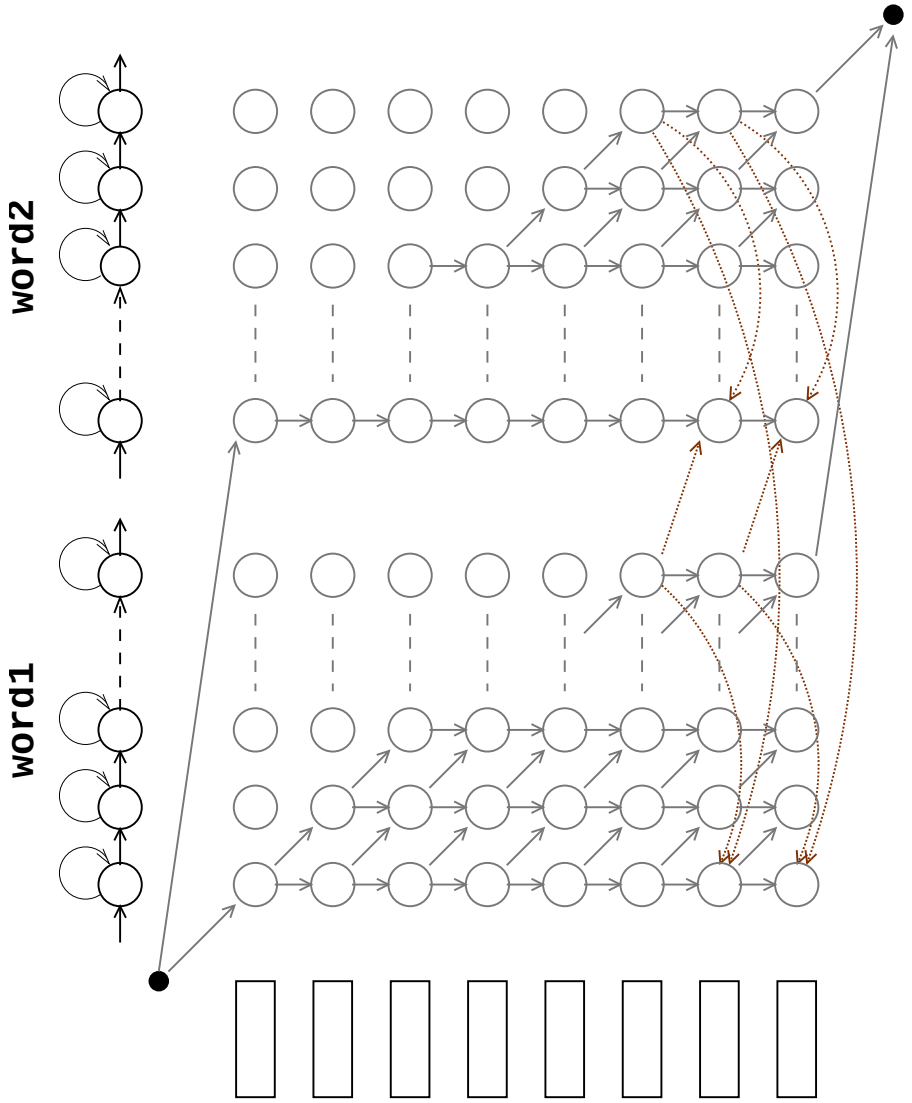

```
REF: quatre millions d' électeurs ******* en ****** sept ans
HYP: quatre millions d' électeurs envolés en voyant ces  temps
```

HMM for sequence classification



Modélisation par mots

HMM for joint segmentation and classification



The three problems [Rabiner 89]

1. **Find out the most likely state sequence**

   Given a model λ_N, how to efficiently compute the state sequence $\mathbf{s} = s_1, \ldots, s_T$ for which the probability of a given observation sequence $\mathbf{o} = o_1, \ldots, o_T$ is maximum.

2. **Evaluate the probability of an observation sequence**

   Given a model λ_N, how to efficiently compute the probability of a given observation sequence $\mathbf{o} = o_1, \ldots, o_T$?

3. **Parameter estimation**

   Given a set of training sequences, how to efficiently estimate the parameters of a model λ_N according to the maximum likelihood criterion.

PARAMETER ESTIMATION

Notations

- Training samples

$$\mathbf{o}^{(r)} = \{o_1^{(r)}, \ldots, o_{n_r}^{(r)}\} \quad \text{for} \quad r \in [1, R]$$

- Parameters to estimate = θ
  - π – initial state distribution
  - A – state transition probabilities
  - B – (discrete) state conditional probabilities

$$\widehat{\theta} = \arg\max_{\theta} \prod_{r=1}^{R} P(\mathbf{o}^{(r)}; \lambda_N(\theta))$$

Empirical estimation

Assuming we know the hidden state sequences $\mathbf{s}^{(r)} = \{s_1^{(r)}, \ldots, s_{n_r}^{(r)}\}$ for each training example, we can use empirical estimators:

$$\widehat{\pi}_i = \frac{\sum_r \mathbb{I}_{(s_1^{(r)}=i)}}{R}$$

$$\widehat{a}_{ij} = \frac{\sum_r \sum_{t=2}^{n_r} \mathbb{I}_{(s_{t-1}^{(r)}=i, s_t^{(r)}=j)}}{\sum_r \sum_{t=1}^{n_r-1} \mathbb{I}_{(s_t^{(r)}=i)}} \qquad \widehat{b}_{ik} = \frac{\sum_r \sum_{t=1}^{n_r} \mathbb{I}_{(s_t^{(r)}=i)} \mathbb{I}_{(o_t^{(r)}=k)}}{\sum_r \sum_{t=1}^{n_r} \mathbb{I}_{(s_t^{(r)}=i)}}$$

For continuous state conditional densities, replace $\widehat{b}_{ik}$ by the adequate empirical estimators based on the obervations assoaciated to state i.

Segmental k-means algorithm

1. Start with some initial values θ_0 for the parameters
2. Find out the best state sequence for each training sample

$$\widehat{\mathbf{s}}^{(r)} = \arg\max_{\mathbf{s}} P[o_1^{(r)}, \ldots, o_T^{(r)} | s_1^{(r)}, \ldots, s_T^{(r)}; \lambda_N(\theta_n)] P[s_1^{(r)}, \ldots, s_T^{(r)}; \lambda_N(\theta_n)]$$

3. Compute new estimates θ_{n+1} of the parameters based on the empirical estimators given the alignments
4. Repeat step 2 and 3 until happy

$\Rightarrow$ EM algorithm to compensate for the missing variables!

The Expectation-Maximization principle

The Expectation-Maximization (EM) principle compensates for missing (aka latent) data, replacing them by their expectations.

EM Iterative principle

1. estimate the missing variables given a current estimate of the parameters
2. estimate new parameters given the current estimate of the missing variables
3. repeat steps 1 and 2 until convergence

Note: this principle applies to many problems, not only for maximum likelihood parameter estimation!

The auxiliary function

The EM algorithm aims at maximizing an auxiliary function defined as

$$Q(\theta, \widehat{\theta}) = E[\ln f(\mathbf{z}, \mathbf{x}; \theta) | \mathbf{x}; \widehat{\theta}]$$

where $f(z, x; \theta)$ is the likelihood function of the complete data.

Estimation step

E compute the expected quantities in $Q(\theta, \widehat{\theta})$ (given $\widehat{\theta} = \theta_n$)

Maximization step

M maximize the auxiliary function w.r.t. the (true) parameters θ (given the expected quantities) to obtain a new estimate $\widehat{\theta} = \theta_{i+1}$, *i.e.*

$$\theta_{i+1} = \arg\max_{\theta} Q(\theta, \theta_i)$$

Auxiliary function for HMMs

For a single sample $\mathbf{o}$, the density of the complete data $\mathbf{o}, \mathbf{s}$ is given by

$$\ln p(\mathbf{o}, \mathbf{s}) = \ln(\pi_{s_1}) + \sum_{t=2}^{T} \ln(a_{s_{t-1} s_t}) + \sum_{t=1}^{T} \ln(b_{s_1}(o_1)) \ ,$$

which can be rewritten as

$$\ln p(\mathbf{o}, \mathbf{s}) = \sum_{i=1}^{N} \ln(\pi_i) \mathbb{I}_{(s_1=i)} + \sum_{t=2}^{T} \sum_{i,j=1}^{N} \ln(a_{ij}) \mathbb{I}_{(s_{t-1}=i, s_t=j)} + \sum_{t=1}^{T} \sum_{i=1}^{N} \ln(b_i(o_k)) \mathbb{I}_{(s_t=i)}$$

and therefore the EM auxiliary function is given by

$$\begin{aligned} Q(\theta, \theta_n) &= \sum_{i=1}^{N} \ln(\pi_i) E[\mathbb{I}_{(s_1=i)} | \mathbf{o}; \theta_n] + \sum_{i,j=1}^{N} \ln(a_{ij}) \left(\sum_{t=2}^{T} E[\mathbb{I}_{(s_{t-1}=i, s_t=j)} | \mathbf{o}; \theta_n] \right) \\ &+ \sum_{i=1}^{N} \sum_{t=1}^{T} \ln(b_i(o_t)) E[\mathbb{I}_{(s_t=i)} | \mathbf{o}; \theta_n] \end{aligned}$$

ML parameter estimation

Respecting the constraints $\sum_i \pi_i = 1, \sum_j a_{ij} = 1$ and $\sum_k b_{ik} = 1$, we have

$$\widehat{\pi}_i = \frac{\sum_r \gamma_1^{(r)}(i)}{R}$$

$$\widehat{a}_{ij} = \frac{\sum_r \sum_{t=2}^{n_r} \xi_t^{(r)}(i,j)}{\sum_r \sum_{t=1}^{n_r - 1} \gamma_t^{(r)}(i)} \qquad \widehat{b}_{ik} = \frac{\sum_r \sum_{t=1}^{n_r} \mathbb{I}_{(o_t^{(r)}=k)}\ \gamma_t^{(r)}(i)}{\sum_r \sum_{t=1}^{n_r} \gamma_t^{(r)}(i)}$$

where

$$\gamma_t^{(r)}(i) = E[\mathbb{I}_{(s_t^{(r)}=i)} | \mathbf{o}; \theta_n] \qquad \xi_t^{(r)}(i,j) = E[\mathbb{I}_{(s_{t-1}^{(r)}=i, s_t^{(r)}=j)} | \mathbf{o}; \theta_n]$$

Computation of the expectations

Problem is now how to efficiently compute the two quantities

$$\begin{aligned}\gamma_t^{(r)}(i) &= E\left[\mathbb{I}_{(s_t=i)}|\mathbf{o};\theta_n\right] \\ &= \text{probability of the event } S_t = i\end{aligned}$$

and

$$\begin{aligned}\xi_t^{(r)}(i,j) &= E\left[\mathbb{I}_{(s_{t-1}^{(r)}=i,s_t^{(r)}=j)}|\mathbf{o};\theta_n\right] \\ &= \text{joint probability of the events } S_{t-1} = i \text{ and } S_t = j\end{aligned}$$

$\Rightarrow$ forward-backward algorithm

Computation of the expectations

The expectations can be expressed as a function of the two forward and backward probabilities:

- **Forward probability**

$$\begin{aligned} \alpha_t(i) &= \text{probability of observing } o_1 \text{ up to } o_t \text{ and being in state } i \text{ at } t \\ &= P[o_1, \ldots, o_t, S_t = i; \lambda_N(\theta_n)] \end{aligned}$$

- **Backward probability**

$$\begin{aligned} \beta_t(i) &= \text{probability of observing } o_{t+1} \text{ up to } o_T \text{ given that we're in state } i \text{ at } t \\ &= P[o_{t+1}, \ldots, o_T | S_t = i; \lambda_N(\theta_n)] \end{aligned}$$

Note that we have

$$P[\mathbf{o}; \lambda_N(\theta)] = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t) \quad \forall t \in [1, T]$$

Computing the expectations (cont'd)

- **State occupancy statistics**

$$
\begin{aligned}
\gamma_t(i) &= P[S_t = i | o_1, \ldots, o_T] \\
&= \frac{P[o_1, \ldots, o_t, S_t = i] P[o_{t+1}, \ldots, o_T | S_t = i]}{P[o_1, \ldots, o_T]} \\
&= \frac{\alpha_i(t)\beta_i(t)}{\sum_{i=1}^{N} \alpha_i(t)\beta_i(t)}
\end{aligned}
$$

Computing the expectations (cont'd)

- **State transition statistics**

$$
\begin{aligned}
\xi_t(i,j) &= P[S_{t-1}=i, S_t=j | o_1, \ldots, o_T] \\
&= \frac{P[o_1^{t-1}, S_{t-1}=i] P[o_t, S_t=j | S_{t-1}=i] P[o_{t+1}^T | S_t=j]}{P[o_1, \ldots, o_T]} \\
&= \frac{\alpha_i(t-1) a_{ij} b_j(o_t) \beta_j(t)}{\sum_{i=1}^{N} \alpha_i(t) \beta_i(t)}
\end{aligned}
$$

Forward algorithm

The forward quantity can be recursively computed according to

$$P[o_1, \ldots, o_t, S_t = j] = \sum_{i=1}^{N} P[o_1, \ldots, o_{t-1}, S_{t-1} = i] P[o_t, S_t = j | S_{t-1} = i]$$

1. **Initialization**: $\alpha_i(1) = \pi_i b_i(o_1) \quad \forall i \in [1, N]$
2. **Recursion** (for $t = 2, \ldots, T$ and $j = 1, \ldots, N$)

$$\alpha_j(t) = b_j(o_t) \sum_{i=1}^{N} \alpha_i(t-1) a_{ij}$$

Note that $P[o_1, \ldots, o_t; \lambda_N(\theta)] = \sum_{i=1}^{N} \alpha_i(T)$ (problem 2).

Backward algorithm

The backward quantity can be recursively computed according to

$$P[o_{t+1}, \ldots, o_T | S_t = i] = \sum_{j=1}^{N} P[o_{t+1}, S_{t+1} = j | S_t = i] P[o_{t+2}, \ldots, o_T | S_{t+1} = j]$$

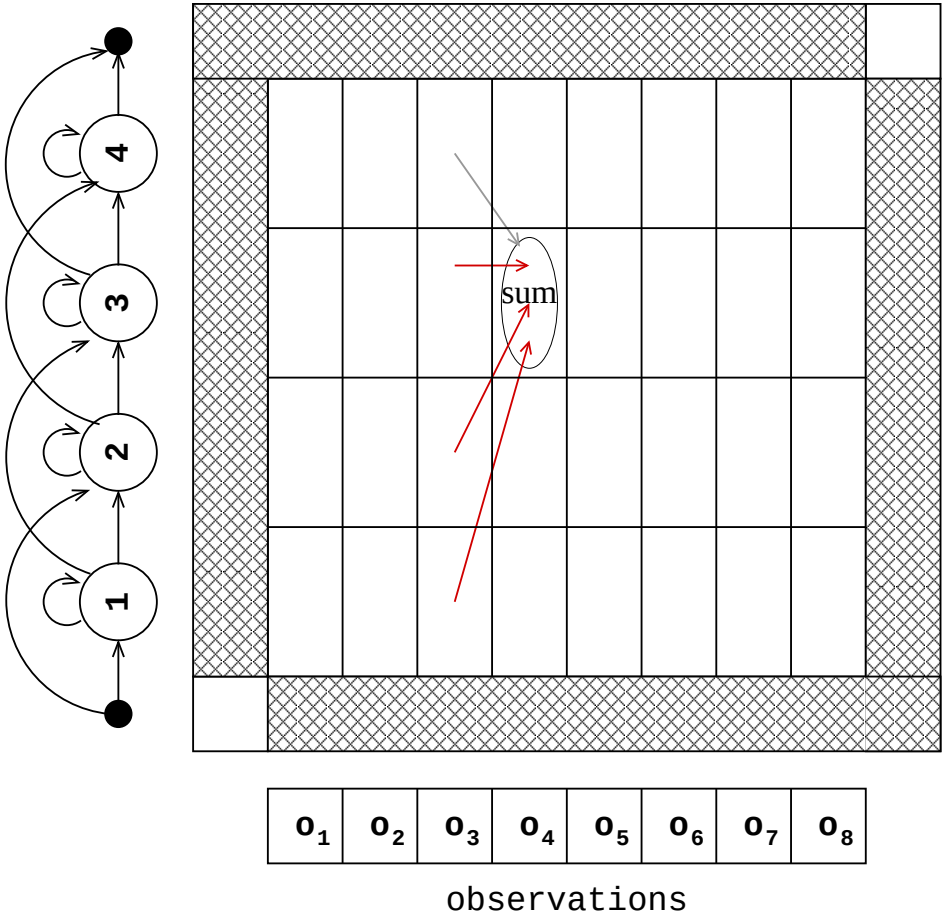
1. **Initialization**: $\beta_i(T) = 1 \quad \forall i \in [1, N]$
2. **Recursion** (for $t = 2, \ldots, T$ and $j = 1, \ldots, N$)

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_j(t+1)$$

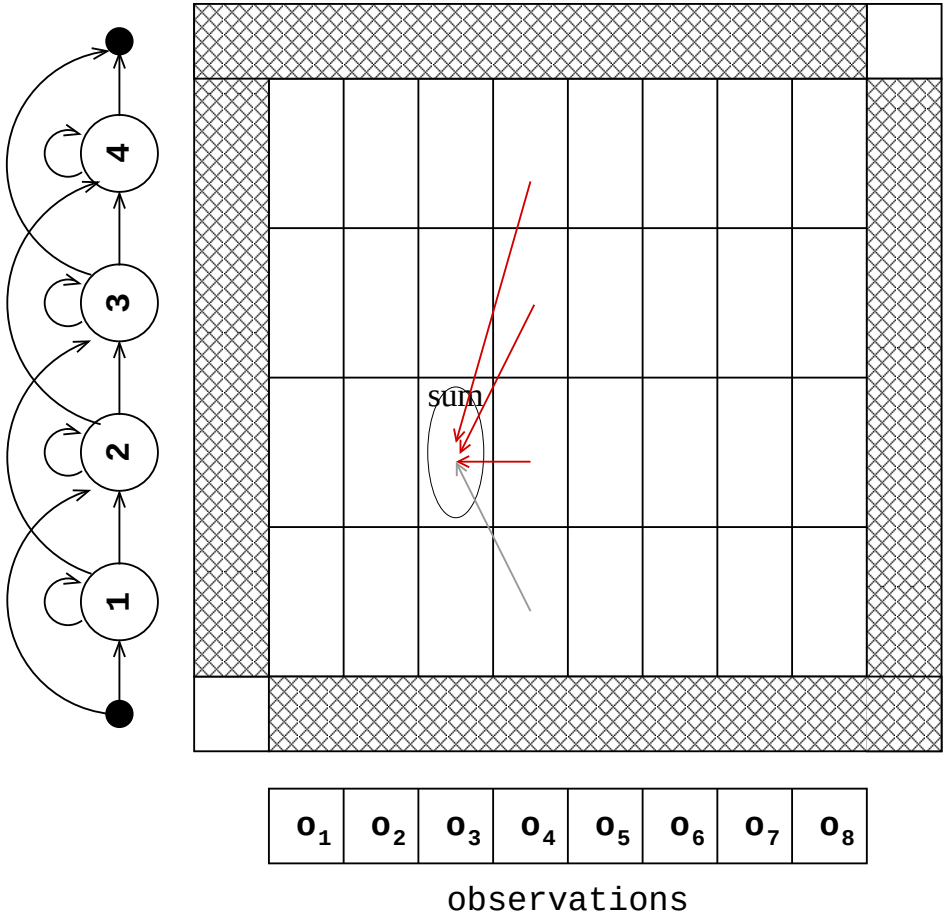
Note that $P[o_1, \ldots, o_t; \lambda_N(\theta)] = \sum_{i=1}^{N} \pi_1 \beta_i(1)$.

Treillis implementation

Forward



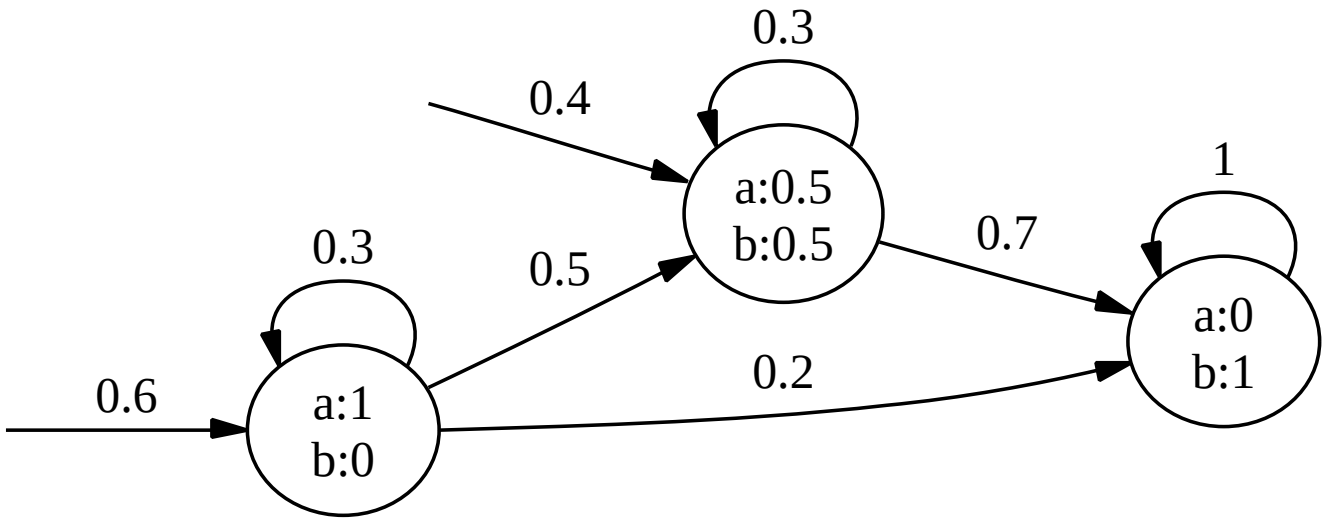
Backward



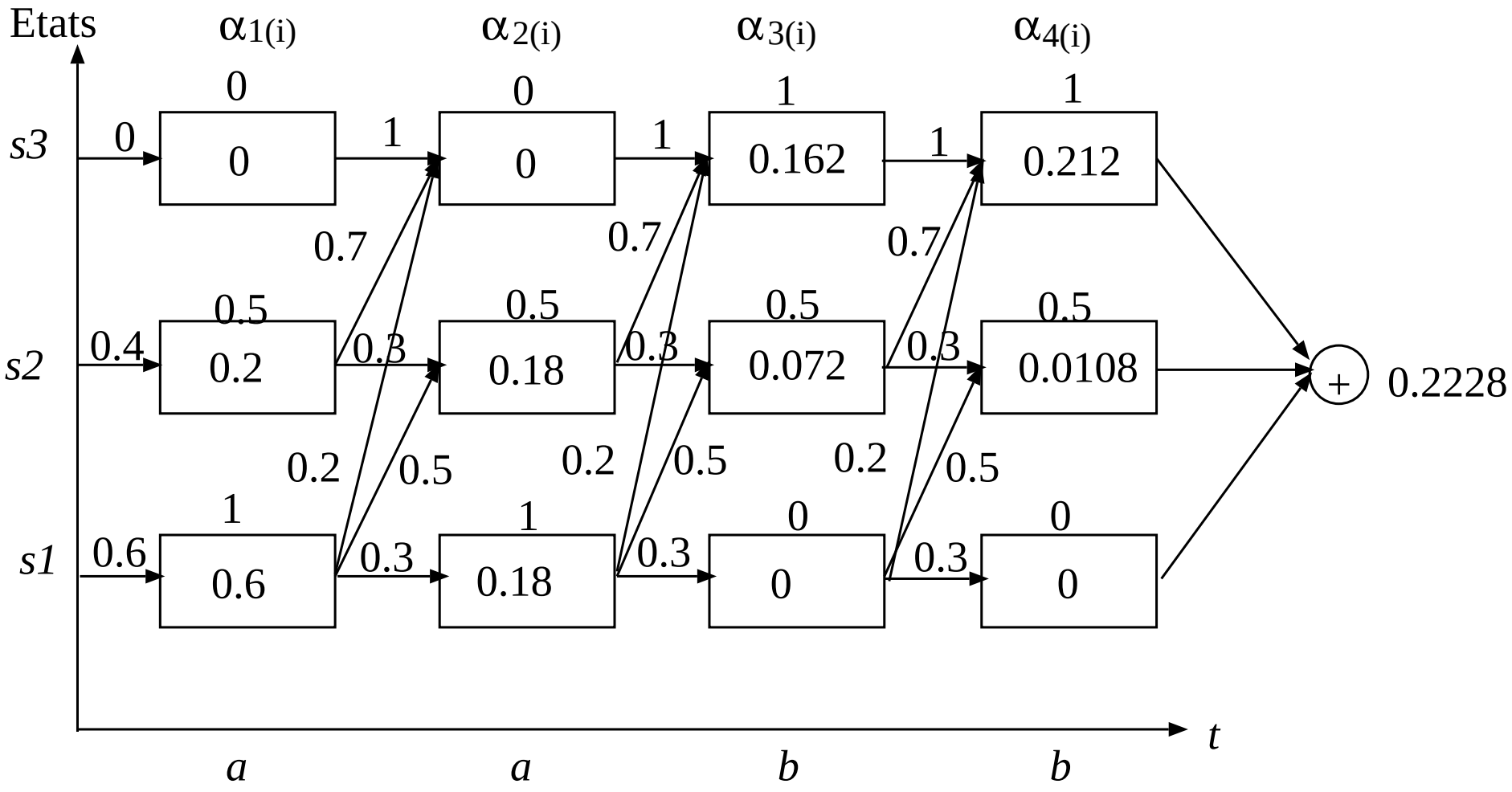
Example of the forward algorithm

Let $\lambda_3 = (A, B, \pi)$ be a 3 state HMM with an alphabet of 2 symbols a and b.

$$A = \begin{pmatrix} 0.3 & 0.5 & 0.2 \\ 0 & 0.3 & 0.7 \\ 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0.5 & 0.5 \\ 0 & 1 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.6 \\ 0.4 \\ 0 \end{pmatrix}$$



Example of the forward algorithm (cont'd)



Example of the forward algorithm (cont'd)

$$
\begin{aligned}
\alpha_1(1) &= \pi_1 b_1(a) = 0.6 \times 1 = 0.6, \\
\alpha_2(1) &= \pi_2 b_2(a) = 0.4 \times 0.5 = 0.2, \\
\alpha_3(1) &= \pi_3 b_3(a) = 0 \times 0 = 0,
\end{aligned}
$$

$$
\alpha_1(2) = (\alpha_1(1)a_{11} + \alpha_2(1)a_{21} + a_{31})b_1(a)
$$

$$
\begin{aligned}
\alpha_3(1) &= (0.6 \times 0.3 + 0.2 \times 0 + 0 \times 0) \times 1 \\
&= (0.18) \times 1 = 0.18,
\end{aligned}
$$

$$
\begin{aligned}
\alpha_2(2) &= (\alpha_1(1)a_{12} + \alpha_2(1)a_{22} + \alpha_3(1)a_{32})b_2(a) \\
&= (0.6 \times 0.5 + 0.2 \times 0.3 + 0 \times 0) \times 0.5 \\
&= (0.36) \times 0.5 = 0.18.
\end{aligned}
$$

$\ldots$ $\ldots$

EM algorithm for HMM

[also known as the Baum-Welsh algorithm]

1. Start with some initial values θ_0 for the parameters
2. For each training sample
   (a) compute the forward and backward quantities, $\alpha_i(t)$ and $\beta_i(t)$
   (b) compute the sufficient statistics $\gamma_t(i)$ and $\xi_t(i,j)$
3. Compute new estimates of the parameters θ_{n+1}
4. Repeat until happy

The EM at work

Starting with $\lambda_3(\theta_0)$ with the following parameters

$$A = \begin{pmatrix} 0.45 & 0.35 & 0.20 \\ 0.10 & 0.50 & 0.40 \\ 0.15 & 0.25 & 0.60 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.5 & 0.5 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.5 \\ 0.3 \\ 0.2 \end{pmatrix}$$

and a single training sample

$$P(a\ b\ b\ a\ a; \lambda_3(\theta_0)) = 0.0278$$

The EM at work (2)

The Baum-Welsh algorithm increases the likelihood of the training data.

After 1 iteration, we have:

$$A = \begin{pmatrix} 0.346 & 0.365 & 0.289 \\ 0.159 & 0.514 & 0.327 \\ 0.377 & 0.259 & 0.364 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.631 & 0.369 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 0.656 \\ 0.344 \\ 0.0 \end{pmatrix}$$

$$P(a\ b\ b\ a\ a; \lambda_3(\theta_1)) = 0.0529$$

The EM algorithm (3)

After 15 iterations, we have

$$A = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.212 & 0.788 & 0.0 \\ 0.0 & 0.515 & 0.485 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.969 & 0.031 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

which gives

$$P(a\ b\ b\ a\ a; \lambda_3(\theta_{15})) = 0.2474$$

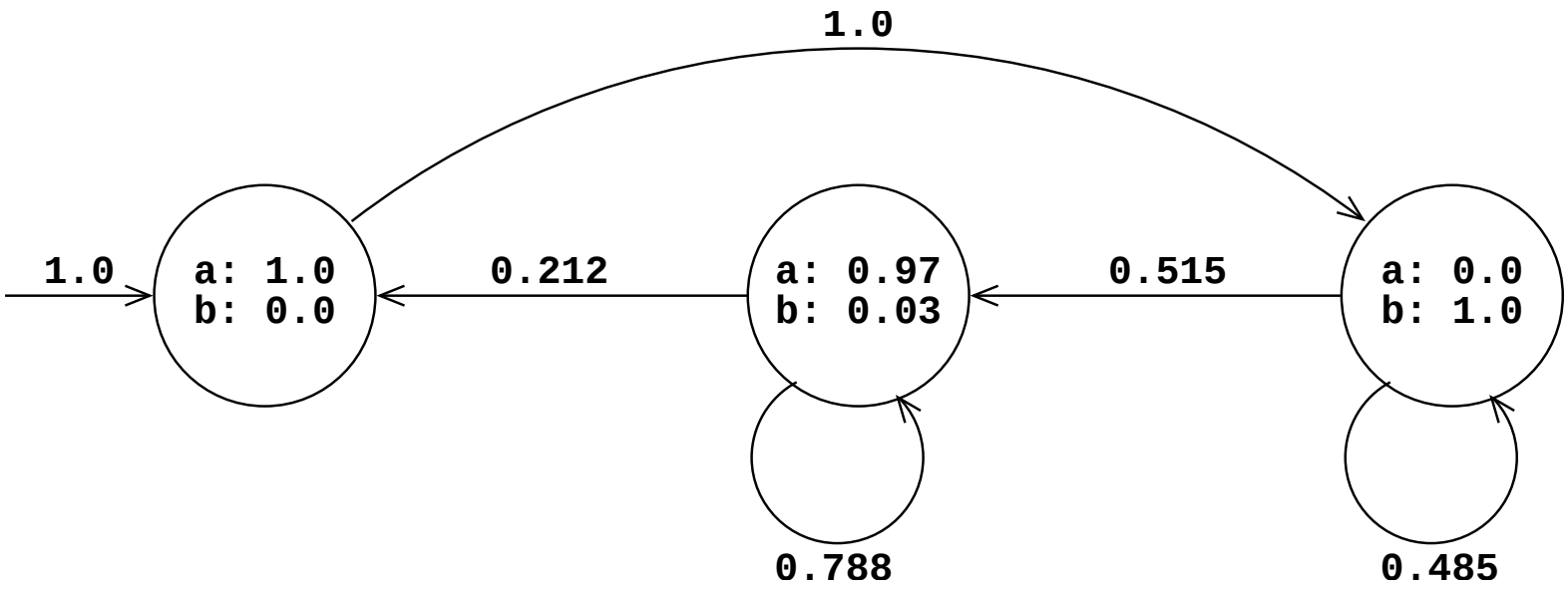
The EM algorithm (4)

After 150 iterations, we have

$$A = \begin{pmatrix} 0.0 & 0.0 & 1.0 \\ 0.18 & 0.82 & 0.0 \\ 0.0 & 0.5 & 0.5 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

which gives

$$P(a\ b\ b\ a\ a; \lambda_3(\theta_{150})) = 0.25$$



The EM algorithm (5)

In the case of an initial HMM with 5 states, the EM estimation converges towards

$$A = \begin{pmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \quad B = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \end{pmatrix} \quad \pi = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}$$

which gives

$$P(a\ b\ b\ a\ a; \lambda_5(\pi, A, B)) = 1.0$$

Mean estimators with continous densities

Gaussian density

$$\mu_i = \frac{\sum_{t=1}^{T} \gamma_t(i)\ o_t}{\sum_{t=1}^{T} \gamma_t(i)}$$

Gaussian mixture

$$\mu_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)\ o_t}{\sum_{t=1}^{T} \gamma_t(i,j)}$$

with

$$\gamma_t(i,j) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{i=1}^{N} \alpha_i(t)\beta_i(t)} \quad \frac{w_{ij}\mathcal{N}(o_t;\mu_{ij},\sigma_{ij})}{\sum_{k=1}^{K} w_{ik}\mathcal{N}(o_t;\mu_{ik},\sigma_{ik})}$$

Bibliography

- A. A. Markov. An exemple of statistical investigation in the test of ’Eugene Onyegin’ illustrating coupling of tests in chains. Proc. Acad. Sci. St. Petersburg, 7:153–162, 1913.
- A. J. Viterbi. Error bounds for convolutional codes and asymptotically optimal decoding algorithm. IEEE Information Theory 13:260–269, April, 1967.
- L. R. Bahl and F. Jelinek. Decoding for channels with insertions, deletions and subtsitutions with applications to speech recognition. IEEE Information Theory, 21:404–411, July, 1975.
- L. Rabiner and B.-H. Juang. Fundamentals of speech recognition. Prentice Hall, 1993.
- Frederick Jelinek. Statistical Methods for Speech Recognition. MIT Press, 1998.