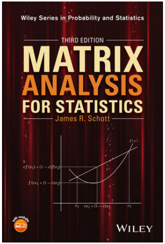
Forward-backward algorithm with matrix calculus

Alain Franc

INRAE BioGeCo & INRIA Pleiade

INCA project, Masemo, Paris, July 1-3, 2025



Hidden Markov Chain: notations

Dependencies between hidden (•) and observed (∘) variables

$$\begin{tikzcd}
y_t \arrow[r] & {} & \circ & \circ & \circ & \circ \\
z_t \arrow[r] & {} & \bullet \arrow[u, "B"'] \arrow[r, "A"] & \bullet \arrow[u, "B"'] \arrow[r, "A"] & \bullet \arrow[u, "B"'] \arrow[r, "A"] & \bullet \arrow[u, "B"'] \\
& & t=0 & t=1 & t=2 & t=3
\end{tikzcd}$$

$$A[z,z'] = \mathbb{P}(Z_t = z \mid Z_{t-1} = z'); \quad B[y,z] = \mathbb{P}(Y_t = y \mid Z_t = z)$$

Joint law of hidden and observed variables: $p(\boldsymbol{z}, \boldsymbol{y})$

$$p(\boldsymbol{z}, \boldsymbol{y}) = \mathbb{P}(\boldsymbol{Z} = \boldsymbol{z}, \boldsymbol{Y} = \boldsymbol{y}) = \left[\prod_{t=T}^{1} B(y_t, z_t)\, A(z_t, z_{t-1})\right] B(y_0, z_0)\, \pi_0(z_0)$$

Main issues for HMM (Rabiner, 1986)

knowing	compute or infer	algorithm
observations, parameters	probability of the observations	fw (-bw)
- id -	smoothing, filtering	fw-bw
- id -	most likely hidden states	Viterbi ($\simeq$ fw)
observations	parameters	EM

Objective of this presentation

- ◇ provide a unified automatic approach (instead of case by case)
- ◇ to write a fw and fw-bw algorithm, compute their complexity
- ◇ for a diversity of HMM based models:
  - multichains,
  - semi-Markov,
  - their combinations.

A probability distribution as a tensor

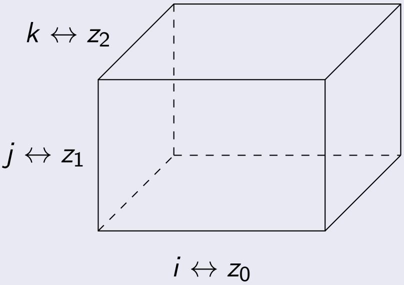
Distributions

$p(z_0, \ldots, z_T) \quad z_t \in \mathcal{K} \quad \mathcal{K} = \{1, \ldots, K\} \subset \mathbb{N} \quad \Omega_Z = \mathcal{K}^{T+1}$

Tensors as multi-ways arrays ($T = 2$)

$$p \longrightarrow \mathcal{A} \in \mathbb{R}^{K \times \ldots \times K} = \left(\mathbb{R}^K\right)^{\otimes (T+1)}$$

$$\mathcal{A} \in \mathbb{R}^{K \otimes K \otimes K}$$



$$p(z_0, z_1, z_2) \updownarrow \mathcal{A}[i, j, k]$$

New object: Un-normalized Heterogeneous Markov based Distribution

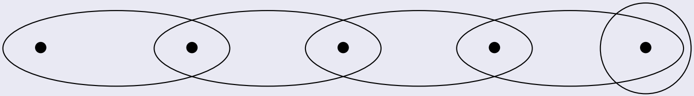
Definition: it is a tensor denoted $\mathcal{A}$

given $T \in \mathbb{N}^*$
$K \in \mathbb{N}^*$
$\pi_0 \in \mathbb{R}_+^K$
$A_1, \ldots, A_T \in \mathbb{R}^{K \times K}$

define $\mathcal{A} \in (\mathbb{R}^K)^{\otimes (T+1)}$

by $\mathcal{A}[z_0, \ldots, z_T] = A_T[z_T, z_{T-1}] \ldots A_1[z_1, z_0]\, \pi_0[z_0]$

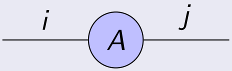
It is a pairwise graphical model along a line (ellipses are factors)



Why tensors? $\rightarrow$ Tensor networks, coupling by contractions

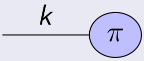
Nodes of the network

matrix



$(a_{ij})_{i,j} \in \mathbb{R}^{K \times K}$

vector

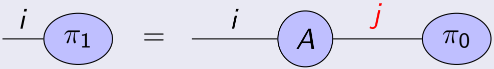


$(\pi_k)_k \in \mathbb{R}^K$

scalar

$\alpha \in \mathbb{R}$

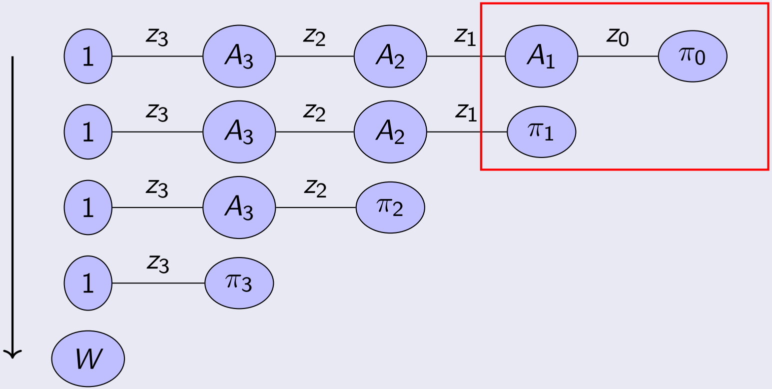
Matrix $\times$ vector product: i: free, j: summed upon (contracted)



$$\pi_1[i] = \sum_{j} A[i, j]\, \pi_0[j]$$

Computation of the normalizing constant

Variable elimination (visual)



Extensions

- on trees ("message passing")
- Dynamic Bayesian Networks ? Graphical models with low tree width?

Elimination as an algorithm on an UHMD

Objective

compute $W = \sum_{z_0=1}^{K} \ldots \sum_{z_T=1}^{K} \mathcal{A}[z_0, \ldots, z_T]$, $\qquad K^{T+1}$ sums

---

Algorithm 1 Elimination algorithm elim() on an UHMD

1: **input** $T, K, (A_t)_{1 \leq t \leq T}$ with $A_t \in \mathbb{R}^{K \times K}$, $\pi_0 \in \mathbb{R}^K$
2: **for** $t \in \{1, \ldots, T\}$ **do**
3: $\quad$ **compute** $\pi_t = A_t \, \pi_{t-1}$ $\qquad$ with $\qquad \pi_t[z] = \sum_{z'=1}^{K} A_t[z, z'] \pi_{t-1}[z']$
4: **end for**
5: **compute** $W = \sum_{z \in \mathcal{K}} \pi_T[z]$
6: **return** W

Complexity (as number of products)

lines 2 (loop on t) and 3 (matrix $\times$ vector product) $\implies \mathcal{O}(TK^2)$

What is behind: why selecting linear algebra

Main tool: matrix $\times$ vector product

The complexity of a matrix $\times$ vector product is on $\mathcal{O}(K^2)$

$$y = Ax \qquad \begin{bmatrix} x_1 \\ \vdots \\ x_K \end{bmatrix}$$

K^2 products

$$i \rightarrow \quad \begin{bmatrix} a_{i1} & \cdots & a_{iK} \end{bmatrix} \quad \begin{bmatrix} y_i \end{bmatrix}$$

$$y_i = \sum_{j=1}^{K} a_{ij} x_j$$

Sparsity of transition matrix

A is a $K \times K$ matrix

Observation

if $\#A$ is the number of terms $a_{ij} \notin \{-1, 0, 1\}$

then $y = Ax$ has complexity $\mathcal{O}(\#A)$

Simple example with $K = 3$

$$A = \begin{pmatrix} a & 0 & 1 \\ -1 & 0 & b \\ c & 1 & 0 \end{pmatrix}, \quad x = \begin{pmatrix} u \\ v \\ w \end{pmatrix} \quad \Longrightarrow Ax = \begin{pmatrix} au + w \\ -u + bw \\ cu + v \end{pmatrix}$$

Complexity in $\mathcal{O}(\#A)$

$y = Ax$ costs 3 multiplications instead of 9: $\mathcal{O}(\#A)$ instead of $\mathcal{O}(K^2)$

One among the three Rabiner's problem

Compute the probability of the observations knowing the parameters

$$p(\boldsymbol{y}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}, \boldsymbol{y})$$

Joint law as an UHMD

$\boldsymbol{y}$ fixed	$\Longrightarrow$ define $A_{t}[z_t, z_{t-1}] = B[y_t, z_t]\, A[z_t, z_{t-1}]$
joint law	$p(\boldsymbol{z}, \boldsymbol{y}) = \left(\prod_{t=T}^{1} A_t[z_t, z_{t-1}]\right) \pi'_0[z_0]$
define	UHMD $\mathcal{A}_{hmm}[z_0, \ldots, z_T] = A_T[z_T, z_{T-1}] \ldots A_1[z_1, z_0]\, \pi'_0[z_0]$

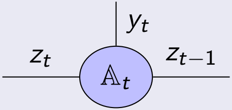
Proposition

- Algorithm elim() on the UHMD $\mathcal{A}_{hmm}$ permits to compute $p(\boldsymbol{y}) = W(\mathcal{A}_{hmm})$
- $\Longrightarrow$ The complexity of the calculation is in $\mathcal{O}(TK^2)$

Probabiity of the observations as a tensor train

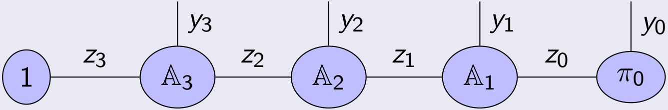
Define

$$\mathbb{A}_t[z_t, y_t, z_{t-1}] = B[y_t, z_t]\, A[z_t, z_{t-1}], \qquad \mathbb{A}_t \in \mathbb{R}^{K \times O \times K}$$



Probability of the observations as a tensor network: tensor train

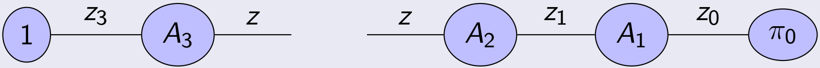
$p(\boldsymbol{y}) =$

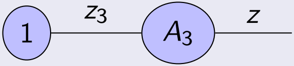


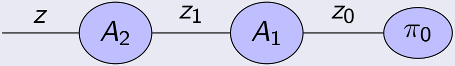
Unary marginals (visual with tensor networks)

$$t = 2, \quad z_2 = z, \qquad \mathfrak{m}(t,z) = \sum_{z_0, z_1, z_3} A[z_3, {\color{red}z}]\, A[{\color{red}z}, z_1]\, A[z_1, z_0]\, \pi_0[z_0]$$

Variable separation



$\alpha_t[z] =$ 

$\beta_t[z] =$ 

$$\mathfrak{m}(z,t) = \alpha_t[z]\, \beta_t[z]$$

Marginal's calculation for an UHMD (smoothing)

Definition: slicing for a tensor

$$\mathfrak{m}(t, z) = \sum_{\boldsymbol{z}\,:\,z_t = z} \mathcal{A}[\boldsymbol{z}]$$

Visualization of the algorithm (forward-backward)

$$\begin{array}{ccccccccc}
\alpha_0 & \xrightarrow{\times A_1} & \alpha_1 & \xrightarrow{\times A_2} & \alpha_2 & \xrightarrow{\times A_3} & \alpha_3 & \xrightarrow{\times A_4} & \alpha_4 \\
\odot & & \odot & & \odot & & \odot & & \odot \\
\beta_4 & \xleftarrow[\times A_4^T]{} & \beta_3 & \xleftarrow[\times A_3^T]{} & \beta_2 & \xleftarrow[\times A_2^T]{} & \beta_1 & \xleftarrow[\times A_1^T]{} & \beta_0 \\
= & & = & & = & & = & & = \\
\mathfrak{m}(0) & & \mathfrak{m}(1) & & \mathfrak{m}(2) & & \mathfrak{m}(3) & & \mathfrak{m}(4)
\end{array}$$

$\alpha_0 \xrightarrow{\times A_1} \alpha_1$ means that $\alpha_1 = A_1\alpha_0$; $\odot$ is the Hadamard product.
$\mathfrak{m}(0) = \alpha_0 \odot \beta_4, \ \ldots, \qquad \alpha_0 = \pi_0, \qquad \beta_0 = (1, \ldots, 1)$

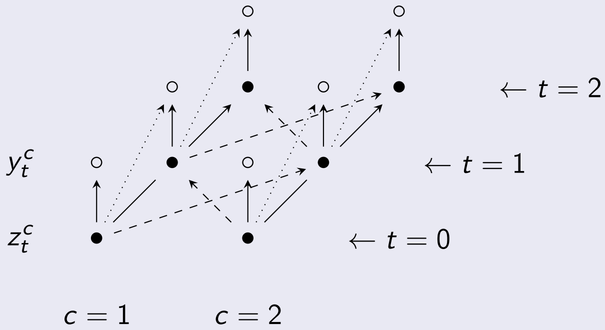
Pseudo-code for marginalisation: complexity in $\mathcal{O}(TK^2)$

Algorithm 2 Marginalisation algorithm margin() in an UHMD

```
input  T, K, (A_t)_{1≤t≤T} with A_t ∈ ℝ^{K×K}, π_0 ∈ ℝ^K
set α_0 = π_0
for t ∈ {1, ..., T} do
    compute α_t = A_t α_{t−1},        (matrix × vector product, in O(K^2))
end for
set β_T = 1
for t ∈ {T − 1, ..., 0} do
    compute β_t = A_{t+1}^T β_{t+1}        (matrix × vector product, in O(K^2))
end for
for t ∈ {0, ..., T} do
    compute m_t = α_t ⊙ β_{T−t}
    return  m_t
end for
```

Extension to multichain and richer dependency structures

An example



General Multichain HMM

Definition

- hidden variables $\boldsymbol{Z}_t \in \mathcal{K}^C$ C chains
- observed variables $\boldsymbol{Y}_t \in \mathcal{S}^O$ O observations

and $(\boldsymbol{Z}_t, \boldsymbol{Y}_t)_t$ is a Markov chain

$$A_G[(\boldsymbol{z}, \boldsymbol{y}); (\boldsymbol{z}', \boldsymbol{y}')] = \mathbb{P}(\boldsymbol{Z}_t = \boldsymbol{z}, \boldsymbol{Y}_t = \boldsymbol{y} \mid \boldsymbol{Z}_{t-1} = \boldsymbol{z}', \boldsymbol{Y}_{t-1} = \boldsymbol{y}')$$

Joint law

$$p(\boldsymbol{z}_{0:T}, \boldsymbol{y}_{0:T}) = A_G[(\boldsymbol{z}_T, \boldsymbol{y}_T); (\boldsymbol{z}_{T-1}, \boldsymbol{y}_{T-1})] \times \ldots \times$$
$$A_G[(\boldsymbol{z}_1, \boldsymbol{y}_1); (\boldsymbol{z}_0, \boldsymbol{y}_0)] \times \pi'_0[z_0, y_0]$$

Emission and Transition

$$A_G[(\boldsymbol{z}, \boldsymbol{y}); (\boldsymbol{z}', \boldsymbol{y}')] = \underbrace{B[\boldsymbol{y}, (\boldsymbol{y}', \boldsymbol{z}, \boldsymbol{z}')]}_{\text{emission}} \times \underbrace{A[\boldsymbol{z}, (\boldsymbol{z}', \boldsymbol{y}')]}_{\text{transition}}$$

General approach for multichains

Reminder

$$A_G[(\boldsymbol{z},\boldsymbol{y});(\boldsymbol{z}',\boldsymbol{y}')] = \underbrace{B[\boldsymbol{y},(\boldsymbol{y}',\boldsymbol{z},\boldsymbol{z}')]}_{\text{emission}} \times \underbrace{A[\boldsymbol{z},(\boldsymbol{z}',\boldsymbol{y}')]}_{\text{transition}}$$

$\boldsymbol{y}_{0:T}$ being fixed

- define $A_t[\boldsymbol{z}_t,\boldsymbol{z}_{t-1}] = B[\boldsymbol{y}_t,(\boldsymbol{y}_{t-1},\boldsymbol{z}_t,\boldsymbol{z}_{t-1})] \times A[\boldsymbol{z}_t,(\boldsymbol{z}_{t-1},\boldsymbol{y}_{t-1})]$
- Then the joint law is $\pi_0'[\boldsymbol{z}_0]\prod_{t=T}^{1} A_t[\boldsymbol{z}_t,\boldsymbol{z}_{t-1}]$
- define an UHMD $\mathcal{A}_G$ with $\mathcal{A}[\boldsymbol{z}_{0:T}] = p(\boldsymbol{z}_{0:T},\boldsymbol{y}_{0:T})$
- run elimination and marginalisation algorithm on it.

This leads, in a systematic and unified way, to:

- how to write the fw and fw-bw algorithm
- compute their complexity

an example: 1to1-MHMM-CI model

Structure

•	A HMM model	HMM
•	multichain	M
•	with conditional independence	CI
•	bijection between hidden and observed variables	1to1

Conditional independence (simplified)

$$p(z_t^1, z_t^2 \mid z_{t-1}^1, z_{t-1}^2) = p_1(z_t^1 \mid z_{t-1}^1, z_{t-1}^2) \times p_2(z_t^2 \mid z_{t-1}^1, z_{t-1}^2)$$

written as

$$A[(z_t^1, z_t^2); (z_{t-1}^1, z_{t-1}^2)] = A^1[z_t^1; (z_{t-1}^1, z_{t-1}^2)] \times A^2[z_t^2; (z_{t-1}^1, z_{t-1}^2)]$$

leading to

$$A[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}] = \prod_{c=1}^{C} A^c[z_t^c; \boldsymbol{z}_{t-1}] \;\Rightarrow\; A_t[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}]$$

Forward algorithm for a 1to1-MHMM-CI

Algorithm 3 Forward algorithm for a 1to1-MHMM-CI

input $T, K, C, \pi_0 \in \mathbb{R}^{K^C}$, $\quad \pi_0[\boldsymbol{z}] = \mathbb{P}(\boldsymbol{Z}_0 = \boldsymbol{z})$

input A_t^c, B_t^c for $1 \leq t \leq T$ and $1 \leq c \leq C$, with $A_t^c, B_t^c \in \mathbb{R}^{K \times K^C}$,

 # preparation

for $t \in \{1, \ldots, T\}$ **do**

  compute $A_t^{ci} \in \mathbb{R}^{K^C \times K^C}$ $\qquad \mathcal{O}(T(C-1)K^{2C})$

  with $A_t^{CI}[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}] = \prod_{c=1}^{C} B_t^c[\boldsymbol{z}_t, \boldsymbol{z}_{t-1}] A_t^c[z_t^c, \boldsymbol{z}_{t-1}]$

end for

 # elimination

compute $W = \mathrm{elim}(T, K^C, (A_t^{CI})_t, \pi_0)$, $\qquad \mathcal{O}(TK^{2C})$

return W

Complexity: preparation + elimination

$\mathcal{O}(T(C-1)K^{2C}) + \mathcal{O}(TK^{2C}) = \mathcal{O}(TCK^{2C})$

A Semi-Markov Model as a Markov Model

SMM (reminder)

SMM: $(J_n, X_{n+1})_n$ is a Markov chain with

- J_n is state z at jump n
- X_{n+1} is sojourn duration is state J_n

$(Z_t, R_t, E_t)_t$ is a MM with

- R_t is the remaining time until next jump
- E_t is the elapsed time since last jump

if $r_t > 0$	no jump	$z \to z$	$r_t \to r_t - 1$	$e_t \to e_t + 1$
if $r_t = 0$	jump	$z \to z' \neq z$	$r_{t+1} = A_J(.. \mid z_t, e_t)$	$e_{t+1} = 1$

HSMM

$\Longrightarrow$ A HSMM can be translated into a HMM

- hidden state space $\mathcal{K} \times \mathcal{D} \times \mathcal{D} \Rightarrow$ transition matrix is $KD^2 \times KD^2$

fw and fw-bw algorithm for a HSMM

naively ...

- Hidden state space has cardinality KD^2
- so elimination algorithm is in $\mathcal{O}(T(KD^2)^2)$

Sparsity

A_{zre}: transition matrix $(z_t, r_t, e_t) \longrightarrow (z_{t+1}, r_{t+1}, e_{t+1})$

- at a jump: $A_{zre}[(z_t, r_t, 1), (z_{t-1}, 0, e_{t-1})] = A_j[(z_t, r_t + 1), (z_{t-1}, e_{t-1})]$
- out of a jump: $A_{zre}[(z_t, r_t, e_t), (z_{t-1}, r_{t-1}, e_{t-1})] \in \{0, 1\}$

$\Rightarrow \quad \#A_{zre} = K^2 D^2$

$\Longrightarrow$ complexity of fw or fw-bw algorithm

$$\mathcal{O}(T(KD)^2) \text{ instead of } \mathcal{O}(T(KD^2)^2)$$

Challenge

How to write an efficient algorithm for computing $\pi_t = A_{zre}\pi_{t-1}$

$A_{zr} := A_{zre}$ on an example: ED-HMM, $K = 3$, $D = 3$

$. := 0; \qquad \times :\neq 0, 1; \qquad ij := (z = i \in \{1,2,3\}, r = j \in \{0,1,2\})$

	10	11	12	20	21	22	30	31	32
10	.	1	.	$\times$	.	.	$\times$	.	.
11	.	.	1	$\times$	.	.	$\times$	.	.
12	.	.	.	$\times$	.	.	$\times$	.	.
20	$\times$	.	.	.	1	.	$\times$	. .	
21	$\times$	.	.	.	.	1	$\times$	.	.
22	$\times$	.	.	.	.	.	$\times$	.	.
30	$\times$	.	.	$\times$	.	.	.	1	.
31	$\times$	.	.	$\times$	.	.	.	.	1
32	$\times$	.	.	$\times$	.	.	.	.	.

$$\left.\begin{array}{l} KD \text{ rows} \\ \text{per row: } K-1 \text{ terms } \notin \{0,1\} \end{array}\right\} \Longrightarrow K(K-1)D \text{ products}$$

$\Longrightarrow$ Complexity in $\mathcal{O}(TK^2D)$

Multichain HSMM

Setting: J-MHSMM

Interactions between chains occur at jumps • only

For one chain c

$$
\begin{array}{ccccccccccccc}
 & t & & t+1 & & t+2 & & t+3 & & t+4 & & t+5 & \\
\cdots & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \cdots \\
 & z & & z' \neq z & & z' & & z' & & z' & & z'' \neq z' & \\
 & r = 0 & & r' = 3 & & r' = 2 & & r' = 1 & & r' = 0 & & r'' = \ldots & \\
 & e = \ldots & & e' = 1 & & e' = 2 & & e' = 3 & & e' = 4 & & e'' = 1 &
\end{array}
$$

- ◇ At time $t+1$, (z', r') depends on the states $z_{t+1}^{c'}$ of other chains
- ◇ At times $t+2, t+3, t+4$, the transitions are compulsory $(z', r', e') \longrightarrow (z', r'-1, e'+1)$

$\Longrightarrow$ the transition matrix is sparse

Forward algorithm for a J-MHSMM

States

$(\boldsymbol{z}_t, \boldsymbol{r}_t, \boldsymbol{e}_t)_t$ with $\boldsymbol{z}_t = (z_t^1, \dots, z_t^C)$, $\boldsymbol{r}_t = (r_t^1, \dots, r_t^C)$ and $\boldsymbol{e}_t = (e_t^1, \dots, e_t^C)$

Transition matrix

$A[(\boldsymbol{z}_t, \boldsymbol{r}_t, \boldsymbol{e}_t), (\boldsymbol{z}_{t-1}, \boldsymbol{r}_{t-1}; \boldsymbol{e}_{t-1})] = p(\boldsymbol{z}_t, \boldsymbol{r}_t, \boldsymbol{e}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{r}_{t-1}, \boldsymbol{e}_{t-1})$
$z_t^c \in \mathcal{K}$ and $|\mathcal{K}| = K$
$r_t^c \in \mathcal{D}$ and $|\mathcal{D}| = D$
$e_t^c \in \mathcal{D}^*$ and $|\mathcal{D}^*| = D$
$\Longrightarrow A$ is a $(KD^2)^C \times (KD^2)^C$ matrix

Complexity

- $\diamond$ expected complexity $\mathcal{O}(T(K^2 D^4)^C)$
- $\diamond$ because of sparsity $\mathcal{O}(T(K^2 D^2)^C)$
- $\diamond$ for a J-M-ED-HMM $\mathcal{O}(T(K^2 D)^C)$

UHMD on semi-rings

Observation

- Matrix $\times$ vector product ($y_i = \sum_{j=1}^{K} a_{ij}x_j$) does not imply any inversion ($x \rightarrow \frac{1}{x}$)

$\Rightarrow$ it can be developped on semi-rings

Semi-ring ($\mathbb{R} \cup \{-\infty\}$, max, $\times$)

- $+ \mapsto \max$
- $\times \mapsto \times$

$$a + 0 = a \mapsto \max(a, -\infty) = a$$

$$a(b + c) = ab + ac \mapsto a(\max(b, c)) = \max(ab, ac)$$

$W(\mathcal{A})$ of an UHMD $\mathcal{A}$ on a semi-ring

On $\mathbb{R}$, $\quad W(\mathcal{A}) = \sum_{\mathbf{z} \in \Omega_Z} \mathcal{A}[z_0, \ldots, z_T]$

On semi-ring ($\mathbb{R} \cup \{-\infty\}$, max, $\times$) $\quad W(\mathcal{A}) = \max_{\mathbf{z} \in \Omega_Z} \mathcal{A}[z_0, \ldots, z_T]$

Matrix product $y = A \odot x$ in $(\mathbb{R} \cup \{-\infty\}, \max, \times)$

$$y = A \odot x \quad \Longrightarrow \quad y_i = \bigoplus_{j=1}^{K} (a_{ij} \odot x_j) = \max_{1 \leq j \leq K} a_{ij} x_j$$

$$y = A \odot x \qquad \boxed{\begin{matrix} x_1 \\ \vdots \\ x_K \end{matrix}}$$

K^2 products

$$i \rightarrow \quad \boxed{\begin{matrix} a_{i1} & \cdots & a_{iK} \end{matrix}} \quad \boxed{y_i} \qquad y_i = \max_j \{a_{ij} x_j \mid 1 \leq j \leq K\}$$

Complexity

The complexity of computing $y = A \odot x$ is in $\mathcal{O}(K^2)$.

Elimination algorithm in $(\mathbb{R} \cup \{-\infty\}, \max, \times)$

Algorithm 4 Elimination algorithm for an UHMD in $(\mathbb{R} \cup \{-\infty\}, \max, \times)$

1: **input** $T, K, (A_t)_{1 \leq t \leq T}$ with $A_t \in \mathbb{R}^{K \times K}$, $\pi_0 \in \mathbb{R}^K$
2: **for** $t \in \{1, \ldots, T\}$ **do**
3: **compute** $\pi_t = A_t \odot \pi_{t-1}$ with $\pi_t[z] = \max\limits_{1 \leq z' \leq K} A_t[z, z']\pi_{t-1}[z']$
4: **end for**
5: **compute** $W = \max\limits_{1 \leq z \leq K} \pi_T[z]$
6: **return** W

Complexity

Its complexity is in $\mathcal{O}(TK^2)$ as well (lines 2 and 3)

Viterbi algorithm for a HMM

Finding the most likely hidden state of a HMM knowing the observations and the parameters.

How to ...

1. Let $p(\boldsymbol{z}, \boldsymbol{y})$ be the joint law between hidden and observed variables
2. write $p(\boldsymbol{z}, \boldsymbol{y})$ as an UHMD $\mathcal{A}[z_0, \ldots, z_T]$
3. compute $M = \max_{\boldsymbol{z}} \ \mathcal{A}[\boldsymbol{z}]$ with elimination algorithm on $\mathcal{A}$ in $(\mathbb{R} \cup \{-\infty\}, \max, \times)$
4. run the traceback

Can be extended as a procedure to

1. Multichain HMM
2. HSMM
3. Multichain HSMM with interactions at jumps

What can be next?

∃ different models for a MHMM

1. general MHMM
2. MHMM with conditional independence
3. one to one MHMM with CI

∃ different models for a HSMM

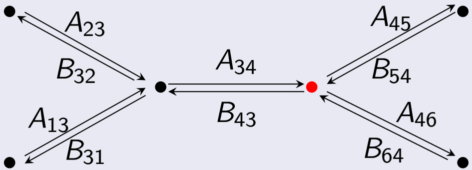
1. general HSMM
2. standard HSMM
3. ED-HMM

What is reachable

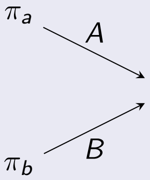
1. develop fw and fw-bw algorithm as a unified procedure for all these models
2. cross together choices for MHMM and for HSMM
3. extend to EM

Message passing on a tree

an example (the hidden layer only is shown)



How to associate incoming messages



$$\pi_c = (A\pi_a) \odot (B\pi_b)$$

Ackowledgements (warm ones ...)

Working groups on monochain HSMM and multichain HMM & HSMM of INCA project

Hanna, Irène, Jean-Baptiste, Nathalie, Nicolas, Régis, Sandra, Vlad

Anwar PhD's thesis

Nathalie, Anwar, Olivier for links between tensor algebra and statistical models (on WSBM) and the use of tensor networks

Last but not least ... Inca project