HMM: Introduction and segmentation

J. Lember

University of Tartu, Estonia

6.12.2019

(Finite) Markov chain

Let S be a finite or countable set – state space; let for every $t = 1, \ldots, n$, Y_t be a random variable taking values in S (random state).

The random variables $Y_1, \ldots, Y_n$ form **Markov chain** when for every $y_i \in S$ and $t = 2, \ldots, n-1$

$$P(Y_{t+1} = y_{t+1} | Y_t = y_t, \ldots, Y_1 = y_1) = P(Y_{t+1} = y_{t+1} | Y_m = y_t)$$

equivalently $\quad P(y_{t+1} | y_t, \ldots, y_1) = P(y_{t+1} | y_t)$

MC is sometimes denoted

$$Y_1 \to Y_2 \to \cdots \to Y_n.$$

Thus $Y_1 \to Y_2 \to \cdots \to Y_n$ iff for every $y_1, \ldots, y_n$ such that $y_t \in S$

$$P(y_1, \ldots, y_n) = P(y_1) p(y_2 | y_1) P(y_3 | y_2) \cdots P(y_n | y_{n-1}).$$

Elementary properties:

1) reversed MC is also a MC:

$$\text{if} \quad Y_1 \to Y_2 \to \cdots \to Y_n, \ \text{then} \quad Y_n \to Y_{n-1} \to \cdots \to Y_1$$

2) Every sub-chain of MC is a MC:

$$\text{if} \quad Y_1 \to Y_2 \to \cdots \to Y_n, \quad \text{then} \quad Y_{n_1} \to Y_{n_2} \to \cdots \to Y_{n_k}.$$

3) If $Y_1 \to Y_2 \to \cdots \to Y_n$, then for every $t < n$ and $y_i \in S_i$

$$P(y_n, \ldots, y_{t+1} | y_t, \ldots, y_1) = P(y_n, \ldots, y_{t+1} | y_t).$$

4) $Y_1 \to \cdots \to Y_n$ iff for every $t = 2, \ldots, n-1$ the random variables $Y_1, \ldots, Y_{t-1}$ and $Y_{t+1}, \ldots, Y_n$ are conditionally independent given Y_t: for every $y_t \in S_t$,

$$P(y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_n | y_t) = P(y_1, \ldots, y_{t-1} | y_t) P(y_{t+1}, \ldots, y_n | y_t).$$

Examples: 1) Independent random variables $Y_1, \ldots, Y_n$.
2) Simple random walk. Let $\xi_1, \ldots, \xi_n$ be independent tosses of fair coin, i.e. $P(\xi_i = -1) = P(\xi_i = +1) = 0.5$, let

$$Y_1 = \xi_1, \quad Y_2 = \xi_1 + \xi_2, \ldots, Y_k = \sum_{i=1}^{k} \xi_i, \quad k = 1, \ldots, n.$$

In general, sums of independent random variables is MC. Here $S = \mathbb{Z}$.

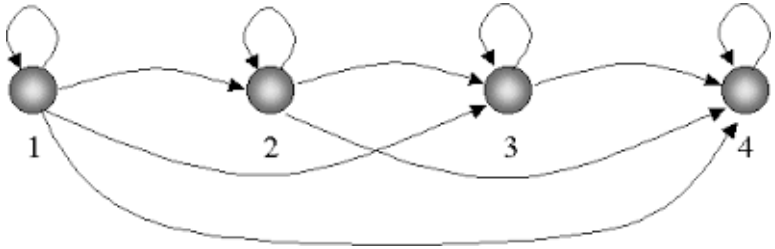
3) Polya urn. In a urn, there is m_b black balls and m_w white balls. Let $Y_0 = m_b$. Then take a random ball and return the ball along with another ball with the same color. Let Y_1 be now the number of black balls; repeat the procedure etc. Here $S = \mathbb{N}$.

4) Birth-death processes. Y_t is the size of population at time t, every time t there is a birth and death probability p_t and q_t:

$$P(Y_{t+1} = k+1 | Y_t = k) = p_t, \quad P(Y_{t+1} = k-1 | Y_t = k) = q_t,$$
$$P(Y_{t+1} = k | Y_i = k) = 1 - q_t - p_t.$$

Polya urn is a birth-process

5) Left-right MC. Let $a_0, a_1, \ldots, a_m$ be states (letters) and if chain is at a_k, then it can: stay or jump to the next or second next (given it exists) state. Models duration from a_0 to a_m. Used in speech recognition for modelling the length of a phonem.



6) Bayesian model. Random parameter θ (prior probability), data X and a statistic $T(X)$ (function of data). Thus $\theta \to X \to T(X)$.

Homogenous MC

For any time t, the transition matrix $\mathbb{P}_t$ is $|S| \times |S|$-matrix with entries being the transition probabilities

$$p_t(i,j) := P(Y_{t+1} = s_j | Y_t = s_i).$$

MC is homogeneous if transition probabilities do not depend on t, i.e. $\mathbb{P}_t = \mathbb{P}$ for every t. In this case, we speak about transition matrix of Markov chain.

Examples: Polya urn. Not homogenuous MC.

Simple random walk. Homogeneous MC, $S = \mathbb{Z}$ and

$$p(i,j) = \left\{ \begin{array}{ll} \frac{1}{2}, & \text{if } j = i+1; \\ \frac{1}{2}, & \text{if } j = i-1; \\ 0, & \text{else;} \end{array} \right\} \quad \mathbb{P} = \left(\begin{array}{cccccccc} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 0 & 1/2 & 0 & 1/2 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1/2 & 0 & /2 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 1/2 & 0 & 1/2 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{array} \right)$$

Two state MC. $S = \{0, 1\}$, transition matrix is

$$\left(\begin{array}{cc} 1-\alpha & \alpha \\ \beta & 1-\beta \end{array} \right)$$

If $\alpha < \beta < 0.5$ are small, then typical outcome is

0000000000111111111000000001111000000000000000001111110000...

Expected length of 0-block is $\frac{1}{\alpha}$ and expected length of 1-block is $\frac{1}{\beta}$.
The proportion of ones is asymptotically?

The asymptotic proportion of 0 and 1 is according to stationary distribution $(\pi(0), \pi(1))$. MCMC method are based on that fact. In our case

$$\pi(1) = \frac{\alpha}{\alpha + \beta}.$$

Three state "triangular" MC. $S = \{0, 1, 2\}$, transition matrix is

$$\begin{pmatrix} 0.9 & 0.1 & 0 \\ 0 & 0.5 & 0.5 \\ 0.1 & 0 & 0.9 \end{pmatrix}$$

Being in a state, the chain either stays or jumps to the next state. Typical outcome

00000000111222222222000000000000012222....

Hidden Markov Model (HMM)

$Y_1, \ldots, Y_n$ – homogeneous Markov Chain with finite alphabet $S = \{1, \ldots, K\}$, and transition probabilities $p(i, j)$ and initial probabilities $\pi(i) := P(Y_1 = i)$.

MC is sometimes called as the regime.

To each state $y \in S$ corresponds an emission distribution P_y taking values on set $\mathcal{X}$. The set $\mathcal{X}$ can be anything, typically either $\mathbb{R}^d$ or a finite or countable set. In what follows, we assume that every P_y has a density f_y w. r. t. some reference measure dx (typically Lebesgue or counting measure).

HMM:

To any realization $y_1, y_2, \ldots$ of Y corresponds a sequence of independent random variables $X_1, X_2, \ldots, X_n$, where $X_t \sim P_{y_t}$.

Hence:

1) Given state sequence $y_1, \ldots, y_n$, the random variables $X_1, \ldots, X_n$ are conditionally independent;
2) The distribution of X_t depends on y_t (independent of the rest of the states).

Hence for every state sequence $y_1, \ldots, y_n$ and (measurable) subsets $A_1, \ldots, A_n \subset \mathcal{X}$, it holds:

$$P(X_1 \in A_1, \ldots, X_n \in A_n | Y_1 = y_1, \ldots Y_n = y_n) \overset{1)}{=}$$
$$\prod_{t=1}^{n} P(X_t \in A_t | Y_1 = y_1, \ldots Y_n = y_n) \overset{2)}{=} \prod_{t=1}^{n} P(X_t \in A_t | Y_t = y_t) =$$
$$\prod_{t=1}^{n} P_{y_t}(A_t) = \prod_{t=1}^{n} \int_{A_t} f_{y_t}(x) dx.$$

Exercise: Write down full probability:

$$P(X_1 \in A_1, \ldots, X_n \in A_n; Y_1 = y_1, \ldots Y_n = y_n).$$

The process X is sometimes called observation process or hidden Markov process. Can be taken as measurements of MC Y with error.

Some general properties (prove them as excercise):

1) In general, $X_1, \ldots, X_n$ is not a MC. It has longer memory. In particular, given a realization of X, it is difficult to prove/disprove that they are from a HMM;

2) 2D-process $(X_1, Y_1), \ldots, (X_n, Y_n)$ is a Markov process, when $\mathcal{X}$ is countable then a Markov chain. Thus, given a realization of (X, Y), it is very easy to prove/disprove that they are from a HMM;

3) Conditionally on a realization $x_1, \ldots, x_n$ of X, the Y-process (regime) is inhomogenuous MC:

$$P(y_{t+1}|y_t, \ldots, y_1; x_1, \ldots, x_n) = P(y_{t+1}|y_t; x_1, \ldots, x_n).$$

Obviously the opposite is true as well: given $y_1, \ldots, y_n$, the observations $X_1, \ldots, X_n$ are independent, thus Markov process.

Example: GpC-islands.

Let the regime be a 2-state MC with transition matrix ($S = \{0, 1\}$)

$$\begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix},$$

where $\alpha, \beta < 0.5$. Let the random variables take values on $\mathcal{X} = \{a, t, c, g\}$ with emission probabilities:

$$P_0(a) = P_0(t) = P_0(c) = P_0(g) = 0.25,$$
$$P_1(a) = P_1(t) = 0.1, \quad P_1(c) = P_1(g) = 0.4.$$

Thus, in regime 0 all letters are equiprobable, in regime 1 g and c have much larger probability. Typical outcomes

$a\,c\,g\,t\,a\,c\,g\,t\,a\,a\,g\,t\,c\,c\,g\,t\,c\,c\,a\,g\,t\,t\,g\,c\,a\,a\,c\,g\,c\,t\,c\,g\,t\,g\,a\,a\,t\,c\,t\,t\,a\,g\,c\,a\,g\,g\,c\,c\,g\,a\,c\,g\,g\,t.$

0000000000 111111111 00000000 1111 000000000000000 111111 00..

Example: pHMM – profile HMM (for pairwise alignments).

Three states: match (m), insertion (i), deletion (d).
Emissions: state match emits pairs (for example (a,g)); insertion emits residue against gap (for example (-,g)): deletion emits residue against gap (for example (a,-)).

An example of outcome is

a	t	t	g	g	a	t	-	g	t
a	a	-	g	c	-	t	g	c	t
m	m	d	m	m	d	m	i	m	m

Every outcome – alignment (table) – has a probability (score) . Hence pHMM is just a way to measure the goodness of alignments.

Forward - backward recursions. We are now study the probabilities

$$P(Y_t = j | X_1 = x_1, \ldots, X_n = x_n) =: p(y_t = j | x_1, \ldots, x_n).$$

The probabilities $p(y_t = j | x_1, \ldots, x_n)$ are called:

smoothing (posterior) probabilities, when $t < n$;
filtering (posterior) probabilities, when $t = n$;
prediction (posterior) probabilities, when $t > n$.

These probabilities are often the basis of the inference, but how to calculate the efficiently? For that, et us for every $j \in S$ define **forward** and **backward** variables

$$\alpha(j, x_1, \ldots, x_t) := p(x_1, \ldots, x_t, y_t = j),$$

$$\beta(x_{t+1}, \ldots, x_n | j) := \begin{cases} 1, \text{if } t = n \\ p(x_{t+1}, \ldots, x_n | y_t = j), \quad \text{if } t < n \end{cases}$$

The α- and β-variables can be calculated recursively:

$$\begin{aligned}
\alpha(j, x_1, \ldots, x_{t+1}) &= p(y_{t+1} = j, x_1, \ldots, x_{t+1}) \\
&= \sum_{i \in S} p(y_{t+1} = j, y_t = i, x_1, \ldots, x_{t+1}) \\
&= \sum_{i \in S} p(y_{t+1} = j, x_{t+1} | y_t = i, x_1, \ldots, x_t) p(y_t = i, x_1, \ldots, x_t) \\
&= \sum_{i \in S} p(y_{t+1} = j, x_{t+1} | y_t = i, x_t) \alpha(i, x_1 \ldots, x_t) \\
&= \sum_{i \in S} p(y_{t+1} = j, x_{t+1} | y_t = i) \alpha(i, x_1 \ldots, x_t) \\
&= \sum_{i \in S} p(i, j) f_i(x_t) \alpha(i, x_1 \ldots, x_t).
\end{aligned}$$

Here $=$ follows from the fact that (X, Y) is a Markov process. The obtained recursion:

$$\alpha(j, x_1, \ldots, x_{t+1}) = \sum_{i \in S} p(i, j) f_i(x_t) \alpha(i, x_1 \ldots, x_t)$$

is called forward recursion (calculate α-variables for $t = 1$, then update for $t = 2$ and so on until $t = n$).

Similarly, there is backward recursion for calculating β-variables:

$$\beta(x_t, \ldots, x_n|j) = \sum_{i \in S} \beta(x_{t+1}, \ldots, x_n|i) p(j,i) f_i(x_t).$$

With $\alpha(j, x_1, \ldots, x_t)$ and $\beta(x_{t+1}, \ldots, x_n|j)$, obviously

$$p(y_t = j, x_1, \ldots, x_n) = p(x_1, \ldots, x_t, y_t = j) p(x_{t+1}, \ldots, x_n | y_t = j, x_1, \ldots, x_t) =$$
$$p(x_1, \ldots, x_t, y_t = j) p(x_{t+1}, \ldots, x_n | y_t = j) = \alpha(j, x_1, \ldots, x_t) \beta(x_{t+1}, \ldots, x_n|j).$$

Here $=$ comes from Markov property of (X, Y) and definition of HMM.
Thus

$$p(y_t = j | x_1, \ldots, x_n) = \frac{p(y_t = j, x_1, \ldots, x_n)}{p(x_1, \ldots, x_n)} = \frac{p(y_t = j, x_1, \ldots, x_n)}{\sum_{i \in S} p(y_t = i, x_1, \ldots, x_n)}$$
$$= \frac{\alpha(j, x_1, \ldots, x_t) \beta(x_{t+1}, \ldots, x_n|j)}{\sum_{i \in S} \alpha(i, x_1, \ldots, x_t) \beta(x_{t+1}, \ldots, x_n|j)}.$$

Hence, to calculate smoothing probabilities, run forward and backward recursion (both linear in n) and the apply the formula above.

In practice, there might happen numerical underflow – both α and β values are too small for computer. Then several rescaled versions of the α and β-variables can be used. Most important: the recursions still hold.

Segmentation (decoding)

A realization $x_1, \ldots, x_n$ of $X_1, \ldots, X_n$ is observed. They are sample, data, observations.

The corresponding regime $:= y_1, \ldots, y_n$, the realization of $Y_1, \ldots, Y_n$ is not observed (Y is hidden).

The problem of segmentation: To find out / prognose / estimate the hidden state sequence $y_1, \ldots, y_n$ – the hidden path.

Warning: There is no way to find out exact hidden path! All meaningful methods give a path which is the best in some sense.

Example: In GpC-island example, thus only the DNA-sequence

$$a\,c\,g\,t\,a\,c\,g\,t\,a\,a\,g\,t\,c\,c\,g\,t\,c\,c\,a\,g\,t\,t\,g\,c\,a\,a\,c\,g\,c\,t\,c\,g\,t\,g\,a\,a\,t\,c\,t\,t\,a\,g\,c\,a\,g\,g\,c\,c\,g\,a\,c\,g\,g\,t$$

is observed. The goal is to (find a good method to) estimate the islands.

Example: DNA copy number alternations. States: DNA copy numbers (typically 2).

Emissions: comparative genomic hybridization (CGH). Typically modeled via Gaussian mixtures. The observations: measurements CGH, the goal is to (find a good method to) estimate copy numbers.

Example: Secondary structure of a protein. States: types of foldings. Emissions: 20 amino-acids. The goal is to guess the folding (secondary) structure of a protein (sequence of amino acids).

Framework of statistical learning (pattern recognition). Formally, we are looking for a function – classifier –

$$g = (g_1, \ldots, g_n) : \mathcal{X}^n \to S^n,$$

that maps the observed sequence into state sequence. What is the best g ?

Notation: $x^n := x_1, \ldots, x_n$, $s^n = s_1, \ldots, s_n$, $y^n := y_1, \ldots, y_n$, so on.

Given $x^n = x_1, \ldots, x_n$ define the conditional risk

$$R(\cdot|x^n) : \quad S^n \to [0, \infty]$$

so that $R(s^n|x^n)$ measures the goodness of state sequence s^n given observations x^n. The best classifier is that with the minimal risk over all possible classifiers. It can be obtained by minimizing the conditional risk:

$$g^*(x^n) := \arg \min_{s^n \in S^n} R(s^n|x^n).$$

The conditional risk $R(\cdot|x^n)$ depends on the task, often it is meaningful to define it via loss function

$$L : S^n \times S^n \to [0, \infty],$$

where $L(a^n, b^n)$ is loss, when the actual state sequence is a^n and the prognose is b^n. The conditional expectation

$$R(s^n|x^n) := E[L(Y^n, s^n)|X^n = x^n]$$

is the conditional risk of y^n. Then minimal-risk (or best) classifier is

$$g^*(x^n) = \arg \min_{s^n \in S^n} \sum_{a^n \in S^n} L(y^n, s^n) P(Y^n = y^n | X^n = x^n).$$

Usually no (or few) training data - unsupervised (or semisupervised) learning.

Standard alignments

Viterbi alignment. Let L be

$$L(a^n, b^n) = \begin{cases} 1, & \text{when } a^n \neq b^n; \\ 0, & \text{when } a^n = b^n. \end{cases} \tag{1}$$

Then the conditional risk (in this case denoted by R_∞) is

$$R_\infty(s^n|x^n) = 1 - P(Y^n = s^n|X^n = x^n)$$

and the minimal-risk classifier – for (1) denoted by v – maps every sequence of observations into sequence s^n with maximum (posterior) probability

$$v(x^n) := \arg \max_{s^n \in S^n} P(Y^n = s^n|X^n = x^n).$$

The maximum-prob state sequence $v(x^n)$ is known as Viterbi alignment (path). It inherits its name from the Viterbi algorithm – a dynamic programming algorithm for finding $v(x^n)$. Viterbi algorithm is very simple and Viterbi alignment is by far most popular classifier in segmentation.

Viterbi algorithm. How to find a path (or all of them) that maximizes the probability $p(s^n, x^n)$ (or, equivalently, $p(s^n|x^n)$) over all possible paths? For given s^n, the joint probability (or conditional probability) is easy to evaluate (linear in n), but there are K^n possible paths. It is a big number even for a moderate n (there are about $10^{80} \approx 2^{270}$ atoms in the universe).

Due to the Markov property (again) (Bellman's) optimality principle holds and a very elementary dynamic programming algorithm does the work.

Let, for every $t = 1, \ldots, n$ and every $j \in S$,

$$\delta_t(j) := \max_{y_1,\cdots,y_{t-1}} p(\overbrace{y_1, \ldots, y_{t-1}}^{y^{t-1}}, y_t = j, \overbrace{x_1, \ldots, x_t}^{x^t}) = \max_{y^{t-1} \in S^{t-1}} p(y^{t-1}, y_t = j, x^t).$$

Thus $\delta_1(j) = p(y_1 = j, x_1) = \pi(j) f_j(x_1)$.

Now observe that

$$\begin{aligned} p(y^{t-1}, y_t = i, y_{t+1} = j, x^{t+1}) &= p(y^{t-1}, y_t = i, x^t) p(y_{t+1} = j, x_{t+1} | y_t = i) \\ &= p(y^{t-1}, y_t = i, x^t) p(i, j) f_j(x_{t+1}). \end{aligned}$$

Hence

$$\delta_{t+1}(j) = \max_{y^t \in S^t} p(y^t, y_{t+1} = j, x^{t+1}) = \max_{i \in S} \left(\max_{y^{t-1} \in S^{t-1}} p(y^{t-1}, y_t = i, y_{t+1} = j, x^{t+1}) \right)$$

$$= \max_{i \in S} \left(\overbrace{\max_{y^{t-1} \in S^{t-1}} p(y^{t-1}, y_t = i, x^t)}^{\delta_t(i)} p(i, j) f_j(x_{t+1}) \right) = \max_{i \in S} \left(\delta_t(j) p(i, j) \right) f_j(x_{t+1})$$

Let

$$i_t(j) := \arg\max_{i \in S} \delta_t(j) p(i, j).$$

Viterbi algorithm:

1) **Initialize:** For every $j \in S$, define $\delta_1(j) := \pi_j f_j(x_1)$;

2) **Do for** $t = 1, \ldots, n-1$**:**

Update $\delta_{t+1}(j) = \max_i \Big(\delta_t(i) p(i,j)\Big) f_j(x_{t+1})$;

Record $i_t(j) := \arg\max_i \delta_t(i) p(i,j)$

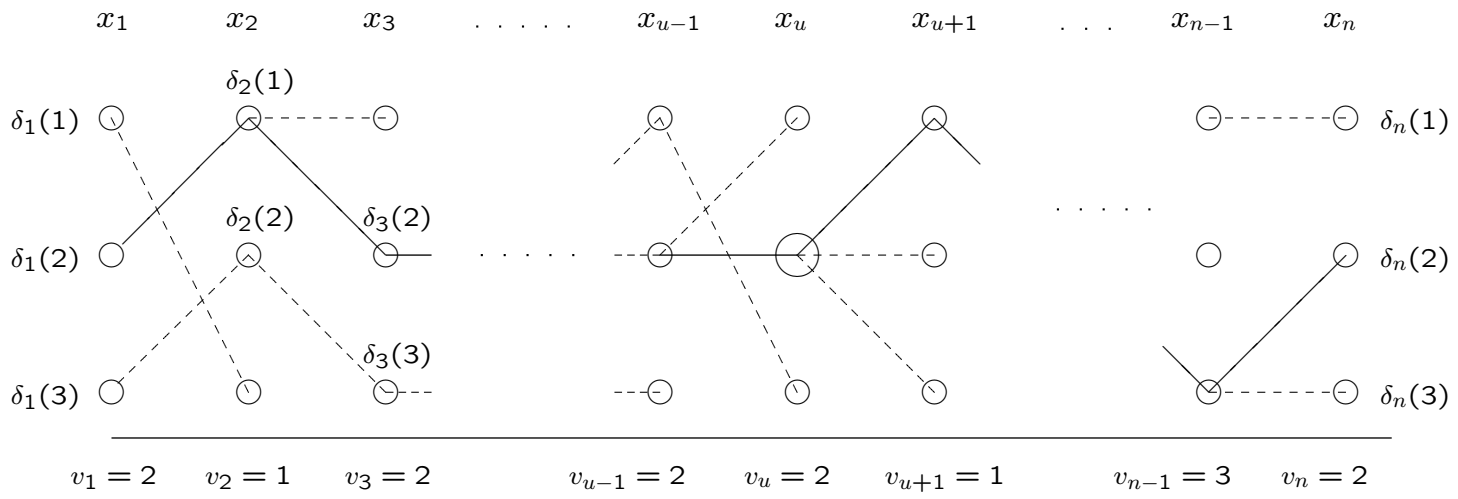
3) **Output:** Find Viterbi alignment v^n by backtracking:

$$v_n := \arg\max_j \delta_n(j), \quad v_t = i_t(v_{t+1}), \quad t = n-1, \ldots, 1.$$

The numerical underflow can be the issue again. A possible solution is to use $\ln \delta_t(j) =: \tilde{\delta}_t(j)$ so that the recursion is

$$\tilde{\delta}_{t+1}(j) = \max_i \left(\tilde{\delta}_t(i) + \ln p(i,j) \right) + \ln f_j(x_{t+1}).$$

The following picture illustrates Viterbi algorithm in action. The solid lines indicates the output, the dashed lines indicate $i_t(j)$.



The Viterbi algorithm is extremely easy to program and understand, therefore it is very popular. The output of it is the path with (conditional) probability How does it look in practice?

Consider a MC with $S = \{0, 1\}$ $\pi(0) = 3/13, \quad \pi(1) = 10/13$ (stationary distribution) and transition matrix

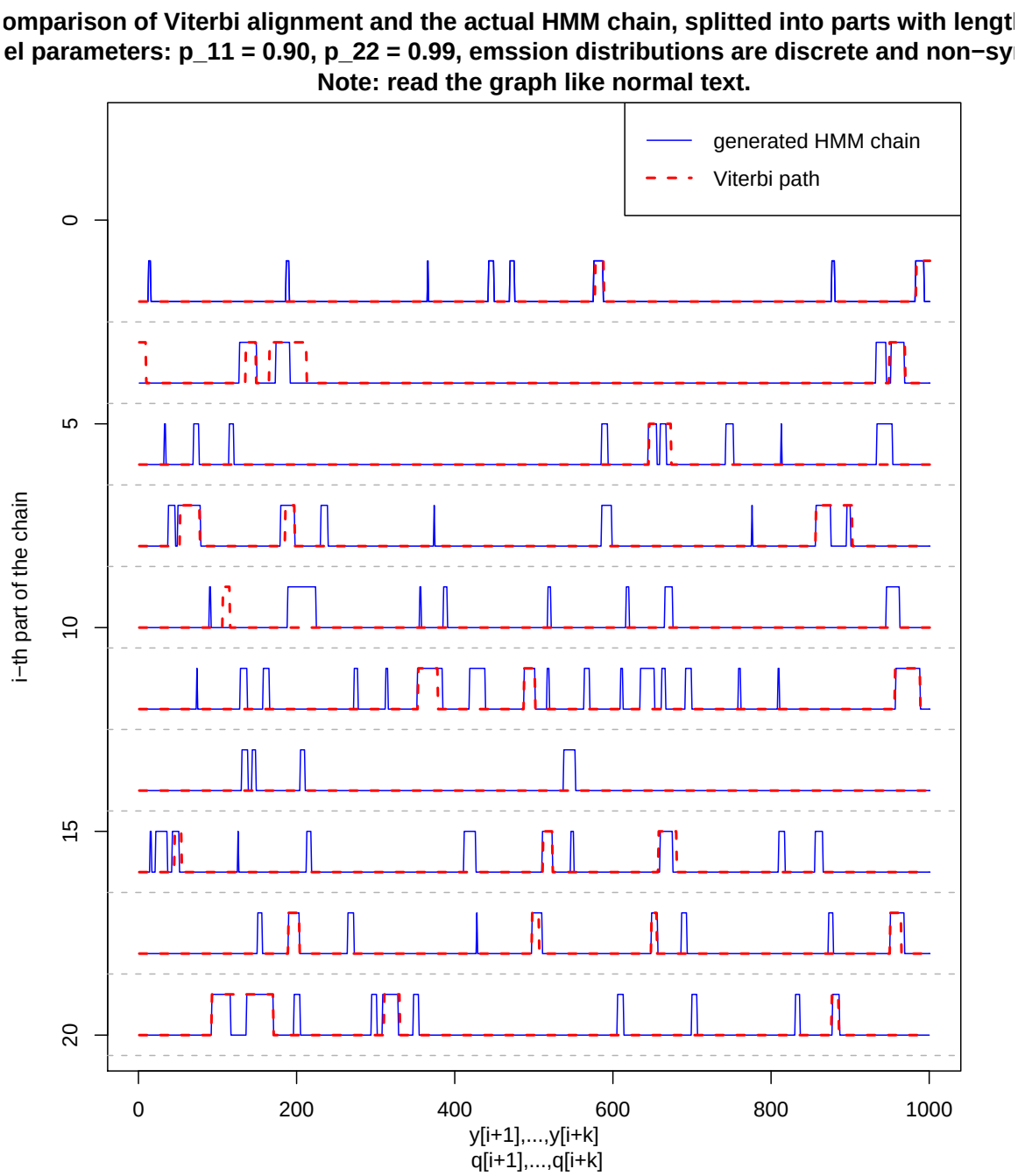
$$\begin{pmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{pmatrix}$$

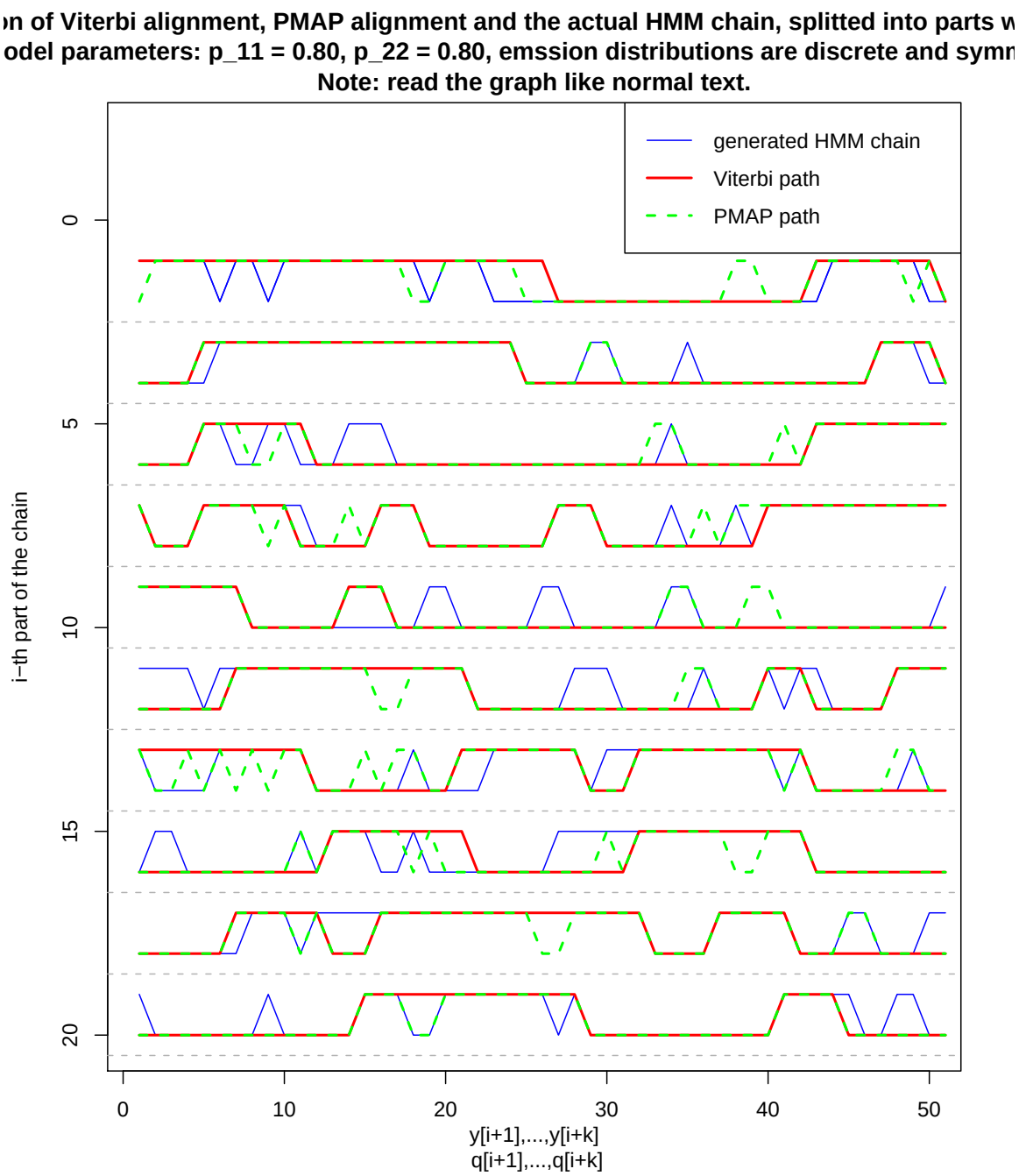
The typical realization of this chain consists of independent blocks o 0's and 1's having lengths distributed as $G(0.4)$ and $G(0.3)$, respectively

$$\overbrace{00}^{G(0.4)} \quad \overbrace{111}^{G(0.3)} \quad 00000 \quad 11 \quad 000 \quad 1111 \ldots$$

Hoe does the path with maximal probability look like?

Here are no data. With data the path changes but the long block structure prevails. Hence no wonder that Viterbi path (for models like this) overestimates the lengths of the blocks. It is highly untypical.


omparison of Viterbi alignment and the actual HMM chain, splitted into parts with lengt
el parameters: p_11 = 0.90, p_22 = 0.99, emssion distributions are discrete and non−sy
Note: read the graph like normal text.
generated HMM chain
Viterbi path
i−th part of the chain
0
5
10
15
20
0
200
400
600
800
1000
y[i+1],...,y[i+k]
q[i+1],...,q[i+k]


ɔn of Viterbi alignment, PMAP alignment and the actual HMM chain, splitted into parts w
odel parameters: p_11 = 0.80, p_22 = 0.80, emssion distributions are discrete and symn
Note: read the graph like normal text.
generated HMM chain
Viterbi path
PMAP path
i−th part of the chain
0
5
10
15
20
0
10
20
30
40
50
y[i+1],...,y[i+k]
q[i+1],...,q[i+k]

PMAP alignment. The loss (1) penalizes all differences alike: no matter whether a^n and b^n differ from one entry or all entries, the penalty is one. Often it is natural to penalize every different entry. This can be done by pointwise loss-function

$$l : S \times S \to [0, \infty) \qquad \text{where } l(s, s) = 0, \quad \forall s \in S. \tag{2}$$

Using l, we can define loss-function L as follows

$$L(a^n, b^n) := \frac{1}{n} \sum_{t=1}^{n} l(a_t, b_t).$$

Then the minimal-risk classifier is obtained pointwise: $g^* = (g_1^*, \ldots, g_n^*)$, where

$$g_t^*(x^n) = \arg\min_{s \in S} E[l(Y_t, s) | X^n = x^n]. \tag{3}$$

Exercise: Show (3).

Counting errors. If

$$l(a,a') = \begin{cases} 0, & \text{if } a = a'; \\ 1, & \text{if } a \neq a'. \end{cases},$$

then the loss-function L counts the differences between a^n and b^n when compared pairwise. Thus the conditional risk measures the expected number of misclassification errors of s^n given the observations are x^n and can be calculated as follows:

$$R_1(s^n|x^n) := 1 - \frac{1}{n}\sum_{t=1}^{n} P(Y_t = y_t|X^n = x^n).$$

So the minimal-risk classifier – let us denote it by u – minimizes the expected number of misclassification errors, from above we see that it can be calculated pointwise, indeed:

$$u_t(x^n) = \arg\max_{s\in S} P(Y_t = s|X^n = x^n) = \arg\max_{s\in S} p(y_t = s|x^n), \quad t = 1,\ldots,n.$$

We shall call u as PMAP (pointwise maximum aposteriori) -alignment (path). Other names encountered: *marginal posterior mode*, *maximum posterior marginals*, *optimal symbol-by-symbol detection*, *symbol-by-symbol MAP estimation*, *MAP-state estimation*.

Typically in practice K is small so that maximizing the smoothing probabilities $p(y_t = s|x^n)$ over S is not a problem. We also know that calculating the smoothing probabilities $p(y_t = s|x^n)$ is not a problem, because it can be done by forward-backward recursions. Thus, finding PMAP path is computationally feasible and from computational point of view it has the same complexity as Viterbi path.

In general, PMAP and Viterbi paths are not equal. Thus Viterbi path is not the best for misclassification errors point of view!

Since PMAP path is purely local, then in the presence of 0-s in transition matrix, it might have probability 0. Thus, it might happen that $p(u^n|x^n) = 0$. Such paths are inadmissible or forbidden and (again) highly untypical (never occur).

Consider our MC again: $S = \{0, 1\}$, $\pi(0) = 3/13$, $\pi(1) = 10/13$ (stationary distribution) and transition matrix $\begin{pmatrix} 0.6 & 0.4 \\ 0.3 & 0.7 \end{pmatrix}$. What is PMAP path in this case (without data)?

Logarithmic versions of standard risks:

$$\overline{R}_\infty(s^n|x^n) := -\frac{1}{n}\ln P(Y^n = s^n|x^n) = -\frac{1}{n}\ln p(s^n|x^n)$$

$$\overline{R}_1(s^n|x^n) := -\frac{1}{n}\sum_{t=1}^{n}\ln P(Y_t = s_t|x^n).$$

Clearly

Viterbi alignment $v(x^n)$ minimizes $\overline{R}_\infty(\cdot|x^n)$

PMAP alignment $u(x^n)$ minimizes $\overline{R}_1(\cdot|x^n)$.

Between PMAP and Viterbi. If the aim is to minimize the expected number of errors, then one should use PMAP-alignment u^n. Unfortunately, in the presence of forbidden transitions and/or emissions, it might be that $p(u^n|x^n) = 0$, i.e. u is inadmissible.

The simplest solution: restricted optimization:

$$\min_{s^n:p(s^n|x^n)>0} R_1(s^n|x^n) \quad \Leftrightarrow \quad \max_{s^n:p(s^n|x^n)>0} \sum_{i=1}^{n} P(Y_i = s_i|x^n). \tag{4}$$

Because of the restrictions, (4) is not equivalent to problem:

$$\min_{s^n:p(s^n|x^n)>0} \overline{R}_1(s^n|x^n) \quad \Leftrightarrow \quad \max_{s^n:p(s^n|x^n)>0} \sum_{t=1}^{n} \ln P(Y_t = s_t|x^n). \tag{5}$$

The solution of (5): posterior Viterbi decoding (PVD).

Note: the solution of (5) does not change if $p(s^n|x^n) > 0$ is replaced by $p(s^n) > 0$, but (4) is in general not equivalent to

$$\min_{s^n:p(s^n)>0} R_1(s^n|x^n) \quad \Leftrightarrow \quad \max_{s^n:p(s^n)>0} \sum_{t=1}^{n} P(Y_t = s_t|x^n). \tag{6}$$

In fact, the solution of (6) can still be inadmissible!

Algorithm for restricted optimization (4):

1) **Initialize:** For every $1 \leq t \leq n$ and $j \in S$ compute $p(y_t = j|x^n)$ with forward-backward recursions. Set $\delta_1(j) := p(y_1 = j|x^n)$.

2) **Do for** $t = 1, \ldots, n-1$**:**

For every $j \in S$ update

$$\delta_{t+1}(j) = \Big(\max_i \delta_t(i) r_{ij} + p(y_{t+1} = j|x^n) \Big) r_j^{t+1},$$

where $r_{ij} := \mathbb{I}_{\{p(i,j)>0\}}$, $r_j^t := \mathbb{I}_{\{p(y_t=j|x^n)>0\}}$.

Record $i_t(j) := \arg\max_i \delta_t(i) r_{ij}, \quad t = 1, \ldots, n-1.$

3) **Output:** Find the optimal alignment s^n by backtracking:

$$s_n := \arg\max_j \delta_n(j), \quad s_t = i_t(s_{t+1}), \quad t = n-1, \ldots, 1.$$

Algorithm for PVD (5):

1) **Initialize:** For every $1 \leq t \leq n$ and $j \in S$ compute $p(y_t = j|x^n)$ with forward-backward recursions. Set $\delta_1(j) := \ln\Big(p(y_1 = j|x^n)\Big)$.

2) **Do for** $t = 1, \ldots, n-1$**:**

For every $j \in S$ update

$$\delta_{t+1}(j) = \Big(\max_i \delta_t(i) + \ln(r_{ij})\Big) + \ln\Big(p(y_{t+1} = j|x^n)\Big);$$

Record $i_t(j) := \arg\max_i \delta_t(i) + \ln(r_{ij}), \quad t = 1, \ldots, n-1.$

3) **Output:** Find the optimal alignment s^n by backtracking:

$$s_n := \arg\max_j \delta_n(j), \quad s_t = i_t(s_{t+1}), \quad t = n-1, \ldots, 1.$$

The solutions of constrained problems can still have small probability. This suggests to consider instead of (4) the following more general penalized optimization problem:

$$\min_{s^n} \Big[R_1(s^n|x^n) + Ch(s^n, x^n)], \tag{7}$$

where $C > 0$ and $h(s^n, x^n)$ or $h(s^n)$ is some penalty term.
Some possibilities for h:

* $h(s^n) = I_{\{p(s^n)=0\}}$. For C large, we get (6) .

* $h(s^n) = -\frac{1}{n} \ln p(s^n)$ (log-prior). Then (7) is

$$\max_{s^n} \Big[\sum_{i=1}^{n} P(Y_i = s_i|x^n) + C \ln p(s^n) \Big]$$

and for $C > 0$ small we get (6)

* $h(s^n, x^n) = \overline{R}_\infty(s^n|x^n)$ (log-posterior). Then (7) is

$$\min_{s^n}[R_1(s^n|x^n) + C\overline{R}_\infty(s^n|x^n)] \quad \Leftrightarrow \quad \max_{s^n}\big[\sum_{i=1}^{n} P(Y_i = s_i|x^n) + C \ln p(s^n|x^n)\big] \qquad (8)$$

and for $C > 0$ small we get (4);

With $\overline{R}_1$-risk instead of R_1-risk, the $\overline{R}_1$ counterpart of (8) is

$$\min_{s^n}[\overline{R}_1(s^n|x^n) + C\overline{R}_\infty(s^n|x^n)] \quad \Leftrightarrow \quad \max_{s^n}\big[\sum_{t=1}^{n} \ln P(Y_t = s_t|x^n) + C \ln p(s^n|x^n)\big] \qquad (9)$$

Small C gives the solution of (5): PVD.

The solutions of (8) and (9) are in general not the same, but both problems naturally interpolate between the two standard alignments: the case $C = 0$ corresponds to PMAP and C very large – Viterbi.

If the alignment s^n has likelihood 0, then $\overline{R}_\infty(s^n|x^n) = \infty$. Thus for every $C > 0$, the solution has positive (posterior, hence prior) likelihood.

Block PMAP alignment

The idea (Rabiner): Instead of maximizing (over all $s^n \in S^n$)

$$\sum_{t=1}^{n} P(Y_t = s_t | x^n)$$

(PMAP alignment) maximize (here $a_r^l := (a_r, \ldots, a_l)$

$$P(Y_1^k = s_1^k | x^n) + P(Y_2^{k+1} = s_2^{k+1} | x^n) + \cdots + P(Y_{n-k+1}^n = s_{n-k+1}^n | x^n), \tag{10}$$

where k is the size of block. The case $k = 1$ corresponds to the PMAP-alignment, the bigger k, the "closer" to the Viterbi alignment.

Minimizes the expected number of correctly guessed k-blocks. For example, when $k = 2$, then the minimizer gives a path that with highest expected number of correct transitions.

Warning: Can be of 0 likelihood even for $k > 1$. (To prove that is an easy exercise).

A modification: Replace $+$ by $\times$. Denote, for $k = 1, 2, \ldots$

$$\overline{R}_k(s^n|x^n) := -\frac{1}{n}\sum_{t=1-k}^{n} \ln P\Big(Y_{(t+1)\vee 1}^{(t+k)\wedge n} = s_{(t+1)\vee 1}^{(t+k)\wedge n}\Big|x^n\Big).$$

So, for $k > 1$, $-n\overline{R}_k(s^n|x^n)$ is basically the sum

$$\ln P(Y_1^k = s_1^k|x^n) + \ln P(Y_2^{k+1} = s_2^{k+1}|x^n) + \cdots + \ln P(Y_{n-k+1}^n = s_{n-k+1}^n|x^n).$$

* PMAP alignment $u(x^n)$ minimizes $\overline{R}_1(\cdot|x^n)$;
* Let $u_k(x^n)$ minimize $\overline{R}_k(\cdot|x^n)$, hence $u_1(x^n) = u(x^n)$.

Lemma Let $n \geq k > 1$. Then, for every state sequence s^n:

$$\boxed{\overline{R}_k(s^n|x^n) = (k-1)\overline{R}_\infty(s^n|x^n) + \overline{R}_1(s^n|x^n).}$$

Thus u_k is the solution of (9):

$$\min_{s^n}[\overline{R}_1(s^n|x^n) + C\overline{R}_\infty(s^n|x^n)]$$

for $C = k - 1$. Gives a nice interpretation to (9) and relates it to Rabiner's k-block idea.

When k increases, then
$\overline{R}_\infty(u_k)$ decreases (likelihood increases)
$\overline{R}_1(u_k)$ risk increases.
for every $k > 1$, the alignment $u_k(x^n)$ has positive likelihood.

Generalizations

The risk

$$\overline{R}_1(s^n|x^n) + C\overline{R}_\infty(s^n|x^n)$$

can be obviously generalized to

$$C_1\overline{R}_\infty(s^n|x^n) + C_2\overline{R}_1(s^n|x^n)$$

($C_1 \geq 0, C_2 \geq 0$) and, to incorporate prior information more,

$$C_1\overline{R}_\infty(s^n|x^n) + C_2\overline{R}_1(s^n|x^n) + C_3\overline{R}_\infty(s^n) + C_4\overline{R}_1(s^n),$$

where

$$\overline{R}_\infty(s^n) := -\frac{1}{n}\ln P(Y^n = s^n) \quad - \text{ prior likelihood}$$

$$\overline{R}_1(s^n) := -\frac{1}{n}\sum_{t=1}^{n}\ln P(Y_t = s_t) \quad - \text{ prior pseudolikelihood.}$$

Consider

$$\boxed{\arg\min_{s^n}[C_1\overline{R}_\infty(s^n|x^n)+C_2\overline{R}_1(s^n|x^n)+C_3\overline{R}_\infty(s^n)+C_4\overline{R}_1(s^n)]}.$$

$C_1>0, C_2=C_3=C_4=0$ – Viterbi alignment;
$C_1=0, C_2>0, C_3=C_4=0$ – PMAP alignment;
$C_1=C_2=0, C_3>0, C_4=0$ – the alignment with maximal (prior) probability;
$C_1=C_2=C_3=0, C_4>0$ – the alignment with (average) minimal number of errors when no data;
$C_1>0, C_2>0, C_3=C_4=0$ – generalized k-block alignment;
$C_1>0, C_2=0, C_3>0, C_4=0$ – generalized Viterbi alignment (with prior probabilities);
$C_1=0, C_2>0, C_3>0, C_4=0$ – generalized PMAP alignment (with prior probabilities – has always admissible solution!)
...

How to find the solution? There is a recursion that can be nicely implemented by a (Viterbi like) dynamic programming algorithm.

Algorithm for optimization:

1) **Initialize:** For every $1 \leq t \leq n$ and $j \in S$ compute $p(y_t = j|x^n)$ with forward-backward recursions and set

$$g_t(j) := C_1 \log p(y_t = j|x^n) + C_2 \log f_j(x_t) + C_3 \log p(y_t = j).$$

Set $\delta_1(j) = C_1 \log p(y_1 = j|x^n) + (C_2 + C_3 + C_4) \log \pi_j + C_2 \log f_j(x_1)$.

2) **Do for** $t = 1, \ldots, n-1$**:**

For every $j \in S$ update

$$\delta_{t+1}(j) = \max_i \Big(\delta_t(i) + (C_2 + C_4) \log(p(i,j))\Big) + g_{t+1}(j).$$

Record $i_t(j) := \arg\max_i \delta_t(i) r_{ij}, \quad t = 1, \ldots, n-1.$

3) **Output:** Find s^n by backtracking ($t = n-1, \ldots, 1$):

$$s_n := \arg\max_j \max_i \Big(\delta_t(i) + (C_2 + C_4) \log(p(i,j))\Big), \quad s_t = i_t(s_{t+1}).$$

Example.

$$\mathbb{P} = \begin{pmatrix} 0.99 & 0.01 & 0 \\ 0.3 & 0.3 & 0.4 \\ 0 & 0.02 & 0.98 \end{pmatrix}, \qquad \pi = \begin{pmatrix} 0.5882 \\ 0.0196 \\ 0.3922 \end{pmatrix},$$

Emission distributions given by

$$p_1 = (0.3, 0.2, 0.2, 0.3), \quad p_2 = (0.1, 0.3, 0.3, 0.3), \quad p_3 = (1/6, 1/6, 1/6, 1/2).$$

Note: transitions $1 \leftrightarrow 3$ not allowed.

Chain mostly stays in state "one", producing "islands" of other states, mostly of state "three".

An outcome is generated and different alignment compared.

Consider the alignments: Viterbi, PMAP,

PairPMAP (Rabiner's 2-block): $\max_{s^n}[\sum_{t=1}^{n-1} P(Y_t^{t+1} = s_t^{t+1}|x^n)]$

PVD: $\min_{s^n:p(s^n|x^n)>0} \overline{R}_1(s^n|x^n)$,

Constrained PMAP: $\min_{s^n:p(s^n|x^n)>0} R_1(s^n|x^n)$

HybridK2: $\min_{s^n} \overline{R}_2(s^n|x^n) = \min_{s^n}[\overline{R}_\infty(s^n|x^n) + \overline{R}_1(s^n|x^n)]$.

True: 111111111233333333333333211111111111111111111···11111111111111111111111111111111

Viterbi: 111111111111111111111111111111111111111111111···11111111111111111111111111111111

PMAP: 111111133333333333333333333333333333333311111111···11133333333333333333333333333333311

PairMAP: 111111111333333333333333333333333333333311111111···11133333333333333333333333333333311

PVD: 111112333333333333333333333333333333321111111111···11111233333333333333333333333333332 1

CstrPMAP: 111111233333333333333333333333333333321111111···11123333333333333333333333333333321

HybridK2: 111112333333333333333333333333333333333211···11111111111111111111111111111111111

Experiments with real data from Protein Data Bank

Data: $N = 25713$ proteins. Each of them is modeled as a (non-stationary) HMM. Hidden sates are 6 type of structure motifs (α-helix, β-helix, coil and their modifications). Many impossible transitions. Emissions: 20 amino-acids.

We have N realizations of (x^{n_j}, y^{n_j}), $j = 1, \ldots, N$. For every i, the parameters are estimated from remaining $N - 1$ sequences and using these parameters, the sequence x^{n_j} is segmented using different methods (alignments). For every method, we calculated: the overall averaged error rate

$$\frac{\sum_{j=1}^{N} \sum_{i=1}^{n_j} I_{\{g_i(x^{n_j}) \neq y_i^{n_j}\}}}{\sum_{j=1}^{N} n^j}$$

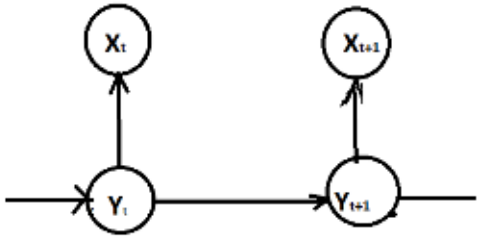
and averaged alignment probability

$$\frac{1}{N} \sum_{i=1}^{n} \left(P(g(x^{n_i})|x^{n_i}) \right)^{\frac{1}{n_i}}.$$

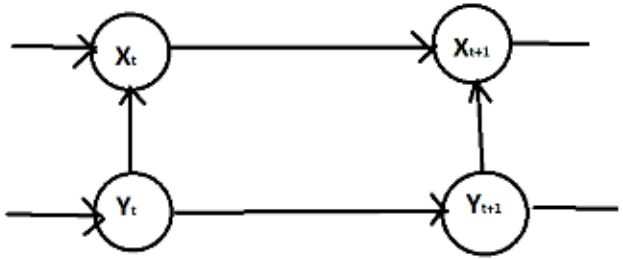
Pairwise Markov models (PMM) is a two-dimensional Markov chain

$$(X_1, Y_1), (X_2, Y_2), \ldots,$$

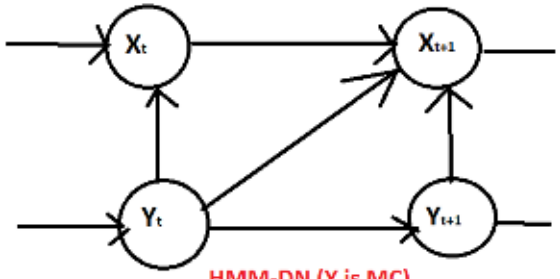
where, as previously, Y_t takes values in a finite set $\mathcal{Y}$ and X_t takes values in $\mathcal{X}$ (in general $\mathbb{R}^d$). Introduced by W. Piezcynski. PMM is a very rich class of models, allowing dependence between observations (given the state sequence), HMM is just a narrow subclass.



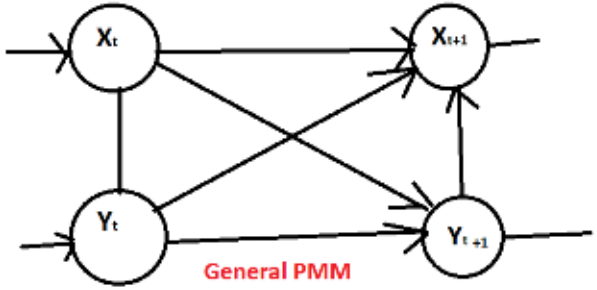
HMM



Markov switching model (Y is MC)



HMM-DN (Y is MC)



General PMM

In general neither Y nor X is a Markov chain, but in special cases one of the might be. Given X, the Y-sequence is a MC and vice versa.

Example: A Markov chain can be modeled as follows: start with a state y_1 chosen according to to initial probabilities, then generate duration from Geometrical distribution G_{y_1}, where for every $y \in S$

$$G_y(k) = (p_{yy})^k(1 - p_{yy}), \quad k = 0, 1, \ldots$$

Then change the state ($y_1 \to y_2$) with transition probabilities

$$q_{yy'} = \frac{p_{yy'}}{1 - p_{yy}}, \quad y \neq y', \quad q_{yy} = 0,$$

then generate a duration from G_{y_2} and so on.

When instead of Geometrical distribution, the duration is modeled via another distribution P_y, one obtains a semi-Markov chain.

It is not hard to see that semi-Markov process is a PMM.

Allows, among others, to model dependence between random sequences (DNA-sequences, for example). The following example shows how to jointly model two binay (1/2 valued) MC.

Example: Let $\mathcal{X} = \mathcal{Y} = \{1, 2\}$, and we want that X and Y were both Markov chains with the same transition matrix $\begin{pmatrix} p & 1-p \\ q & 1-q \end{pmatrix}$

It can be achieved by considering PMM Z with transition matrix

$$\mathbb{Q} = \begin{array}{c|cccc} & (1,1) & (1,2) & (2,1) & (2,2) \\ (1,1) & p\lambda_1 & p(1-\lambda_1) & p(1-\lambda_1) & 1+p\lambda_1-2p \\ (1,2) & p\lambda_2 & p(1-\lambda_2) & q-p\lambda_2 & 1+p\lambda_2-q-p \\ (2,1) & q\mu_1 & q(1-\mu_1) & p-q\mu_1 & 1+q\mu_1-p-q \\ (2,2) & q\mu_2 & q(1-\mu_2) & q(1-\mu_2) & 1+q\mu_2-2q \end{array}$$

where $\lambda_1, \lambda_2, \mu_1, \mu_2$ are the parameters to control the dependence between X and Y. They are independent, if $\lambda_1 = \mu_1 = p$ and $\lambda_2 = \mu_2 = q$; the bigger λ_i and ν_i, the more they are positively corr; the smaller parameters, the bigger neg. corr.

Examples of generated realizations of (X, Y) with different dependence (different λ_i and μ_i, $p = q = 0.7$)

p=q=0.7

Max

1001111111010110110000101111111101111001111001111101101101100100101101011111111010110110010100101111111101111001111101111011111011000001010

Min

1111111111111001001110011111111111011000111100111011101110011101111011111001001110001111111000111000110011100111100011101101101111101111000100

Ind

0011101101011110101100111111010111111111101111110010110101011011011100011110110111111111111011011000101101111111011101111111101111010011110111

Triplet Markov model (TMM) is a three-dimensional Markov chain

$$(X_1, Y_1, U_1), (X_2, Y_2, U_2), \ldots,$$

where typically U_t takes values in a finite set. Introduced by W. Piezcynski, allows deep modeling. Now, the pair (X, Y) – observations and classes – might not be a MC any more.

The third component allows to change the transition matrix and so the triplet model is very flexible to model inhomogeneous dynamics.

Example: A HMM with changing transition matrix can be modeled as TMM. Suppose we have two transition matrices

$$\mathbb{P}_A = \begin{pmatrix} p_A & 1-p_A \\ q_A & 1-q_A \end{pmatrix}, \quad \mathbb{P}_B = \begin{pmatrix} p_B & 1-p_B \\ q_B & 1-q_B \end{pmatrix},$$

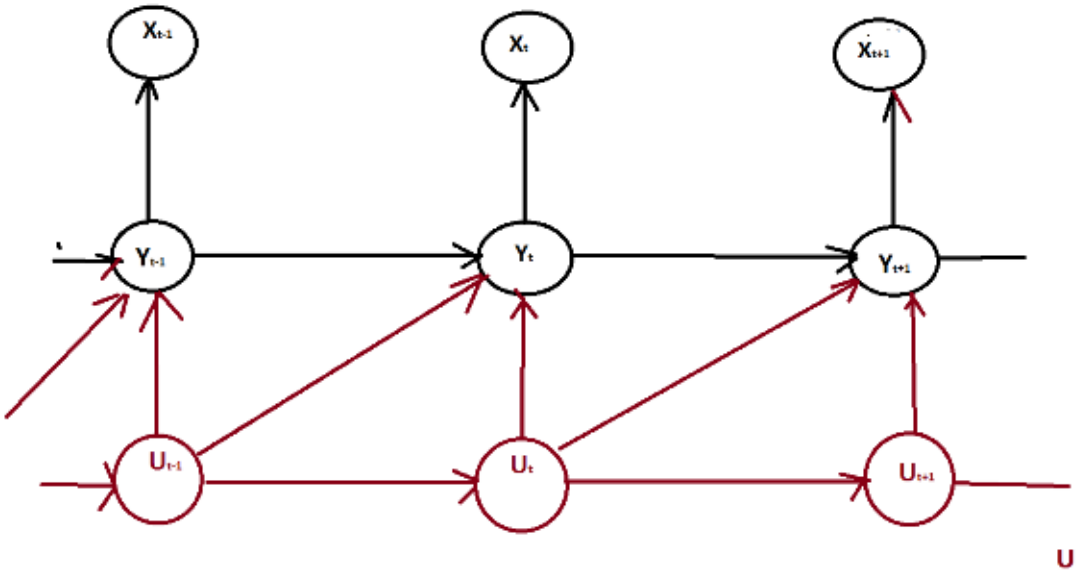
and with probability $\epsilon > 0$ they change. This can be modelled as a PMM (Y, U) as follows U takes values A and B, Y takes values 0 and 1 and the transition matrix of (Y, U) is

	$(0,A)$	$(1,A)$	$(0,B)$	$(1,B)$
$(0,A)$	$(1-\epsilon)p_A$	$(1-\epsilon)(1-p_A)$	$\frac{\epsilon}{2}$	$\frac{\epsilon}{2}$
$(1,A)$	$(1-\epsilon)q_A$	$(1-\epsilon)(1-q_A)$	$\frac{\epsilon}{2}$	$\frac{\epsilon}{2}$
$(0,B)$	$\frac{\epsilon}{2}$	$\frac{\epsilon}{2}$	$(1-\epsilon)p_B$	$(1-\epsilon)(1-p_B)$
$(1,B)$	$\frac{\epsilon}{2}$	$\frac{\epsilon}{2}$	$(1-\epsilon)q_B$	$(1-\epsilon)(1-q_B)$

Conditionally on $U_1 = u_1, \ldots, U_n = u_n$ $(u_t \in \{A, B\})$, the process $Y_1, \ldots, Y_n$ is a inhomogenuous MC with transition matrix

$$P(Y_{t+1} = j | Y_t = i, U_1 = u_1, \ldots, U_n = u_n) = \begin{cases} p_A(i,j), & \text{if } u_t = u_{t+1} = A; \\ p_B(i,j), & \text{if } u_t = u_{t+1} = B; \\ 1/2, & \text{if } u_t \neq u_{t+1}; \end{cases}$$

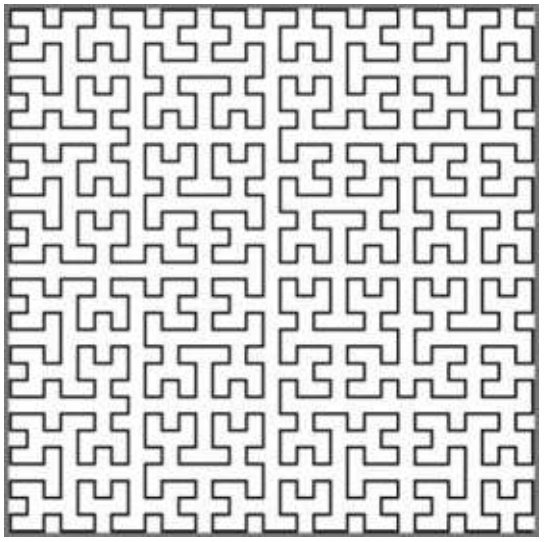
Like for HMM, let the observations be conditionally (given U and Y values) independent and depending only on Y_t. We have the following fairly simple TMM:



Many more examples possible, a way larger class of models than PMM. It can be that none of the processes: X, Y, U, (X,Y), (Y,U), (U,X) MC, only (X,Y,U) is. Conditionally on U, (X,Y) is a MC and so on.

2D case. 2D generalization of MC is Markov random field (MRF), with independent noise hidden MRF (HMRF). PMM and TMM are just two or three-component MRFs. Problem with fields: no dynamics! In particular – no Viterbi algorithm, no forward-backward algorithms, no EM-algorithms.

A solution: map 2D (or higher dim) field into 1D via Hilbert-Peano curve. Then model (HMM, PMM, TMM), estimate parameters, segment and reconstruct picture.

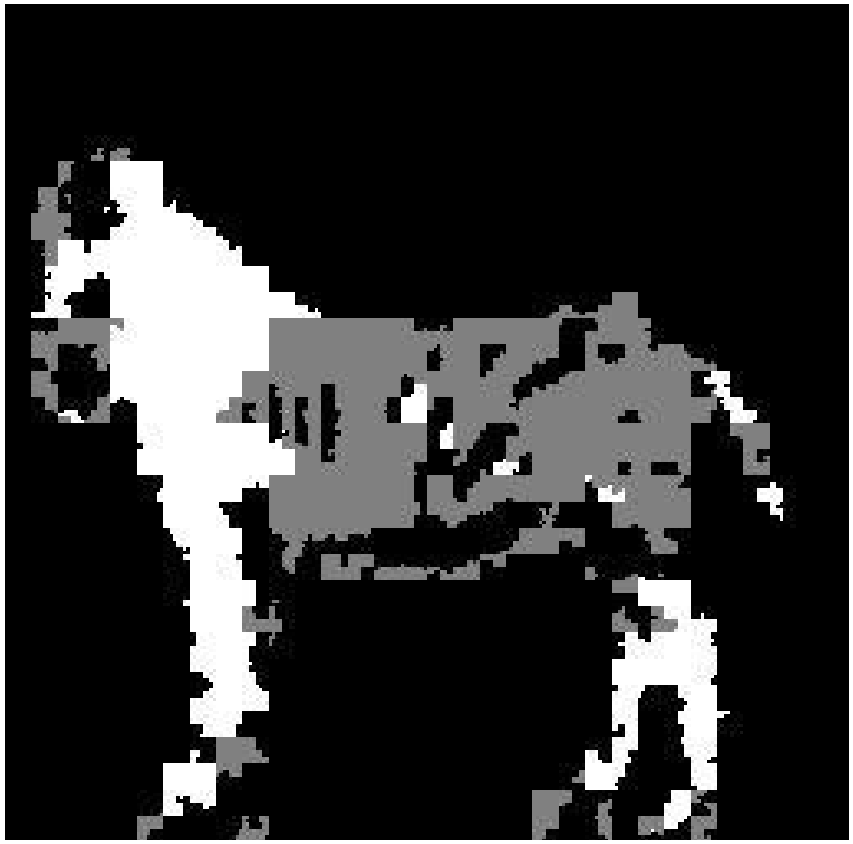


Denoising a zebra with HMM and PMM (by W. Pieczynski).

The noise is independent Gaussian, thus HMM or TMM with changing transitions (as above) are feasible. The picture is scanned with HP curve, then model parameters are estimated and PMAP segmentation is performed. After that the picture is reconstructed.

HMM reconstruction

TMM reconstruction



Estimates of U

Some (selected) references

About (basic of) HMM:

L. Rabiner, "A Tutorial in Hidden Markov Models and Selected Applications in Speech Recognition", Prodeedings of the IEEE, 1989;

T. Koski, "Hidden Markov models for bioinformatics", Kluwer, 2002;

O. Cappe, E. Moulines, T. Ryden, "Inferences with Hidden Markov models", Springer, 2005;

...

About segmentation:

J. Lember, K. Kuljus, A. Koloydenko, "Theory of segmentation", In: Hidden Markov Models: Theory and Applications (P. Dymarksi (Ed)), InTech, 2011;

J. Lember, A. Koloydenko, "Bridging Viterbi and Posterior Decoding: A Generalized Risk Approach to Hidden Path Inference Based on Hidden Markov Models", Journal of Machine Learning Research, 2014;

About PMM and TMM:

W. Pieczynski, "Pairwise Markov chains", IEEE Trans. on Pattern Analysis and Machine Intelligence, 2003.

S. Derrode and W. Pieczynski, "Unsupervised data classification using pairwise Markov chains with automatic copulas selection", Comp. Stat. and Data Anal., 2013.

D. Benboudjema and Benboudjema, "Unsupervised Statistical Segmentation of Nonstationary Images Using Triplet Markov Fields", IEEE Transactions on Pattern analysis and Machine Inteliignece, 2007.

...