



HAL
open science

Data Analysis - Lecture 9 : Time series analysis, Hidden Markov Models

Jérémie Sublime

► **To cite this version:**

Jérémie Sublime. Data Analysis - Lecture 9 : Time series analysis, Hidden Markov Models. Engineering school. France. 2022. hal-03845332

HAL Id: hal-03845332

<https://hal.science/hal-03845332>

Submitted on 9 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Data Analysis - Lecture 9

Time series analysis: Hidden Markov Models

Dr. Jérémie Sublime

LISITE Laboratory - DaSSIP Team - ISEP
LIPN - CNRS UMR 7030

jeremie.sublime@isep.fr

Plan

- 1 Introduction to HMM
- 2 The decoding Problem
- 3 The learning Problem
- 4 Conclusion

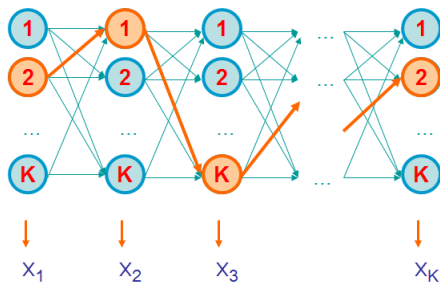
Outline

- 1 Introduction to HMM
- 2 The decoding Problem
- 3 The learning Problem
- 4 Conclusion

Introduction to HMM

- Hidden Markov models (HMMs) are a ubiquitous tool for modeling time series data.
- Far from being limited to time series, they are widely used with all sorts of sequential data:
 - Speech recognition systems
 - Genome sequence analysis
 - Pattern recognition
 - Object tracking and computer vision
 - Etc.

Introduction to HMM



Definition

A hidden Markov model is a tool for representing probability distributions over sequences of observations. It is assumed that each observation X_t was generated by some process whose state $S_t \in [1..K]$ is unknown (hence the name hidden).

Introduction to HMM

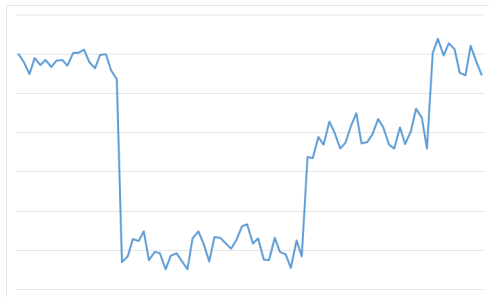


Figure: Example of a univariate sequence with 3 hidden states

HMM: Properties

- Each observation X_t depends only on the current state Q_t
- Markov property: Given the value of the state Q_{t-1} , the current state Q_t is independent from all states prior to $t - 1$.

Definition of a HMM

A HMM is made of 5 key elements:

Definition of a HMM

A HMM is made of 5 key elements:

- **An alphabet** $\Sigma = \{o_1, \dots, o_M\}$ which defines the form that the observations can take:
 - $\Sigma = \mathbb{R}^d$, $d \in \mathbb{N}^*$ for multi-dimensional observations
 - $\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice rolls
 - etc.

Definition of a HMM

A HMM is made of 5 key elements:

- **An alphabet** $\Sigma = \{o_1, \dots, o_M\}$ which defines the form that the observations can take:
 - $\Sigma = \mathbb{R}^d$, $d \in \mathbb{N}^*$ for multi-dimensional observations
 - $\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice rolls
 - etc.
- **A set of states** $Q = \{1, \dots, K\}$

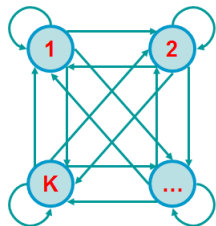


Figure: Example of a state transition diagram

Definition of a HMM

A HMM is made of 5 key elements:

- **An alphabet** $\Sigma = \{o_1, \dots, o_M\}$ which defines the form that the observations can take:
 - $\Sigma = \mathbb{R}^d$, $d \in \mathbb{N}^*$ for multi-dimensional observations
 - $\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice rolls
 - etc.
- **A set of states** $Q = \{1, \dots, K\}$
- **A transition probability matrix** $A = (a_{ij})_{K \times K}$, $\forall i \sum_j a_{ij} = 1$
 - a_{ij} is the probability of transition from state i to state j .

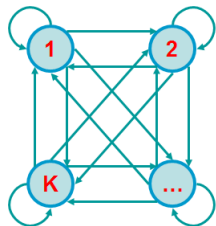


Figure: Example of a state transition diagram

Definition of a HMM

A HMM is made of 5 key elements:

- **An alphabet** $\Sigma = \{o_1, \dots, o_M\}$ which defines the form that the observations can take:

- $\Sigma = \mathbb{R}^d$, $d \in \mathbb{N}^*$ for multi-dimensional observations
- $\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice rolls
- etc.

- **A set of states** $Q = \{1, \dots, K\}$

- **A transition probability matrix** $A = (a_{ij})_{K \times K}$, $\forall i \sum_j a_{ij} = 1$

- a_{ij} is the probability of transition from state i to state j .

- **Emission probabilities** within each state:

$$e_i(x) = P(x|Q = i), \quad \forall i \in [1..K] \sum_{x \in \Sigma} e_i(x) = 1$$

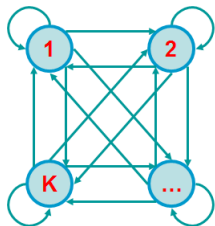


Figure: Example of a state transition diagram

Definition of a HMM

A HMM is made of 5 key elements:

- **An alphabet** $\Sigma = \{\sigma_1, \dots, \sigma_M\}$ which defines the form that the observations can take:

- $\Sigma = \mathbb{R}^d$, $d \in \mathbb{N}^*$ for multi-dimensional observations
- $\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice rolls
- etc.

- **A set of states** $Q = \{1, \dots, K\}$

- **A transition probability matrix** $A = (a_{ij})_{K \times K}$, $\forall i \sum_j a_{ij} = 1$
 - a_{ij} is the probability of transition from state i to state j .
- **Emission probabilities** within each state:

$$e_i(x) = P(x|Q = i), \quad \forall i \in [1..K] \sum_{x \in \Sigma} e_i(x) = 1$$

- **Starting probabilities:** π_1, \dots, π_K , $\sum_{i=1}^K \pi_i = 1$

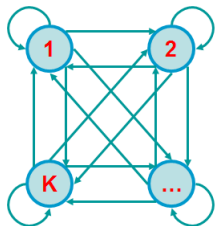


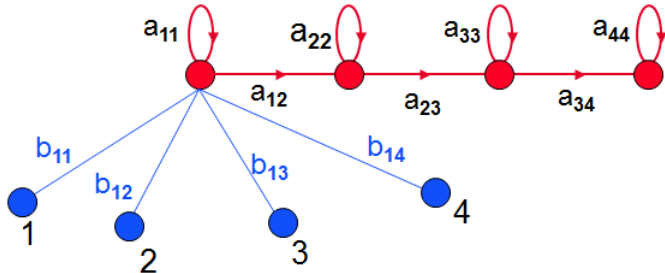
Figure: Example of a state transition diagram

Definition of a HMM

Notations

A HMM model is denoted $M = \{A, B, \pi\}$, where:

- A is the transition probability matrix
- B contains the emissions probability laws $e_i(x)$
- π the starting probabilities.



Question 1: Evaluating

The Evaluation Problem

- **Given:** a sequence of observations $X = \{x_1, \dots, x_L\}$ and a model $M = \{A, B, \pi\}$
 - **Question:** How do we efficiently compute $P(X|M)$, the probability of observing this sequence X knowing the model M .
-
- The probability $P(X|M)$ can be viewed as a measure of quality to evaluate the model M .
 - It can be used to discriminate or select among alternative models M_1, M_2, M_3, \dots

Question 2: Decoding

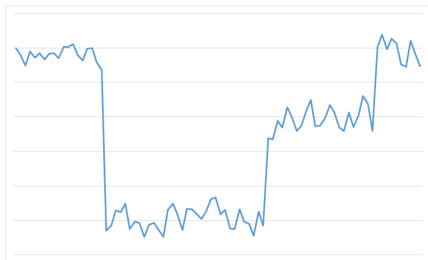
The Decoding Problem

- **Given:** a sequence of observations $X = \{x_1, \dots, x_L\}$ and a model $M = \{A, B, \pi\}$
- **Question:** How do we compute the most probable sequence(s) of states $Q = \{Q_1, \dots, Q_L\}$?

Question 2: Decoding

The Decoding Problem

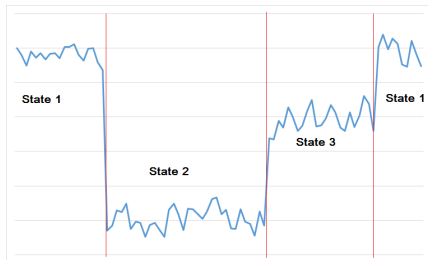
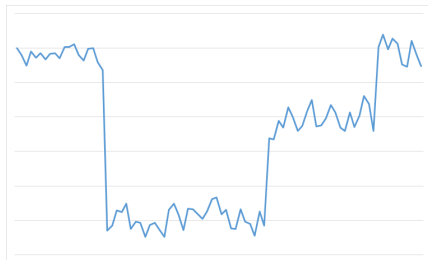
- **Given:** a sequence of observations $X = \{x_1, \dots, x_L\}$ and a model $M = \{A, B, \pi\}$
- **Question:** How do we compute the most probable sequence(s) of states $Q = \{Q_1, \dots, Q_L\}$?



Question 2: Decoding

The Decoding Problem

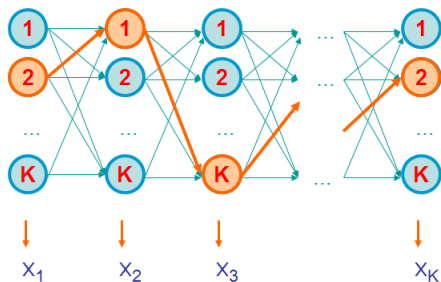
- **Given:** a sequence of observations $X = \{x_1, \dots, x_L\}$ and a model $M = \{A, B, \pi\}$
- **Question:** How do we compute the most probable sequence(s) of states $Q = \{Q_1, \dots, Q_L\}$?



Question 3: Learning

The Learning Problem

- **Given:** Just the sequence of observations $X = \{x_1, \dots, x_L\}$
- **Question:** How do we learn $M = \{A, B, \pi\}$ and then figure out Q ?

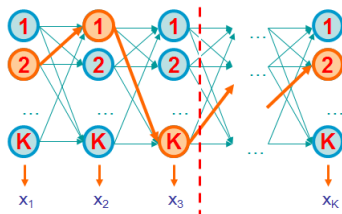


Outline

- 1 Introduction to HMM
- 2 The decoding Problem**
- 3 The learning Problem
- 4 Conclusion

The decoding problem

- Finding the sequence $\mathbf{Q} = \{Q_1, \dots, Q_N\}$ of hidden states is the most common task with HMM.
- Given the observations $X = \{x_1, \dots, x_N\}$, we want to find \mathbf{Q} that maximizes $P(X, \mathbf{Q})$



Let us denote $V_k(t)$ the optimal sequence of state knowing that at time t we are in the state k :

$$V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$, how do we define $V_l(t+1)$?

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$, how do we define $V_l(t+1)$?

$$V_l(t+1) = \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l)$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$,
how do we define $V_l(t+1)$?

$$\begin{aligned} V_l(t+1) &= \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l) \\ &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | x_1 \dots x_t, Q_1 \dots Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \end{aligned}$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$,
how do we define $V_l(t+1)$?

$$\begin{aligned} V_l(t+1) &= \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l) \\ &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | x_1 \dots x_t, Q_1 \dots Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\ &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \end{aligned}$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$, how do we define $V_l(t+1)$?

$$\begin{aligned}
 V_l(t+1) &= \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | x_1 \dots x_t, Q_1 \dots Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_k \left[P(x_{t+1}, Q_{t+1} = l | Q_t = k) \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k) \right]
 \end{aligned}$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$, how do we define $V_l(t+1)$?

$$\begin{aligned}
 V_l(t+1) &= \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | x_1 \dots x_t, Q_1 \dots Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_k \left[P(x_{t+1}, Q_{t+1} = l | Q_t = k) \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k) \right] \\
 &= \max_k \left[P(x_{t+1} | Q_{t+1} = l) P(Q_{t+1} = l | Q_t = k) V_k(t) \right]
 \end{aligned}$$

Decoding: Main Idea

Viterbi algorithm: Main idea

The idea of the Viterbi algorithm is to recursively build the $V_k(i)$

- Knowing that $V_k(t) = \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k)$, how do we define $V_l(t+1)$?

$$\begin{aligned}
 V_l(t+1) &= \max_{Q_1 \dots Q_t} P(x_1 \dots x_{t+1}, Q_1 \dots Q_t, Q_{t+1} = l) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | x_1 \dots x_t, Q_1 \dots Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_{Q_1 \dots Q_t} P(x_{t+1}, Q_{t+1} = l | Q_t) P(x_1 \dots x_t, Q_1 \dots Q_t) \\
 &= \max_k \left[P(x_{t+1}, Q_{t+1} = l | Q_t = k) \max_{Q_1 \dots Q_{t-1}} P(x_1 \dots x_t, Q_1 \dots Q_{t-1}, Q_t = k) \right] \\
 &= \max_k [P(x_{t+1} | Q_{t+1} = l) P(Q_{t+1} = l | Q_t = k) V_k(t)] \\
 &= e_l(x_{t+1}) \max_k [a_{kl} V_k(t)]
 \end{aligned}$$

The Viterbi Algorithm

Initialization:

- $V_0(0) = 1$
- $\forall k > 0 \quad V_k(0) = 0$

Iteration:

- $V_k(t) = e_j(x_t) \max_q [a_{qk} V_q(t-1)]$
- $Ptr_k(t) = \operatorname{argmax}_q [a_{qk} V_q(t-1)]$

Termination:

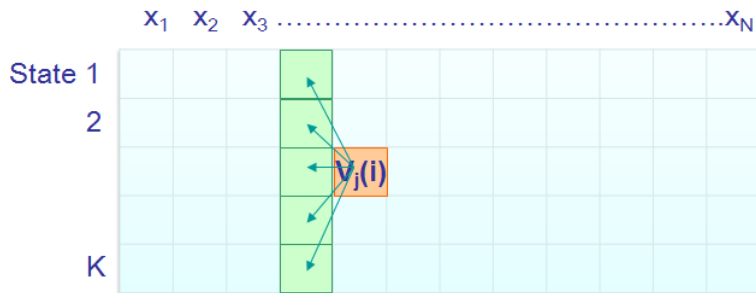
$$P(X, \mathbf{Q}^*) = \max_k V_k(N)$$

Traceback:

$$Q_N^* = \operatorname{argmax}_k V_k(N)$$

$$Q_{t-1}^* = Ptr_{Q_t^*}(t)$$

The Viterbi Algorithm



complexity of the Viterbi algorithm

- Time complexity $O(K^2N)$
- Space complexity $O(KN)$

The Viterbi Algorithm

An important remark on a practical detail

Underflows are a major problem with the Viterbi algorithm.

$$P(x_1, \dots, x_t, Q_1, \dots, Q_t) = \pi_{Q_1} \cdot a_{Q_1, Q_2} \cdot a_{Q_2, Q_3} \cdots a_{Q_{t-1}, Q_t} \cdot e_{Q_1}(x_1) \cdots e_{Q_t}(x_t)$$

These numbers are extremely small, thus leading to a quick underflow.

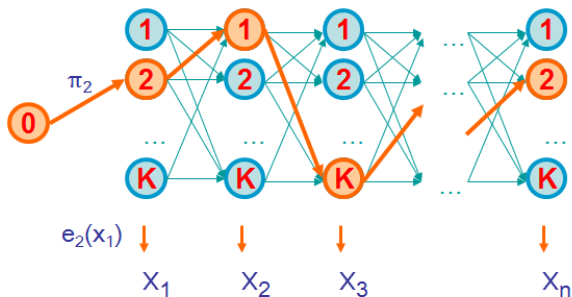
A good solution is to use the logarithm of all values, e.g:

$$\log V_k(t) = \log e_j(x_t) + \log (\max_q [a_{qk} \times V_q(t-1)])$$

Generating a sequence by the model

Given a HMM, we can generate a sequence of length n as follows:

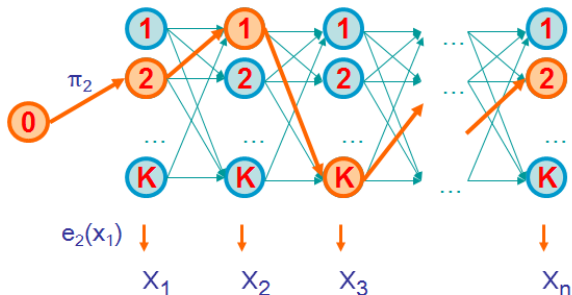
- 1 Start at state Q_1 according to π_k
- 2 Emit observation x_1 according to $e_{Q_1}(x_1)$
- 3 Go to state Q_2 according to a_{Q_1, Q_2}
- 4 ... until emitting x_n



Evaluation

We want an algorithm that can compute:

- $P(X)$ the probability of X given the model
- $P(x_i \cdots x_j)$ the probability of a substring of X given the model
- $P(Q_i = k | X)$ the posterior probability that the i^{th} state is k , given X



The Forward Algorithm

We want to calculate $P(X)$, the probability of the whole sequence given the HMM.

We can sum all possible ways of generating X :

$$P(X) = \sum_{\mathbf{Q}} P(X, \mathbf{Q}) = \sum_{\mathbf{Q}} P(X|\mathbf{Q})P(\mathbf{Q})$$

To avoid summing over an exponential number of paths \mathbf{Q} , we define the **forward probability**:

$$f_k(t) = P(x_1 \cdots x_t, Q_t = k)$$

It is the probability of observing the sequence $x_1 \cdots x_t$ and having the t^{th} state being k .

The Forward Algorithm: Derivation

$$f_k(t) = P(x_1 \cdots x_t, Q_t = k)$$

The Forward Algorithm: Derivation

$$\begin{aligned} f_k(t) &= P(x_1 \cdots x_t, Q_t = k) \\ &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) P(x_t | Q_t = k) \end{aligned}$$

The Forward Algorithm: Derivation

$$\begin{aligned}f_k(t) &= P(x_1 \cdots x_t, Q_t = k) \\&= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) P(x_t | Q_t = k) \\&= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) e_k(x_t)\end{aligned}$$

The Forward Algorithm: Derivation

$$\begin{aligned}
 f_k(t) &= P(x_1 \cdots x_t, Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) P(x_t | Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) e_k(x_t) \\
 &= \sum_l \sum_{Q_1 \cdots Q_{t-2}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-2}, Q_{t-1} = l) a_{lk} e_k(x_t)
 \end{aligned}$$

The Forward Algorithm: Derivation

$$\begin{aligned}
 f_k(t) &= P(x_1 \cdots x_t, Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) P(x_t | Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) e_k(x_t) \\
 &= \sum_l \sum_{Q_1 \cdots Q_{t-2}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-2}, Q_{t-1} = l) a_{lk} e_k(x_t) \\
 &= \sum_l P(x_1 \cdots x_{t-1}, Q_{t-1} = l) a_{lk} e_k(x_t)
 \end{aligned}$$

The Forward Algorithm: Derivation

$$\begin{aligned}
 f_k(t) &= P(x_1 \cdots x_t, Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) P(x_t | Q_t = k) \\
 &= \sum_{Q_1 \cdots Q_{t-1}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-1}, Q_t = k) e_k(x_t) \\
 &= \sum_l \sum_{Q_1 \cdots Q_{t-2}} P(x_1 \cdots x_{t-1}, Q_1 \cdots Q_{t-2}, Q_{t-1} = l) a_{lk} e_k(x_t) \\
 &= \sum_l P(x_1 \cdots x_{t-1}, Q_{t-1} = l) a_{lk} e_k(x_t) \\
 &= e_k(x_t) \sum_l f_l(t-1) a_{lk}
 \end{aligned}$$

The Forward Algorithm

We can compute all the $f_k(t)$ using dynamic programming:

Initialization:

- $f_0(0) = 1$
- $\forall k > 0 \quad f_k(0) = 0$

Iteration:

$$f_k(i) = e_k(x_t) \sum_l f_l(t-1) a_{lk}$$

Termination:

$$P(X) = \sum_k f_k(N)$$

The Backward Algorithm

Motivation

To unlock the sequence of hidden states, want to compute $P(Q_t = k|X)$ the probability distribution on the t^{th} position given X .

Since we have $P(Q_t = k|X) = \frac{P(Q_t=k, X)}{P(X)}$, we start by computing $P(Q_t = k, X)$:

The Backward Algorithm

Motivation

To unlock the sequence of hidden states, want to compute $P(Q_t = k|X)$ the probability distribution on the t^{th} position given X .

Since we have $P(Q_t = k|X) = \frac{P(Q_t=k, X)}{P(X)}$, we start by computing $P(Q_t = k, X)$:

$$P(Q_t = k, X) = P(x_1 \cdots x_t, Q_t = k, x_{t+1} \cdots x_N)$$

The Backward Algorithm

Motivation

To unlock the sequence of hidden states, want to compute $P(Q_t = k|X)$ the probability distribution on the t^{th} position given X .

Since we have $P(Q_t = k|X) = \frac{P(Q_t=k,X)}{P(X)}$, we start by computing $P(Q_t = k, X)$:

$$\begin{aligned} P(Q_t = k, X) &= P(x_1 \cdots x_t, Q_t = k, x_{t+1} \cdots x_N) \\ &= P(x_1 \cdots x_t, Q_t = k)P(x_{t+1} \cdots x_N | Q_t = k, x_1 \cdots x_t) \end{aligned}$$

The Backward Algorithm

Motivation

To unlock the sequence of hidden states, want to compute $P(Q_t = k|X)$ the probability distribution on the t^{th} position given X .

Since we have $P(Q_t = k|X) = \frac{P(Q_t=k,X)}{P(X)}$, we start by computing $P(Q_t = k, X)$:

$$\begin{aligned}P(Q_t = k, X) &= P(x_1 \cdots x_t, Q_t = k, x_{t+1} \cdots x_N) \\ &= P(x_1 \cdots x_t, Q_t = k)P(x_{t+1} \cdots x_N | Q_t = k, x_1 \cdots x_t) \\ &= P(x_1 \cdots x_t, Q_t = k)P(x_{t+1} \cdots x_N | Q_t = k)\end{aligned}$$

The Backward Algorithm

Motivation

To unlock the sequence of hidden states, want to compute $P(Q_t = k|X)$ the probability distribution on the t^{th} position given X .

Since we have $P(Q_t = k|X) = \frac{P(Q_t=k,X)}{P(X)}$, we start by computing $P(Q_t = k, X)$:

$$\begin{aligned}P(Q_t = k, X) &= P(x_1 \cdots x_t, Q_t = k, x_{t+1} \cdots x_N) \\&= P(x_1 \cdots x_t, Q_t = k)P(x_{t+1} \cdots x_N | Q_t = k, x_1 \cdots x_t) \\&= P(x_1 \cdots x_t, Q_t = k)P(x_{t+1} \cdots x_N | Q_t = k) \\&= f_k(t)b_k(t)\end{aligned}$$

Let us note $b_k(t)$ the backward probability.

The Backward Algorithm: Derivation

$$b_k(t) = P(x_{t+1} \cdots x_N | Q_t = k)$$

The Backward Algorithm: Derivation

$$\begin{aligned} b_k(t) &= P(x_{t+1} \cdots x_N | Q_t = k) \\ &= \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} \cdots Q_N | Q_t = k) \end{aligned}$$

The Backward Algorithm: Derivation

$$\begin{aligned} b_k(t) &= P(x_{t+1} \cdots x_N | Q_t = k) \\ &= \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} \cdots Q_N | Q_t = k) \\ &= \sum_l \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} = l, Q_{t+2} \cdots Q_N | Q_t = k) \end{aligned}$$

The Backward Algorithm: Derivation

$$\begin{aligned}
 b_k(t) &= P(x_{t+1} \cdots x_N | Q_t = k) \\
 &= \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} \cdots Q_N | Q_t = k) \\
 &= \sum_l \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} = l, Q_{t+2} \cdots Q_N | Q_t = k) \\
 &= \sum_l P(x_{t+1} | Q_{t+1} = l) a_{kl} \sum_{Q_{t+2} \cdots Q_N} P(x_{t+2} \cdots x_N, Q_{t+2} \cdots Q_N | Q_{t+1} = l)
 \end{aligned}$$

The Backward Algorithm: Derivation

$$\begin{aligned}
 b_k(t) &= P(x_{t+1} \cdots x_N | Q_t = k) \\
 &= \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} \cdots Q_N | Q_t = k) \\
 &= \sum_l \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} = l, Q_{t+2} \cdots Q_N | Q_t = k) \\
 &= \sum_l P(x_{t+1} | Q_{t+1} = l) a_{kl} \sum_{Q_{t+2} \cdots Q_N} P(x_{t+2} \cdots x_N, Q_{t+2} \cdots Q_N | Q_{t+1} = l) \\
 &= \sum_l e_l(x_{t+1}) a_{kl} \sum_{Q_{t+2} \cdots Q_N} P(x_{t+2} \cdots x_N, Q_{t+2} \cdots Q_N | Q_{t+1} = l)
 \end{aligned}$$

The Backward Algorithm: Derivation

$$\begin{aligned}
 b_k(t) &= P(x_{t+1} \cdots x_N | Q_t = k) \\
 &= \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} \cdots Q_N | Q_t = k) \\
 &= \sum_l \sum_{Q_{t+1} \cdots Q_N} P(x_{t+1} \cdots x_N, Q_{t+1} = l, Q_{t+2} \cdots Q_N | Q_t = k) \\
 &= \sum_l P(x_{t+1} | Q_{t+1} = l) a_{kl} \sum_{Q_{t+2} \cdots Q_N} P(x_{t+2} \cdots x_N, Q_{t+2} \cdots Q_N | Q_{t+1} = l) \\
 &= \sum_l e_l(x_{t+1}) a_{kl} \sum_{Q_{t+2} \cdots Q_N} P(x_{t+2} \cdots x_N, Q_{t+2} \cdots Q_N | Q_{t+1} = l) \\
 &= \sum_l e_l(x_{t+1}) a_{kl} b_l(t+1)
 \end{aligned}$$

The Backward Algorithm

We can compute all the $b_k(t)$ using dynamic programming:

Initialization:

- $\forall k \quad b_k(N) = 0$

Iteration:

$$b_k(i) = \sum_l e_l(x_{t+1}) a_{kl} b_l(t+1)$$

Termination:

$$P(X) = \sum_l \pi_l e_l(x_1) b_l(1)$$

Computational complexity

Complexity analysis for the Forward and Backward algorithms

- Time complexity $O(K^2N)$
- Space complexity $O(KN)$

Like for the Viterbi algorithm, underflows can be a problem:

- Method 1: Use sums of log
- Method 2: Rescale every few positions by multiplying by a constant

Posterior decoding

Using both Forward and backward algorithms, we can now calculate:

$$P(Q_t = k|X) = \frac{f_k(t)b_k(t)}{P(X)}$$

Posterior decoding

Using both Forward and backward algorithms, we can now calculate:

$$P(Q_t = k|X) = \frac{f_k(t)b_k(t)}{P(X)}$$

And from there we can deduce the most likely hidden state at a position i of a sequence X :

$$\widehat{Q}_t = \operatorname{argmax}_k P(Q_t = k|X)$$

Posterior decoding

Using both Forward and backward algorithms, we can now calculate:

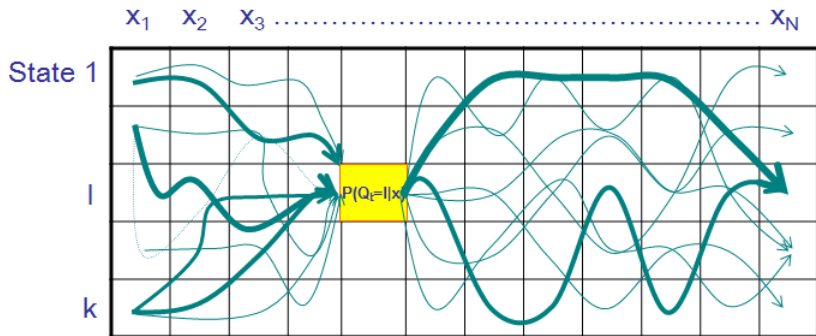
$$P(Q_t = k|X) = \frac{f_k(t)b_k(t)}{P(X)}$$

And from there we can deduce the most likely hidden state at a position i of a sequence X :

$$\widehat{Q}_t = \operatorname{argmax}_k P(Q_t = k|X)$$

- From there, the posterior decoding gives us a likelihood curve of states for each position.
- This is sometimes more informative than Viterbi path Q^*

Posterior decoding



Outline

- 1 Introduction to HMM
- 2 The decoding Problem
- 3 The learning Problem**
- 4 Conclusion

The learning problem

- If the hidden states are known, it is easy to guess the parameters:
 - Count the transitions to guess A
 - Count or measure the distribution of the observations in each states to guess the emission parameters B .
- If the parameters A , B and π are known, the sequence of hidden states can be guessed using Viterbi or the Forward-Backward algorithm.

The learning problem

- If the hidden states are known, it is easy to guess the parameters:
 - Count the transitions to guess A
 - Count or measure the distribution of the observations in each states to guess the emission parameters B .
- If the parameters A , B and π are known, the sequence of hidden states can be guessed using Viterbi or the Forward-Backward algorithm.
- We have a “Chicken and egg” problem.

The learning problem: Use the EM algorithm

This kind of problem can be solved using the EM algorithm (Cf. Lecture 6) to alternatively optimize both the model and the most likely path.

- **Initialization:** Guess the initial HMM parameters $M = \{A, B, \pi\}$
- Repeat until convergence:
 - **E-Step:** Compute the distribution over paths.
 - **M-Step:** Compute the maximum likelihood parameters.

The Baum-Welch algorithm

The Forward-Backward algorithm can be adapted into an EM like procedure to learn the model. This is known as the Baum-Welch algorithm.

Repeat until convergence:

- Compute the probability of each state at each position using forward and backward probabilities
 - This gives the expected distribution of the observations for each state using Bayes Theorem.
- Compute the probability of each pair of states at each pair of consecutive positions t and $t + 1$ using $\text{forward}(t)$ and $\text{backward}(t+1)$
 - This gives the expected transition counts.

$$\text{Count}(k \rightarrow l) = \frac{\sum_i f_k(t) a_{kl} b_l(t+1)}{P(X)}$$

Outline

- 1 Introduction to HMM
- 2 The decoding Problem
- 3 The learning Problem
- 4 Conclusion**

ARMA or HMM, when to use which ?

Type of data

- If your observations/data seem (or are known) to depend upon previous observations, use ARMA.
- If your observations/data are independent in time, use HMM.
- For HMM series, each state is assumed to have stationary properties. With ARMA series, it depends on the parameters.
- With cyclic data, ARMA models (and their evolved ARIMA version) are usually better.
- For stationary data with a single observable event, HMM are usually a bit too much and ARMA models are enough.
- For multi-dimensional data, HMM should be preferred.

ARMA or HMM, when to use which ?

Type of tasks

- ARMA models are dedicated to modeling time dependent events and to predict the evolution of a phenomenon in time.
- HMM usually perform poorly in long term predictions.
- For time data segmentation and modeling, HMM should be used.
- Both ARMA and HMM can be used to fill in missing data.
- For data sequences analysis other than time data, HMM are usually preferred.