World Journal Control Science and Engineering, 2013, Vol. 1, No. 1, 9-14
Available online at http://pubs.sciepub.com/wjcse/1/1/2

DOI:10.12691/wjcse-1-1-2

Some Algorithms for Large Hidden Markov Models

Sanaa Chafik*, Daoui Cherki

University Sultan Moulay Slimane, Laboratory of modelisation and calcul, Béni Mellal
*Corresponding author: san.chafik@gmail.com

Received June 30, 2013; Revised August 25, 2013; Accepted August 27, 2013

Abstract The Hidden Markov Model (HMM) has become increasingly popular in the last several years because it is used in a wide range of applications. There are some inherent limitations of this type of statistical model. The major limitation of HMM is large hidden state space, which limits their practical purview. The objective of this work is to reduce the task of solving some classical algorithms (Forward, Backward, Baum-Welch) by review of their theoretical aspects, offering faster improved algorithms based on the decomposition technique which represent a general approach to solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately.

Keywords: *Hidden Markov Model, Forward, Backward, Baum Welch, large hidden state space, divide and conquer, decomposition, communicating class, graph theory*

Cite This Article: Sanaa Chafik, and Daoui Cherki, "Some Algorithms for Large Hidden Markov Models." *World Journal Control Science and Engineering* 1, no. 1 (2013): 9-14. doi: 10.12691/wjcse-1-1-2.

1. Introduction

A Hidden Markov Model (HMM) is a doubly stochastic process, one of whose components is an unobservable Markov chain, the HMM have been successfully used for pattern recognition, speech recognition [1,2], Handwriting recognition [3,4,5], computational biology [6], Machine translation [7]. During the use of HMMs we are led to treat three fundamental problems Evaluation, Decoding and Learning [8].

The HMMs fall most often on a large dimension state space that makes interesting use of the algorithmic technique of divide and conquer their principle based on dividing a large problem into several similar problems which avoids the curse of dimensionality. In this direction, we will propose a decomposition method and improved algorithms to solve large HMMs.

This paper is organized as follows: Section 1 present a general introduction to Hidden Markov Models, their fundamental problems, and some classical algorithms algorithms (Forward, Backward, Viterbi, Baum-Welch). Section 2 details the curse of dimensionality problem and gives the principle of our proposed solution. Section 3 presents a new version of some HMM algorithms. Section 4 summarizes the complexity of the different algorithms.

2. Hidden Markov Model

2.1. Definition

The elements of a HMM are [9,10]: The Model formed by N states $S=\{S_1,S_2,...,S_N\}$; separate observations according to M symbols $O=\{O_1,O_2,...,O_M\}$; the matrix of transition probabilities is denoted by $A=[a_{ij}]$ where $a_{ij}=P(s_{t+1}=S_j \mid s_t=S_i)$ and $\sum_{j=1}^{N} a_{ij}=1$, $\forall i \in [1,N]$, that specifies the probability of transitioning from state S_i to state S_j ; the observation probability matrix or emission probability is denoted by $B=[b_j(m)]$ where $b_j(m)=P(o_t=O_m \mid s_t=S_j)$ that represents the probability of symbols emission O_m at the instant t by the state S_j ; the initial state distribution is denoted by $\Pi=[\pi_i]$ where $\sum_{i=1}^{N}\pi(i)=1$, that specifies the probability of being in state S_i at time zero.

Given appropriate values of N, M, A, B and Π , the HMM can be used as generator to give an observation sequence $O=\{O_1,O_2,...,O_T\}$, where T is the number of observations in the sequence.

We denote by λ the model parameters, where λ = (A, B, Π).

2.2. Fundamental Problems of HMM

There are three basic HMM problems [8] of interest that must be solved for the model to be useful in real-world application. These problems are the following:

Evaluation: Given the observation $O=O_1,O_2,...,O_T$ and an HMM model $\lambda=\{\Pi,A,B\}$, how do we compute the probability of O given the model $\lambda=\{\Pi,A,B\}$? Several techniques are developed in this direction: direct evaluation method, "Forward-Backward" procedure and Viterbi Algorithm.

Decoding: Given the observation $O = O_1, O_2, ..., O_T$ and an HMM model $\lambda = \{\Pi, A, B\}$, how do we find the state sequences that best explain the observation? There are several solutions to this problem: The local criterion, the global criterion and the Viterbi algorithm.

Learning: How do we adjust the model parameters $\lambda = \{\Pi, A, B\}$, to maximize $P(O \mid \lambda)$? The task is usually to derive the maximum likelihood estimate of the parameters of the HMM given the set of output sequences. No tractable algorithm is known for solving this problem exactly, but a local maximum likelihood can be derived efficiently using the Baum-Welch algorithm or the Baldi-Chauvin algorithm [11]. The Baum-Welch algorithm is a special case of the expectation-maximisation algorithm.

2.3. Classical Algorithms

Let $O = O_1, O_2, ..., O_T$ the sequence of observation wehre T represent the number of possible observations.

2.3.1. Forward-Backward Algorithm

To calculate the probability to engender the sequence of observation O given the model λ illustrated in equation (1), we make recourse to two procedures (Forward $\alpha_t(i)$ and Backward $\beta_t(i)$).

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \tag{1}$$

For each pair (state i, time t) we associate the Forward variable $\alpha_t(i)$ (2) which represents the probability of the partial observation sequence $\{O_1, ..., O_t\}$ (until time t) and state S_i at time t, given the model λ:

$$\alpha_t(i) = P(o_1 = O_1, ..., o_t = O_t, s_t = S_i \mid \lambda) \tag{2}$$

Algorithm 2.1.

***Step1*:** Initialization, for each $i \in [1, N]$

$$\alpha_1(i) = \pi_i \times b_i(o_1)$$

***Step 2*:** Induction, for $t \in [1, T-1]$ and $j \in [1, N]$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \times a_{ij}\right] b_j(o_{t+1}) \tag{3}$$

***Step 3*:** Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4}$$

Analogous to the Forward variable, just in the other direction, we define the Backward variable $\beta_t(i)$ given in (5) which represents the partial probability to observe all future events $\{O_{t+1}, ..., O_T\}$ knowing that we are in state S_i, at time t and given the model λ.

$$\beta_t(i) = P(o_{t+1} = O_{t+1}, ..., o_T = O_T \setminus s_t = S_i, \lambda) \tag{5}$$

Algorithm 2.2.

***Step 1*:** Initialization, for $i \in S$

$$\beta_T(i) = 1 \tag{6}$$

***Step 2*:** Induction, for $t \in [T-1, 1]$ and $i \in [1, N]$

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)\right] \tag{7}$$

2.3.2. Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence $S^* = \{S_1^*, ..., S_T^*\}$ of hidden states.

Algorithm 2.3.

***Step 1*:** Initialization, for t = 1 and $i \in [1, N]$

$$r_1(i) = \pi_i \times b_i(o_1) \text{ and } \psi_1(i) = 0 \tag{8}$$

***Step 2*:** Recursion, for $2 \le t \le T$ and $j \in [1, N]$

$$\begin{aligned} r_t(j) &= \max_{1 \le i \le N} r_{t-1}(i) \times b_j(o_t) \times a_{ij} \\ \psi_t(j) &= \arg\max_{1 \le i \le N} r_{t-1}(i) a_{ij} \end{aligned} \tag{9}$$

***Step 3*:** Termination

$$S_T^* = \arg\max_{1 \le i \le N} r_T(i) \tag{10}$$

***Step 4*:** Optimal state sequence backtracking

$$S^* = \{S_1^*, ..., S_T^*\} \tag{11}$$

Where $S_t^* = \psi_{t+1}(S_{t+1}^*), t = T-1, ..., 1$

2.3.3. Baum-Welch Algorithm

This is re-estimation algorithms that iteratively try to estimate the model parameters by maximizing the likelihood.

Baum-Welch algorithm [12] is summarized in three steps: first forward passes and backward pass followed by a smoothing pass.

To reestimate the values of the transition matrix A, the observation matrix B and the probability initialization Π we need to define two more auxiliary variables $\xi_t(i, j)$ given in (12) and $\gamma(i, t)$ given in (13), in addition to the Forward and Backward variables defined in a previous section. The variable $\gamma(i, t)$ represents the probability of being in state S_i at time t, given the observation sequence and the model, and the variable $\xi_t(i, j)$ represents the probability of being in state S_i at time t and in state S_j at t+1.

$$\xi_t(i, j) = P(s_t = S_i, s_{t+1} = S_j \mid O, \lambda) \tag{12}$$

$$\gamma(i, t) = \sum_{j=1}^{N} \xi(i, j) \tag{13}$$

The variables $\xi_t(i, j)$ and $\gamma(i, t)$ can be expressed as:

$$\gamma(i, t) = P(s_t = i / O) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \tag{14}$$

$$\xi_t(i,j)=\frac{\alpha(i,t)\times b_i(o_{t+1})\times a_{ij}\times\beta(j,t+1)}{\sum_{k=1}^{N}\alpha(k,t)\beta(k,t)} \quad (15)$$

Using the above formulas we can give a method for reestimation of the parameters (π, A, B) of an HMM:

$$\pi_1(i)=\gamma(i,1) \quad (16)$$

$$a_{ij}(t)=\frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma(i,t)} \quad (17)$$

$$b_i(o_m)=\frac{\sum_{t=1\cap o_t=o_m}^{T}\gamma(i,t)}{\sum_{t=1}^{T}\gamma(i,t)} \quad (18)$$

3. Problematic and Solution

3.1. The Curse of Dimensionality

The statistical learning algorithms such as those dedicated to hidden Markov chains they are suffering from the exponentially increase of the cost when the volume of data grows, which is known as the curse of dimensionality [13]. In this direction we thought using the "Divide and Conquer" algorithmic technique to alleviating the curse of dimension Divide and Conquer.

It is a Latin proverb "to destroy a group of enemies, split into subsets that are crushed one by one, the fusion of the results while not posing much difficulty".

The term Divide and Conquer algorithmic technique [14,15] yields elegant, simple and very efficient algorithms, their principle is based on dividing a large problem into several similar problems which avoids the curse of dimensionality. It is preceded in three steps:

***Step 1*:** Partitioning the object of size N in P sub-objects.

***Step 2*:** Solving the problem for each sub-objects.

***Step 3*:** Merging the results obtained to get a solution of the initial problem.

3.2. Principe of Decomposition

Decomposition technique [9] consists of the following steps. First, the algorithm of decomposition to levels is applied, thereafter the restricted HMMs are constructed, eventually, we combine all the partial solutions in order to construct the global solution of the HMM.

In this section, we consider HMM, Let G=(S, U) be the graph associated with the HMM, that is, the state space represents the set of nodes and $U=\{(i,j)\in[1,N]^2 : a_{ij}=P(s_{t+1}=S_j \mid s_t=S_i)>0\}$ the set of directed arcs.

3.2.1. Decomposition into Levels

The state space can be partitioned into strongly connected classes $C_1, C_2, ..., C_H$. Note that the strongly connected classes are defined to be the classes with respect to the relation on G defined by: S_i is strongly connected to S_j if and only if $S_i = S_j$ or there exist a directed path from S_i to S_j and directed path from S_j to S_i. There are many good algorithms in graph theory for the computations of such partition, e.g., see [16]. Now we construct by induction the levels of the graph G. The level L_0 is formed by all classes C_i such that C_i is closed, that is, any arc emanating from C_i has both nodes in C_i. The path level L_p is formed by all classes C_i such that the end of any arc emanating from C_i is in some level $L_{p-1}, L_{p-2}, ..., L_0$.

Remark 3.1. Let C_i be strongly connected class in the level L_p then C_i is closed with respect to the restricted HMM to the state space $S-(L_{p-1}, L_{p-2}, ..., L_0)$.

It is clear that, from Remark 3.1, the following algorithm finds the levels.

Algorithm 3.1.

Beginning

$\Omega \leftarrow S;\ p \leftarrow 0;\ \mathrm{L}_p = \{C_i : C_i \text{ closed at } \Omega\}$

Repeat

$\Omega \leftarrow \Omega \setminus \mathrm{L}_p$

If $\Omega \neq \varnothing$ then

$\mathrm{L}_{p+1} = \{C_i : C_i \text{ closed for HMM restricted to } \Omega\}$

$\mathrm{p} \leftarrow p+1$

End if

Until $\Omega=\varnothing$

Example 3.1.

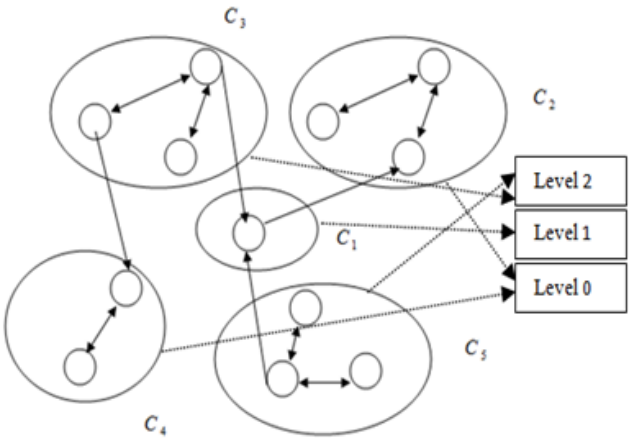

Figure 1. Construction of the levels

3.2.2. Construction of the Restricted HMM

Let (C_{pk}), $k\in\{1,2,...,K(p)\}$ be the K^{th} strongly connected class corresponding to the nodes in level p, where K(p) represent the maximum of the classes included in the p^{th} level.

Remark 3.2. The construction of restricted HMM for each level depends on the type of HMM problems; we will details this point in the sequel.

3.2.2.1. Restricted HMM for Decoding Problem Using Forward Algorithm

Construction of the restricted HMM in level the L_n : For each $k = 1,2,\ldots K(n)$, we denote by HMM_{nk} the restricted HMM corresponding to the class C_{nk} that is the restricted HMM in which state space restricted is $S_{nk} = C_{nk}$; the same M symbols $O_1, O_2, ..., O_M$; the matrix of transition probabilities and the matrix of observation probability restricted to S_{nk}.

Construction of the restricted HMM in level L_p , $p = n-1,...,0$: For each $k = 1,2,\ldots K(p)$, we denote by HMM_{pk} the restricted HMM corresponding to the class C_{pk} .

Let $\Gamma^{-1}(C_{pk})$ be the set of predecessors for each state $i \in C_{pk}$. The restricted HMM_{pk} defined by:

State space: $S_{pk} = C_{pk} \cup \Gamma^{-1}(C_{pk})$.

Matrix of transition probabilities $A_{pk} = \left[a_{ij}\right]_{pk}$ for each $j \in C_{pk}$, $i \in S_{pk}$; where $a_{ij} = P(s_{t+1} = S_j \mid s_t = S_i)$.

The same symbols $O = \{O_1, O_2, ..., O_M\}$.

The probability distribution of the initial state $\Pi_{pk} = \left[\pi_i\right]_{pk}$; for ach $i \in C_{pk}$ where $\pi_i = P(s_1 = S_i)$.

Matrix of observation probability $B_{pk} = \left[b_j(m)\right]_{pk}$; for each $j \in C_{pk}$ where $b_j(m) = P(o_t = O_m \mid s_t = S_j)$.

3.2.2.2. Restricted HMM for Decoding Problem Using Backward Algorithm

Construction of the restricted HMM in level L_p, $p > 0$: For each $k = 1,2,\ldots K(p)$, we denote by HMM_{pk} the restricted HMM corresponding to the class C_{pk} . Let $\Gamma(C_{pk})$ be the set of successors for each state $i \in C_{pk}$ .The restricted HMM_{pk} defined by:

State space: $S_{pk} = C_{pk} \cup \Gamma(C_{pk})$.

Matrix of transition probabilities $A_{pk} = \left[a_{ij}\right]_{pk}$: for each $i \in C_{pk}$, $j \in S_{pk}$, where $a_{ij} = P(s_{t+1} = S_j \mid s_t = S_i)$.

The same symbols: $O = \{O_1, O_2, ..., O_M\}$

Matrix of observation probability $B = \left[b_j(m)\right]_{pk}$; for each $j \in S_{pk}$ where $b_j(m) = P(o_t = O_m \mid s_t = S_j)$.

4. Improved Algorithms

4.1. Improved Forward Algorithm

We denote by $\alpha_{t,pk}(i)$, $t \in \{1,...,T\}$, p=n,...,0 and k={1,..,K(p)} the Forward variable in state $i \in C_{pk}$.

Lemma 4.1. Let $j \in C_{pk}$, the Forward variable for state j at time t+1 is defined by:

$$\alpha_{t+1,pk}(j) = \left[\sum_{i \in S_{pk}} \alpha_{t,pk}(i) \times a_{ij}\right] b_j(o_{t+1}) \qquad (19)$$

Proof. From equation (3) to calculate the value $\alpha_{t+1}(j)$ we need only the states i such as $P(x_{t+1} = j \mid x_t = i) \neq 0$, $\forall\ j \in C_{pk}$, these states belongs to the original set states of the class C_{pk} or $i \in \Gamma^{-1}(C_{pk})$.

Remark 4.1. To calculate the Forward variable $\alpha_{t+1,pk}(j)$ we need the Forward variable $\alpha_{t,pk}(i)$ for each $i \in \Gamma^{-1}(C_{pk})$, therefore, we always need some values that have been already calculated in the upper levels.

Algorithm 4.1.

***Step1*:** Initialization

For p=n,...,0 and $k = 1,2,\ldots K(p)$; let $i \in C_{pk}$

$$\alpha_{1,pk}(i) = \pi_i \times b_i(o_1) \qquad (20)$$

***Step 2*:** Iteration

For $t \in [1,\ T-1]$, p=n,...,0 and $k = 1,2,\ldots K(p)$; let $j \in C_{pk}$

$$\alpha_{t+1,pk}(j) = \left[\sum_{i \in S_{pk}} \alpha_{t,pk}(i) \times a_{ij}\right] b_j(o_{t+1}) \qquad (21)$$

***Step 3*:** Termination

$$P(O \mid \lambda) = \sum_{p=0}^{n} \sum_{k=1}^{K(p)} \sum_{i \in C_{pk}} \alpha_{T,pk}(i) \qquad (22)$$

4.2. Improved Backward Algorithm

We denote by $\beta_{t,pk}(i)$; $t \in \{1,...,T\}$, p=0,...,n and k={1,..,K(p)} the Backward variable in state $i \in C_{pk}$.

Lemma 4.2. Let $i \in C_{pk}$, The Backward variable for the state i at time t is defined by:

$$\beta_{t,pk}(i) = \left[\sum_{j \in S_{pk}} a_{ij} b_j(o_{t+1}) \beta_{t+1,pk}(j)\right] \text{ if } i \in C_{pk} \qquad (23)$$

Proof. From equation (7) to calculate the value $\beta_t(i)$ we need only the states j such as $P(x_t = j \mid x_{t-1} = i) \neq 0 \quad \forall\ i \in C_{pk}$, these states belongs to the original set states of the class C_{pk} or $i \in \Gamma(C_{pk})$.

Remark 4.2. To calculate the Backward variable $\beta_{t,pk}(i)$ we need the Backward variable $\beta_{t+1,pk}(j)$ for each $j \in \Gamma(C_{pk})$, Therefore, we always need some values that have been already calculated in the lower levels.

Algorithm 4.2.

***Step1*:** Initialization

For p=0,...,n and $k = 1,2,\ldots K(p)$; let $i \in C_{pk}$

$$\beta_{T,pk}(i) = 1 \quad (24)$$

***Step 2*:** Iteration

For $t \in [T-1,1]$, p=0,...,n and $k = 1,2,\ldots K(p)$; let $i \in C_{pk}$

$$\beta_{t,pk}(i) = \left[\sum_{j \in S_{pk}} a_{ij} b_j(o_{t+1}) \beta_{t+1,pk}(j) \right] \quad (25)$$

4.3. Improved Baum Welch Algorithm

To describe the Baum-Welch algorithm, we need to define two more auxiliary variables. These variables that are defined in a previous section can be expressed in terms of the forward and backward variables, so it is clear that the improvements made on Forward and Backward algorithms remain useful for the algorithm Baum Welch.

5. Complexity

The classical Forward algorithm generate N (N +1) (T-1) + N multiplications and N (N-1) (T-1) additions, it takes on the order of N^2T computations. The same for Backward algorithm takes on the order of $2N^2T$ computations, which represents the quadratic complexity as a function of the number of hidden state.

In general the complexity of the algorithm decomposed into sub-problems is equal to:

$$T(N) = P(N) + \sum_{i=1}^{E} T_i(\frac{N}{b_i}) + F(N) \quad (26)$$

Where:

$T(N)$: The complexity of algorithm.

$F(N)$: The time required to fuse the solutions.

$P(N)$: The time required to partition the object of size N in sub-problems.

N: The size of the problem.

$T_i(\frac{N}{b_i})$: The execution time for each sub-problem of size $\frac{N}{b_i}$

E: The number of the sub-problems

Using our method the fusion time is negligible and the decomposition time is proportional to the size N of the problem, then $F(N) + P(N)$ on the order N. There are several techniques to solve the equation (26): Master Theorem [17], Akra Bazzi theorem [18].

The Akra Bazzi theorem is best adapted to our case, because he is the most general.

Theorem 5.1.

$$T(N) = \sum_{i=1}^{k} a_i T(b_i N) + g(N) \quad (27)$$

Let p be the unique real number for which

$$\sum_{i=1}^{k} a_i b_i^p = 1 . \quad (28)$$

$a_i > 0$: represents the number of sub-problem that have the size $b_i N$, where $b_i \in (0,1)$. Then:

$$T(N) = \Theta(N^p (1 + \int_1^N \frac{g(u)}{u^{p+1}} du)) \quad (29)$$

To calculate the complexity $T(N)$ given in equation (27), we must find the value of p that satisfies the relation in equation (28).

In general, the calculates of the Forward variable $\alpha_{t+1,pk}(j)$ or the variable Backward $\beta_{t,pk}(i)$ is done at the original states of each class, then the relation illustrated in equation (27) does ensure that if and only if p = 1.

$$T(N) = \Theta(N \ (1 + \int_1^N \frac{u}{u^2} du)) = \Theta(N + N \int_1^N \frac{1}{u} du) = N \log(N)$$

Which represents a quasi-linear complexity as a function of the number of state.

6. Conclusion

In this paper we have attempted to present the theory of hidden Markov models, it has been our purpose to focus on the curse of dimensionality problem; hence we have used the algorithmic technique of divide and conquer in order to propose some faster improved algorithms to solve the fundamental problems of large HMMs.

References

[1] Pellegrini T., Duée R., "Suivi de Voix Parlée grace aux Modèles de Markov Cachés," *IRCAM*, PARIS, 2003.

[2] Juang B. H.; Rabiner L. R., "Hidden Markov Models for Speech Recognition," *Technometrics*, Vol. 33, No. 3. pp. 251-272. Aug 1991.

[3] Ben Amara N., Belaïd A., Ellouze N., "Utilisation des modèles markoviens en reconnaissance de l'écriture arabe état de l'art," *Colloque International Francophone sur l'Ecrit et le Document (CIFED'00)*, Lyon, France, 2000.

[4] Nakai M., Akira N., Shimodaira H., Sagayama S., "Substroke Approach to HMM-based On-line Kanji Handwriting Recognition," *Sixth International Conference on Document Analysis and Recognition (ICDAR 2001).* pp.491-495. 2001.

[5] Ramy Al-Hajj M., Mokbel C., Likforman-Sulem L., "Reconnaissance de l'écriture arabe cursive: combinaison de classifieurs MMCs à fenêtres orientées," Université de Balamand, Faculté de Génie, pp 1-6. 2006.

[6] Krogh A. ,Mian I. S., Haussler D., "A Hidden Markov model that finds genes in E.coli DNA," *Nucleic Acids Research*, Vol. 22, No. 22. 1994.

[7] Morwal S., Jahan N., Chopra D., "Named Entity Recognition using Hidden Markov Model (HMM)," *International Journal on Natural Language Computing (IJNLC)* Vol. 1, No.4. 2012.

[8] RABINER R., "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceding of the IEEE*, Vol. 77, No. 2. 2010.

[9] Daoui C., "Decomposition des problèmes de decision markoviens," Thesis of doctorat in Faculty of science Rabat , pp 64-69. 2002.

[10] Jaafer S. A. , Mohamed I. O., Elhafiz M. M., "Text-Independent Speaker Identification Using Hidden Markov Model," *World of Computer Science and Information Technology Journal (WCSIT)*, ISSN: 2221-0741, Vol. 2, No. 6, 203-208. 2012.

[11] Baldi P., Chauvin Y., "Smooth On-Line Learning Algorithms for Hidden Markov Models," *Neural Computation 6*, 307-318. 1994.

[12] Khreich W., Granger E, Miri A., Sabourin R., "On the memory complexity of the forward–backward algorithm," *Pattern Recognition Letters 31*. 2010.

[13] Rust J., "Using Randomization to Break the Curse of Dimensionality," *Econometrica*, Vol. 65, No. 3, pp. 487-516. May 1997.

[14] Canet L., "Algorithmique, graphes et programmation dynamique," pp 23-25. 2003.

[15] Van Caneghem M., "Algorithmes recursifs Diviser pour régner,". Janvier, 2003.

[16] Gondran M., Minoux M., "Graphes et Algorithmes," Paris 2nd edition. 546 p. 1990.

[17] ROURA S., "Improved Master Theorems for Divide-and-Conquer Recurrences," *Journal of the ACM*, Vol. 48, No. 2, pp. 170-205. March 2001.

[18] Drmota M., Szpankowski W., "A Master Theorem for Discrete Divide and Conquer Recurrences," Austrian Science Foundation FWF Grant No. S9604. 2011.