Viterbi Algorithm and Its Application to Indonesian Speech Recognition

Z Hatala* and F Puturuhu

1Politeknik Negeri Ambon

e-mail: dzulqarnaenhatala@gmail.com

Abstract. An algorithm used to extract HMM parameters is revisited. Most parts of the extraction process are taken from implemented Hidden Markov Toolkit (HTK) program under the name Hinit. The HMM model is introduced briefly based on the theory of Markov Chain. We schematically outline the Viterbi method implemented by Hinit. The iterative formal definition of the method which directs computer implementation is reviewed. We also illustrate the method calculation precisely using manual calculation and extensive graphical illustration. The distribution of observation probability used is simply independent Gaussians. The performance of the algorithm for phone recognition rate on small Indonesian vocabulary is 80%, while the result is near perfect 95% for word recognition.

Keywords: Viterbi, algorithm, speech recognition

1. Introduction

Hidden Markov Toolkit (HTK) [1] is state of the art in speech recognition. HTK is released with its source code open that provide advantages to all researchers who want to study the implementation of ASR. HTK is written in ANSI C and runs on any modern OS. It is currently used in many speech laboratories around the world and it is also used for teaching in numbers of Universities. Even its source code is available but HTK is considerable a big project with thousands of lines of C code. An effort must be done to use the libraries of HTK and to extend its functionalities. Thus, by studying directly into its C code intensively, researchers can effectively apply HTK to a new language or even correcting errors encounter while applying. Looking deep inside into the existing C code of HTK is mandatory to master and understand its data structures and code sequences. HTK is an integrated suite of software tools for building and manipulating continuous density Hidden Markov Models (HMMs). It consists of a set of library modules and a set of tools (executable programs). One of the programs in HTK is Hinit, which employs the Viterbi algorithm to estimate the parameter of HMM. Viterbi algorithm is introduced in [2] and it was originally used to decode convolutional code in fields of telecommunication theory. Since then, the method has implemented in many fields of applications including speech recognition to train parameter of Hidden Markov Model [3].

The method also is known in the literature as Viterbi extraction, Viterbi segmentation, Baum Viterbi, segmental K-means [4], classification EM, hard EM[5], MAP path estimator [6], etc. The convergence property of this algorithm is also elaborated in [6]. To the best of our knowledge, the
Viterbi algorithm is never graphically illustrated in connection with the parameter estimation of HMM. It is not even further tested in Indonesian speech recognition especially in noisy conditions. It is important to present research where an algorithm works and solves the problem sufficiently. In the next section, we formalize the notation and theory of HMM and the Viterbi algorithm. Then we illustrate graphically the parameter estimation of HMM using MFCC[7] data. Later we present the performance of the algorithm when applied to noisy Indonesian speech recognition. Finally, we conclude this writing by the important results and notes found in this text.

2. Definitions and terms

2.1. Hidden Markov Model
HMM[3] is a discrete-time Markov chain (DTMC) [8] that emits an observation when sitting in one of its states. An HMM is completely defined by the initial and transition probability of the Markov chain (MC) and conditional observation probabilities. The distribution of observation only depends on the current state of the underlying Markov chain. Set of states that emit an observation is called emitting states, while non-emitting states could be an entry or start state, absorption or exiting state and intermediate or delaying state. The entry state is the state with initial probability 1. HMM has applications in many fields such as thermodynamics [9] and speech recognition[3]. The observations of HMM are used to model the sequence of speech feature vectors like Mel Frequency Cepstral Coefficients (MFCC) [7]. MFCC is assumed to be generated by emitting states of the underlying DTMC. According to [3] an HMM is completely determined by parameter set:

$$\lambda = (A, B, \pi)$$  \hspace{1cm} (1)

Where A is the transition probability matrix, \(\pi\) is the initial probability vector of DTMC and B is the observation probability vector.

2.2. Viterbi algorithm
As taken from [3], the iterative algorithm for finding the best state sequence is divided into stages:

- **Initialization**, set scores from the initial state into states that generate first observation:

$$\delta_1(j) = \pi_j b_j(O_1)$$  \hspace{1cm} (2)

- **Recursion**, propagating next observation and finding the maximum score for each destination state \(j\):

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t),$$  \hspace{1cm} (3)

And save state entries in backtrack matrix:

$$\psi_t(j) = \arg \max_i \max_j \delta_{t-1}(i) a_{ij},$$  \hspace{1cm} (4)

$$2 \leq t \leq T - 1, \hspace{0.5cm} 1 \leq j \leq S$$

- **Termination**, finding the best score and the end state of the best path:

$$P^* = \max_i \delta_T(i)$$  \hspace{1cm} (5)

$$q_T^* = \arg \max_i \delta_T(i)$$  \hspace{1cm} (6)

- **Backtracking**, recursively aligns path sequence from best path end state:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \hspace{0.5cm} t = T - 1, T - 2, ..., 1$$  \hspace{1cm} (7)

The time complexity of the algorithm for a fixed number of states and \(T\) number of observations is \(O(T)\). The memory requirement for this algorithm has a size of \(T \times S\).
2.3. Hidden Markov Toolkit
The components of HTK are depicted in Figure 1. In the outer block, it is outlined libraries of HTK which were written in ANSI C. In the central block, HTKTools are ready to use programs such as HInit, HCopy, HRest, HVite and so on. The operation of HInit is summarized in Figure 2 and further explained in section 3.

![Figure 1. Components of HTK](image)

![Figure 2. Viterbi algorithm on HInit](image)

3. Illustration of the algorithm
To illustrate the Viterbi method, let’s consider a model of HMM that is popularly used in HTK. The underlying DTMC is depicted in Figure 3.

![Figure 3. HTK’s HMM model](image)

This is a four states HMM with a start state 1, emitting states 2, 3 and an absorption state 4. This HMM is used to model a sequence of MFCC associated with a single subword or phoneme in speech recognition. For example, the Indonesian digit ‘satu’ (English: one) can be broken into its subwords
as s-ah-t-uh. In this case, subword s has its own HMM definition, so are ah, t and uh. The HMM has property that the initial probability of state 1 is 1, and the only transition from state 1 is limited only into state 2.

3.1. Observation data
Consider the observations data in Table 1. At each time \( t=1,2,\ldots,12 \), a vector:

\[
\mathbf{x} = \begin{bmatrix} x_{t1} & x_{t2} & x_{t3} \end{bmatrix}, \quad t = 1, \ldots, 12
\]

is assumed to be generated, either by state 2 or state 3 of DTMC in Figure 3.

| \( t \) | \( x_{t1} \) | \( x_{t2} \) | \( x_{t3} \) |
|---|---|---|---|
| 1 | -1.115696192 | -1.014122963 | -0.244220227 |
| 2 | -0.971390247 | -0.823073566 | -0.661046565 |
| 3 | -0.399597883 | -0.510152936 | -0.782005250 |
| 4 | 0.652983546 | 0.032239955 | -0.676724792 |
| 5 | 1.174029231 | 0.492249459 | -0.531797767 |
| 6 | 1.049691796 | 0.873368561 | -0.658340454 |
| 7 | 0.582641065 | 1.254104257 | -0.705632925 |
| 8 | -0.179972363 | 1.284092903 | -0.751379013 |
| 9 | -0.513338625 | 0.562875986 | -0.449841380 |
| 10 | -0.466798663 | -0.551046491 | -0.449841380 |
| 11 | -0.309872597 | -1.142085671 | -0.048693269 |
| 12 | -0.206728458 | -1.149199367 | 0.414609402 |

Data in Table 1 are taken from 3 last MFCC’s coefficients generated by HTK program HCopy operated on a recorded sound file. We choose 3 out of 39 coefficients merely to simplify graphical illustration and has no connection with any applications.

3.2. Observation distribution
Each item in the filtered MFCC vector is assumed to be Gaussian. By independency, the observation probability for the MFCC vector is simply the multiplication of the item. In this case, the observation parameter in equation (1) which can be written as:

\[
\mathbf{B} = \begin{bmatrix} b_2(t) & b_3(t) \end{bmatrix}
\]

Which the logarithm of its entry is simply calculated as in (10) and (11):

\[
\log(b_j(t)) = -\frac{1}{2} \left[ G_j + \sum_i \left( \frac{x_{ij} - \mu_j}{\sigma^2_{ji}} \right)^2 \right]
\]

\[
G_j = 3\log(2\pi) + \sum_i \log \sigma^2_{ji}
\]

\[
j = 2,3, \quad i = 1,2,3, \quad t = 1,2,\ldots,12
\]

Equation (10) can also be formulated using MATLAB’s notation:

\[
\log_{\_\_j \_t} = \text{sum} \left( \log \left( \text{normpdf} \left( \mathbf{x}_{\_t}; \text{mu}_j, \text{var}_j \right) \right) \right)
\]

In the next sections, we’re going to extract the parameter set for HMM with DTMC in Figure 3, based on observation in Table 1 using procedures at Figure 2.
3.3. Uniform segmentation and parameter initialization

The initial probability \( \pi \) in (1) is not estimated since at initial time \( t=0 \) the DTMC in Figure 3 always starts at state 1 with probability 1. Back to Figure 2 again, the initial value of transition probability matrix \( A \) in (1) is needed. Let assume this matrix as in (13).

\[
A = \begin{bmatrix}
0.0 & 1.0 & 0.0 & 0.0 \\
0.0 & 0.1 & 0.4 & 0.5 \\
0.0 & 0.8 & 0.1 & 0.1 \\
0.0 & 0.0 & 0.0 & 0.0
\end{bmatrix}
\]  

(13)

All entries of the 4th row in (13) are zero because state 4 is the absorption state. Uniform segmentation states that the number of observations assumed to be generated equally between states. Since we have only 2 emitting states for 12 observations, then the first 6 observations are segmented into state 2 and the rest is assumed to be generated by state 3. By this segmentation, observation distribution in (9), means and variances of Gaussians are calculated as in Table 2:

| state \( j \) | Means \( \hat{\mu}_j \) | Variances \( \hat{\sigma}_j^2 \) |
|-------------|-----------------|-----------------|
| 2           | 0.0650 -0.1583 -0.5923 | 0.8717 0.4701 0.0295 |
| 3           | -0.1823 0.0432 -0.3820 | 0.1322 1.0758 0.1880 |

The parameter initialization step is complete since all parameters \( A, B \) in (1) is identified. In the next subsection, we illustrate the Viterbi algorithm based on these initial values.

3.4. Viterbi Extraction

An illustration of the estimation process is presented in Figure 4 for time \( t=1 \) to \( t=12 \). Values inside circles are representing delta in (3). Green circles are for state 2 and red circles are for state 3, black circles are starting and absorption states.

Initialization, the cost for the first propagation comes only for observation generated by state 2.
\[
\delta_1(2) = \log\left( P(x_1 | q_1 = 2) \right)
\]
\[
= -4.1817
\]

The only backtrack entry exists, is for state 2:
\[
\psi_1(2) = 1
\]

Recursion, for all next observations from time \( t=2 \) until \( t=12 \):
\( t=2 \): At this time, observation can be generated by staying in state 2 or moving to state 3 with total cost each:
\[
\delta_2(2) = -7.1013, \ \delta_2(3) = -8.3930
\]

The entries in backtrack matrix each:
\[
\psi_2(2) = 2, \ \psi_2(3) = 2
\]

\( t=3 \): at these times propagating into states 2 or 3 can be reached either from states 2 or 3. For these two alternatives, we select the source with the best probability or minimal cost to calculate (3). We remove not survival path components \( q_2 = 2, q_3 = 2 \) and \( q_2 = 3, q_3 = 3 \). Eliminated path components are shown in Figure 4 as a black arrow crossed by a red line. Now we have values for (3) and (4) as:
\[
\delta_3(2) = -10.59, \ \delta_3(3) = -10.81 \quad \psi_3(2) = 2, \ \psi_3(3) = 2
\]

By iteratively continuing in this fashion we have all values for (3) and (4) for every time index from \( t=3 \) until \( t=12 \). Again, all these steps are illustrated in Figure 4.

Termination: maximum probability or minimum cost when generating last observation at time \( t=12 \) and also entering exit state 4 is achieved via state 3.
Figure 4. Viterbi iteration for time $t=1$ to $t=12$
Backtracking: The best path sequence can be traced back from (14) using (7) by following all values inside the generated backtrace matrix, which is shown in Table 3.

Table 3. Backtrace matrix

| t  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|
| state | 2 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | -  |
| 3 | - | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | -  |
| 4 | - | - | - | - | - | - | - | - | - | - | - | 3  |

Backtracking is illustrated as following green arrows in figure 4. The same goal is achieved by following the grey shaded cell in Table 3 from the last time index t=13. Using these ways, at this iteration, we have for (7) our best path sequence of states in Table 4.

Table 4. Optimal states sequence

| t  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|
| state | 1 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 3 | 3 | 4 |

3.5. HMM Parameter Updating

Using state segmentation results in Table 4 and observation data in Table 1, HMM parameters now can be updated. From Table 4 the transition between state can be counted and be normalized into a new transition probability which is shown in (15)

$$ A_{t+1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & \frac{3}{7} & \frac{4}{7} & 0 \\ 0 & \frac{3}{5} & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} $$

(15)

Gaussian parameters can be determined the same way entries of Table 2 is calculated. But instead of using uniform segmentation, now Viterbi segmentation in Table 4 is used. Updated parameters are shown in Table 5.

Table 5. Viterbi Segmentation : Gaussian parameters

| state j | Mean $\hat{\mu}_j$ | Variances $\sigma^2_j$ |
|---------|-------------------|----------------------|
| 2       | 0.0204 0.0419 -0.5676 | 0.7633 0.6644 0.0260 |
| 3       | -0.1694 -0.1969 -0.3745 | -0.1516 0.9162 0.2292 |

Since this is the first Viterbi iteration, based on Figure 2 we have to perform an iteration at least once again. After the second iteration then we can decide if the whole process is convergent based on (16):

$$ |P_{k+1}^* - P_k^*| \leq \varepsilon $$

(16)

In (16), $P_k^*$ is the value of (5) at kth iteration. The Viterbi process is terminated whenever this probability difference between the two consecutive iterations is less than $\varepsilon$. The next iterations of Viterbi segmentation are performed the same way which is illustrated in figure 4. That is the parameters estimated from the previous iteration are used to segment the states of the next iteration. In our example of HMM with underlying DTMC on figure 2, we will stop at iteration 5th if:

$$ \varepsilon = 0.0001 $$

Each iteration values of (16) for our example is presented in Table 6.
Table 6. Iterations convergence values

| kth iteration | $P_k^*$ | $P_k^* - P_{k-1}^*$ |
|---------------|--------|---------------------|
| 1st Iteration | -44.08260 | -                   |
| 2nd Iteration | -35.15588 | 8.9267              |
| 3rd Iteration | -29.51328 | 5.6426              |
| 4th Iteration | -18.87455 | 10.6387             |
| 5th Iteration | -18.87455 | Less than 0.0001    |

4. Results
The algorithm is tested on a small set Indonesian corpus which is recorded on the moderated noisy real daily environment. This system is trained to recognize limited commands, which later parsed to toggle four high voltage electrical switches. Hinit is used to estimate the parameters of HMM and HVite, which is also from HTK is used as a recognizer. Two recognition performances are collected. For the monophonic subword system, the phone recognition rate achieves 80%. And for word recognition rate we have nearly 95% score. I.e. when applied to speech based electrical switch application, the trained system can recognize all commands completely.

5. Conclusions
In this tutorial, the detail of the Viterbi Algorithm to estimate HMM parameters are presented. The method adopted the implementation used by HTK program HInit. The data used is limited to small vocabulary Indonesian speech recognition. The parameters estimated are robust even under moderated noisy environments. Nevertheless, we have shown the case where the algorithm suitably fit the application of interest.

Acknowledgements
This work was supported by the Ministry of Research, Technology and Higher Education, Ristekdikti, Indonesia.

References
[1] Young S, Gunnar E, Mark G, Hain T and Kershaw D 2015 The HTK Book version 3.5 alpha (Cambridge University)
[2] Forney D 1973 The Viterbi Algorithm Proc. IEEE 61 268–78
[3] Rabiner L R 1989 A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition Proceedings of the IEEE vol 77 (IEEE) pp 257–86
[4] Juang B H and Rabiner L R 1990 The segmental K-means algorithm for estimating parameters of hidden Markov models Acoust. Speech Signal … 38 1639–41
[5] Allahverdyan A and Galstyan A 2011 Comparative Analysis of Viterbi Training and Maximum Likelihood Estimation for HMMs Neural Information Processing Systems
[6] Caliebe A and Rösler U 2002 Convergence of the Maximum A Posteriori Path Estimator in Hidden Markov Models IEEE Trans. Inf. Theory 48 1750–8
[7] Davis S B and Mermelstein P 1980 Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences IEEE Trans. Acoust. 28 357–66
[8] Osaki S 2002 Applied Stochastic System Modeling
[9] Bechhoefer J 2015 Hidden Markov models for stochastic thermodynamics IOPScience New J. Phys.