Viterbi algorithm

Tobias Schellien, Philipp Copei, Robert Koczula, Akanchha Choudhary

Abstract—This paper gives an overview about the Viterbi algorithm. It also talks about how the Hidden Markov Model (HMM) has been used to analyse the algorithms result, using a test sequence of observed values over a period. This paper also talks about the occurrence of errors and how these can be minimized using proper verification of the problematic sets of emissive values and calculated states. The Viterbi algorithm was invented by Andrew J. Viterbi in 1967 for decoding convolution codes along with analysing the error probability of such codes. One must apply this method because the Viterbi algorithm is highly useful for achieving low error rates during data transmission by using a maximum likelihood algorithm which matches the most expected pattern from the received data, hence giving the most optimal result. There are real time applications where the algorithm can be applied for example such as speech recognition, finding health conditions of patients, event detection in video etc. We have seen how the Viterbi algorithm takes a set of hidden states as input and how the expected outcoming sequence can be analysed for its proximity to the true state.

Index Terms—Viterbi algorithm, Viterbi, algorithm, HMM, IEEE, journal, paper

1 Introduction

The Viterbi algorithm is considered one of the most widely used dynamic programming algorithms for determining the expected sequences of hidden states in a given Hidden Markov Model (HMM) and an observed sequence of symbols. This path of a sequence is also known as the Viterbi path. The name 'Viterbi' was derived from the inventor of the algorithm, Andrew J. Viterbi. It is the most commonly used method to decode convolution codes. It is recommended that the trellis diagram be followed before the Viterbi algorithm is applied as it helps determine the best path to the received data bit sequence. It is most useful when one wants to calculate the most likely path through the state transitions of these models over time. These also play an important role in other fields like bioinformatics since they can be used, among other things, to deduce the actual sequence of a DNA segment for possible hidden states. It can be used almost anywhere for pattern recognition. The Markov model helps to know how to move between the states of automation. In HMM, the state is not directly visible, but the output/observations, dependent on the state, are visible. Each state has a probability distribution over the possible output. The sequence of observations generated by an HMM gives some information about the sequence of states, which further helps determine the most desired path.

1.1 Motivation

Suppose someone plans a journey for the route from the West Coast to the East Coast of the United States. There are four possible cities they can start from on the East Coast. Seattle, Newport, San Francisco, and Los Angeles. They only want to drive less than 600 miles per day. On the 1st day, it causes them to end up in four possible cities—Boise, Salt Lake, Las Vegas, or Tusan. On the 3rd day, they end up covering Casper, Denver, Albuquerque, and El Paso. So this will take them approx. 7 days with the target of covering less than 600 miles. Hence, to solve this huge time-taking problem, they try to find the shortest way to cover all the cities. One way to find the shortest path is to make a long list of all the possible routes that they can take and then take the sum of their one-day mileage to get the total mileage for each of the routes. But, as you can see, this is just not the convenient way because there are lots of paths and routes, which would be a very tedious task to do. Here, the Viterbi

algorithm can play a role in finding the shortest path. There are many such challenges in the world of communications as well as in decoding convolution codes, which are used both in CDMA, GSM digital cellular dial-up modems, satellites, deep-space communication, and other wireless LANs over noisy digital communication links. These motivate us to get deeper into such algorithms, which help in resolving such challenges where we can make use of the Viterbi algorithm. [5]

2 State-of-the-art

The Viterbi algorithm was invented by Andrew J. Viterbi in 1967 [7] and is used to decode convolution codes—for example, in communication networks or environments where we face large state sequence problems [7]. A specific example for a use case in communication networks is reducing errors during mobile communication and transmitting data with WLAN [12]. The Viterbi algorithm is also used to reduce the errors on persistent data on a hard drive, which is interesting for data recovery and backup systems [9]. In research areas for speech recognition, the Viterbi algorithm is used to recognize speech or to spot keywords within a given sequence of words.
The publication from Alfonsus Raditya Arsadjaja and Achmad Imam Kistijantoro describe an improvement in the Viterbi algorithm in Automatic Speech Recognition (ASR). They built a parallel ASR system to work on the Graphics Processing Unit (GPU) instead of the Central Processing Unit (CPU) [2]. In general, the audio signal has to be preprocessed to extract features and put them into a model like the Gaussian mixtures model (GMM), which is used in [2]. The idea is to speed up the evaluation of GMMs with the higher amount of cores in the GPU.
In mobile devices, ASR is becoming more attractive. It is used by different kinds of speech recognition software like Siri, Cortana, or Alexa. Mobile devices are very energy sensitive; hence, the ASR has to consume less power and be efficient. An approach to create a power-saving solution for ASR is described in the paper [14] where the authors use the GPU from the mobile device to determine a low power ASR.
In addition, the Viterbi algorithm finds some use cases in large state sequence problems occurring in bioinformatics.Jason Bobbin describes in his paper an incremental approach of the Viterbi algorithm for solving large state problems. The incremental version of the Viterbi algorithm reduces the memory usage of the long state sequences [3].
Another use case for the Viterbi algorithm is presented in the paper by Yuan, Illindala, and Khalsa, where they present a modified Viterbi algorithm to determine a system restoration strategy to ensure network stability [4].

Some additional papers, which use the same or similar solution approaches for HMM, as described above, consist of the following works: 'On efficient Viterbi decoding for hidden semi-Markov models' [11] 'presenting algorithms for improved Viterbi decoding for the case of hidden semi-Markov models' [11], which focuses on improvements and observations of the efficiency for using the Viterbi algorithm on 'fully connected models as well as restrictive topologies and state duration conditions' [11]. 'Data analyzing and daily activity learning with hidden Markov model' [15], in which the Viterbi algorithm is used to 'translate and reduce the raw data to state data. Secondly, using the hidden Markov model, forward algorithm, and the Viterbi algorithm to analyze the data' [15] for the purpose of 'observing and analyzing a person's daily activities' [15]. 'Network Anomaly Detection' [1], which focuses on 'a new approach using Two-dimensional Hidden Markov Model (HMM) based Viterbi algorithm' [1], aims to 'detect anomalous behaviors but also identify the intention behind them' [1]. Lastly, 'Efficient computation of the hidden Markov model entropy for a given observation sequence' [6], which describes an implementation of the Viterbi algorithm which 'is based on a trellis structure resembling that of the Viterbi algorithm, and permits the efficient computation of the entropy with a complexity linear in the

number of observations' [6].

3 PROBLEM

This section describes the specification of the problem that the Viterbi algorithm tries to solve. Furthermore, to analyse the problem, a test sequence will be defined.
The algorithm gets as input an HMM, which is modelled as a Markov chain, where the states are unobservable (hidden) and, so, they are unknown [13]. However, the output or otherwise observation or emission at each time is visible, and gives stochastically a hint of the hidden state. A possible transition from one state to another is given by a transition probability. The characteristic of a transition is that the transition probability from one state to the next depends only on the present state (Markov Assumption) [7].

$$P(s_i|s_1..s_{i-1}) = P(s_i|s_{i-1})$$

Figure 1 shows an example of an HMM that illustrates the transition between observed and hidden states. In this case, the HMM has a state

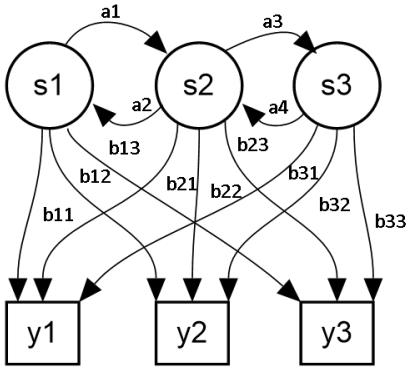

Fig. 1: HMM with observed states

space $S = \{s_1, s_2, ..s_N\}$, an observation space $O = \{o_1, o_2, ..o_K\}$, a transition matrix A of size $N \times N$, where A_{ij} is the transition probability from state a_i to state s_j, an emission matrix B of size $N \times K$, where B_{ij} is the probability that observed state o_j is from state s_i and an array of initial probabilities at time $t = 1$ of size N where π_i is the probability of state s_i. Furthermore, an output shows the observation sequence $y_1, y_2, ..y_T$.

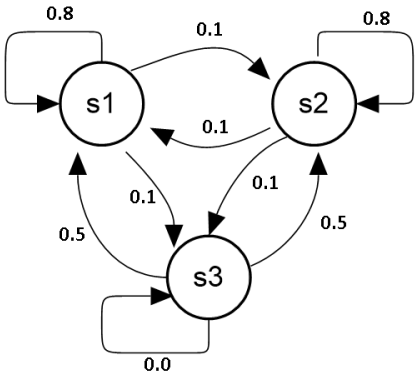

Fig. 2: Markov chain (cf. [8])

To analyse the Viterbi algorithm, this paper will use the following Markov chain:

Thus, the state space can be summarized as $S = \{A, B, C\}$ and the transition matrix as

$$A = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.5 & 0.5 & 0 \end{pmatrix} . \text{ [8]}$$

In the following, when the Viterbi algorithm will be used, the states are supposed to be unobservable. Hence, the state space S will be treated as hidden. To estimate the hidden state, the observation space $O = \{1, 2, 3\}$ with the emission matrix

$$B = \begin{pmatrix} 0.9 & 0.1 & 0.0 \\ 0.1 & 0.8 & 0.1 \\ 0.0 & 0.1 & 0.9 \end{pmatrix} \text{ [8]}$$

will be used. The start probability of the state space S will be

$$\pi = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix} . \text{ [8]}$$

To analyse the problem and the correctness of the Viterbi algorithm, the observed states Y with the true hidden states will be shown next. Here, it is again mentioned that in practice, the hidden states are unknown.

```
AAAAAABBBCBBBBBBCBCAAAAAABBBBBAAABBBBAAAAAAACAAAAAAAAAA
11111122322222323111112322311122221211111312212111111
```

Fig. 3: test sequence [8]

The challenge of the Viterbi algorithm is to classify or predict the true state by only using

the given information of the observed states and transition probabilities.

4 Solution

In regards to problems described in the previous section, the Viterbi algorithm aims to solve these issues by attempting to recreate the original sequence of states from the provided HMM. This recreated sequence can then be used to determine falsely recognized states as well as causes for the occurrence of such states. In order to recreate the original sequence, the Viterbi algorithm calculates the path of state transitions with the highest probability to produce the given sequence of emissive values. These are based on the HMM described by matrix A and the matrix $B$of emission probabilities per state. The formulas described in the following paragraphs originate from Page 272 of the Viterbi algorithm [7] paper. Meanwhile, the specified formula notations are used from the German Wikipedia article under the section 'Algorithmus', extracted from the lecture 'Spezielle Musteranalysesysteme' by E. G. Schukat-Talamazzini [10]. In the initial step, the algorithm starts by calculating the highest possible probability to reach each state during the first timestep $t = 1$ and follows up by calculating the highest probability path for timesteps $t > 1$.

$$\vartheta_1(i) = \pi_i * B_{i1} \text{ [7]}$$

During the first timestep $t = 1$, the probabilities to start with each state are calculated with the initial probabilities from set π. Here π_i for $i = \{1, 2, 3\}$ describes the probability to initialize the HMM with a specific state. In order to calculate the reachability probability with the Viterbi algorithm, the initial probabilities are multiplied with the emission probability B_i1. This describes the probability to emit the symbol o_1 at timestep $t = 1$ from matrix B for the state s_1 at index 1. The calculated reachability probability is then saved within the variable matrix ϑ at $\vartheta_1(i)$, which contains the maximum reachability probabilities for each state from the HMM A and for each timestep $t \geq 1$.

$$\vartheta_t(i) = B_{it} * \max_{1 \leq j \leq |S|} (A_{ji} * \vartheta_{t-1}(i)) \text{ [7]}$$

Continuing after this, the probabilities to reach each state at timestep $t > 1$ are calculated by determining the highest probability, for transitioning from $\vartheta_{t-1}(i)$ for $i = \{1, 2, 3\}$ to the current state at timestep t. In order to determine the highest transition probability, the algorithm calculates the max value over the set S of states from the HMM. The check for the highest transitions probability per state is then calculated by multiplying the highest probability to reach $i = \{1, 2, 3\}$ at timestep $t - 1$in $\vartheta_{t-1}(i)$, with the probability to transition from state $j = \{1 \leq j \leq | S |\}$ to state i in A_{ji}. Afterwards, the highest calculated transition probability is then multiplied with the emission probability B_{it}.

$$\psi_t(i) = \underset{1 \leq j \leq |S|}{\operatorname{argmax}}(A_{ji} * \vartheta_{t-1}(i)) \text{ [7]}$$

Once this transition has been calculated, the state from timestep $t - 1$, used to calculate the transition from $\vartheta_{t-1}(i)$ $\vartheta_t(i)$, is saved within the variable matrix ψ at $\psi_t(i)$. This matrix contains the predecessor states that were used to calculate $\vartheta_t(i)$ for each state from the HMM A and for each timestep $t \geq 1$. In order to calculate the proper predecessor states, the previously used calculation for the highest transition probabilities is now executed using $argmax$ instead of max.

The approaches described above are then continued until the probabilities to reach each state and their corresponding contributors, which have been calculated for each timestep $t \geq 1$.

$$y_T = \underset{1 \leq j \leq |S|}{\operatorname{argmax}}(\vartheta_T(j)) \text{ [7]}$$

After the above steps have been concluded, the highest probability path can be determined. The path is started by choosing the state with argmax at the final timestep T, with value $T = 55$ from the example, using the highest probability $\vartheta_T(j)$ for states $j = \{1 \leq j \leq | S |\}$. $j = \{1 \leq j \leq | S |\}$.

$$y_t = \psi_{t+1}(y_{t+1}) \text{ [7]}$$

Lastly, the path can be built by following the remembered contributor states ψ_{t+1}, starting

from the chosen final state. This is done from end to start by back propagating, until an initial state at timestep $t = 1$ has been reached. Once the path of states with the highest reachability probability has been determined and saved as the set Y, the work of the Viterbi algorithm itself is concluded.

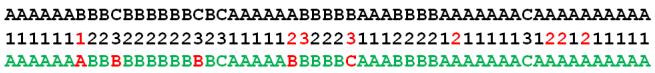

Fig. 4: Recreation of sequence [8]

At this point, the new sequence can now be used to check for errors that occurred during the recreation process described by Figure 4. The referenced Figure is based on the example described in the 'bissantz' [8] article provided for 'finding states in sequences based on HMM'. The previously mentioned errors consist of states within the recreated sequence that differ from their original state at a specific timestep. The occurrence of such errors would, for instance, allow the validation of a data transmission process. Here, emissive values that occur at a similar timestep as the determined errors can now be viewed as transmitted results, which would be prone to cause falsely interpreted data at the receiving end. Generally, the Viterbi algorithm allows one to determine problematic data parts with a high rate of success, thereby explaining its widespread use in more practical cases.

5 Summary

In this paper, the Viterbi algorithm was analysed by using a test sequence. The algorithm tries to identify the true state of an HMM with an estimation by observing emission states. To analyse the accuracy of the estimation, this paper used an example with a Markov chain. Unlike an HMM, the true states in this experiment were known but assumed to be hidden. During the evaluation, the estimated states were compared with the true states.

The use of the implemented variant of the Viterbi algorithm on the original data sequence of emissive values, described by the numeric values in Figure 3, resulted in the same sequence of states as provided in the example. The example result is hereby given by the new sequence of states from Figure 4. Therefore, one can conclude that the implementation is compatible with the description of the Viterbi algorithm. From this, one can also conclude that the algorithm can, in fact, not guarantee the resultant sequence of states, to be free from possible errors. As described in the provided article [8], one would need to mark problematic sets of emissive values and calculated states, as in Figure 4, in order to properly verify the results. This would be done so that errors caused by inconsistent emission processes and falsely recognized states could be identified and reduced to a minimum. For instance, as described in the article [8], this could be used to improve the quality of signal-based data transmissions.

All in all, the Viterbi algorithm cannot accurately identify the true hidden state of an HMM, which is highlighted by the test sequence and other papers. Despite that, the algorithm still allows observers to recognize unwanted behaviours and points of interests for optimization and indications based on their provided data sequences and the calculated estimates of true states.

References

[1] S. Alhaidari and M. Zohdy. Network anomaly detection using two-dimensional hidden markov model based viterbi algorithm. April 2019.
[2] A. R. Arsadjaja and A. I. Kistijantoro. Online speech decoding optimization strategy with viterbi algorithm on gpu. 2018.
[3] J. Bobbin. An incremental viterbi algorithm. 2017.
[4] M. S. I. Chen Yuan and A. S. Khalsa. Modified viterbi algorithm based distribution system restoration strategy for grid resiliency. 2017.
[5] K. Chugg. "viteri algorithm." university of south carolina, via youtube. 2017.
[6] G. C. D. Hernando, V. Crespi. Efficient computation of the hidden markov model entropy for a given observation sequence. June 2005.
[7] G. Forney. The viterbi algorithm. 1973.
[8] B. . C. GmbH. Hidden-markov-modelle: So bekommt man zustände! December 2014.

[9] K. M. Kilavo Hassan and S. I. Mrutu. Performance of soft viterbi decoder enhanced with non-transmittable codewords for storage media. *ELECTRICAL and ELECTRONIC ENGINEERING*, 2018.
[10] E. G. S.-T. lecture WS 2012/13 Universitity in Jena. Chapter 5 Slide 39 ff. www.minet.uni-jena.de/fakultaet/schukat/MAS/Scriptum/lect05-HMM.pdf, 2019.
[11] J. H. Ritendra Datta and B. Ray. On efficient viterbi decoding for hidden semi-markov models. December 2008.
[12] S. Singhal and M. Gilani. https://www.eetimes.com/document.asp?doc_id=1277544#, 2002.
[13] G. W. Slade. The viterbi algorithm demystified. *Dec*, 2012.
[14] R. Yazdani, A. Segura, J.-M. Arnau, and A. González. Low-power automatic speech recognition through a mobile gpu and a viterbi accelerator. 2017.
[15] G. Yin and D. Bruckner. Data analyzing and daily activity learning with hidden markov model. November 2010.