Viterbi algorithm

Tobias Schellien, Philipp Copei, Robert Koczula, Akanchha Choudhary

Abstract—- what is the generel content of the paper ? (viterbi algorihm, used to reduce errors in state sequences) - who invented it? - why do someone want to use the algorithm? (one sentence to motivation) - what are the generall use cases for the viterbi algorithm? - what is our use case? - which results did we get? - how can we interprete the results of the algorithm? - what can we conclude from it?

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

1 Introduction

The Viterbi algorithm is considered to be one of the most used dynamic programming algorithms for determining the expected sequences of hidden states in a given Hidden Markov Model and an observed sequence of symbols. This path of a sequence 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 to follow the trellis diagram before applying the viterbi algorithm as it helps in determining 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. They also play an important role in other fields such as bioinformatics as 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. Markov model helps to know how to move between the states of the 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 a HMM gives some information about the sequence of states, which further helps in determining the most desired path.

1.1 Motivation

Suppose someone plans a journey for the route from the east coast to the west coast of the United States. There are four possible cities, they can start from on the east coast. Seattle, New Port, San-Francisco and Los-Angeles. They only want to drive less than 600 miles per day. On the 1st day it leads them to end up reaching four possible cities Boise, Salt Lake, Los Vegas or Tusan. On the 3rd day they end up covering Casper, Denver, Albaquerque, and Elpaso. So this will take them approx. 7 days with the target of covering less than 600 miles. So to solve this huge time taking problem they try to finds out the shortest way to cover all the cities. Now 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 you 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 communication worlds as well as decoding convolution codes, which are used both in CDMA, GSM digital Cellular dial-up-modems, satellite, deep-space communication and other wireless LANs over noisy digital communication Links, which motivates us to get deeper into such algorithms, that help in resolving such challenges where we can make use of the Viterbi algorithm. The used example is also explained in detail under the referenced link [1].

2 State of the art

The viterbi algorithm was invented by from Andrew J. Viterbi 1967[2] and is used to decode convolution codes for example in communication networks or environments where we force large state sequence problems [2]. A specefic example for an use case in communication networks is reducing errors during mobile communication and transmitting data with WLAN [13]. 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 [10]. In ares of research for speech recognition the viterbi algorithm is used to recognize speech or spotting keywords within a given sequence of words.
The publication from Alfonsus Raditya Arsadjaja and Achmad Imam Kistijantoro describe an improvement of the viterbi algorithm in Automatic Speech Recognition (ASR). They build a parallel ASR system to work on the Graphics Processing Unit (GPU) instead of the Central Processing Unit (CPU) [4]. 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[4]. The idea is to speedup the evaluation of GMMs with the higher amount of cores in the GPU.
On mobile devices ASR is becoming more attracted. It is used by different speech recognition software like Siri, Cortana or Alexa. Mobile devices are very energy sensitive so the ASR has to be less power consuming and 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. Json 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 [5].
Another use case for the viterbi algorithm is presented in the paper from Yuan, Illindala and Khalsa where they present a modified viterbi algorithm to determine a system restoration strategy to ensure network stability [6].

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” [12] ”presenting algorithms for improved Viterbi decoding for the case of hidden semi-Markov models” [12], 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” [12]. ”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 hidden Markov model, forward algorithm, and Viterbi Algorithm to analyze the data” [15] for the purpose of ”observing and analyzing a person’s daily activities” [15]. ”Network Anomaly Detection” [3] which focuses on ”a new approach using Two-dimensional Hidden Markov Model (HMM) based Viterbi algorithm” [3] that aims to ”detect anomalous behaviors but also identifying the Intention behind them” [3]. Lastly ”Efficient computation of the hidden Markov model entropy for a given observation sequence” [7] which describes an implementation of the viterbi algorithm that ”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” [7].

3 Problem

This section describes the specification of the problem that the Viterbi algorithm tries to solve. Furthermore, to analyze the problem, a test sequence will be defined.
The algorithm gets as input a Hidden Markov

Model (HMM) which is modeled as a Markov-chain where the states are unobservable (hidden) and thus they are unknown []. However, the output or otherwise observation or emission at each time is visible and gives a stochastically 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)[].

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

Figure 1 shows an example of a HMM that illustrate the transition between observed and hidden states.

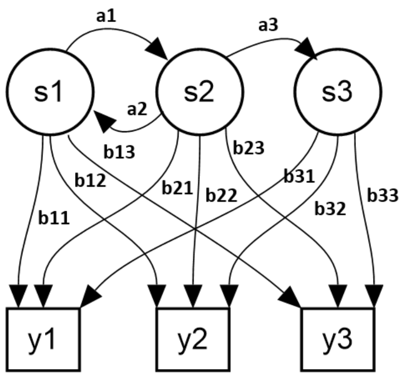

Fig. 1: Hidden Markov Model with observed states []

In this case the HMM has a state 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 a_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. Further, an output shows the observation sequence $y_1, y_2, ..y_T$.

To analyze 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

$$A = \begin{pmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.5 & 0.5 & 0 \end{pmatrix}. \text{ [9]}$$

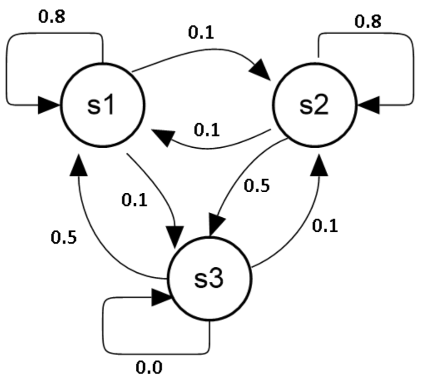

Fig. 2: Markov chain [9]

In the following when the Viterbi algorithm will be used, the states are supposed unobservable. Thus, 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

$$B = \begin{pmatrix} 0.9 & 0.1 & 0.0 \\ 0.1 & 0.8 & 0.1 \\ 0.0 & 0.2 & 0.9 \end{pmatrix} \text{ [9]}$$

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

$$\pi = \begin{pmatrix} 1/3 \\ 1/3 \\ 1/3 \end{pmatrix}. \text{ [9]}$$

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

```
AAAAAABBBCBBBBBBCBCAAAAAAABBBBBAAABBBBAAAAAAAACAAAAAAAAAAA
111111122322222223231111123222311122222121111131221211111
```

Fig. 3: test sequence [9]

The challenge of the Viterbi algorithm is to classify or to predict the true state by only using the given information of the observed states and transition probabilities.

4 Solution

In regards to problems, that were 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 emmissive values. These are based on the HMM described by matrix A and the matrix B of emmission probabilities per state. The formulas described in the following paragraphs, originate from Page 272 of the viterbi algorithm [8] paper. Meanwhile the specificied formula notations are used from the german wikipedia article [2] under the section "Algorithmus", that were extracted from the lecture "Spezielle Musteranalysesysteme" by E. G. Schukat-Talamazzini [11]. 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 $t > 1$.

$$\vartheta_1(i) = \pi_i * B_{i1} \text{ [8]}$$

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 mulitplied with the emmission probability $B_i 1$. 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 $t \geq 1$.

$$\vartheta_t(i) = B_{it} * \max_{1 \leq j \leq |S|} (A_{ji} * \vartheta_{t-1}(i)) \text{ [8]}$$

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 state $i = \{1, 2, 3\}$ at timestep $t - 1$ in $\vartheta_{t-1}(i)$, with the probability to transition from state $j = \{1 \leq j \leq\mid S \mid\}$ to state i in A_{ji}. Afterwards the highest calculated transition probabilty is then multiplied with the emmission probability B_{it}.

$$\psi_t(i) = \underset{1 \leq j \leq |S|}{\text{argmax}}(A_{ji} * \vartheta_{t-1}(i)) \text{ [8]}$$

Once this transition has been calculated, the state from timestep $t - 1$, that was used to calculate the transition from $\vartheta_{t-1}(i)$ to $\vartheta_t(i)$ 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, have been calculated for each timestep $t \geq 1$.

$$y_T = \underset{1 \leq j \leq |S|}{\text{argmax}}(\vartheta_T(j)) \text{ [8]}$$

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 $j = \{1 \leq j \leq\mid S \mid\}$.

$$y_t = \psi_{t+1}(y_{t+1}) \text{ [8]}$$

Lastly the path can be build 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.

```
AAAAAABBBCBBBBBBCBCAAAAAABBBBBAAABBBBAAAAAAACAAAAAAAAAA
111111122322222223231111123222311122222121111131221211111
AAAAAAABBBBBBBBBBBCAAAAAABBBBBCAAABBBBAAAAAAACAAAAAAAAAA
```

Fig. 4: recreation of sequence [9]

At this point the new sequence can now be used to check for errors that occured during the recreation process described by Figure 4. The referenced Figure is based on the example, that is described in the provided ”bissantz” [9] article 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 emmissive values, that occur at a similar timestep as the determined errors, can now be viewed upon as transmitted results, that would be prone to cause falsely interpreted data on the receiving end. Generally the viterbi algorithm allows one to determine problematic data parts with a high rate of success, hence explanining its widespread use in more practical cases.

5 Summary

In this paper the Viterbi algorithm was analyzed by using a test sequence. The algorithm tries to identify the true state of a HMM with an estimation by observing emission states. To analyze the accuracy of the estimation, this paper used an example with a Markov Chain. Unlike a 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.

Using the implemented variant of the viterbi algorithm on the original data sequence of emmissive values, described by the numeric values in Figure 3, resulted in the same sequence of states as provided in the example. Hereby the example result is 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 resulting sequence of states, to be free from possible errors. As described in the provided article [9], one would need to mark problematic sets of emmissive values and calculated states as in Figure 4, in order to properly verify the results. This would be done, so that errors caused due to inconsistent emmission processes and falsely recognized states could be identified and reduced to a minimum. For instance, as described in the article [9], this could be used to improve the quality of signal based data transmissions.

All in all, the Viterbi algorithm can not accurately identify the true hidden state of a 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 indication based on their provided data sequences and the calculated estimates of true states.

References

[1] https://www.youtube.com/watch?v=6JVqutwtzmo.

[2] https://de.wikipedia.org/wiki/Viterbi-Algorithmus, 2018.

[3] S. Alhaidari and M. Zohdy. Network anomaly detection using two-dimensional hidden markov model based viterbi algorithm. April 2019.

[4] A. R. Arsadjaja and A. I. Kistijantoro. Online speech decoding optimization strategy with viterbi algorithm on gpu. 2018.

[5] J. Bobbin. An incremental viterbi algorithm. 2017.

[6] M. S. I. Chen Yuan and A. S. Khalsa. Modified viterbi algorithm based distribution system restoration strategy for grid resiliency. 2017.

[7] G. C. D. Hernando, V. Crespi. Efficient computation of the hidden markov model entropy for a given observation sequence. June 2005.

[8] G. Forney. The viterbi algorithm. 1973.

[9] B. . C. GmbH. Hidden-markov-modelle: So bekommt man zustände! December 2014.

[10] 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.

[11] 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.

[12] J. H. Ritendra Datta and B. Ray. On efficient viterbi decoding for hidden semi-markov models. December 2008.

[13] S. Singhal and M. Gilani. https://www.eetimes.com/document.asp?doc_id=1277544#, 2002.

[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.