Hidden Markov Models for Speaker Diarisation

Term Project for
Statistical Foundation for AI and ML (AI61004)

Indian Institute of Technology Kharagpur

by

Prabhav Patil 20MA20042
Koushiki Dasgupta Chaudhury 18MA20020
Shivam 18MA20042
Varun Gupta 18MA20050
Radhika Gupta 18MA20035

Under the supervision of

Prof. Adway Mitra

Indian Institute of Technology Kharagpur

Spring Semester, 2023

April 27, 2023

Abstract

Speaker diarisation is the process of segmenting an audio recording into speaker-specific segments, thereby identifying the different speakers in a multi-speaker environment. Hidden Markov Models (HMMs) are a powerful tool for speaker diarisation, as they can model the temporal dynamics of speech and the variability in speaker characteristics. In this approach, HMMs are used to model the acoustic features of the speech signal, and these models are then combined with a clustering algorithm to group similar segments of speech into speaker-specific segments. HMM-based speaker diarisation has been shown to be effective in a variety of settings, including telephone conversations, meeting recordings, and broadcast news. Despite the challenges posed by background noise and speaker overlap, HMM-based diarisation remains an important and active area of research, with ongoing efforts to improve its accuracy and scalability.

One of the key advantages of HMMs for speaker diarisation is their ability to handle the variability in speaker characteristics over time, such as changes in pitch, speaking rate, and pronunciation. This is achieved by using a set of HMMs to model the different speaker characteristics, and then combining these models to create a joint HMM for each speaker. Additionally, HMM-based speaker diarisation can be adapted to different acoustic conditions by adjusting the model parameters to account for variations in the audio signal.

Overall, HMM-based speaker diarisation is a powerful technique that has shown great promise in a wide range of applications, from forensic investigations to speech transcription and indexing. As the demand for automatic speaker diarisation continues to grow, there is a need for further research to develop more accurate and robust HMM-based approaches.

In this term project, we implement three different variants of HMMs on a speaker diarisation dataset, so as to compare and judge the efficiency of all of the three different types.

Contents

Chapter 1

Data Source

1.1 Data Source

We used the popular **AMI Corpus** library for implementing the HMM models, from which we extracted **IB4010** conference recording and trained our parameters on the models.

1.2 AMI Meeting Corpus

The AMI Meeting Corpus is a multi-modal data set consisting of 100 hours of meeting recordings.Around two-thirds of the data has been elicited using a scenario in which the participants play different roles in a design team, taking a design project from kick-off to completion over the course of a day. The rest consists of naturally occurring meetings in a range of domains.

Although the AMI Meeting Corpus was created for the uses of a consortium that is developing meeting browsing technology, it is designed to be useful for a wide range of research areas. The downloads on this website include videos that are suitable for most purposes, but higher resolution videos are available for researchers engaged in video processing

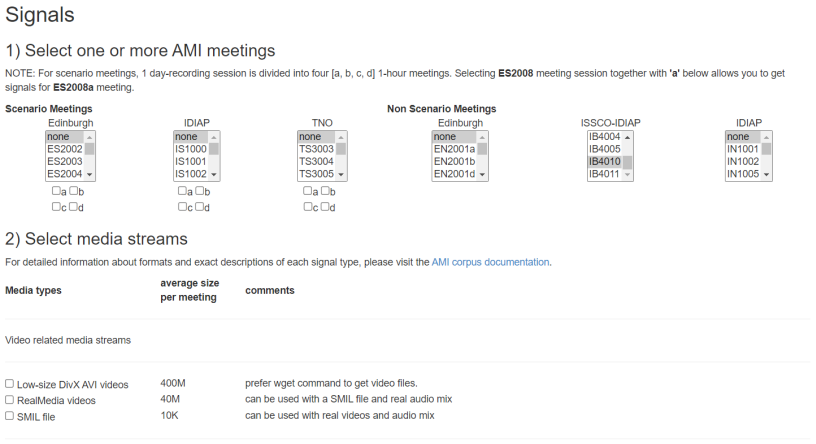

FIGURE 1.1: AMI CORPUS

Chapter 2

Models and its Variants

2.1 Overview

There are several variants of HMMs that have been developed to better model different types of data such as:

1. **Gaussian HMMs**:This variant assumes that the observations are continuous and follow a Gaussian distribution, which is what we consider the observations to be in our project.
2. **Hierarchial HMMs**: This variant models the HMM as a tree structure, where each level represents a different time scale.
3. **Switching HMMs**:This variant allows for the HMM to switch between different sets of parameters, allowing for more flexibility in modeling complex data.
4. **Time-varying HMMs**:In this variant, the parameters of the HMM are allowed to change over time, allowing for more dynamic modeling.
5. **Factorial HMMs**:This variant allows for multiple independent HMMs to be combined to model different aspects of the data.

In this project, we have implement three variants of Hidden Markov Models on the speaker diarisation dataset. These include:

2.2 Variants of HMM

2.2.1 Standard

The model consists of three main sets of parameters: the state transition probabilities, the emission probabilities, and the initial state probabilities. The state transition probabilities represent the probability of moving from one hidden state to another, while the emission probabilities represent the probability of generating a particular observed symbol given

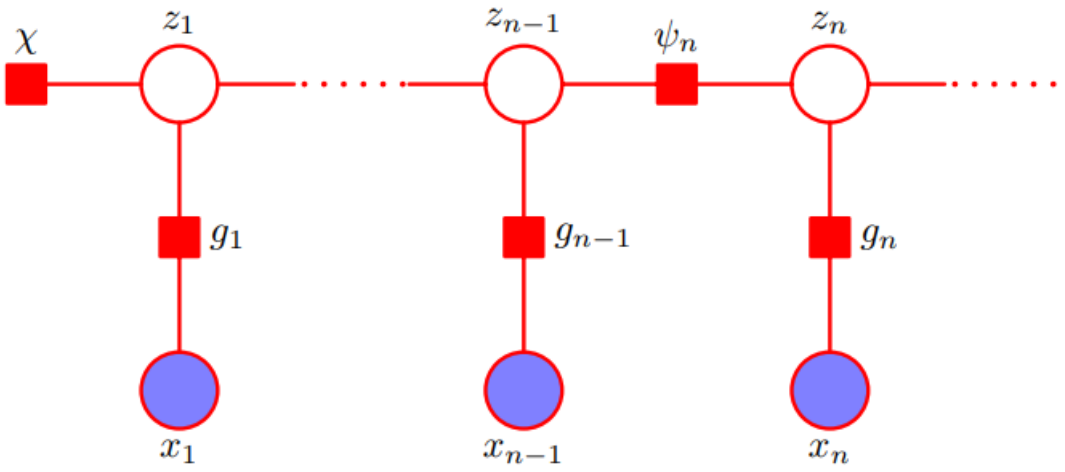

FIGURE 2.1: Standard HMM

a hidden state. The initial state probabilities represent the probability of starting in a particular hidden state.

The joint probability of the observed symbols and the corresponding hidden states in an HMM can be computed using the forward-backward algorithm. The forward algorithm computes the probability of observing a sequence of symbols given the model parameters, while the backward algorithm computes the probability of being in a particular hidden state at a particular time given the observed sequence of symbols.

The key feature of this type of HMM is that the observation at the current time-step is dependent only on the immediate observation at the previous time-step.

2.2.2 Auto-regressive

Autoregressive Hidden Markov Models (AR-HMMs) are a type of Hidden Markov Model (HMM) that incorporate the concept of autoregression into the model. Autoregression refers to the idea that a variable depends on its own previous values.

In an AR-HMM, the hidden states and observed symbols are modeled as a function of their previous values. This means that the current state and observation depend not only on the previous state and observation but also on a set of previous states and observations. This allows for more complex dependencies between the hidden states and observed symbols, as compared to standard HMMs.

The parameters of an AR-HMM include the state transition probabilities, the emission probabilities, and the autoregressive parameters. The state transition probabilities and emission probabilities are similar to those in standard HMMs, representing the probability of transitioning from one state to another and the probability of emitting a particular symbol from a particular state, respectively. The autoregressive parameters represent the dependence of the current state and observation on their previous values.

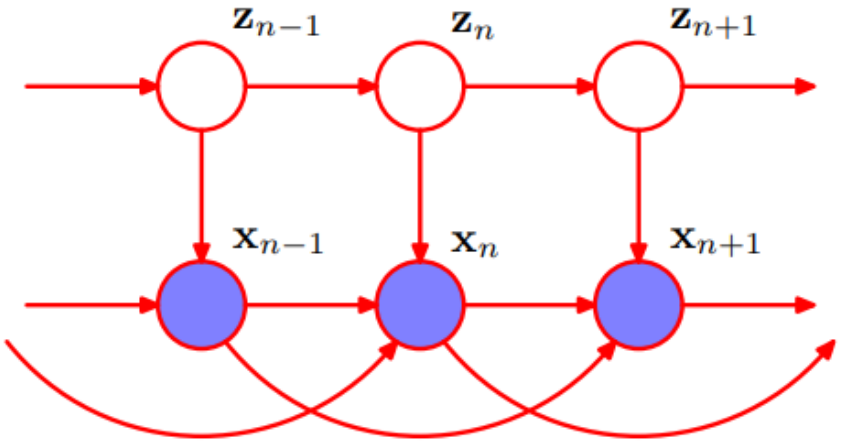

FIGURE 2.2: Auto-Regressive HMM

2.2.3 Sticky

Sticky Hidden Markov Models (SHMMs) are a type of Hidden Markov Model (HMM) that incorporate a concept known as "stickiness" into the model. Stickiness refers to the idea that there is a tendency for the hidden state to stay in its current state for a certain period of time before transitioning to a new state.

In a SHMM, the state transition probabilities are modified to allow for stickiness. Specifically, the probability of transitioning from a state i to a state j is decreased if the current state is not i. This means that if the hidden state is currently in state i, there is a higher probability of staying in state i for a longer period of time before transitioning to another state.

The motivation behind SHMMs is to model situations where the underlying process is expected to exhibit long periods of stability or persistence in a particular state, followed by occasional transitions to new states. This is often seen in real-world systems, such as in speech recognition, where certain phonemes are more likely to be repeated multiple times before transitioning to a different phoneme.

2.3 Data Preprocessing

To implement the variants of HMM on the data, there is a need to pre-process the dataset and segment the 'speech-part' of the data into short time-intervals, to make it discrete.

Generally sound produced by humans is shaped by the vocal cords in the mouth, human vocal tract give out the envelope of the short time power-spectrum (frequency v/s power). Then we use the Mel Frequency Cepstral Coefficient (MFCC) to accurately represent this envelope of spectrum.

The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies

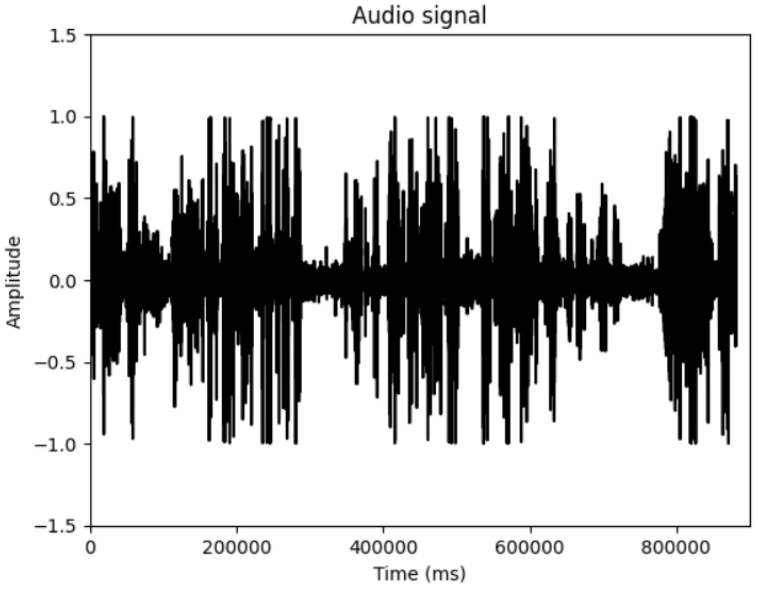

FIGURE 2.3: Normalised Amplitude v/s Time of Audio Signal

than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear. This feature extraction then generates the discrete observations which can now be used to train and implement the above Hidden Markov Models using the Baulm-Welch algorithm, so as to train the parameters of the variants.

Chapter 3

Results and Comparison

3.1 Extracting MFCC Features from the Dataset

To extract the features from the dataset, the following steps are needed to be taken care of:

```
n_mfcc = 15
mfcc = np.zeros((mfcc_features.shape[0],n_mfcc))
mfcc = np.reshape(mfcc_features,(-1,15))
#print(mfcc)
print(mfcc)
```

```
[[-1.91188431e+02  1.14908188e+02  3.33354263e+01  3.20853081e+01
   2.37144127e+01  1.95490742e+01  1.49306116e+01  1.34298286e+01
   9.76234818e+00  8.64542961e+00  5.85106659e+00  4.85556889e+00
   4.40967083e+00  2.81921196e+00  2.55116129e+00]
 [-2.34292542e+02  1.19434731e+02  3.43508186e+01  4.44593048e+01
   2.95988579e+01  2.26247387e+01  1.83001766e+01  1.50811024e+01
   1.12472191e+01  1.22145643e+01  4.43048477e+00  3.77753687e+00
   6.47163391e+00  1.39662504e+00  1.49968565e+00]
 [-2.87496307e+02  1.20276932e+02  2.62622643e+01  4.95057602e+01
   2.94703197e+01  2.80988445e+01  2.34328480e+01  1.19372377e+01
   9.05604172e+00  1.97867928e+01  6.15141678e+00  1.70767212e+00
   4.76325703e+00  8.56988072e-01  2.61529350e+00]
 [-2.29974472e+02  7.89308167e+01  5.06460991e+01  4.37240906e+01
   3.51106567e+01  3.37583542e+01  1.78713531e+01  9.54522705e+00
   1.96799812e+01  1.07837620e+01  1.42703066e+01 -1.21966515e+01
   7.38692999e+00  3.81279230e-01 -8.61713171e-01]
 [-2.04179016e+02  7.82406540e+01  5.19683762e+01  5.14875641e+01
   3.61136398e+01  3.54860725e+01  1.94310951e+01  1.09775066e+01
   2.66275043e+01  1.63793335e+01  1.68740158e+01 -1.22451954e+01
   4.60388088e+00 -1.73872542e+00 -1.87235630e+00]
 [-2.40594742e+02  9.24932098e+01  4.11769562e+01  6.00674019e+01
   4.09448814e+01  3.35217552e+01  1.91920357e+01  1.35373182e+01
   2.40730629e+01  2.11883049e+01  1.76664314e+01 -7.13149738e+00
   6.34476376e+00  1.89604366e+00 -4.29467678e+00]
 [-2.73593384e+02  1.08350800e+02  3.43692131e+01  6.39645653e+01
   4.86588287e+01  2.76871376e+01  1.57403126e+01  1.71056213e+01
   1.60921822e+01  2.36595230e+01  1.76361446e+01  1.61360097e+00
   1.06181993e+01  1.74681628e+00 -2.80116701e+00]
 [-2.85573975e+02  1.09959671e+02  3.06668358e+01  6.50440216e+01
```

FIGURE 3.1: Extracted MFCC features of the audiofile

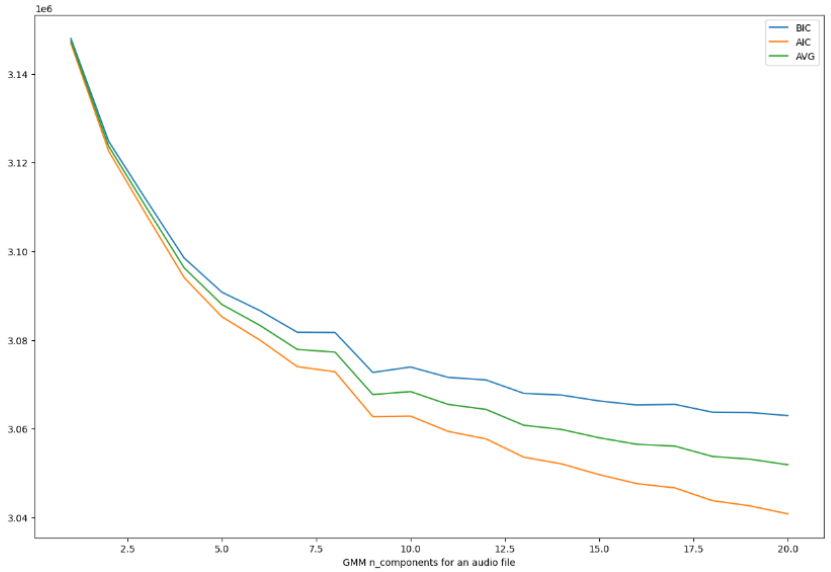

FIGURE 3.2: BIC/AIC Plots for Comparison of parameters

1. **Voice Activity Detection**: We use a Voice Activity Detection to separate out speech from non-speech. This is required to trim out silences and non-speech part from your audio recording.

2. **Speech Segmentation**: We extract out small segments of your audio call (typically about 1 second long) in a continuous manner. Speech segments typically contain just one Speaker.

3. **Embedding**: We find the MFCC (Mel Frequency Cepstral Coefficient) of the audio segment. These are basically feature coefficients which capture the variations in the speech like pitch, quality, intone of the voice in a much better way. We extract about 15 of features of MFCCs for our project.

4. **Clustering**: Embeddings of the segments belonging to same speakers are part of one cluster, and assigned the label of the speaker. This can be done by clustering techniques such as Gaussian Mixtures.

The optimal number of clusters to be generated is done by comparing the AIC and BIC plots as shown in Figure 3.2.

3.2 Evaluation of the Models

1. **AutoRegressive HMM**: We cannot directly apply a standard HMM to an autoregressive time series since the observations at different time steps are dependent on each other. In an autoregressive time series, each observation is a function of the previous observations, and thus the dependencies between the observations need to be explicitly modeled. Hence, to implement an autoregressive HMM in Python, we created a lagged dataset where each observation is a function of the previous observations. Then, we can use the lagged dataset to train and predict using an autoregressive HMM.

```
self.model.fit(X)

A = self.model.transmat_
B = self.model.means_
C = self.model.covars_
pi = self.model.startprob_

self.sticky_A = (1 - self.kappa) * A + self.kappa * np.eye(self.n_components)
self.sticky_B = B
self.sticky_C = C
self.sticky_pi = pi
```

FIGURE 3.3: Sticky HMM Implementation

2. **Sticky HMM**: We compute the sticky transition matrix, which is a weighted average of the learned transition matrix and the identity matrix. Identity matrix corresponding to the tendency of remaining in the same state, and κ being the stickiness parameter, which will influence the sticky transition matrix.

After having generated the variants of the Hidden Markov Models, we split the dataset into a 70:30 split, of which the first 70 percent is used to train the parameters of the Hidden Markov Models through the Baulm-Welch Algorithm, while the rest 30 percent of the data is used to test the Hidden Markov Models through the use of Viterbi algorithm.

3.3 Comparison of the Models

An interesting question associated with HMMs is the following: Given two HMMs, X1 and X2, what is a reasonable measure of the similarity of the two models? A key point here is the similarity criterion. Even when the two models, λ_1, and λ_2, look ostensibly very different, statistical equivalence of the models can occur. We can introduce a sense of distance measure to compare the models, which can then be evaluated to find out the difference in the similarity in both the given models. Several interpretations of distance measure exist in terms of cross entropy, or divergence, or discrimination information.

$$\textbf{Distance Measure } D(\lambda_1, \lambda_2) = \frac{1}{T}[\log P(O^{(2)}|\lambda_1) - \log P(O^{(2)}|\lambda_2)]$$

where $O^{(2)} = O_1 O_2 ... O_T$ is a sequence of observation generated by model λ_2. In other words, this is a measure of how well model λ_l matches observations generated by model λ_2, relative to how well model λ_2 matches observations generated by itself.

3.4 Limitations and Future Scope

There is a vast scope for future work in this area, Some of them are as follows:

1. **Limited modeling of speaker variability**:HMMs are based on the assumption that the underlying process generating the observed data is a Markov process, which means that the current state depends only on the previous state. However, this assumption may not hold in real-world scenarios, where speaker variability can affect the acoustic properties of the speech signal. This can lead to errors in speaker diarisation, especially when there is significant overlap between speakers.

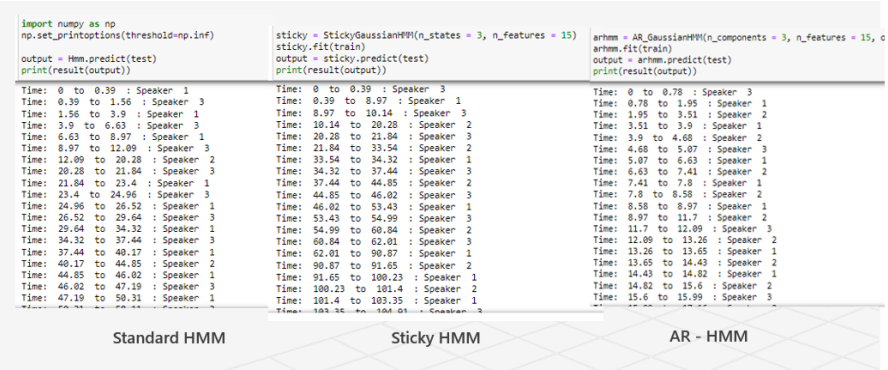

FIGURE 3.4: Results / Output for variants of Hidden Markov Models

2. **Dependence on the number of speakers**:HMM-based speaker diarisation systems typically require the number of speakers to be known in advance. This can be a significant limitation in scenarios where the number of speakers is not known beforehand or can vary over time.

3. **Limited robustness to noise and other sources of interference**:HMM-based speaker diarisation systems can be sensitive to noise and other sources of interference in the speech signal. This can lead to errors in speaker diarisation, especially in noisy environments or when the speech signal is degraded.

4. **Comparison of the Models**:Given an observation sequence which is likely to occur with respect to more than one HMM model, the observation sequence generated from one HMM is likely to fit another HMM as well. This implies a statistical equivalence in the models. Hence, judging and comparing different HMM models based on various criterion such as Cross-Entropy, Likelihood, measures etc. is a possible future scope.

5. **Integration with other machine learning techniques**:HMMs are just one of many machine learning techniques that can be used for speaker diarisation. Future research could explore ways to integrate HMMs with other techniques such as neural networks, deep learning, and reinforcement learning, to improve the accuracy and robustness of speaker diarisation systems.

Appendix A

Appendix

A.1 Mel Frequency Cepstral Coefficient

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip. The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal spectrum.

This frequency warping can allow for better representation of sound, for example, in audio compression that might potentially reduce the transmission bandwidth and the storage requirements of audio signals.

The basic procedure for MFCC calculation is given as follows:

1. Logarithmic filter bank outputs are produced and multiplied by 20 to obtain spectral envelopes in decibels.
2. MFCCs are obtained by taking Discrete Cosine Transform (DCT) of the spectral envelope.
3. Cepstrum coefficients are obtained as:

$$ci = \sum_{n=1}^{Nf} S_n \cos\left[i(n-0.5)\frac{\pi}{Nf}\right]$$

where $c_i = ith$ MFCC coefficient, Nf is the number of traingular filters in the filter bank, Sn is log energy output of nth filter coefficient and L is the number of MFCC coefficients that we want to calculate.

A.2 Baulm-Welch Algorithm

The Baum-Welch algorithm is a statistical algorithm used for training hidden Markov models.

The Baum-Welch algorithm is based on the Expectation-Maximization (EM) algorithm, which is a general-purpose optimization algorithm used to estimate the parameters of statistical models. The EM algorithm consists of two steps: the E-step, which computes the expected value of the likelihood function, and the M-step, which maximizes the likelihood function with respect to the parameters of the model.

In the case of HMMs, the Baum-Welch algorithm performs the E-step by computing the forward and backward probabilities, which represent the probability of being in a particular state at a particular time given the observations up to that time. The M-step then updates the parameters of the model based on the expected values of the state transitions and emissions.

To compute the E-step and the M-step, we use the forward and backward algorithm so as to update the parameters of the Hidden Markov Models in each iteration.

A.2.1 Forward Algorithm

The forward algorithm is a dynamic programming algorithm used to compute the probability of a given observation sequence in a hidden Markov model (HMM).

In an HMM, we have a set of hidden states, and we observe emissions from those states. The forward algorithm calculates the probability of observing a sequence of emissions, given a particular HMM model. It does so by computing the probability of being in each possible hidden state at each point in time and then summing over all possible hidden state sequences that could have produced the observed sequence.

The algorithm works by recursively calculating the probability of being in each hidden state at time t, given the observations up to time t. This probability is called the forward variable, denoted as $\alpha_t(i)$, where i is the index of the hidden state at time t. The forward variable $\alpha_t(i)$ is defined as:

$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = i|\lambda)$

where $O_1, O_2, ..., O_t$ are the observed emissions up to time t, q_t is the hidden state at time t, and λ is the HMM model.

A.2.2 Backward Algorithm

The Backward Algorithm is a dynamic programming algorithm used in Hidden Markov Models (HMMs) to compute the probability of a sequence of observations given a model. It is one of the two fundamental algorithms used in HMMs, the other being the Forward Algorithm.

In the Backward Algorithm, we start at the end of the sequence of observations and work backwards. The algorithm computes the probability of observing the remaining observations given that the model is in a particular state. The probability of observing

Input: HMM specified by $\Theta = (\mathcal{A}, A, C, \mathcal{B}, B)$
Observation sequence $O = (o_1 = \beta_{k_1}, o_2 = \beta_{k_2}, \ldots, o_N = \beta_{k_N})$
Output: Optimal state sequence $S^* = (s_1^*, s_2^*, \ldots, s_N^*)$

Procedure: Initialize the $(I \times N)$ matrix $\mathbf{D}$ by $\mathbf{D}(i,1) = c_i b_{ik_1}$ for $i \in [1:I]$. Then compute in a nested loop for $n = 2, \ldots, N$ and $i = 1, \ldots, I$:

$$\begin{aligned} \mathbf{D}(i,n) &= \max_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1)\right) \cdot b_{ik_n} \\ \mathbf{E}(i,n-1) &= \operatorname{argmax}_{j \in [1:I]} \left(a_{ji} \cdot \mathbf{D}(j, n-1)\right) \end{aligned}$$

Set $i_N = \operatorname{argmax}_{j \in [1:I]} \mathbf{D}(j,N)$ and compute for decreasing $n = N-1, \ldots, 1$ the maximizing indices

$$i_n = \operatorname{argmax}_{j \in [1:I]} \left(a_{ji_{n+1}} \cdot \mathbf{D}(j,n)\right) = \mathbf{E}(i_{n+1}, n).$$

The optimal state sequence $S^* = (s_1^*, \ldots, s_N^*)$ is defined by $s_n^* = \alpha_{i_n}$ for $n \in [1:N]$.

FIGURE A.1: Psuedocode for Viterbi Algorithm

the complete sequence of observations is then obtained by summing over all possible initial states.

A.3 Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for obtaining the maximum a posteriori probability estimate of the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events.

The psuedocode of Viterbi Algorithm is as shown in Figure A.1.

Packages, Libraries, and Frameworks

Numpy, Matplotlib, Tensorflow, Sci-kit Learn, Sci-py

1. Link to Github Repository here
2. Link to Code in Google Colab here
3. Dataset Source here

References

[1] LR Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition" - Proceedings of the IEEE, 1989

[2] C Bishop, "Pattern Recognition and Machine Learning" - pp. 617-635, 2006

[3] M Yeluri, "Understanding and Implementing Speech Recognition using Hidden Markov Model" - Medium, 2018

[4] Ephraim, "Revisiting Autoregressive Hidden Markov Modeling of Speech Signals" - Proceedings of the IEEE, vol 12 pp. 166-169, 2005

[5] Lafferty, J., Pereira, F. C., Lee, K, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data" -Proceedings of the 18th International Conference on Machine Learning (ICML-01) pp. 282-289, 2001