Understanding the Acoustic Model

Exploring the Probabilistic Models for Speech Recognition

Team 4

2019102004 (Abhishek Mittal)

2019113004 (Shreyas Pradhan)

2019113005 (Arihanth Srikar Tadanki)

2019113007 (Sriram Devata)

2019113022 (Hrishi Narayanan)

Probability and Random Processes

Contents

1 Introduction

The study of speech processing and recognition has gained momentum over the recent years with the development of new technologies that depend on speech (audio) as input. Smart devices, speech-to-text, real-time speech-to-speech translations are few of the several cutting edge technologies that take in speech (audio signals) as an input. Two major technical challenges are faced while taking audio signals as input. First challenge is 'cleaning' the audio signal; the noise in the signal must be removed to receive the correct input. The second challenge is the recognition of the speech obtained as input, so that it can be used efficiently. This project will be focusing on the latter aspect of speech processing, the recognition of speech signals.

Speech recognition is done by using several methods. One approach is to the problem is by using statistical and probabilistic models understand and predict the received input better. Such models include the Acoustic Model, the Lexicon (Phonetic/Pronunciation) Model, and Language Model. The Acoustic Model involves establishing statistical representations for the feature vector sequences computed from the speech waveform. The Lexicon Model helps compute probability of the sound or the "phone states" for the given the words, which may be modeled as simple a dictionary of pronunciations, or some other complex model. Meanwhile, the Language Model helps calculate the probability distribution over sequences of words.

In this project, we will be focusing on the Acoustic Model. Acoustic Model in a nutshell represent the relationship between a captured audio signal and the corresponding phonemes or other linguistic units that make up speech. Acoustic models are "trained" by considering a very large number of audio recordings of speech, along with their corresponding transcription. This is then used to create a statistical representations of the sounds that make up each word. From the Laws of Large Numbers (LLNs) it follows that a model trained with very large duration transcribed audio will be much accurate than a model trained with limited audio. Including a broad array of different and varying sounds is also a necessary factor for better accuracy.

Acoustic Model is usually done employing one of the two statistical models, namely Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM).

2 Gaussian Mixture Models (GMM)

Gaussian mixture models are probabilistic models for representing normally distributed samples over the entire sample space. Mixture models in general don't require knowing which a data point belongs to which subset within the sample state, allowing the model to learn the subsets automatically. It is also used for supervised learning or classification to learn the boundary of subsets of the sample space. One of the interesting application is the modelling of audio, which we will be discussing ahead in the project.

When we analyse our audio samples, we will come across various features which we will plot as points on a graph. We must classify these points in order to determine which phoneme they correspond to. The process of classifying these points is done by the GMM along with the EM (Expectation Maximization) algorithm.

2.1 Spectogram

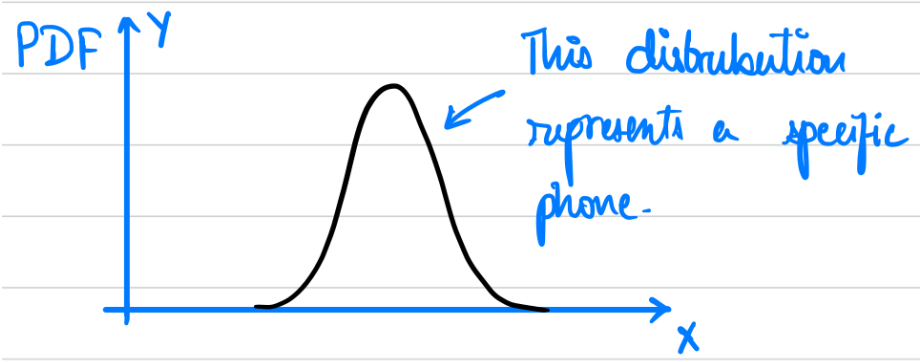
Whenever we think of a signal, the picture that comes up in our mind is the amplitude plotted against time. This is called the time domain representation of a signal and is the most intuitive way to look at it. But analysing the time domain signals does not provide us with all the information about the signal. Hence we change the domain representation of the signal to be able to better process the information in the signal. We choose the frequency domain representation where the signal is now just a linear combination of frequencies. In this representation though the information about the position of the frequency components disappears. We can no longer say when a high pitch noise occurred in the time domain representation of the signal. So we conserve this informa-

tion by taking the Fourier transform of the signal over intervals of time. A spectogram is a plot of the frequencies plotted along the Y axis and the time along the X axis. The colour of the plot represents the square of the magnitude of the Fourier coefficients and gives an idea of the energy of the signal over various frequencies plotted over time. This gives us the MFCC's.

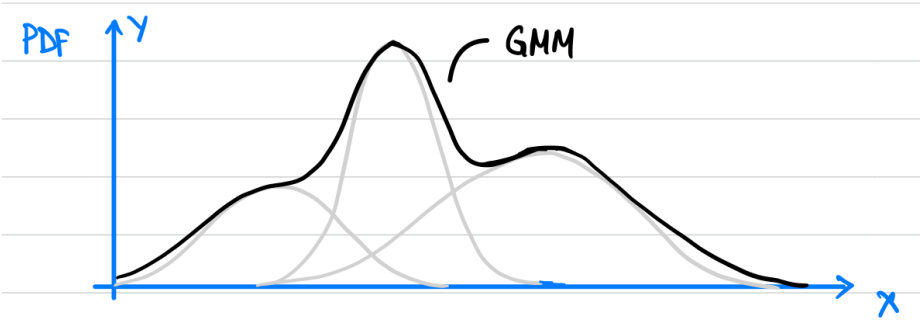
2.2 Mel-frequency Cepstral Coefficients (MFCC)

With the help of MFCC (Mel-frequency Cepstral Coefficients), we can extract the features from an audio sample. These features must be classified. This classification is done with existing data on phonetics that contains information/differentiation between all the consonants and vowels upon looking at their spectral diagrams.

Each feature of an audio sample can be modelled as a Gaussian distribution. For example,



We can also look at multiple features and the classification between those features. We can visualize some 1-D Gaussian variables as



The PDF of a Gaussian random variable is denoted by $\mathcal{N}(\mu, \Sigma)$, where μ is the mean and Σ is the variance.

$$\mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{2\pi\Sigma}} e^{-\frac{1}{2\Sigma}(x-\mu)^2}$$

In speech recognition, each feature is given by a Gaussian random variable, so we will have a sequence of Gaussian random variables $X_1, X_2, X_3, \ldots, X_n$ representing n features. Therefore, we will consider a Gaussian random vector

$$X = (X_1, X_2, X_3, \ldots, X_n)^T$$
$$P(X) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Now, let us consider a feature j among a total of n features, the likelihood of the observed feature in the feature vector is

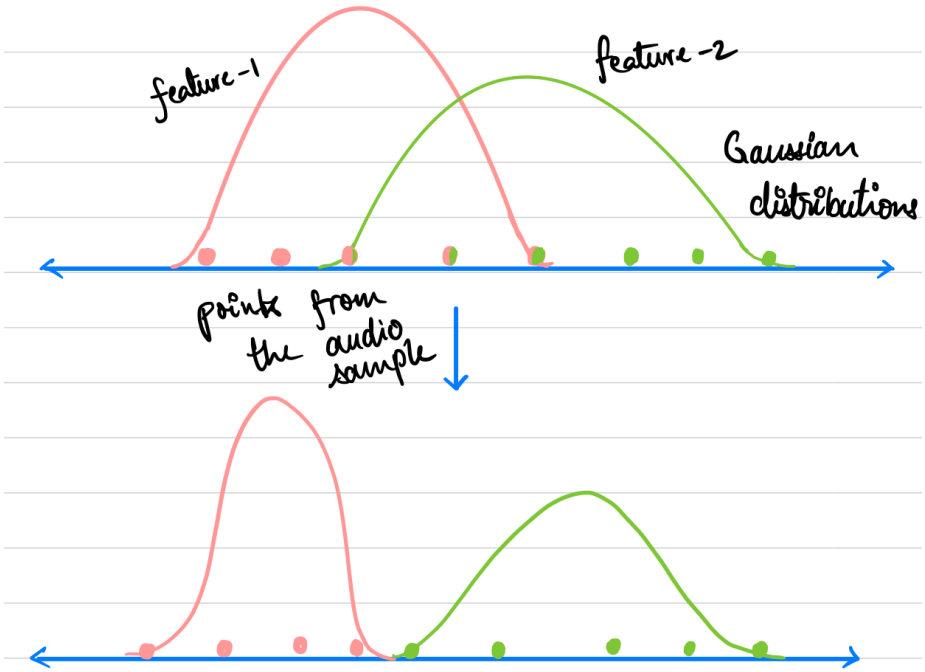
$$b_j(x) = P(x|S = j) = \sum_{m=1}^{n} C_{jm} \mathcal{N}(x : \mu_{jm}, \Sigma_{jm})$$

where C_{jm} is the weight for the Gaussian distribution m. These parameters are then determined by the EM algorithm.

2.3 Expectation Maximisation (EM) Algorithm

An EM algorithm has two steps: the expectation step (E-Step) and the maximisation step (M-step). These two steps are repeated iteratively until the change in the parameters is negligible. Based on the combination of the Gaussian distribution in the graph, we can find the probability of the sampled point belonging to each Gaussian random variable.

To understand this, we can model the problem in the diagram. Here, we assume that the points can be classified into two features with probabilities a_i and b_i.



$$b_i = P(b|x_i) = \frac{P(x_i|b)P(b)}{P(x_i|b)P(b) + P(x_i|a)P(a)}$$
$$a_i = 1 - b_i$$

The new mean and variance of the distribution will be given by,

$$\mu_a = \frac{a_1x_1 + a_2x_2 + \ldots + a_nx_n}{a_1 + a_2 + \ldots + a_n},\ \mu_b = \frac{b_1x_1 + b_2x_2 + \ldots + b_nx_n}{b_1 + b_2 + \ldots + b_n}$$
$$\sigma_a^2 = \frac{a_1(x_1 - \mu_1)^2 + \ldots + a_n(x_n - \mu_n)^2}{a_1 + a_2 + \ldots + a_n},\ \sigma_b^2 = \frac{b_1(x_1 - \mu_1)^2 + \ldots + b_n(x_n - \mu_n)^2}{b_1 + b_2 + \ldots + b_n}$$

We then repeat this process of finding the values of a_i, b_i and $\mu_a, \mu_b, \sigma_a^2, \sigma_b^2$. After doing this iteratively, we will get a final Gaussian distribution that classifies all the points extracted from the audio sample.

Since the values a_i, b_i are not strictly restricted to $\{0,1\}$, but take values in the range $[0,1]$, there is a soft assignment of parameters. In the above example,

there are n features, but each of these points belong to only two different clusters of features for simplicity. We can extend the same concept to n features belonging to m clusters, where we will have the probabilities $a_i, b_i, c_i, \ldots, n_i$ where $a_i + b_i + \ldots + n_i = 1$. This likelihood of each feature being related to a phoneme is used as the emission probability of an internal state in a HMM.

The general equation for the following are,

$$c_{m(j+1)} = \frac{1}{N} \sum_{t=1}^{N} h_{mj}(t),$$

$$\mu_{m(j+1)} = \frac{\sum_{t=1}^{N} h_{mj}(t) x^t}{\sum_{t=1}^{N} h_{mj}(t)}$$

$$\Sigma_{m(j+1)} = \frac{\sum_{t=1}^{N} h_{mj}(t)[x^t - \mu_{mj}][x^t - \mu_{mj}]^T}{\sum_{t=1}^{N} h_{mj}(t)}$$

$$h_{mj}(t) = \frac{c_{mj}\ \mathcal{N}(x^t; \mu_{mj}, \Sigma_{mj})}{\sum_{i=1}^{n} c_{ij}\ \mathcal{N}(x^t; \mu_{ij}, \Sigma_{ij})}$$

3 HMM (Hidden Markov Models)

The study of how multiple events influence each other can become cumbersome through just analytical descriptions. Hence we use something called the Markov models which simplifies our understanding of these transitions from one event to the other.

Markov models are graphical representations of how likely it is for an event to be caused by any other event. The model can be represented by a matrix where each element is the probability of event i causing event j. It can also be visualised as a graph where each node represents an event and the weight of the directed edge between 2 nodes represents the probability with which one event causes the other. This probability is called transition probability and is represented by a throughout the paper. These events are observable in the sense that we can measure/observe these events directly, whereas sometimes

these events are not directly observable. For example if the mood of a person depends on the weather, then the event of the person being happy is not directly observable but the weather is observable. These kind of situations are described by Hidden Markov Models.

A Hidden Markov Model is an extension to the usual Markov Model, where we not only have our desired events (states) but also observable events that help us in deducing the desired events. This requires an additional probability matrix which represents the probabilities of an observable event getting triggered when the desired event occurs. These are called emission probabilities and will be represented by b in the remainder of the paper.

3.1 Mapping Phonemes

Given a set of T feature vectors of an audio sample $n_1, n_2, \ldots, n_T$ and a learned HMM, our objective is to map each of the T vectors to a *phoneme* from our HMM. A learned HMM consists of

- n states $S_1, S_2, \ldots, S_n$ that represents different phonemes in a language
- Transition probabilities a_{ij} such that

  $$a_{ij} = P(q_t = S_i | q_{t-1} = S_j),$$

  where q_t is a state at time t
- Emission probabilities $b_i(x_t)$, where $b_i(x_t)$ is the probability that state S_i generates the feature vector x_t

The process of how we actually obtain a trained Markov Model will be explained later with the Baum-Welch algorithm.

We need to map our observation vector $X = (x_1, x_2, \ldots, x_T)$ that we got from the audio sample windows to a phoneme sequence W^* such that

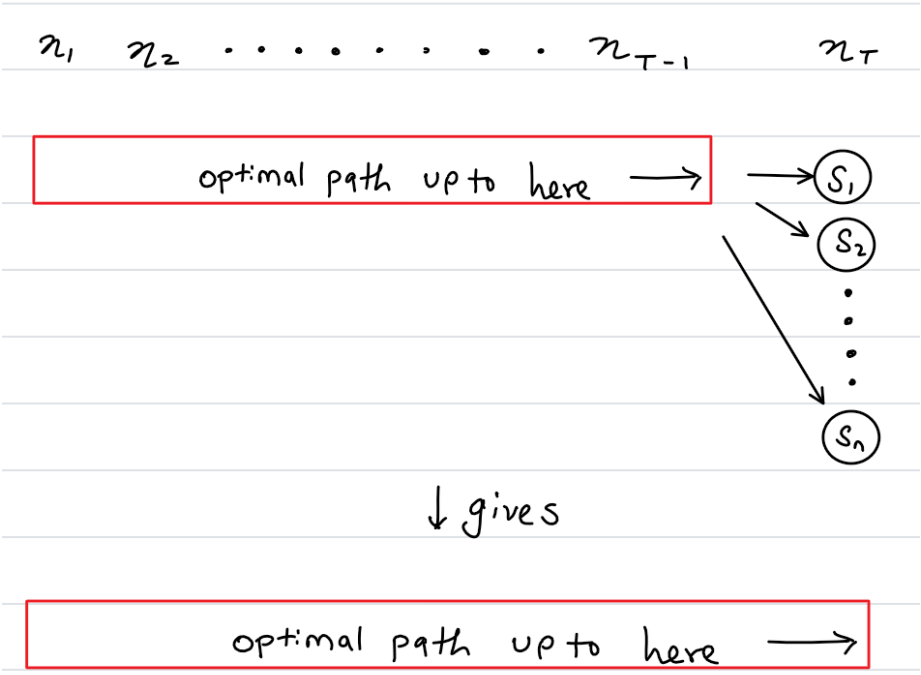
$$W^* = \underset{w}{\operatorname{argmax}}\, P(w|X),$$

where w is a phoneme sequence of length T.

Here, the idea is to iterate over all possible phoneme sequences of length T and find the one with the highest probability of generating the observation vector X. The time complexity of this brute force approach is $O(n^T)$, which is exponential and is unacceptable for our purpose. However, using the Viterbi algorithm, we can calculate state paths recursively, making this a solution that uses dynamic programming.

3.2 Viterbi Algorithm

The essence of this algorithm is to find an optimal path of states for the audio windows from $t = 1$ to $t = T$.



As shown in the figure, using the optimal path till time $t - 1$, we will calculate the optimal path till time t. Hence, we need to define a recursive relation:

$$V_t(j) = \max_{q_1,q_2,\ldots,q_{t-1}} P(x_1, x_2, \ldots, x_t, q_1, q_2, \ldots, q_{t-1}, q_t = S_j|\lambda), \text{ where } \lambda$$

is the set of transition probabilities and emission probabilities of the HMM

Here $V_t(j)$ is the joint probability of observing x_t from the state S_j at time t

with an optimal sequence of states till time $t-1$.

$$V_t(j) = \max_{i=1}^{n} V_{t-1}(i) \cdot a_{ij} \cdot b_j(x_t)$$

For the base case of this recursion, we will use the initial probability distribution.

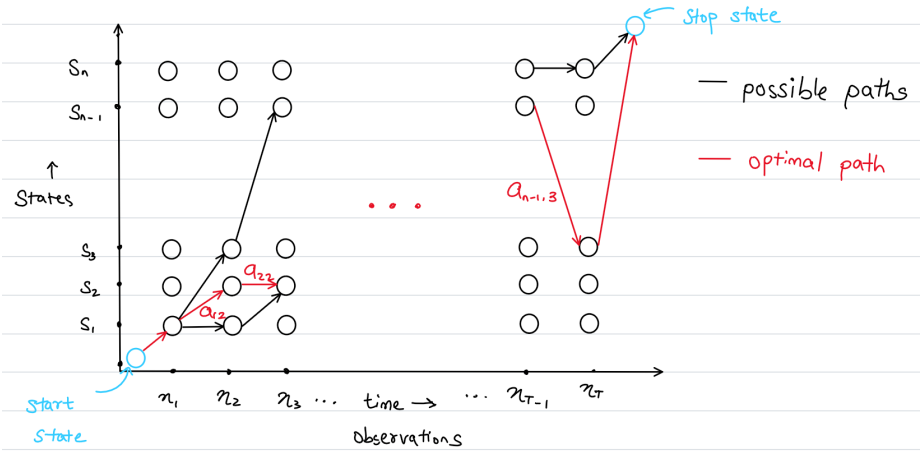
$$V_1(j) = \pi_j b_j(x_1)$$

where π_j comes from the initial state distribution of the HMM

Using the above recursive relation, we can find the values of $V_t(j)$, where $t = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, n$. To find the final optimal path, let $z(t)$ represent the index of the optimal state at time t,

$$z(t) = \operatorname*{argmax}_i V_t(i)$$

The optimal state at time t is $S_{z(t)}$.



Hence, the optimal path or the sequence of the states that is most likely to generate the observation vector $X = (x_1, x_2, \ldots, x_T)$ is,

$$S_{z(1)} S_{z(2)} S_{z(3)} \ldots S_{z(T)}$$

To conclude the algorithm, we found the sequence of phonemes that most likely correspond to the given audio signal using the Viterbi algorithm that has a time complexity of $O(n^2 T)$, which is a great improvement compared to the brute force method with an exponential complexity.

3.3 Baum-Welch Algorithm

The Baum-Welch algorithm is used to build the HMM that is used later in the speech recognition process. It is similar to solving an optimisation problem. For example, if there exists a function $f(x, y)$ defined as,

$$f(x, y) = ax^2 + by^2 + 2hxy$$

Optimising/minimizing the function in this case through conventional methods is not possible as x and y are independent. So, in such cases, we first fix one variable and then optimize the function and continue doing so for every variable. This is an iterative solution. In the example we took above, it can easily be done by mathematical formulas like gradient descent. In such methods we fix all variables but one variable and try to find the direction of the change of the function by partially differentiating with respect to that one variable, and continue to do this for the rest of the variables.

To understand the problem, we have to find the state distribution at time t (probability distribution of states at each time). But this depends on the transition and emission probabilities of the HMM. But to find these probabilities, we need the state distribution at each time. Hence, we use the iterative solution to obtain our system parameters (a, b). We fix a and b and vary the state distributions at each time, and then we vary (a, b) itself.

Now the question arises about our objective function. What are we trying to minimize/maximize? How do we decide whether a solution is better or worse than the other. This objective function is inherited from the EM algorithm. Baum-Welch Algorithm is in fact a special case of the EM algorithm.

The objective function can be intuitively thought of as a performance index function which needs to be maximised. The performance index is a measure of the similarities between the observations that are given in the training set and the observations obtained from the system parameters(a,b). Where a is the matrix of transition probabilities and b is the emission probabilities.

Let us now try to mathematically formulate the relations between the state distributions and the system parameters.

$$\alpha_i(t) = P(Y_1 = y_1, \ldots, Y_t = y_t, X_t = i|\theta), \text{ where}$$

$$\alpha_i(t) \text{ is the forward probability,}$$

$$(Y_1 = y_1, \ldots, Y_t = y_t) \text{ is the probability of all observations till time } t,$$

$$X_t = i \text{ states that the internal state at time } t \text{ is } i,$$

$$\theta \text{ is the system parameters}$$

$$\beta_i(t) = P(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T|X_t = i, \theta), \text{ where}$$

$$\beta_i(t) \text{ is the backward probability,}$$

$$(Y_{t+1} = y_{t+1}, \ldots, Y_T = y_T) \text{ is the probability of all observations after time } t,$$

$$X_t = i \text{ is the internal state at time t is } i,$$

$$\theta \text{ is the system parameters}$$

$$\gamma_i(t) = P(X_t = i|Y, \theta), \text{ where}$$

$$\gamma_i(t) \text{ is the probability of state to be i at time} t,$$

$$Y \text{ and } \theta \text{ are the observations and system parameters}$$

$$\varepsilon_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta), \text{ where}$$

$$\varepsilon_{ij}(t) \text{ is the probability that the state is i and j at times t and t+1,}$$

3.3.1 Initializing, Forward and Backward Algorithms for a HMM

Now, let us calculate $\gamma_i(t)$,

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{P(X_t = i, Y|\theta)}{P(Y|\theta)}$$

The transition probability for all the observations $\varepsilon_{ij}(t)$ is,

$$\begin{aligned}\varepsilon_{ij}(t) &= P(X_t = i, X_{t+1} = j|Y, \theta) \\ &= \frac{P(X_t = i, X_{t+1} = j, Y|\theta)}{P(Y|\theta)} \\ &= \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{i=1}^{N}\sum_{i=1}^{N}\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}\end{aligned}$$

	Forward	Backward
Initialization	$\alpha_1(j) = \pi_j b_j(x_1)$, where π_j is over the initial state distribution and $b_j(x_1)$ is the emission probability	$\beta_i(T) = 1$, which means that there are no more observations to make in the future
Each time step	$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j x_t$	$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_j(t+1)$
$P(X\|\theta)$	$P(X\|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$, where $P(X\|\lambda)$ is the probability of a sequence of observations and $\alpha_T(i)$ is the final step	$P(Y\|\lambda) = \sum_{j=1}^{N} \pi_j b_j(y_1) \beta_j(1)$, where $\beta_j(1)$ is the final step

We notice that both α, β depend on γ, ε and vice versa,

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \varepsilon_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_{ij}^*(v_k) = \frac{\sum_{t=1}^{T} \mathcal{I}(y_t = v_k) \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

We now present the iterative solution. We take an initial guess of the HMM parameters and then vary the state distributions to optimize it locally. Then, we update our system parameters and again fix them and locally optimize them. We do this till we converge to a solution. The objective function is

$\theta_1^* = \arg\max_{\theta_1} \sum_{x \in \mathcal{D}} E_{\theta_2} P(x, \theta_2, \theta_1)$, where the M-step finds θ_1 and the E-step establishes P(γ, ε|x,a,b)

$\theta_2^* = \gamma, \varepsilon \; \theta_1 = a, b$

Using all the observations, we can train our HMM model using the Baum-Welch Algorithm.

4 Conclusion

The Acoustic Model provides a sufficiently efficient algorithm to model audio signals and recognise the corresponding phones or speech patterns, using probabilistic models. This is done in many ways, including the two commonly used methods, namely Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM).

However, with the advent of newer and more efficient technologies like Deep Learning (DL), these models have taken a backseat. The performance of GMM as a classifier is not impressive compared with other conventional classifiers such as K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree and Naive Bayes. Efficient hybrid systems that combine HMM techniques with Deep Learning are also in use now. Many generative models, however, yet have HMM as the principle method.

5 Contributions

- **Hrishi Narayanan** and **Sriram Devata** explored the different models of Speech Recognition and the possible workflows. Out of multiple approaches, they chose the specific algorithms to explore in the project. They were also responsible for compiling the document and maintaining the narrative of the document.

- **Arihanth Srikar Tadanki** explained the working of GMM and EM algorithm which classifies the phones present in the audio signal, forming the base for the acoustic model.

- **Abhishek Mittal** and **Shreyas Pradhan** explained the mathematics behind training and using the HMM. Abhishek Mittal wrote about the Baum-Welch algorithm, and explained both the forward and backward algorithms in detail, while Shreyas Pradhan wrote about the process of decoding the HMM (involving the Viterbi algorithm), which is used to recognize the phonemes in the audio signal.

References

[1] Tan, T.-P., Besacier, L., & Lecouteux, B. (2014). Acoustic model merging using acoustic models from multilingual speakers for automatic speech recognition. 2014 International Conference on Asian Language Processing (IALP). doi:10.1109/ialp.2014.6973492

[2] Yu, C. K., & Ching, P. C. (1991). A probability decision criterion for speech recognition. Computer Speech & Language, 5(1), 11–17. doi:10.1016/0885-2308(91)90015-i

[3] Yaman, S., Li Deng, Ye, D., Wang, Y.-Y., & Acero, A. (2007). A Discriminative Training Framework using N-Best Speech Recognition Transcriptions and Scores for Spoken Utterance Classification. 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07. doi:10.1109/icassp.2007.367149

[4] Khademi, A. (2017). Hidden Markov Models for Time Series: An Introduction Using R (2nd Edition). Journal of Statistical Software, 80(Book Review 1), 1 - 4. doi:http://dx.doi.org/10.18637/jss.v080.b01

[5] Jaya Kumar, A., Schmidt, C., & Köhler, J. (2017). A knowledge graph based speech interface for question answering systems. Speech Communication, 92, 1–12. doi:10.1016/j.specom.2017.05.001

[6] Microsoft. (2008, June 20). Statistical Speech Recognition: A Tutorial . Retrieved from https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MC_He_Ch02.pdf

[7] Jonathan Hui. (2019, Sep 9). Speech Recognition — GMM, HMM . Retrieved from https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196

[8] Jonathan Hui. (2019, Sep 16). Speech Recognition — Acoustic, Lexicon & Language Model. Retrieved from https://jonathan-hui.medium.com/speech-recognition-acoustic-lexicon-language-model-aacac0462639

[9] Hastie & Tibshirani (2008, Nov 12). Gaussian mixture models. Retrieved from https://statweb.stanford.edu/ tibs/stat315a/LECTURES/em.pdf