Single Word Speech Recognition using Hidden Markov Models with Gaussian Mixture emissions

Ronak Panchal
Department of Computer Science
University at Buffalo
Person #:50205775
ronakhem@buffalo.edu

Jayaram Kuchibhotla
Department of Computer Science
University at Buffalo
Person #:50208766
jayaramk@buffalo.edu

Abstract

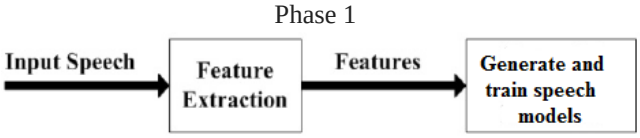
Speech Recognition is the process of converting spoken words or sentences into text that is clear and understandable. This paper describes about the application of Hidden Markov Models with Gaussian Mixture emissions (HMM GMM) to identify single word from one audio file. The audio file has the utterance of the name of fruit. The description has the following steps: (1) The process involved in speech identification, (2)Explanation of reason for using HMM GMM models to identify speech (2) Feature Extraction from the audio input dataset, (3) Details about Implementation and Training the models, (4) Inference of text content from new audio files using the trained models.

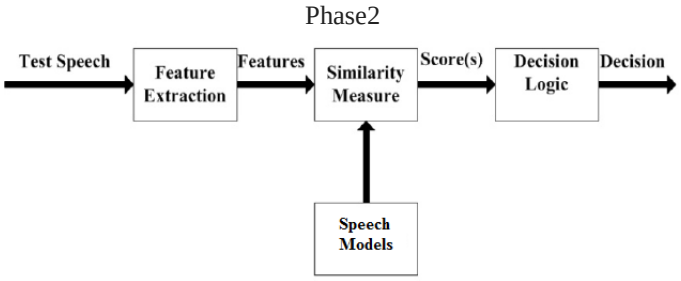
1 Introduction

The achievement of high accuracy in converting speech to text has led to increased creation and usage of Speech recognition systems like Amazon Alexa, Google Now, Apple Siri and Microsoft Cortana. In a pursuit to understand speech recognition, we have come up with idea of implementing Hidden Markov Models (HMM) along with Gaussian Mixture emissions (GMM) .The dataset that we have chosen for this project belongs to Google Code Archive. This dataset consists of 105 audio files each containing utterance of one fruit name made by a single speaker. These 105 audio files are equally divided into seven different categories of fruit names i.e. one category consists of 15 audio files. The whole dataset is partitioned into training (84 samples) and test dataset(21 samples). The API 'HMMGMM' of hmmlearn library is used to implement the models mentioned above in abstract section. In addition, the function 'score' is used for inference. Samples are drawn from the model is using the function 'sample'.

2 Process of Speech Identification

A speech recording is expected to contain meaningful sound in the human audible range 20 Hz to 20,000 Hz.These recordings are stored in different file formats such as .wav ,.mp3. The files in the dataset used for our project are present in .wav format. The speech recognition consists of two phases as shown in the block diagram below:
1. Creation and Training of Speech Models using labelled speech
2. Identification of test speech using trained models





In Phase1, the speech models can be made using either 1.Combination of Deep Neural Network (DNN) and Hidden Markov Model (HMM) or 2.Combination of Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM). The latter is used in our implementation. Seven different models are created and trained in phase 1

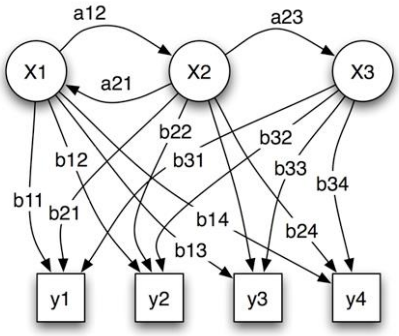
In Phase2, the test speech is scored against all the models; the label associated with the model, which has scored maximum, is picked up.

3 HMM and GMM usage for speech recognition

3.1 Hidden Markov Models

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. An HMM can be presented as the simplest dynamic Bayesian network. Hidden Markov models are generative models based on stochastic finite state networks. Speech is considered to be a Markov process. Hence, HMMs are used for speech recognition

HMM mainly consists of 1.Hidden States 2.Starting probabilities of hidden states 3.Transition probabilities and 4.Emission probabilities 5.Observations



In the above diagram, x1,x2 and x3 are the hidden states; a12,a23,a21 are the transition probabilities; b11,b12,b13,b14,b21,b22,b23,b24,b31,b32,b33,b34 are the emission probabilities ; y1,y2,y3 and y4 are the observations. The basic unit of sound represented by the acoustic model is the phone. For example, the word “bat” is composed of three phones /b/ /ae/ /t/. About 40 to 46 such phones are required for English. In our implementation, we have considered to model a single word utterance using 3 phonemes .Each hidden state represents a phoneme, which implies three hidden states in a single model.

3.2 Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior mode

The Gaussian mixture model for speech representation assumes that a M component mixture model with component weights $\boldsymbol{P(w_m)}$ and parameters $\boldsymbol{\theta_m}$ can represent the spectral shape. The general form of a univariate mixture model is

$$P(x|\theta) = \sum_{m=1}^{M} P(w_m)\, P(x|w_m, \theta_m)$$

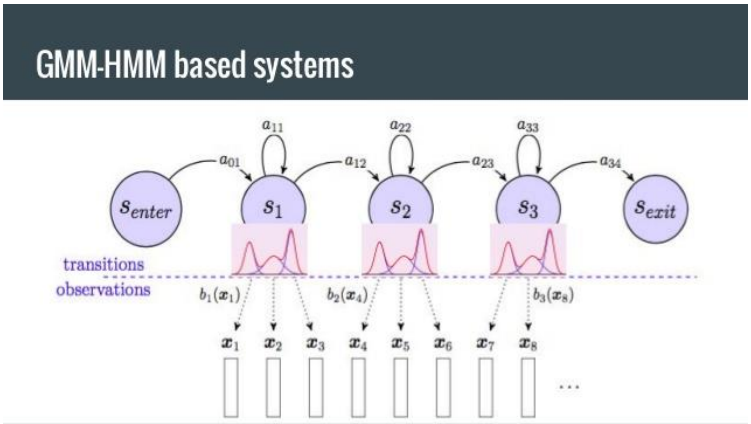
where $\boldsymbol{P(w_m)}$ is the prior probability of component $\boldsymbol{m}$ and $\boldsymbol{\theta_m}$ the component parameters. The mixture components in this case are Gaussian, and hence can be described by

$$P(x|wm, \theta m) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(\frac{(x-\mu)^2}{2\sigma^2})$$

where μ is the mean and σ is the standard deviation for component w_m . The first step in estimating a GMM from the speech is to form a continuous PDF based on the spectral representation. The distance between the spectral PDF and the GMM is then minimized with respect to the GMM parameters to yield the optimal GMM parameters.

3.3 Application of GMM and HMM Combination

Firstly, the speech signal is converted to 'mfcc' vector having 6 features. These mfcc features are represented using the Gaussians. The Gaussians are clustered together to form a Gaussian Mixture Model which represents one hidden state of the Hidden Markov Model. The observations y1, y2, y3 and y4 are considered probabilities of the hidden states or phonemes drawn from the multi variate Gaussian distribution or the Gaussian Mixture Model. These observations are sequenced together using a dynamic programming algorithm 'Viterbi' to form a single word



4 Feature Extraction

The feature extraction stage seeks to provide a compact representation of the speech waveform. This form should minimize the loss of information that discriminates between words, and provide a good match with the distributional assumptions made by the acoustic models. For example, if diagonal covariance Gaussian distributions are used for the state-output distributions then the features should be designed to be Gaussian and uncorrelated. Feature vectors are typically computed every 10 ms using an overlapping analysis window of around 25 ms. one of the simplest and most widely used encoding schemes is based on mel-frequency cepstral coefficients (MFCCs). These are generated by applying a truncated discrete cosine transformation (DCT) to a log spectral estimate computed by smoothing an FFT with around 20 frequency bins distributed non-linearly across the speech spectrum. The nonlinear frequency scale used is called a mel scale and it approximates the response of the human ear. The DCT is applied in order to smooth the spectral estimate and approximately decorrelate the feature elements. After the cosine transform the first element represents the average of the log-energy of the frequency bins.

Typically, 13 mfcc coefficients are used. In the current implementation, we have considered 6 coefficients which yielded better results

The API mfcc(d, nwin=int(samplerate * 0.03), fs=samplerate, nceps= 6)[0] is used for feature extraction , which is part of the library sickits

5 Implementation and Training of speech models

The API of hmmlearn library,

model = GMMHMM(n_components, n_mix, transmat_prior, startprob_prior, covariance_type, n_iter)

1. n_components is the number of hidden states in the HMM
   The value considered = 3 after trial and error

2. n-mix is the number of Gaussian mixtures to form one GMM
   The value considered = 2 after trial and error

3. transmat_prior is the matrix of transition probability prior values
4. startprob_prior is the matrix of start probability model values

   These matrices values are decides using the Bakis Model

   Calculated value are:

   Start Probability matrix
   [1. 0. 0.]

   Transition Probability matrix
   [[0.5 0.5 0.]
   [0. 0.5 0.5]
   [0. 0. 1.]]

5. covariance_type is the string describing the type of covariance parameters to use
   Diagonal covariance matrix is considered in our implementation

Seven models are built using the above API for seven different labels. The mfcc features belonging to same label type are combined together a 2D numpy array. In total, there are 7 different arrays. Each array is passed to model. fit() for training one single model

5.1 Sampling and Calculation of mean, entropy and relative entropy parameters

Each of the seven models are sampled using model.sample() .In a single model, for every column or mfcc feature the parameters mean, entropy are calculated both for sampled and original values. The last column in the tables represents the relative entropy between sampled and original values

Speech Label :orange					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.257	12.4	12.111	2.186	-1.591
2	3.39	-0.736	3.04	2.628	-77.94
3	0.466	0.065	1.075	0.635	-125.58
4	1.14	0.454	1.971	1.525	-119.669
5	0.959	0.172	1.538	1.353	-2.216
6	0.655	-0.788	3.278	0.946	-1.648

Speech Label :pineapple					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.649	11.933	8.731	2.249	-1.31
2	0.832	1.889	2.088	1.302	-0.523
3	0.659	-0.12	1.722	1.037	-28.394
4	0.59	-0.725	3.674	0.949	-1.527
5	0.67	0.03	1.418	0.975	-78.564
6	0.607	-1.233	5.62	0.933	-1.431

Speech Label :banana					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.257	12.654	13.02	2.127	-1.691
2	0.777	2.782	4.113	1.131	-1.061
3	0.303	0.199	0.31	0.235	-0.961
4	0.51	-1.19	6.046	0.682	-1.517
5	0.553	-0.362	2.108	0.761	-3.988
6	0.332	-0.555	3.896	0.222	-1.565

Speech Label :lime					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.834	13.855	9.55	2.322	-1.534
2	0.721	2.051	2.756	1.079	-0.821
3	0.817	-0.062	1.658	1.142	-252.342
4	0.748	-0.869	3.171	1.022	-2.065
5	0.829	0.216	1.586	1.291	-32.227
6	0.8	-1.05	3.497	1.2	-1.767

Speech Label : kiwi					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	1.625	12.129	22.909	1.975	-1.826
2	1.333	-0.262	2.471	1.725	-20.644
3	0.702	0.85	1.127	1.023	-0.151
4	0.609	1.342	1.843	0.907	-0.629
5	0.914	0.546	1.429	1.326	-0.865
6	0.77	-0.788	2.973	1.09	-2.293

Speech Label : peach					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.111	12.911	14.719	2.176	-1.645
2	1.877	-2.424	4.429	2.046	-1.732
3	0.764	0.126	1.423	1.114	-9.597
4	0.639	1.261	1.457	0.966	-0.441
5	0.491	-0.115	1.381	0.681	-42.98
6	0.349	-0.68	4.639	0.342	-1.468

Speech Label : apple					
mfcc feature index	Mean of input data	Mean of sampled data	Entropy of input data	Entropy of sampled data	Relative Entropy
1	2.784	13.824	10.499	2.332	-1.564
2	1.104	1.361	1.526	1.499	-0.039
3	0.962	-0.569	2.685	1.381	-3.781
4	0.608	-0.971	4.237	0.944	-1.47
5	0.392	0.271	0.557	0.408	-0.61
6	0.442	-1.146	6.898	0.597	-1.487

6 Inference:

Each sample of the dataset is passed into model.score() method of each of the 7 trained models. This method returns the log probability value. The model which scores the maximum is associated with the sample . As the trained model is already associated with the speech label, it helps in identifying the word or label for a sample

The tabulation of scores and the maximum score, the predicted label for each of the sample in test dataset is shown below

Test Data Sample No.	orange	pineapple	banana	lime	kiwi	peach	apple	Max score	Predicted label
1	-339.136	-901.777	-1038.74	-1302.209	-231.462	-121.659	-1202.504	-121.659	peach
2	-247.259	-144.64	-77.568	-154.07	-694.932	-684.008	-284.393	-77.568	banana
3	-263.899	-134.822	-389.866	-260.6	-667.558	-684.448	-296.999	-134.822	pineapple
4	-170.339	-750.377	-905.36	-1127.705	-885.663	-929.824	-881.798	-170.339	orange
5	-317.47	-152.514	-362.661	-320.722	-847.949	-870.576	-317.689	-152.514	pineapple
6	-346.25	-888.134	-996.574	-1309.966	-295.595	-144.861	-1097.867	-144.861	peach
7	-396.56	-333.933	-425.375	-453.376	-716.646	-623.698	-140.099	-140.099	apple
8	-500.579	-565.499	-676.347	-570.631	-109.53	-221.97	-910.704	-109.53	kiwi
9	-302.062	-216.759	-96.495	-254.158	-792.557	-779.378	-374.807	-96.495	banana
10	-414.742	-542.793	-629.052	-492.204	-123.171	-292.312	-846.479	-123.171	kiwi
11	-280.267	-235.269	-88.343	-240.339	-985.17	-896.647	-396.475	-88.343	banana
12	-167.405	-881.236	-1020.852	-1289.835	-1049.311	-948.462	-1014.869	-167.405	orange
13	-316.697	-915.392	-1048.457	-1335.41	-246.375	-119.479	-1134.547	-119.479	peach
14	-455.786	-541.971	-640.248	-499.804	-106.776	-210.238	-833.969	-106.776	kiwi
15	-282.204	-146.44	-255.733	-138.275	-540.795	-601.063	-253.391	-138.275	lime
16	-299.871	-899.572	-1029.785	-1242.149	-208.662	-134.91	-1201.934	-134.91	peach
17	-524.728	-618.137	-734.793	-618.645	-102.644	-256.146	-939.598	-102.644	kiwi
18	-328.618	-176.798	-382.308	-253.754	-608.058	-578.658	-75.969	-75.969	apple
19	-162.637	-853.499	-1041.674	-1258.075	-945.297	-975.553	-1028.583	-162.637	orange
20	-337.018	-168.709	-359.631	-306.547	-812.462	-859.501	-353.771	-168.709	pineapple
21	-482.384	-597.815	-726.468	-580.981	-106.23	-255.064	-935.721	-106.23	kiwi

6.1 Accuracy

We are able to achieve an accuracy of 100% with 84 training samples and 21 testing samples. When testing dataset is increased to 50 samples and training dataset is decreased to 55 samples, the accuracy was only 95%. Before splitting, the dataset was shuffled to ensure uniform test and training datasets

7 Conclusion

The consideration of HMMGMM was apt for our problem statement as the aim was to identify a single word in audio file. The same project when implemented by 'KasternKyle' ,the accuracy reported was 71.3% . He has implemented HMMGMM , feature extraction from scratch . He has mentioned that when hmmlearn and mfcc apis are used, the accuracy would be higher which is observed by us.

Efficient continuous speech recognition systems employ RNN and CTC or DNN and HMM and the accuracy we have reported may not be achievable in this case if we use if HMMGMM is used

8 References

[1] https://hmmlearn.readthedocs.io/en/latest/api.html
[2] https://www.semanticscholar.org/paper/Isolated-word-speech-recognition-using-hidden-Mark-Sandsmark/916600115c908ee7599b9696e2a8ba5cd9876c02
[3] https://kastnerkyle.github.io/posts/single-speaker-word-recognition-with-hidden-markov-models/