ASR Project Report

150050068 B.Vamsi
150050091 Bhavan Turaka
140050061 Rishabh Chavhan

Speaker Recognition :

HMM Model

Plan:
Let $\mathbf{S_1, S_2, ... S_n}$ be the set of speakers and **O** be an utterance, then we predict the speaker of the utterance **O** as:

$$\mathbf{S_i} = \text{argmax}(\Pr(\mathbf{O}/(\text{HMM}(\mathbf{S_i}))))$$

Building HMM($\mathbf{S_i}$):
Let the utterances spoken by $\mathbf{S_i}$ in the training set be $\mathbf{O_1, O_2,...,O_T}$
Sound is produced when air passes through vocal track. Different shapes of vocal tract. Different shapes of vocal tract are possible while a sound is produced. A sound can be produced from any of the shape of the vocal track and we don't know which shape produced the sound. It has been observed that the order of change of the vocal tract shapes corresponds to the text spoken while the correspondence between the vocal tract shapes and the sounds emitted identifies a speaker. So the shapes of the vocal tract are hidden states. For identifying a speaker, the emission probabilities emitted in these hidden states play an important role.

So we keep our transition probabilities constant. That means if we assume **m** states, the transition probability matrix would be an **mxm** matrix where each entry is **1/m.**

We assume that the emission sound of a shape of vocal tract follows a Gaussian distribution. So we model the emission probabilities as a vector of shape **d,** then
For each state we have $\boldsymbol{\mu_d, \Sigma_d}$ for estimating emission vectors. In total for the HMM we have $\boldsymbol{\Pi = (\Pi_1, \Pi_2,..., \Pi_m)}$ start probabilities and $\boldsymbol{(\mu_i, \Sigma_i)}$ for a state.
i.e for each state we have to estimate $\boldsymbol{(\Pi_i, \mu_i, \Sigma_i)}$.
Given $\mathbf{O = (O_1, O_2,...,O_T)}$, the acceptance probability by the HMM is equal to

$$= \max_i(\Pi_i \text{x } N(\mu_i, \Sigma_i, O_i)) \text{ x } \Pi \max(N(\mu_i, \Sigma_i, O_i))$$

This acceptance probability resembles that of Gaussian Mixture Model with number of a=gaussians as **m.**

We compare the acceptance probabilities across all HMMs and give out the label corresponding to the HMM which gives maximum acceptance probability.

We assume the mfcc's, their deltas and their double deltas are sufficient enough to represent an emission sound frame. Each utterance is divided into frames of length 25ms with astride of 10ms. We extracted 13 values from each frame, Combining these 13 values with their deltas and double deltas gives us a 39 dimensional vector for each frame. We took the number of states in the HMM model as 50. Increasing the number of states overfits model to the training data of the speaker while decreasing the number of states will reduce the speaker specificness of the HMM,
We train using the Expectation Maximization algorithm and the algorithm converges in less than 10 iterations for each speaker. Only 10 utterances per speaker are used as they give good enough results while reducing the training time. Librespeech is used for extracting mfcc's. HMMLearn is used for creating the HMM model.

Experiments table:

For the 96 people testing case training took 1 hour and testing took 11-12 hours

Speakers	Accuracy
2	100%
4	100%
8	100%
24	100%
64	98.14%
96	93.59%

Limiting each speaker utterance to 2 seconds the results are below

Speaker	Accuracy
2	100%
4	100%
8	100%
16	98.27%
24	97.04%
40	94.82%

Summary:

We are surprised that this simple HMM model could produce such good results. While HMM model gave very good results for Speaker Identification, it is very difficult to use it for Speaker Verification. The ratio of acceptance probabilities of an utterance **O** on an HMM model of the actual speaker and HMM models of the other speakers is small. SO threshold of acceptance for verification is very difficult to set. We can increase this ratio by overfitting the model (increase the number of states) to the utterance, then our model won't give good predictions on unseen utterances of the same speaker. For this model we should know all the future speakers beforehand.

Conclusion:

Pros:

- Very fast training.
- Very small training data per speaker is required
- Results are very good.

Cons:

- We cannot identify out-of-domain speakers
- Doing verification is very tough
- Storing HMMs for every speaker consumes a lot of memory

Neural Network Model:

To overcome the speaker verification which was unsolved by the previous mode, we propose to use embeddings, which will be created by our Neural Network for utterances. These embeddings are used to calculate the identities of speakers by measuring their distances from the centres created by K-Means clustering.

Neural Network:

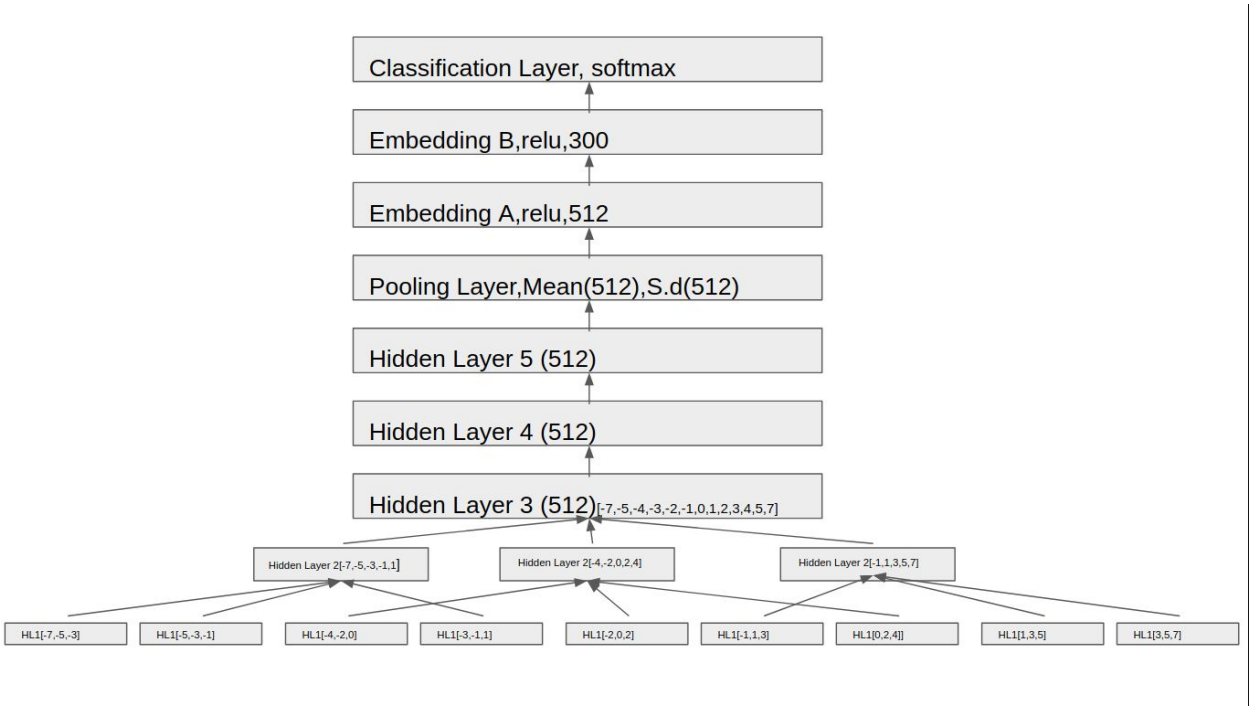
We propose to use a partially connected time-delayed neural network (TDNN). This time-delayed neural network architecture allows us to train the models in parallel and because of partial connections the total number of parameters are greately reduced compared to a fully connected TDNN.

While the RNN uses the entire utterance context, we limit the length of the context for a TDNN. The various partial connections we propose allows us to make the final layers of the TDNN see the supplied time context while drastically reducing the number of parameters compared to a fully connected TDNN.

The TDNN is supplied with a window of size 15 frames. The output from the TDNN are pooled to get mean, standard deviation corresponding to the entire utterance which will be used to

predict the speakers. Embeddings are extracted from the two layers present between the above pooling layer and the softmax classification layer.

Architecture:



Experimental setup:

We used a set of 100 speakers averaging up to 40 utterances per speaker. Each utterance was fixed to a size of 700 frames. MFCC vectors of length 13 along with its deltas and double deltas were taken as the emission vectors per frame of utterance with a stride of 10ms and a frame length of 25 ms

Experiment:

Speaker	Accuracy
2	50.12%
4	27.81%
10	11.2%

(This accuracy is reported on the training set on the neural net)

Summary:

The performance of this model was very bad compared to that of HMM model. There might have been multiple reasons for this.

1. The training data per person for the neural network might have been insufiicient
2. There might have been a bug in our implementation
3. Architecture might not have worked out.

Conclusion:
Pros:

- This could solve the problem of speaker verification
- No need to know all the speakers beforehand
- Performance doesn't decrease as bad as HMMs when speaker numbers are increased

Cons:

- Need large amount of training data to give good results
- Training takes a lot of time

Speech Recognition:

HMM-GMM Model

We plan to solve:

The problem of predicting a word sequence given an utterance

Plan:

A HMM (fst) (D) which takes in accoustic vectors and gives out word sequences can be used to solve the problem.
Instead of building the entire D, we can compose it using **H, C, L, G**

- **H** is an acoustic model
- **C** is a triphone to monophone decoder
- **L** is a Pronunciation model
- **G** is a Language model

Implementation:

G gives out the likelihood of the occurring of a word sequence in the language given the word sequence. We assume that a word in a sequence depends on the **n** preceding words in the word sequence. Based on this **n,** we call the language model as monogram for n=1, bigram for n=2, trigram for n=3..so on

For a n-gram model,

$$\mathbf{Pr(W_1, W_2,...,W_n) = \Pi(Pr(W_i / W_{i-1},W_{i-2},....,W_{i-n+1},))}$$

There will be unseen n-grams since we can't account for all data. Probability of a word sequence can be zero just because of this. To overcome this, we take away mass from the seen n-grams and give them to unseen n-grams. This is called Smoothing.

L is a mapping from word to phone sequence. This will be mostly given by linguists.

C is a direct decoder which decodes from triphone to corresponding monophones.

H is a HMM trained by using utterances in train data and phones corresponding to the word sequence labels for these utterances. The required phones corresponding to the words are obtained from the Pronounciation model (lexicon data)

The training is done using an Expectation Maximizatiuon alogorithm to learn transition, emission probability parameters. The best word sequence for an utterance is decoded using a Veterbi algorithm

Finally, the **D** is created by composing **H, C, L, G**

D = min(det (H o (det (C o (det(L o G))))))

Experimental setup:

We used Kneser-ney discounting (smoothing) for language model

Experiments:

	monophone	triphone
Seen data	26.8%	14.6%
Unseen data	60.4%	81.35

Summary:

Language model should be large enough to handle unseen data.
Training needs considerably large amount of data to get good results on test data
Triphone models give better results compared to monophone models because of better recognition due to context
The graph can be too large

Conclusion:

Pros:

- Better recognition accuracy than end-to-end models
- Faster training compared to end-to-end models

Cons:

- We need linguistic experts help for manually mapping words to phones
- We need to tune the smoothing for our language model as there are unseen n-grams