CS6300 Speech Technology: Assignment 4 Report

-**Team 7:** Akash Reddy A, EE17B001 and Nikhil Mattapally, EE17B138

1 Dynamic Time Warping-based Recognition

Dynamic Time Warping is template-based technique to match two time series which have different sizes, a test sequence and a template sequence. We can find similarity of two sequences which may be of different lengths. Here we are using DTW to match the feature vectors of each audio file to check if both are similar (having same word) or not.

Initially we have extracted Cepstrum and deltaCepstrum features using the given code for each audio file in the train and test data. Then we have sorted all the training data and arranged it such that to test any new sample we can match it with train data. To match the feature vectors between two samples we have found DTW distance between each feature vector and added them to get a final DTW distance.

Steps to check which class a sample belongs to:

1. Match the sample with the entire training data and store the distances.
2. Find 20 training samples with least distances and note class of each sample.
3. Declare the class of the test sample as the class which has highest number of training samples in the 20 samples.

After running this test for all the 60 test data samples we obtained 52 correct, which results in an accuracy of 86.6%. Most errors were for words 'five' and 'zero'.

2 Discrete HMM-based Speech Recognition

This method of speech recognition uses statistical modelling to classify the recording into a word. The word with the highest likelihood of generating the observed sequence of feature vectors is returned as the output.

This likelihood for each word W is generated by estimating the Hidden Markov Model from the feature vectors of all the train examples of W. Since speech is a quasi-stationary process, it can be assumed that throughout each window in which any recording can be assumed to be stationary, it remains in the same state of a Markov Model, after which it transitions into the next state. However, these states and transition probabilities are not known to us, and we have to estimate the parameters of the Markov Model from the training examples that we have.

Since the HMM is discrete in our modelling, we need to make sure that the emitted observations from the feature vectors are discrete. By performing MFCC feature extraction, we obtain continuous real-valued (floating-point) vectors as a feature vector. In order to map each vector to a discrete output, we perform K-Means clustering, and then train our HMM models using these discretised outputs.

Steps:

1. The feature vectors of all recordings of all digits are initially taken together (in our assignment, the feature vectors are 38-dimensional), and K-Means clustering is performed on them. This way, each continuous-valued 38-dimensional feature vector get mapped to a single integer, the cluster index. Since each feature vector belongs to a certain phone, we have used the total number of phones to empirically choose the number of clusters that they belong to. We have:
    - two = /t//u/

- three = /th//r//i/
- four = /f//o//r/
- five = /f//ai//v/
- zero = /z//i//r//o/

and there are 10 unique phones. We have chosen 16 clusters for K-Means Clustering, and it has performed well compared to a few other values. After applying K-Means Clustering and associating each feature vector with the index of the cluster that it belongs to, each digit recording represented initially as a $n \times 38$ sequence of feature vectors is converted to a $n \times 1$ sequence of observations.

2. We store all the length n observation sequences of all recordings of each digit 'two', 'three', 'four', 'five', and 'zero' in separate files in order to train the HMM model for each digit separately. We have chosen the number of states for a certain digit following a trial-and-error method, and found numbers that gave a good accuracy. The HMMs for 'two', 'three', 'four', 'five', and 'zero' have been assigned 5, 5, 4, 5, 10 states respectively

   In general, there is no hard and fast rule to select the number of states for the HMMs. If we pick too many states, and hence parameters, we run the risk of training data not being sufficient to learn the parameters well. Nevertheless, we have obtained good results using these numbers of states, so we have stuck with them.

3. Next, we have trained the HMMs with the clustered feature vectors, number of symbols = 16, and number of states as per the digit. These are fed as arguments to the `train_hmm.cc` file, which uses the Baum-Welch Algorithm to train the HMMs.

4. Similarly, the test feature vectors are also mapped to their clusters, and a sequence of observations (cluster indices) is obtained for each text recording. They are all put into a single file and we obtain the log likelihoods (alpha values from the Forward method of Testing) of each sequence being generated. We repeat this for each model, and store the sequence likelihoods in a single file.

5. Finally, for each sequence, we give the prediction as the digit corresponding to the model with the highest likelihood for each sequence. Since log is a monotonic function, we can directly compare the alpha values to obtain the prediction for each sequence.

The accuracy we have obtained is 91.66%, which is an improvement over DTW-based recognition by 5%.

****