EE 679: Computing Assignment 3

Prajval Nakrani

17D070014

# Methodology

I implemented the **VQ codebook matching** algorithm. The work flow is as follows:

1. Preprocessing:
   1. Speech End-Pointing: We first remove the unwanted silences from the audio signal. That is because the silences in the audio clips are irrelevant to the task of classification and would result in a reduction of accuracy. Hence we extract the exact utterance ignoring the silences. We do this by putting a threshold on the short time energy and removing the frames with energy lower than the threshold. See code for details of implementation.
   2. Rejection: After speech end-pointing, if the resulting signal turns out to be too small, we reject the utterance itself. That is because there should a minimum time that a normal person takes to pronounce a word clearly. The value of threshold may depend on the number of phonemes in the utterance.
   3. Feature Extraction: We mainly use the MFCC features of the extracted audio signals. However, along with the MFCC vector we compute the MFCC\_delta and MFCC\_delta\_delta feature vectors to incorporate temporal information as well in our features. The finally feature vector is the concatenation of the all the 3 feature vectors and hence we obtain a 39-dimensional feature vector for every frame of utterane.
2. Code-Book Generation: For every utterance in the training data of a particular word, we first do speech end-pointing to extract the relevant part of the signal. Then for every frame of the extracted signal we extract a 39-dimensional feature vector as explained above and add that vector to the codebook of that word. We repeat this process for every word and hence we obtain one codebook for every word.
3. K-means Clustering: We don’t directly use the Code-book for searching. That is because it becomes computationally very expensive. We first apply K-means clustering to obtain representative vectors of the K clusters of the respective codebook. This process is called as Vector Quantization. We vector quantize every codebook individually and obtain the representative vectors for each one of them.
4. Inference: For inference, for every test utterance, we first endpoint the signal and extract the frame-wise features as explained above. Now we compute the minimum total distortion of the utterance with respect to every codebook. That is obtained by adding up the minimum distortions of every frame of the utterance with respect to that particular codebook. Finally, the codebook that gives the minimum total distortion for that utterance is predicted to be the word that was spoken. Please note that we have used the same training set for prediction of noisy test utterance as well.

# Hyper-parameter Values:

* Window Length = 30 ms
* Hop Length = 15 ms
* Threshold for End-Pointing of Clean Utterances = 1% (of max energy)
* Threshold for End-Pointing of Noisy Utterances = 10%
* Minimum duration for rejection = 300 ms (for 4 phoneme words)
* Minimum duration of rejection = 200 ms (for 3 phoneme words)
* Minimum duration of rejection = 150 ms (for 2 phoneme words)
* Number of clusters for Vector Quantization = 64

# Results

Task A (Clean Test Utterances):

* Accuracy = 74.09 %
* Confusion Matrix (Clean)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Words** | Down | Go | Left | No | Off | On | Right | Stop | Up | Yes |
| Down | 0.710 | 0.045 | 0.024 | 0.107 | 0.016 | 0.045 | 0.008 | 0.008 | 0.016 | 0.016 |
| Go | 0.053 | 0.604 | 0.024 | 0.136 | 0.024 | 0.014 | 0.014 | 0.009 | 0.107 | 0.009 |
| Left | 0.049 | 0.016 | 0.710 | 0.043 | 0.010 | 0.016 | 0.027 | 0.010 | 0.043 | 0.071 |
| No | 0.12 | 0.115 | 0.026 | 0.68 | 0.004 | 0.013 | 0.004 | 0.004 | 0.031 | 0 |
| Off | 0.025 | 0.008 | 0.025 | 0 | 0.692 | 0.136 | 0.008 | 0.025 | 0.076 | 0 |
| On | 0.048 | 0.017 | 0.004 | 0.013 | 0.08 | 0.760 | 0.017 | 0 | 0.057 | 0 |
| Right | 0.024 | 0.024 | 0.039 | 0.014 | 0.019 | 0.014 | 0.824 | 0 | 0.039 | 0 |
| Stop | 0.032 | 0 | 0.027 | 0.005 | 0.010 | 0.010 | 0.005 | 0.843 | 0.064 | 0 |
| Up | 0.040 | 0.010 | 0.090 | 0.020 | 0.050 | 0.040 | 0.020 | 0.030 | 0.696 | 0 |
| Yes | 0.01 | 0 | 0.015 | 0.025 | 0.020 | 0.005 | 0.040 | 0.010 | 0.010 | 0.863 |

Task B (Noisy Test Utterances):

* Accuracy = 44.10 %
* Confusion Matrix (Noisy)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Words** | Down | Go | Left | No | Off | On | Right | Stop | Up | Yes |
| Down | 0.315 | 0.068 | 0.190 | 0.028 | 0.226 | 0.024 | 0.016 | 0.032 | 0.052 | 0.044 |
| Go | 0.017 | 0.383 | 0.174 | 0.053 | 0.174 | 0.004 | 0.013 | 0.035 | 0.098 | 0.044 |
| Left | 0.008 | 0.017 | 0.535 | 0.030 | 0.234 | 0.0 | 0.022 | 0.022 | 0.053 | 0.075 |
| No | 0.046 | 0.096 | 0.138 | 0.34 | 0.226 | 0.012 | 0.012 | 0.025 | 0.050 | 0.050 |
| Off | 0.005 | 0.026 | 0.186 | 0.015 | 0.563 | 0.037 | 0 | 0.031 | 0.090 | 0.042 |
| On | 0.025 | 0.047 | 0.123 | 0.008 | 0.329 | 0.376 | 0.017 | 0.008 | 0.055 | 0.008 |
| Right | 0.013 | 0.026 | 0.243 | 0.013 | 0.115 | 0.004 | 0.433 | 0.017 | 0.053 | 0.079 |
| Stop | 0.009 | 0.009 | 0.198 | 0.004 | 0.342 | 0.004 | 0 | 0.346 | 0.036 | 0.049 |
| Up | 0.017 | 0.041 | 0.184 | 0.023 | 0.279 | 0.011 | 0.005 | 0.047 | 0.345 | 0.041 |
| Yes | 0.004 | 0.004 | 0.242 | 0.012 | 0.125 | 0 | 0.025 | 0.030 | 0.021 | 0.532 |

Observations:

* We observe that Go and No are highly confused with each other. This is because they have a common vowel “o” and they only have two phonemes and the duration of vowel is usually more than a plosive. Hence the observed confusion.
* Noisy test utterances tend to reduce the accuracy of prediction quite a lot. This is because the added noise changes the spectral properties of the acoustic signal of the utterance and hence causes disturbance in the MFCC vectors which leads to poor accuracy of prediction.

I tried to generate a codebook of noisy test utterances. That is by adding all the background noises to every utterance in the training dataset. By doing so, I obtained a “noisy” training dataset. On this I applied the algorithm as mentioned above, however it resulted in poor performance instead of better. For some reason, the prediction method was predicting the same word for all the test utterances. Hence, I decided to stick with the original clean training set.