COM4511 SPEECH TECHNOLOGY: ISOLATED SPOKEN DIGIT RECOGNITION

Jack Cheng Ding Han

University of Sheffield

ABSTRACT

The project regards implementing, training and evaluating Hidden Markov Models (HMMs) using the Viterbi algorithm for the recognition of isolated spoken digits. The use of HMMs are compared to GMMs. How the performance and results change according to different parameters–features (Mel-frequency cepstral coefficients or filter banks), number of states and mixtures– are all explored.

1. INTRODUCTION

Isolated digit recognition is a form of automatic speech recognition where the audio signals involve the pronunciation of a single digit, zero to nine the English reading of decimal digits. Multiple digits may be spoken with pauses in between each individual utterance (hence "isolated"). A prominent use is in automated telephone services [1].

For this project, the task of isolated digit recognition is performed with the Hidden Markov Model (HMM). A given HMM has the attributes listed in table 1 [2, ch. 5.5.2].

Attribute	Description
$Q = q_1, q_2, \ldots, q_N$	Set of N states
A	a_{ij} is how likely is the transition i to j
$O = [o_1, \ldots, o_T]$	Sequence of T observations
$B = b_i(o_t)$	Emission probabilities
q_0, q_F	Start and finish states

Table 1. Attributes of a HMM.

2. THE VITERBI ALGORITHM

The Viterbi algorithm is used to calculate the likeliest sequence of states given observations. According to Jurafsky & Martin in 2009, the Viterbi algorithm is perhaps the most common decoding algorithm used for HMMs [2, ch. 5.5.3]. During the initialisation stage, likelihood and back-pointer matrices of size $N \times T$ are initialised with zeroes. In the recursion stage, the entry for a given state s and time step t in the likelihood matrix is updated by finding state j that maximises the product of the transition from s to j and the emission probability of state s at time t. The predecessor is stored using the back-pointer matrix. At the termination stage, the most likely path can be recovered by tracing back using through the back-pointer matrix.

3. HMM TRAINING

Within the context of this project, the observations are the features from sampling the signal–with a choice between MFCCs or filter banks, the hidden states must be inferred from the sequence of observations to get predictions of digits..

The following steps occur at each iteration of training:

1. Train by accumulating statistics for each observation provided by a feature list, using the respective Viterbi alignment.
   (a) To get the Viterbi alignment, the log-likelihood and most probable state sequence are calculated by performing Viterbi decoding on the features.
   (b) To accumulate state observations and state transitions, one must:
      i. Count how many times the state in question appears in the state sequence.
      ii. Accumulate the state transitions by adding the count to a state transitions array.
      iii. Find the first index the state has in the state sequence.
      iv. Accumulate the state observations by concatenating the current list of observations and the features at said index.
   (c) Fit each state with its respective observations to update the probability density functions.
   (d) Update log-probabilities in the transition matrix.

In each iteration of training, the log-likelihood generally increases quickly from 0 to 1 and from about 4 to 19 starts to converge. Figure 1 shows the plot for the digit 9.

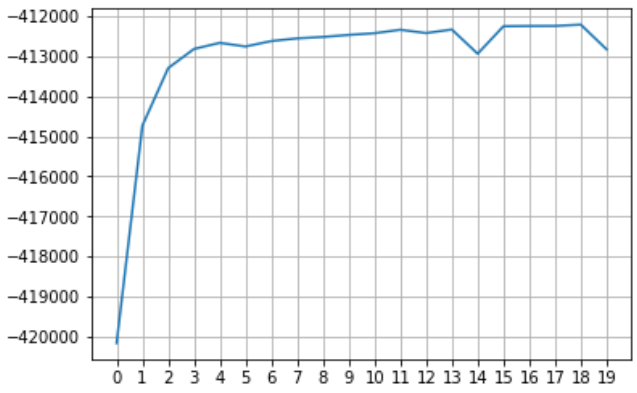

Fig. 1. Plot for digit 9. 10 states, 3 mixtures using MFCCs.

4. TESTING PROCEDURES

For each `.wav` file that is read, its associated target label is obtained. The log-likelihood of each HMM (one of each digit) is computing using Viterbi decoding. With the HMM with the maximum likelihood found, the prediction for the digit can be obtained. The word error rate can be calculated by taking the mean value of how many times the target label did not match the predicted label.

5. RESULTS AND DISCUSSION

5.1. HMMs vs. GMMs

As shown in table 2, for the same number of Gaussian mixtures (3), the GMMs gave higher word error rates than HMMs which had the number of states fixed to 10.

	MFCC	Filter Banks
GMM	22.42%	20.76%
HMM	1.97%	3.63%

Table 2. Word error rates for GMMs and HMMs with 3 mixtures.

At 27 mixtures, the GMM with MFCCs was able to achieve a word error rate of about 3.48%. At 24 and 25 mixtures, the GMM with filter bank was able to achieve a word error rate of about 4.39%.

5.2. MFCC vs. Filter Banks

Table 3 shows how the number of states and mixture components can change the word error rate for MFCCs, table 4 shows the same same for filter banks. With the exception of 10 states and 1 mixture, MFCCs had a lower word error rate than filter banks for the same combination of states and mixtures. 3 states gave higher word error rates for both kinds of features. The greatest word eror rate was for filter banks with 3 states and 1 mixture and the lowest word error rate was for MFCCs with 10 states and 3 mixtures.

Figure 2 shows the log-likelihood trend for digit 9 with 3 states and 1 mixture, notice the higher log-likelihood values and the faster convergence; it settled on -532807.608495589 by the twelfth iteration.

	10	3
3	1.97%	3.79%
1	5.76%	13.33%

Table 3. MFCC word error rates for different state-mixture combinations

	10	3
3	3.63%	6.21%
1	3.79%	32.12%

Table 4. Filter bank word error rates for different state-mixture combinations

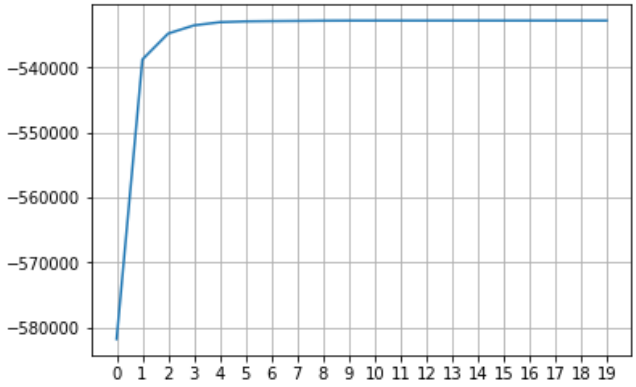

Fig. 2. Plot for digit 9. 3 states, 1 mixture using fbanks.

6. CONCLUSIONS

The HMMs performed better than the GMMs. The HMMs had Gaussian mixtures for each state whereas one GMM is by definition one Gaussian mixture model. The HMMs used for comparison to the GMMs had 10 states.

With a suffience number of mixtures, MFCCs gave better results than filter banks.

7. REFERENCES

[1] B H. Juang and Lawrence Rabiner, "Automatic speech recognition - a brief history of the technology development," 01 2005.

[2] Daniel Jurafsky and James H. Martin, *Speech and Language Processing (2Nd Edition)*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.