EEEM030 Group Assignment 2: Speech Recognition

Assessment: 15% of module

1. Overview

The second coursework assignment for *EEEM030 Speech & audio processing & recognition* is designed to give you an opportunity to experience the machine learning methodology in a small development team, and to practice the key algorithms that are used to extract features and to initialize, train and test your models. In doing so, you will get to the efficient recursive likelihood-based procedures to observe the effects of training and to perform recognition for a simple, isolated-word recognition task.

The aim of the coursework is to develop a simple speech recognizer in Matlab that uses HMMs with a small vocabulary of eleven key words.

This is a group project. You will be assigned to a group with 4-5 members. Your group will submit one joint technical report (max. 5,000 words, plus references and any appendices). The report will include a distinct chapter from each team member, plus other essential parts submitted as a collective contribution (i.e., the abstract, introduction/background, conclusion and references). The whole document must be consistently formatted in a single PDF file that is less than 12MB. It must be legible and must not exceed 50 pages in total. Your final mark will assess both your individual chapter and the quality of the technical report as a whole.

The spokesperson for your group must submit your **PDF file** to the EEEM030 assignment folder in SurreyLearn by the deadline **4pm Tuesday 5th December 2023** (week 11).

The recommended structure for your group technical report is as follows, where each team member is to contribute **one** of the main central chapters highlighted in bold:

- Front page with university crest, module code (EEEM030), name of the assignment, title of the report, list of co-authors and copyright notice
- Abstract
- Table of Contents
- Introduction
- **First contribution chapter**
- **Second contribution chapter**
- **...**
- **Last contribution chapter**
- Summary/conclusion
- References/bibliography
- Appendix

You may use publicly-available source code, where it is relevant, but any code you use (or adapt) that was written by someone outside your team **must be cited** with due reference (to avoid accusations of plagiarism), stating how it has been used. You will need to ensure you understand what it does and how it works in order to complete the assignment successfully, as there can be significant variations (e.g., around pdf normalisation or definition of null states).

As with all formally assessed coursework, you may discuss the concepts associated with the coursework with your peers in other teams, but not the details of any solution that you implement. You cannot share code between different teams. In line with University policy, you may use other sources, such as textbooks, lecture notes, articles, online tutorials and code libraries, but failure to cite them correctly may be viewed as plagiarism and trigger an academic misconduct investigation. So please reference all your sources carefully and thoroughly!

2. Model prototype and speech data

You are asked to initialize, train and test a set of hidden Markov models (HMMs), with one for each of the key words in the vocabulary. Each model is an 8-state HMM with 13-dimensional continuous probability density function (pdf), i.e., N=8 and K=13. The HMM has a strict left-right topology, as illustrated with some typical values of the state-transition probabilities A in Table 1. The output probabilities B are obtained with a 13-D multivariate Gaussian pdf that has a mean $\boldsymbol{\mu}$ and a diagonal covariance matrix $\boldsymbol{\Sigma}$.

0	1	0	0	0	0	0	0	0	0
0	0.8	0.2	0	0	0	0	0	0	0
0	0	0.8	0.2	0	0	0	0	0	0
0	0	0	0.8	0.2	0	0	0	0	0
0	0	0	0	0.8	0.2	0	0	0	0
0	0	0	0	0	0.8	0.2	0	0	0
0	0	0	0	0	0	0.8	0.2	0	0
0	0	0	0	0	0	0	0.8	0.2	0
0	0	0	0	0	0	0	0	0.8	0.2
0	0	0	0	0	0	0	0	0	0

Table 1: State-transition probability matrix, $A = \{\pi_i, a_{ij}, \eta_i\}$, including entry and exit transitions.

The speech data comprise a collection of audio recordings from a variety of speakers. The vocabulary includes the set of key words listed in Table 2. Each audio file contains only one word. The filename indicates the index of the word spoken in that file and spells out the corresponding word.

You are initially provided with development data for training, and will later obtain a set of validation data to test your system.

Index	1	2	3	4	5	6	7	8	9	10	11
Word	“heed”	“hid”	“head”	“had”	“hard”	“hud”	“hod”	“hoard”	“hood”	“who’d”	“heard”

Table 2: Set of isolated words making up the vocabulary for this recognition task.

3. Feature extraction, model initialisation, training and decoding

For feature extraction, use 13 Mel-frequency cepstral coefficients including the zeroth coefficient. Do not include the velocities and accelerations, i.e., the delta features and delta-delta-features. This will provide a 13-dimensional feature vector for each frame of audio data. The frame should take a 30ms block of audio samples, apply a tapered window and a hop size of 10ms, i.e., with a 20ms overlap.

For model initialisation, you are advised to compute the global mean and variance across the whole development set. For the mean, this should give you a 13-D vector of numbers. For the variance, we also want a set of 13 values in order to populate the diagonal of the 13×13 covariance matrix. So you can either compute the variance for each dimension separately, or compute the full covariance and set the off-diagonal values equal to zero. You will use a scaled version of this global variance to set a variance floor in training, i.e., to limit how narrow these distributions can get.

From the number of frames extracted from the audio files, you can compute the average duration of each state, in frames, and hence an estimate for the self-loop transition probability and the probability of transition to the next state. These two values can then be applied to all the state transitions in turn, as depicted for the illustrative example in Table 1, with the self-loop at 0.8 or four fifths, and the onward transition at 0.2 or one fifth. You can initialise the self-loop probability a_{ii} from the average duration τ as: $a_{ii} = \exp(-1 / (\tau - 1))$.

For training, you should apply the Baum-Welch equations to re-estimate the models, as described in detail in your lecture notes. This requires the implementation of the forward and backward procedures to obtain the forward and backward likelihoods, α and β respectively, followed by the calculation of occupation and transition likelihoods, γ and ξ. These are used to increment the accumulators as each training file is processed, and finally these provide the updated parameter values from one complete training epoch, where $\hat{A} = \{\hat{a}_{ij}\}$ and $\hat{B} = \{\hat{\mu}_i, \hat{\Sigma}_i\}$.

For monitoring the training process on the development data, and later for evaluating the trained models with the validation data, you are asked to use the Viterbi algorithm as the basis of the decoder. This is used to perform recognition, to identify the output label for a given test file, and ultimately to score recognition errors, which may also be shown in the form of a confusion matrix.

Note that the likelihood calculations may be computed directly as probabilities or, more efficiently, in the form of log-probs by taking the natural logarithm of each of the probability values. Equally, to maintain the probability estimates within Matlab's range of numerical precision, you may need to re-scale the probabilities. This can either be done each time frame, for example, or using a global scale factor. You should expect to encounter this problem with this assignment, so inspect the probabilities obtained in your calculations and check for overflow/underflow in the values. Matlab may use infinity (Inf) or not-a-number (NaN) to denote such a result.

4. Tasks

This assignment performs isolated word recognition (IWR) by training HMMs according to the Expectation-Maximization method with the Baum-Welch equations, and running the recognizer by decoding with the Viterbi algorithm. You are provided development data for training and some test data for evaluation. You are asked to record and test additional speech data.

As a group, you will need to decide how you are going to manage and coordinate your work. You will need to distribute tasks across the team to enable each individual to make a contribution, according to their abilities and the difficulty of the tasks. You may have to adapt your plan, as work evolves.

The tasks that your group is asked to undertake are as follows:

1. For each member of the team, obtain a recording of every key word in Table 2
2. Initialise a set of prototype HMMs for each word in the vocabulary
    a. Extract MFCC acoustic features from the training data in the development set
    b. Compute flat-start prototype model parameters from the global statistics of the development set
3. Train the HMMs with the training data
    a. Compute the forward and backward likelihoods via the forward and backward procedures
    b. Determine the occupation and transition likelihoods to increment the accumulators
    c. Re-estimate the model parameters at the end of one iteration over the whole training data, and save the models
    d. Repeat up to a total of 15 iterations
4. Evaluate the recognizer at the start and after each iteration on the development data
    a. Compute the maximum cumulative likelihoods via the Viterbi algorithm
    b. Determine the recognition outputs, score the results and calculate the error rate
5. Evaluate the recognizer on the supplied test data
    a. Extract MFCC acoustic features from the training data in the test data
    b. Compute the maximum cumulative likelihoods via the Viterbi algorithm
    c. Score the recognition outputs and derive the confusion matrix
6. Repeat the evaluation on the team's recorded test data, reporting the score and confusions

For this assignment, it is expected that your calculations are coded in Matlab from scratch, using the built-in functions and libraries available to university users. You will acquire the best understanding of the algorithms working this way. Note that, if you employ third party implementations for any of the required components, such as a toolbox, toolkit, library or software downloaded from an online repository, you must highlight the lines of code that have been contributed by others and **provide the appropriate citation to the source in your references**. Any third party code that is not appropriately cited in your report may be regarded as academic misconduct as an act of plagiarism, and will be attributed to the author of the contribution chapter where this occurs.

5. Assessment

Your assignment mark will take into consideration both your individual contribution and the group's overall achievements. Therefore it is critical that each report explains in the introduction how responsibility for the tasks has been divided, stating who is the author of each contribution chapter.

Assessment of your individual contribution will be based primarily on whether the results in your individual contribution chapter are complete and correct. The group's overall achievement including the quality of your group's technical report, which contains the abstract, introduction, planning of human resources over the project, summary of conclusions, bibliography and any appendices, in addition to those individual contribution chapters.

PJBJ