MSc/MEng Data Mining and Machine Learning (2021)
Lab 3 – Speech Recognition using HTK

Introduction

The purpose of this laboratory is to familiarise you with automatic speech recognition. You will use the Hidden Markov Model Toolkit (HTK) to build a connected digit recognition system which takes an acoustic speech signal as input, performs training of the HMM for each digit and evaluate the performance of the system on a provided dataset. The entire HTK consists of several tools (exe-files), each performing a specific operation, e.g., feature extraction, HMM training, etc. Each tool is executed in the Command Prompt window by typing its name together with passing all the required input parameters. The exe-files of the individual HTK tools are included in the `LabASR.zip` file to be downloaded from Canvas. The zip-file also includes the manual for the HTK software – the manual is big but you are going to need it only occasionally and only as a reference in order to find out the meaning of the input/output parameters which are passed when using a specific HTK tool.

Getting started

Download the zip-file `LabASR.zip` from Canvas to your drive. Open the zip-file and copy the entire directory structure to your drive. Run the Command Prompt Window by going to the Windows Start menu and typing '`cmd`' (no quotes). Use the '`cd`' command to set your directory to the place you copied the unzipped file. You are now set to start running some HTK tools.

Dataset

The dataset used in the laboratory contains recording of spoken digit sequences, where a digit is one of the following: one, two, three, four, five, six, seven, eight, nine, zero, oh. The data is split into training part (folder `TRAIN`) and testing part (folder `TEST`). In each (train/test) part, there is a set of clean (noise-free) recordings (folder `CLEAN1`) and a set of recordings corrupted by an additive noise (i.e., noise signal added to the clean signal) at the signal-to-noise ratio (SNR) of 20 dB and 10 dB (folder `N1_SNR20`, `N1_SNR10`, respectively). The additive noise illustrates the effect of a background ambient noise in practice.

Viewing the signal

In this initial exercise you will practice the use of the `HList` tool. This tool allows you to view wav-files or files containing features extracted from wav-files (the feature extraction can be performed using the `HCopy` tool which will be the subject of the next section). Typing the below gives the values of samples in the wav-file and these are stored in the file `logHList_wav`:

```
HTK3.2bin\\HList -h -C config/config_HList_wav
dataAurora2/wavLabDMML/TRAIN/CLEAN1/FAC_13A.wav > logHList_wav
```

You can examine the file containing the MFCC features (after you have created them as described in the next section) by typing:

```
HTK3.2bin\\HList -h -C config/config_HList_mfcc
dataAurora2/specLabDMML/TRAIN/CLEAN1/FAC_13A.mfcc > logHList_mfcc
```

Feature extraction

The `HCopy` tool enables to extract a sequence of feature vectors from a given wav-file. It is capable of extracting several different types of features, e.g., logarithm filter-bank energies,

MFCCs, etc. By typing the below, you can convert the `MAE_12A.wav` file into a file with the same name but extension `.mfcc` which contains the MFCC features (note that the feature file will be located in a different directory):

```
HTK3.2bin\\HCopy -C config/config_HCopy_MFCC_E
dataAurora2/wavLabDMML/TRAIN/CLEAN1/FAC_13A.wav
dataAurora2/specLabDMML/TRAIN/CLEAN1/FAC_13A.mfcc
```

The `HCopy` tool can be used to extract features for a set of files listed in a given text-file. This can be performed by using the `HCopy` as below, where the `listTrainHCopy_LabDMML_CLEAN1.scp` is a text-file containing the list of files (with a full path) to be processed. This file is located in the `list` directory. Open and view this file and you can see that each line contains name of two files (with a full path) – the first is the file to be used as the input and the second is the file to be used as the output. You will need to modify the path here to be the path where your data are located. After you have done the path modifications, type:

```
HTK3.2bin\\HCopy -C config/config_HCopy_MFCC_E -S
list/listTrainHCopy_LabDMML_CLEAN1.scp
```

The option `-S` is used to specify a script file name (`listTrainHCopy_LabDMML_CLEAN1.scp`) that contains the list of files to be converted.

Building the digit recognition system – parameter set-up

In the previous section, we have converted a set of wav-files into files containing the features. Now, you start to build your digit recognition system. You will need the following:

- Vocabulary list – file `wordList_noSP` located under the `lib` directory – this contains the list of words the recogniser is going to be able to recognise. A model will be built for each vocabulary word.
- Dictionary (or pronunciation model) – file `wordDict` located under the `lib` directory – this defines the mapping of words to acoustic units, i.e., how model of each vocabulary word is built using a single (or a sequence of concatenated) HMMs. Since we are using in this example HMMs of whole words, the dictionary contains a repetition of each vocabulary word. Note that this would be different in a case of building HMMs of each phoneme.
- Language model (or grammar) – file `wordNetwork` located under the `lib` directory – this defines (in a specific format) the set of possible sentences that can be recognised, as well as their relative prior probabilities. If needed, it can be written by hand or more conveniently using the tool `HParse`.
- Features extracted for the training / testing data – are located under `dataAurora2` directory.
- Label files for the training / testing data – file `label_LabDMML_noSP.mlf` located under the `label` directory is to be used in the first instance. You can open this text file and see that it contains the labels (i.e., transcription of what have been spoken in terms of the digits) for all the training data.
- Prototype HMM – file `proto_s1d13_st8m1_LabDMML_MFCC_E` located under the `lib` directory. You can open this text file and see that it contains a definition of the type of HMM to be used – it defines the dimension of the features, the number of states in the

HMM, initial values for means, variances and weights for each state (these values are indicative only – they inform about the structure of the HMM), and the transition probability matrix which determines the possible transitions between states (the transitions assigned to zero will not be possible).

- Configuration file for the individual tools – each tool may have different configuration file (containing the parameters of the processing to be performed).

Building the digit recognition system – training the HMMs

1. Create the directory `hmm0` under `hmmsTrained`. The initial parameters of HMMs are going to be estimated using the tool `HCompV`. By executing the following, the initially trained HMM parameters will be located in the file `hmmdef` (and `vFloors`) under the directory `hmmsTrained/hmm0`. Note that you will need to modify the path in the `listTrainFullPath_LabDMML_CLEAN1.scp` file.

```
HTK3.2bin\\HCompV -C config/config_train_MFCC_E -o hmmdef -f 0.01 -m -S
list/listTrainFullPath_LabDMML_CLEAN1.scp -M hmmsTrained/hmm0
lib/proto_s1d13_st8m1_LabDMML_MFCC_E
```

2. Now you will create 2 files (could be done manually but you are provided exe-files which do the work automatically for you).

   Type the below – it will create file with name `models` containing the HMM definition of all the 11 digits and the silence model. The `models` file could be created manually by simply copying the content of `hmmdef` several times (for each vocabulary unit) and replacing the name according to the vocabulary.

```
HTK3.2bin\\models_1mixsil hmmsTrained/hmm0/hmmdef hmmsTrained/hmm0/models
```

   Type the below, which creates the so-called macro-file having basically the same content as the file `vFloors` but slightly modified structure. The value `13` indicates the dimension and `MFCC_E` the type of features – you will need to modify these when using different features/dimension.

```
HTK3.2bin\\macro 13 MFCC_E hmmsTrained/hmm0/vFloors hmmsTrained/hmm0/macros
```

3. The next step is to run several iterations of the Baum-Welch training procedure. This can be done using the tool `HERest`. Among the input parameters for this tool is the input directory containing the current HMM parameters (which is now `hmmsTrained/hmm0`) and the output directory containing the new re-estimated HMM parameters (which is now `hmmsTrained/hmm1`). Thus, you need to create the new directory `hmm1` and then run:

```
HTK3.2bin\\HERest -C config/config_train_MFCC_E -I
label/label_LabDMML_noSP.mlf -t 250.0 150.0 1000.0 -S
list/listTrainFullPath_LabDMML_CLEAN1.scp -H hmmsTrained/hmm0/macros -H
hmmsTrained/hmm0/models -M hmmsTrained/hmm1 lib/wordList_noSP
```

   Altogether, perform three iterations of the `HERest`. Before each iteration, make a new directory (`hmm1`, `hmm2`, and `hmm3`) where the newly trained HMMs are going to be stored. At each iteration, you should not forget to change the corresponding input and output directory names in the above `HERest` command – use the output directory from the current iteration as the input directory in the next iteration.

4. Now create two new directories `hmm4` and `hmm5`. Then copy the content of the directory `hmm3` into the `hmm4` directory.

5. Create the model for a short-pause `sp` by performing the two commands as below:

```
HTK3.2bin\\spmodel_gen hmmsTrained/hmm3/models hmmsTrained/hmm4/models

HTK3.2bin\\HHEd -H hmmsTrained/hmm4/macros -H hmmsTrained/hmm4/models -M
hmmsTrained/hmm5 lib/tieSILandSP_LabDMML.hed lib/wordList_withSP
```

6. Perform another three iterations of the `HERest` (with `sp` this time) – before each iteration, make a new directory where the newly trained HMMs will be stored.

```
HTK3.2bin\\HERest -C config/config_train_MFCC_E -I
label/label_LabDMML_withSP.mlf -t 250.0 150.0 1000.0 -S
list/listTrainFullPath_LabDMML_CLEAN1.scp -H hmmsTrained/hmm5/macros -H
hmmsTrained/hmm5/models -M hmmsTrained/hmm6 lib/wordList_withSP
```

Training finished! – you have now obtained trained models of digits in the folder `hmm8`, each modelled by 10 state HMM with a single Gaussian PDF with diagonal covariance matrices. Let's go to do testing (recognition).

Building the digit recognition system – recognition

1. The tool `HVite` is to be used for testing of the recognition system. This performs the Viterbi decoding and gives the sequence of models which are most likely to produce the given unknown utterance. Among the input parameters to the `HVite` tool are the trained HMMs and the list of testing utterances (from the testing data directory). First, you need to extract features from the testing wav-files using the `HCopy` tool as described at the beginning of the lab (when you created features for the training utterances). Then, you can run the Viterbi decoding using:

```
HTK3.2bin\\HVite -H hmmsTrained/hmm8/macros -H hmmsTrained/hmm8/models -S
list/listTestFullPath_LabDMML_CLEAN1.scp -C config/config_test_MFCC_E -w
lib/wordNetwork -i result/result.mlf -p 0 -s 0.0 lib/wordDict
lib/wordList_withSP
```

2. Tool `HResults` is to be used for analysing the results of the `HVite` and providing the final recognition accuracy of the system. The `-e` option will cause that `sil` and `sp` models will be omitted from counts for the overall recognition performance.

```
HTK3.2bin\\HResults -e "???" sil -e "???" sp -I label/labelTest_LabDMML.mlf
lib/wordList_withSP result/result.mlf >> result/recognitionFinalResult.res
```

HResults provides results on sentence (SENT) level and Word (WORD) level – these indicate how well the entire sentences or words were recognised. In the results, the 'H', 'D', 'S', 'I', and 'N' denote the number of hits, deletions, substitutions, insertions and total number of words/sentences, respectively. If there is a large difference between the number of deletions ('D') and insertions ('I'), this indicates that the recognition system is not well balanced. To improve this balance, there is a parameter referred to as -p flag in the `HVite` command – this is word insertion penalty (WIP), a penalty on transiting from one model to other model. The

WIP can be used to balance the number of deletions and insertions. If needed, change the value from 0 to some other positive or negative value (e.g., in steps of 10).

Perl scripts

In the Lab directory in Canvas you can find the file `perlScripts_LabASR.zip` – this contains several Perl scripts which in a neat way incorporate all the above commands. The `ASR_LabDMML_MFCC_E.pl` script does all the above (feature extraction, training and testing) and the `ASR_LabDMML_onlyTest_MFCC_E.pl` performs testing only (assuming the training has been performed). You will need to change paths inside the Perl scripts. Then you can run the first Perl script by typing `perl ASR_LabDMML_MFCC_E.pl` in the Command Prompt window – it should perform the feature extraction, the entire training and testing. For a reference, an introduction to Perl is located in the Lab directory in Canvas.

Lab Report Tasks:

For all the tasks below, if needed, modify the –p flag (in HVite) to achieve reasonable balance of the number of deletions and insertions.

1. Explore the effect of delta and delta-delta features. Using the provided Perl script, modify the recognition system developed above such that it uses not only the static MFCC features (i.e., `MFCC_E`) but also the delta and delta-delta features (i.e., `MFCC_E_D_A`). You will need to perform modifications at several places. In the `HCopy` config modify the `TARGETKIND` to `MFCC_E_D_A` and set the `DELTAWINDOW=3` and `ACCWINDOW=2`. The `MFCC_E_D_A` features will not be 13 dimensional (as were the `MFCC_E` features) but 39 dimensional – so, you will need to make modifications at places where the feature dimension information appears. You will also need to modify the `TARGETKIND` in `config_train` and `config_test` and will need to use the `proto_s1d39_st8m1_LabDMML_MFCC_E_D_A`. Train the system using the clean training data. Perform experimental evaluations on clean test data. Report and discuss your results. [30 marks]

2. Investigate the effect of improved modelling. Modify the provided Perl scripts (and configuration files) to develop a recognition system that uses the `MFCC_E_D_A` features and employs 3 Gaussian mixture components per state. Train the system using the clean training data. Perform experimental evaluations on clean testing data and compare the results with those obtained using a single Gaussian per state as obtained from Task 1. Report and discuss your results. [30 marks]

3. Explore the effect of noise. [40 marks]
    a. Perform experimental evaluations of the recognition system developed under Task 2 separately on each provided noisy test data (`N1_SNR10`, `N1_SNR20`).
    b. Then develop the final system – this should be as system in Task 2 but trained using a combined set of all the clean and noisy training data, i.e., create a new list file containing all the filenames of all the clean and noisy training data. Perform evaluations of this system separately on clean and each noisy test data (`N1_SNR10`, `N1_SNR20`).

   Report, compare and discuss your results.

Lab Report Submission

You should report concisely on each of the above tasks. Describe clearly what changes you needed to make to perform the task and discuss the obtained results. Your report from this lab

is expected to be no longer than 8 pages and the submission is through Canvas. Standard penalty of 5% per day applies for late submissions.
END