ELEC-E5510 — Exercise 2: Hidden Markov models and isolated word recognition

In this exercise you will be constructing a functional isolated word recognizer using HTK, the Hidden Markov Model Toolkit. It is a toolkit aimed at speech recognition applications and it is widely used in the field of automatic speech recognition. If you want to have a more detailed look at the HTK, you may refer to HTK book available at the HTK website or in the course directory `/work/courses/T/S/89/5150/general/doc`.

Use the submission instructions for returning your answers. **Deadline is Wednesday 8.11.2023 at 23:59**.

When doing these exercises from home use a Maari-B computer.

To connect to a Maari-B computer, first connect to Aalto Kosh with the terminal: *ssh your_aalto_user_name@kosh.aalto.fi*. Then, connect to any Maari-B computer, e.g. *ssh akaatti*. The full list of Maari-B hostnames is available at https://www.aalto.fi/fi/palvelut/it-luokkien-linux-tietokoneiden-nimet

To begin, create a directory for the exercise and into that directory a symbolic link to the course directory for easier access to the toolkit and source data. Create also two directories for initial speech models:

```
mkdir ex2
cd ex2
ln -s /work/courses/T/S/89/5150/general material
mkdir hmm-0
mkdir hmm-1
```

Using HTK to create models for speech recognition requires some repetitive steps which are handled by the scripts provided for this exercise. Also the preprocessing of the data, which usually is the most laborious step in the model training process, has been done already. The training and evaluation sets in HTK format are in directories `material/data/rm1_train` and `material/data/rm1_eval`, respectively. In the latter directory, in addition to the feature files you also have the original audio files for listening. The corpus used in this exercise is the speaker independent isolated word part of the well-known English Resource Management corpus. There are 3597 utterances from 120 speakers in the training set and 570 words from 38 speakers in the evaluation set.

The first step of building the acoustic models is to copy a prototype HMM (`material/ex2/hmm_proto`) to separate phoneme models, each in their own file. This can be done with the following command:

```
material/ex2/proto2hmms.pl material/ex2/hmm_proto material/ex2/monophones hmm-0/
```

The directory `hmm-0` now contains the same three-state left-to-right HMM template for each phoneme model (phonemes are listed in file `material/ex2/monophones`). Initially the models consist only one Gaussian per state.

All the files generated by HTK in this exercise are text files and you are encouraged to view them (use e.g. the unix command `less`). We can also edit them with a text editor (e.g. `emacs` or `nano`) if needed.

The phoneme set includes two special silence models, sp and sil. sp is used for short pauses during the training, sil is used mainly for the beginning and the end of the utterance. To make sure sp model is short enough, we need to convert it into a one-state HMM. So open the model file `hmm-0/sp` with your favorite text editor and remove the two extra states (states 3 and 4). Note that each phoneme model implicitly contains two special states, the initial and the final state, so a model with a single emitting state actually has three states in HMM definition. After deleting the Gaussians, also update the state count and the transition matrix. The probabilities in the matrix are the transitions, you can copy the logic from the 5x5 to the new 3x3.

The next step is to initialize the template models. For this we will be utilizing an existing segmentation of the training data, collected to one master file `material/data/rm1_train/rm1_train.mlf`. You need to call `HInit` for each of the prototype HMM file, e.g. for phoneme /aa/:

```
material/bin/HInit -A -T 1 -C material/ex2/config \
   -S material/data/rm1_train/rm1_train.scp -I material/data/rm1_train/rm1_train.mlf \
   -M hmm-1 -H material/ex2/macros -i 5 -l aa hmm-0/aa
```

This command collects the samples of the given phoneme from the training data, defined by the file list `material/data/rm1_train/rm1_train.scp` and the corresponding transcriptions `material/data/rm1_train/rm1_train.mlf`. The initialized HMMs are written to directory `hmm-1`. You could do this by hand to each HMM file found in `hmm-0` directory, but to ease the task you can as well use the for-loop of your shell. With the default shell zsh or bash,

```
for p in `cat material/ex2/monophones`; do material/bin/HInit -A -T 1 \
   -C material/ex2/config -S material/data/rm1_train/rm1_train.scp \
   -I material/data/rm1_train/rm1_train.mlf -M hmm-1 -H material/ex2/macros \
   -i 5 -l $p hmm-0/$p; done
```

If you have tcsh (you can find out by typing `echo $SHELL`), use this instead:

```
foreach p (`cat material/ex2/monophones`)
  material/bin/HInit -A -T 1 -C material/ex2/config \
     -S material/data/rm1_train/rm1_train.scp -I material/data/rm1_train/rm1_train.mlf \
     -M hmm-1 -H material/ex2/macros -i 5 -l $p hmm-0/$p
end
```

The initialized phoneme models are now in directory `hmm-1`. Once initialized, they can be combined to one model file, which is what the following script does:

```
material/ex2/collect_hmms.pl material/ex2/monophones hmm-1 > hmm-1/hmmdefs
```

In HTK it is customary to keep the HMMs in a file called `hmmdefs` and global options in the file `macros`.

Before entering the iterative process of model training, the statistics of the initial models have to be collected:

```
mkdir hmm-2
material/bin/HERest -T 1 -C material/ex2/config -I material/data/rm1_train/rm1_train.mlf \
   -t 250.0 150.0 1000.0 -S material/data/rm1_train/rm1_train.scp -H hmm-1/macros -H hmm-1/hmmdefs \
   -M hmm-2 -s hmm-2/stats material/ex2/monophones > hmm-2/train.log
```

This uses the HTK program `HERest` that does most of the model training work. In this case it re-estimates the model parameters, writes them to a new directory `hmm-2` and collects the so-called occupancy statistics writing them to file `hmm-2/stats`. Occupancies indicate the number of samples segmented for each HMM state. Note that these are not integers due to Baum-Welch segmentation algorithm! `HERest` writes some additional training statistics to *.log files, you will be monitoring these later in the Question 2.

The initial models have only one Gaussian component in each Hidden Markov model state. The main part of the training consists of iteratively splitting the Gaussians to generate new components and re-estimating the parameters of HMMs. The splitting is controlled by the occupancy statistics so that states with more data get more Gaussians and vice versa. For the splitting the training script provided uses HTK program `HHEd`. Copy the main training script to your own directory in case you need to modify its operation, and run it (this takes about 5 minutes to complete):

```
cp material/ex2/hmm_train.pl .
./hmm_train.pl material/bin material/data/rm1_train/rm1_train.mlf material/data/rm1_train/rm1_train.scp \
   material/ex2/monophones material/ex2/config
```

By default the script splits Gaussians 4 times and between them re-estimates the models 3 times. Each time Gaussians are split a new directory is created so that in the end the directories hmm-3, hmm-4 etc. each contain a trained HMM with increasing number of Gaussian components. The directories also contain some information about the training, e.g. the log likelihood of the models after each training pass. Note that although EM algorithm guarantees that the likelihood of the model does not decrease between iterations, the generation of new Gaussian components does not have such a property so that in the splitting step the likelihood may decrease. In total the script calls `HERest` 12 times to re-estimate the models, each time processing the whole training data. In the end, the models have *on average* 8 Gaussians per state, depending on their frequency in the training set.

The models are now ready to be used. To test the recognizer, you'll need a recognition network, which in this case is derived from a dictionary, or a *lexicon*, as they are often referred. A lexicon defines the allowed words and their pronunciations, i.e. the phoneme sequences of the words. From directory `material/ex2`, take the dictionary w600.dict and build a simple word loop recognition network:

```
cut -f 1 -d ' ' material/ex2/w600.dict > w600.list
material/bin/HBuild -t SILENCE SILENCE -T 1 w600.list w600_loop.htk
```

Note that in this exercise we don't have any grammars or language models to further restrict the recognition.

You can now evaluate the recognition accuracy using the provided evaluation set. This set is independent from the training set, meaning it contains different speakers. The recognition is performed by HTK's simple Viterbi-based decoder:

```
material/bin/HVite -T 1 -w w600_loop.htk -H hmm-6/macros -H hmm-6/hmmdefs -C material/ex2/config \
   -t 250 -S material/data/rm1_eval/rm1_eval_word.scp -i results material/ex2/w600.dict material/ex2/monophones
```

(You may wish to change the file `results` to which the results are written to.) Note that you must give both the recognition network and the dictionary, because the network only contains allowed words and transitions between them, not the pronunciations. To compute the word error rate of the results, use the program `HResults`:

```
material/bin/HResults -h -I material/data/rm1_eval/rm1_eval_word.mlf /dev/null results
```

This compares the word level results with the reference texts in `material/data/rm1_eval/rm1_eval_word.mlf`. The error rate (percent) we are interested in this exercise is depicted under "S. Err". You also see the division of errors to three different categories that are substitution, deletion, and insertion, but you can ignore them for now as they do not make that much of a sense in isolated word recognition. To see the words that were misrecognized, you can add flag -t to `HResults`.

Question 1

a)

The recognition result is very sensitive to the vocabulary of the evaluation data and the lexicon used for recognition. To improve the recognition accuracy, a second lexicon `material/ex2/w150.dict` is provided which contains only those words that appear in the evaluation set. Build a recognition network from that and run the recognition test over the evaluation set. ***Compare the results to those of the larger lexicon. Why is the accuracy improving?***

b)

The recognition network should reflect the recognition task. The task in this exercise contains only single words, but the recognition network contains a loop which allows multiple words to be recognized from a single utterance. Remove the loop by modifying the recognition network (built from w150.dict) with a text editor. Run the recognition test once more and ***compare to previous results. Explain why the accuracy changed.***

Hint: *The recognition network has nodes and transitions. The loop is the only transition pointing backwards to an earlier node. Transitions are lines starting with J=<index>, and have parameters S for the starting node and E for the ending node. As the transitions are indexed, the easiest way is to move the last defined transition instead of the loop transition and update its index. You also need to reduce the count of the transitions in the second line of the file, L=<num>.*

Question 2

a)

An easy way of tracking the HMM training process is to look at the logarithmic likelihood values reported by HERest. You can fetch them with the command `grep "average log prob" hmm-?/train*log`. ***Plot the values as one continuous line.*** *(Check that grep gave them in the right order!)*

b)

Using the modified w150 network, recognize the evaluation set using all the models through hmm-2 to hmm-6. ***Graphically plot the word error rate with respect to the model number.***

Comment on the relationship of the two curves. Do you think further iterations would improve the recognition result?

Question 3

One possible way of improving the recognition accuracy is to build gender dependent acoustic models, that is, separate models for males and females. For this purpose both the training and the evaluation set has been split to two parts. The scp-files for the file lists are otherwise the same as before but the name body contains a suffix _m for male speakers and _f for female speakers. Your task is to build two acoustic models using these subsets and evaluate them.

To be fairly comparable with the previous gender independent model, the new models should have about the same number of parameters (= Gaussians) in total. Moreover, because the training material is not evenly distributed between the genders, but roughly the proportion of females to males is 2:5, ***the number of parameters in the gender dependent models should reflect this.*** *Use the parameter for the average number of Gaussians in `hmm_train.pl` to adjust this for the training. Note that the average number does not have to be an integer.*

Be careful with the files and directories, now that you make multiple models with different training data. Make sure you write the models to different directories and use proper file lists (scp-files) in all places, including the initialization! *The safest thing is to create separate directories for training the different gender models. Note that you can still use the same mlf-file for both gender models, as it is only the scp-file which determines which utterances are actually used for the training.*

a)

Recognize the evaluation set with the gender dependent models, using correct evaluation subsets _f for females and _m for males. Use the modified w150 network and `hmm-6` models. Analyse the possible benefits and drawbacks of gender dependent models and their use as a combined system. Assume our evalution data sets are representative of the real case. Back your analysis with further sufficient experiments; can you prove your statements with results, actual numbers?

NOTE: Given the gender specific result files, you can get the combined results with `HResult` simply by providing both input files at the same time.

b)

Assume that you know the gender of the speaker before hand. Now that you have the male and female gender dependent models, and the gender independent model, what would be the best configuration to use for recognition?

c)

Report the number of Gaussians in the final models. *For a model under `hmm-6`, you can fetch it with command*

```
grep MEAN hmm-6/hmmdefs | wc -L
```