Hidden Markov Models (HTK)

Jan Černocký, FIT VUT Brno

This exercise deals with the isolated-word recognition using HMM. We will use the HTK toolkit from University of Cambridge (UK).

1 HTK

serves to define, train and recognize speech using HMM and contains tools for parameterization (feature extraction), evaluation of results, pronunciation dictionaries, and others. HTK is written in C-language and for non-commercial use, it can be downloaded from:
`http://htk.eng.cam.ac.uk/`
In this exercise, we will use a set of pre-compiled programs that we will run from the command line of Windows operating system:

- **HCopy** - as the name says, it should copy. While copying however, it can perform a conversion of speech data, for example from signal samples to MFCC vectors. The behavior is controlled by a configuration file.
- **HList** - visualizes (as text) a file with feature vectors.
- **HCompV** - initializes parameters of emission probability distribution functions (PDFs) in HMM states to global values for given word.
- **HRest** - retraining of model. It computes the values of the "soft" function assigning vectors to states (state occupation likelihood), followed by re-estimation of model parameters.
- **HParse** - converts human-readable form of recognition network to human-unreadable HTK format.
- **HVite** - Viterbi decoder or recognizer. For an unknown word, it computes the Viterbi probability of all models and finds the maximum. The model that "emitted" the given word with the maximum probability, wins.
- **HResults** - a tool for evaluation of recognition results – based on the correct transcriptions, it computes the word accuracy.

Running any of the programs without parameters shows a brief help.

2 The task

Create a speaker-independent recognizer for isolated words ANO/NE (yes/no in Czech). For the training, use data from 60 speakers from the Czech database "Číslovky" (each speaker has uttered both "ano" and "ne"). For testing, use data from 20 speakers.

Parameterization (feature extraction) should be done using 12 MFCC (Mel-frequency cepstral) coefficients and log-energy (in HTK notation MFCC_E). Complete the feature vector by the approximations of the 1st and 2nd derivative (Δ and $\Delta\Delta$ coefficients, in HTK notation MFCC_E_D_A). Set the frame-length to 25 ms, and the frame shift to 10 ms, you will therefore obtain 100 feature vectors per second.

Models will have left-right architecture, without state-skips. From i-th state, only transitions to i-th state and to $(i+1)$-th state are allowed. The models will have 7 states in total. The first and last are special non-emitting, there will therefore be 5 emitting states. The probability distribution function (PDF) in states will be a single Gaussian with diagonal covariance matrix. One PDF will therefore be described by a vector of 39 mean values and a vector of 39 variances.

3 Solution and comments

This section contains complete solution of the task. Details on the creation of lists, MLF files, etc., are in the enclosed `README` file[1].

[1] Many of the commands in README file will, however, run only under UNIX operating system...

Practical comments

Copy the contents of ..... to arbitrary local directory on your computer. The subdirectories contain:

- `cfg` — configuration files for HTK programs.
- `dics` — dictionaries.
- `net` — word networks for the recognition.
- `lists` — lists of models.
- `proto` — prototypes of models.
- `hmm0` — models initialized using `HCompV`.
- `hmm1` — models retrained using `HRest`.
- `data` — speech in raw format: no header, F_s=8000 Hz, 16-bit lin. Files with MFCC coefficients will be generated to the same directories. Files `*a0.raw` contain ANO, files `*a1.raw` contain NE.
- `mlf` — speech data transcriptions in Master-Label files.
- `scripts` — lists of files for HTK. The name 'scripts' is a bit misleading (scripts are usually batches of commands for operating system, for example `*.bat` under DOS). This notation is unfortunately common in the documentation of HTK, so that we will use it here, too.

Open a window with a command line (here, you will run HTK programs) and a File manager (for modifications and visualizations of text files).

3.1 Parameterization (Feature Extraction)

- Study the configuration file `cfg\hcopy.conf`:

```
BYTEORDER     = VAX        the byte order will be Intel-PC
SOURCEKIND    = WAVEFORM
SOURCEFORMAT  = NOHEAD     header-less files
SOURCERATE    = 1250       sampling period is 1250×100 ns = 1/8000
ZMEANSOURCE   = FALSE      no removal of DC offset

TARGETKIND    = MFCC_E     type of output features: MFCC and log-energy
TARGETFORMAT  = HTK
TARGETRATE    = 100000     sampling period of output feature vectors (frame shift) will be 10 ms
WINDOWSIZE    = 250000.0   frame length 25 ms
NUMCHANS      = 24         number of triangular filters used for the computation of MFCC
ENORMALISE    = TRUE       energy will be normalized.
```

- Study script-files `scripts\train.scp` a `scripts\test.scp`.
- Run the feature extraction for both training and test sets:
  `HCopy -T 1 -C cfg\hcopy.conf -S scripts\train.scp`
  `HCopy -T 1 -C cfg\hcopy.conf -S scripts\test.scp`
- Visualize one of created feature files as text (using `HList`) and in Matlab using `readhtk.m` function.

4 Model training

4.1 Initialization

- Study the prototypes of models in directory `proto`. Note, that allowed and forbidden transitions are "hard-wired" in the matrix of transition probabilities at the end of each model. Mean values are set to 0, variances to 1.
- Study the Master-Label file `mlf\train.mlf` The numbers before the label stand for beginning and end of file in hundreds of ns. For one file, check, if the length recorded in MLF corresponds to the file-size: time[100ns] = # of bytes / 2 / 8000 / 100×10^{-9}.

- Note, that the configuration file for model initialization `cfg\hcompv.conf` contains only one line:
  `TARGETKIND=MFCC_E_D_A`
  This means, that MFCC and energy coefficients (already on the disk in `*.mfc` files), will be on-line completed with Δ a $\Delta\Delta$ coefficients.

- Look at the script-file `scripts\train_htk.scp`

- Run the initialization for both models:
```
HCompV -T 7 -I mlf\train.mlf -l ANO -C cfg\hcompv.conf
       -m -S scripts\train_htk.scp -M hmm0 proto\ANO
HCompV -T 7 -I mlf\train.mlf -l NE -C cfg\hcompv.conf
       -m -S scripts\train_htk.scp -M hmm0 proto\NE
```

- Study resulting models in directory `hmm0`. What has changed?

4.2 Re-training of models

- Note, that the configuration file for re-training `cfg\hrest.conf` contains again only one line:
  `TARGETKIND=MFCC_E_D_A`

- Run the re-training of both models:
```
HRest -T 7 -I mlf\train.mlf -l ANO -C cfg\hrest.conf
      -S scripts\train_htk.scp -M hmm1 hmm0\ANO
HRest -T 7 -I mlf\train.mlf -l NE -C cfg\hrest.conf
      -S scripts\train_htk.scp -M hmm1 hmm0\NE
```

- Study resulting models in directory `hmm1`. What do you observe?

5 Recognition and evaluation

5.1 What else will we need

The results of training are two trained models in directory `hmm1`. We will however need a couple of other things:

- List of models. See file `lists\models`.

- Pronunciation dictionary. This dictionary contains the transcription of words in terms of models. In case we used smaller units (phonemes), it would contain for example: ANO=A N O. In our case, one word is modeled by one model, the pronunciation dictionary is therefore trivial: `dics\dictionary`.

- Recognition network. This network determines allowed sequences of words at the output of the recognizer. For us, this is ANO or NE. Hand-made and human-readable recognition network is in file: `net\oldnetwork`. For HTK, it is necessary to convert it to a human-unreadable form using:
  `HParse  net\oldnetwork net\network`

5.2 Recognition

for unknown files, it produces a transcription and stores it to MLF-file `mlf\testout.mlf`. The recognition is run using:
```
HVite -T 1 -C cfg\hvite.conf -d hmm1 -S scripts\test_htk.scp
      -i mlf\testout.mlf -w net\network dics\dictionary lists\models
```

- Look at the recognition results in the resulting Master Label file.

5.3 Evaluation

We are interested in the quality of the recognizer. In this experiment, we have a reference MLF with the correct transcription of test files: `mlf\test.mlf`. This can be compared to `HVite` output using:
`HResults -I mlf\test.mlf lists\models mlf\testout.mlf`
The most important number in the output of `HResults` is `Acc=` (word accuracy). How many % did you reach?

6 And more...

1. Record yourself a set of 10 WAV-files (8 kHz, 16 bit) containing ANO. For `HCopy`, use configuration file `hcopy_wav.conf`, which allows for reading of WAV-files. Create your own script-files for feature extraction and for the recognition. Create your own reference MLF file (in case MLF is used only as a reference for `HResults`, it is not necessary to fill the beginning and end times). Extract the features using `HCopy` and recognize using `HVite`. Evaluate the recognition accuracy using `HResults`. How many % did you reach?

2. In Matlab, add white noise to your files so that the signal-to-noise ratio (SNR) is 0 dB. You may for example move the original files to `xx_clean.wav` and then use the following sequence of Matlab-commands:

```
SNR = 0;
[s,fs,nbit] = wavread('xx_clean.wav');
s = s' - mean(s);
E = sum(s.^2) / length(s);              % energy of the signal
Enoise = E / 10^(SNR/10);               % energy of the noise
n = randn(1,length(s)) * sqrt(Enoise); % generating the noise
wavwrite (s + n, fs, nbit,'xx.wav');   % writing signal+noise to disk
```

   Listen to the resulting files (note, that for SNR=0 dB, the energy of noise is the same as the energy of signal!). Extract MFCC features, recognize and evaluate the word accuracy. What is your result?