ICT for Health
Laboratory # 8
Parkinson's disease with HMM

Monica Visintin

Politecnico di Torino

2017/18

Dataset

- Load file `data.zip` from folder `materiale`
- Unzip the file: you'll have folder `data` that contains two folders:
    1. `healthy`, that contains 10 files with sound 'a' spoken by healthy people
    2. `parkins`, that contains 10 files with sound 'a' spoken by patients diagnosed with Parkinson's disease

Lab description

A person affected by Parkinson's disease typically has voice impairments. Goal of the lab is to distinguish Parkinson's disease using the voice of the patient and the Hidden Markov Models. This time we use Matlab, since the implementation of HMM's in Python is very poor.
We will use patients from 0 to 4 for the **training phase** and patients from 5 to 9 for the **testing phase**. We will use **two different HMM's**: one trained with 5 voices of healthy people and one trained with 5 voices of people affected by Parkinson's disease. Each test voice is given to both HMM's and the probabilities that the input sequence belongs to the machine patterns are compared: if the HMM trained with healthy voices gives the higher probability, then the test voice is classified as healthy, otherwise it is classified as Parkinson's.
We are using HMM's with `Nstates=8` states

Pre-processing [1]

The pre-processing phase is fundamental. Follow these steps:

1. To import `filename.wav` use
   `[voice, FS]=audioread('filename.wav');`
   which generates the vector `voice` and the value of the sampling frequency `FS`
2. The `.wav` files have been obtained with sampling frequency 44.1 kHz, which can be reduced by 5 times (subsampling) without changing the voice content, and simplifying the subsequent analysis. To perform subsampling, use:
   ```
   Nsub=5;
   FSa=Fs/5;% new sampling frequency
   voice=voice(Nmin:Nsub:N-Nsub);
   ```
   where `Nmin=Nsub*500` and `N=Nsub*4500`; after this, voice is a vector with 4000 samples.
3. Evaluate the mean square value of vector `voice`, and normalize it so that, after normalization, the mean square value is 1

Pre-processing [2]

4. plot the normalized signal to check it:
```
t=[0:length(voice)-1]/FSa;% time axis
plot(t,voice), grid on
```
5. Use Matlab function findpeaks to find the relative maxima of the normalized voice signal:
```
Tmin=1./200;% minimum time interval between two peaks (s)
[PKS,LOCS] = findpeaks(voice,Fsa,'MinPeakDistance',Tmin);
```
`PKS` is the vector with the maxima, whereas `LOCS` is the vector with the times at which the maxima occur
6. Between two subsequent peak times, introduce `Nstates`-1 uniformly spaced samples and generate the corresponding time vector `tt` (if you have 10 maxima, you have `9xNstates` values in `tt`)
7. Interpolate signal `voice` and evaluate its values at the times stored in `tt`:
```
voice1=interp1(t,voice,tt);
```
8. Take only the first 2000 samples of `voice1`.

Pre-processing [3]

9. Voice values are real numbers, and the HMM described in the lecture only manages a finite number of signal values. It is therefore necessary to introduce quantization. Use `Kquant=8` uniformly spaced quantization levels:
```
amax=max(voice1));
amin=min(voice1));
delta=(amax-amin)/(Kquant-1);%quantization interval
ar=round((voice1-amin)/delta)+1;%quantized signal
```

At the end of this pre-processing phase, applied to all the 20 available voice signals, the signals have been stretched so that their period is always equal to `Nstates` samples. Voices of healthy people are more regular (all the periods are equal, more or less), while voices of people affected by Parkinson's disease are irregular (different behaviors inside the period). We have forced the period equal to the number of states of the HMM so that the training phase ideally generates always the same sequence of states (from a given state, there is only one subsequent state, which is reached with probability 1), and from each state only one voice value is generated.

Training the HMM

Matlab has a function that performs the training of the HMM:

```
[ESTTR,ESTEMIT] = hmmtrain(vtrain,TRANS_HAT,EMIT_HAT,
'Tolerance',1e-3,'Maxiterations',200);
```

where

- `vtrain` is a matrix with the quantized voices used for training in the columns (`vtrain` has 2000 rows and 5 columns)
- `TRANS_HAT` is the initial guess of the HMM transition matrix (set it and justify your choice)
- `EMIT_HAT` is the initial guess of the HMM emission matrix (set it and justify your choice)
- the default algorithm used in the training phase is the Baum-Welch algorithm, but you can specify the Viterbi algorithm by including `'ALGORITHM','Viterbi'` in the list of the input parameters.
- `ESTTR` is the (output) estimated transition matrix, whereas `ESTEMIT` is the (output) estimated emission matrix

Testing the HMM

The Matlab function that estimates the probability that an input sequence actually belongs to the set of training sequences used for a given HMM is `hmmdecode`. Read the online help (`help(hmmdecode)` in the Matlab command window) or search the web, and find out how to use it.

Report

In your report include the following

- probabilities that the training sequences are correctly detected
- probabilities that the testing sequences are correctly detected
- all the plots that give information about the system (in your opinion)

using the two available algorithms Baum-Welch and Viterbi.