# Lab sessions Speech Processing - H09K9a - Assignment 4

Hugo Van hamme, KU Leuven / ESAT / PSI

Version 1.1 - 16 November 2021

## 1 Learning objectives and outcomes

The learning objectives and outcomes of these sessions are:

- To thoroughly understand the methods and algorithms involved in modelling sequential data, viz. speech.
- To thoroughly understand the methods and algorithms involved in classifying sequential data, viz. speech.
- To translate formal algorithm descriptions in code.
- To continue working on programming skills.
- To be able to select the correct algorithm to solve a given problem.

## 2 Project description

In this project, you will write code to train and evaluate a hidden Markov model (HMM) for automatic speech recognition (ASR). You will subsequently look for suitable hyper-parameters.

### 2.1 Provided for you: Data

The data are recordings of the 10 digits $0 \dots 9$ spoken in Dutch. Each recording contains just one word. The train set contains 2929 utterances by 435 speakers (not every speaker says all 10 digits). The test set contains 830 utterances by 104 speakers. The speaker sets in the train and test set are disjoint. Data are provided as a Matlab *structure*.

A feed-forward neural network is used to yield the probabilities of the leaves of a phonetic tree (CART) (instead of using Gaussian likelihoods). The network takes a window of almost 150 ms of speech data to fit the CART's leaf probabilities given this window of acoustic data: P(leaf|acoustic data). This network as well as the phonetic decision tree are trained on 200 hours of speech. You don't need to repeat that training: the relevant information is provided. For each frame in each utterance, you are given so-called *bottleneck features*. These are the activations in the $8^{th}$ out of 12 layers of the feed-forward neural network. Since further downstream in the network, P(leaf|acoustic data) is computed from these activations, their value should relate to the phone identity. Hence, you will investigate if the bottleneck features can serve well for ASR, without using the actual phonetic tree.

### 2.2 Model

Model each word in the vocabulary with an $N$-state *left-to-right* HMM model without skips. The HMM parameters are digit-specific, i.e. *word* models, i.e. no tying of states to make things simpler. Also the states that model leading and trailing silence can be word-specific.

The emission model is a single Gaussian with diagonal covariance. All state paths should start in state 1 and end in state $N$. The transition probabilities are not re-estimated, but fixed values (e.g. 0.9 for loop and 0.1 for transition), or you can opt to set all non-zero transition probabilities to 1 (so they vanish from the equations, though this is theoretically not correct, of course).

### 2.3 Method

Use the Viterbi approximation for training. Use the Viterbi approximation for computing the likelihood in recognition.

## 3 Reporting

You should hand in code and a report. You will be asked to provide additional explanation at the oral exam.

The code should contain comments that make it readable. Especially, refer to the slide(s) of the lecture notes a code section is implementing.

The report should motivate your choice of methods and algorithms. It should report the word error rate on the test data.

Additionally, find experimental evidence for answering:

1. What is your preferred method to initialize the HMMs ?
2. Does the use of delta features help to reduce the word error rate ? Explain.
3. Do the HMM parameter values converge during training ? Does the data likelihood increase at very iteration ?
4. What is a reasonable choice for the number of states of the HMM ?

**Bonus question**: model every phoneme by a context-independent 3-state left-to-right HMM with a single diagonal Gaussian as emission density. As before, use fixed transition probabilities. Some emission densities will now be shared across multiple digits as evidenced by these pronunciations:
0 = n Y l
1 = e n
2 = t w e
3 = d r i
4 = v i r
5 = v E+ f
6 = z E s
7 = z e v @ n
8 = A x t
9 = n e g @ n
Is this model more accurate ?