Matlab Exercise 3: Voice activity detection

1 Introduction

In this exercise we implement functions to extract basic speech features suitable for voice activity detection (VAD) and utilize them to train two simple VAD classifiers. These features include: Zero-crossing rate (ZCR), energy, and one-lag autocorrelation. The features are then computed from a test utterance, and combined into a matrix that acts as the input data provider for our classifiers. The addition of delta and delta-delta parameters is also implemented. The code for the classifier training is provided, but you must experiment with different features and properties and report your findings.

In this exercise, you must implement and return the following files:

1. `ex3_main.m` – Complete the main script file to run the experiments. Remember to answer the questions within the file! Report the used parameter values for noteworthy findings.
2. `ex3_energy.m` – Function for frame energy computation.
3. `ex3_one_lag_autocorrelation.m` – Function for one-lag autocorrelation parameter computation.
4. `ex3_zcr.m` – Function for zero-crossing rate computation.
5. `ex3_add_deltas_deltadeltas.m` – Function that adds the delta- and delta-delta features into the input feature matrix.

Encrypted reference files (`*_solution.p`) are provided to show you the "intended" functionality of the exercise code.

The following sections go through the necessary basic theory to implement this exercise.

2 Signal energy

The average energy e of a signal $s[n]$ of length N is defined as.

$$e = \frac{1}{N}\sqrt{\sum_{i=0}^{N-1} s[i]^2} \tag{1}$$

In many cases, it can be desirable to remove the DC component (mean value) of the frame before computing the energy.

3 One-lag autocorrelation

For more details on the autocorrelation function, check out the instructions for exercise 2. The one-lag autocorrelation parameter is obtained by setting $k = 1$, which yields the following formula:

$$r[1] = \sum_{n=1}^{N-1} s[n]s[n-1] \tag{2}$$

4 Zero-crossing rate

The zero-crossing rate is defined as the number of instances that the sign of the product of two consequtive samples within a frame is negative. I.e.,

$$zcr = \sum_{i=0}^{N-1} [sign(s[n]s[n-1]) < 0] \tag{3}$$

5 Delta and delta-delta computation

The delta- and delta-delta features of a signal are defined as their first and second time derivatives. The accurate estimation of time derivatives for discrete signals can be tricky, but for most purposes first-order central finite difference coefficients are sufficient:

$$f'(x_0) \approx -\frac{1}{2}f(x_{-1}) + \frac{1}{2}f(x_{+1}) \tag{4}$$

and

$$f''(x_0) \approx f(x_{-1}) - 2f(x_0) + f(x_{+1}) \tag{5}$$

In practice, the computation of delta- and delta-delta parameters is most easily obtained with the `filter` function by setting the FIR filter coefficients corresponding to those of Equations 4 and 5.

6 Linear classifiers

We will train two different classifiers with the toy-sized data set of a single utterance for out experiments. A linear classifier learns a set of weights w that form a linear combination mapping of an input vector x_k into a target output value y_k:

$$y_k = x_k^T w \tag{6}$$

Classification is done by *thresholding* y according to the zero-level. A simple linear classifier is trained with training data with inputs on matrix $X = [x_0, x_1, \ldots]$ corresponding to a labeled target vector $y = [y_0, y_1, \ldots]$ with the *pseudo inverse* of X:

$$w = (XX^T)^{-1}Xy^T \tag{7}$$

Perceptron is another formulation of a linear classifier. Perceptron also maps the linear combination of the input vector into the target class of a binary value:

$$f(x) = 1, \qquad \text{if} \;\; x^T w > 0 \tag{8}$$
$$= 0, \qquad \text{otherwise} \tag{9}$$

The difference between the least-squares linear classifier and the perceptorn is the learning algorithm. Within the pseudo-inverse classifier, the analytic least-squares solution can be derived by minimizing the prediction error *before* thresholding; within the perceptron model the weights are adjusted *iteratively* based on the *classification error* obtained after thresholding.