Exercises for

Introduction to Machine Learning

Paul Stoewer, Johannes Enk, Changhun Kim, Yu Gao, Haiting

Huang, Mohamad El Hage Ali Task 1: deadline: 17. May 2022



Exercise 01: Sampling

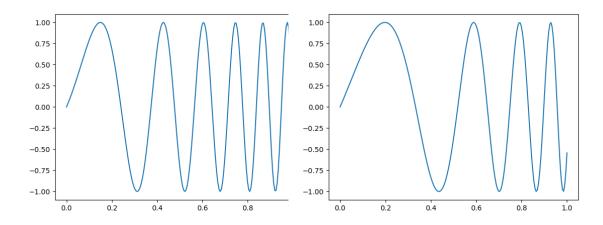
Exercise 1 Sampling Theorem

(a) Create a module *chirp.py*. In this module, create a function

createChirpSignal(sampling rate, duration, freq from, freq to, linear)

where a chirp-signal is generated and returned. The user should be able choose between a linear and an exponential chirp-signal through setting the variable linear to True or False. The sampling rate (samplingrate), the duration in seconds and the starting (freqfrom) and end frequency (freqto) has to be defined. It is not allowed to use the provided function scipy.signal.chirp, but of course you can check your result by comparing it to the signal.

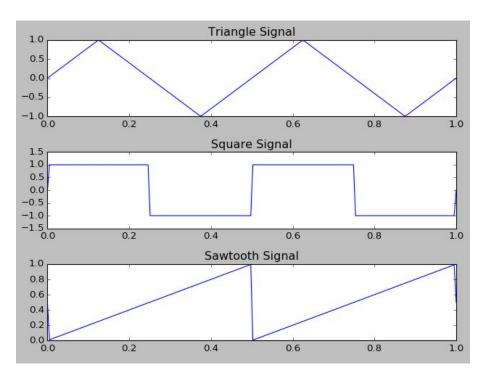
(b) In the existing module main.py (from Task 1): Create a linear and an exponential chirp signal by calling your chirp function. For example: samplingrate = 200, duration = 1, frequency from 1 to 10 Hz.



Exercise 2 Fourier Decomposition In this task, we want to reconstruct a Square, a Triangle and a Sawtooth Signal by implementing their specific Fourier Decomposition functions according to the lecture slides (Chapter 3).

(a) Create a module decomposition.py and define the three functions shown below returning the specific signal function over the time of one second. samples denotes the number of samples, frequency the signals frequency, kMax the last computed k and amplitude the amplitude for the sawtooth signal.

- \bullet createTriangleSignal(samples, frequency, kMax)
- \bullet createSquareSignal(samples, frequency, kMax)
- \bullet createSawtoothSignal(samples, frequency, kMax, amplitude)
- (b) Test your function in the existing module main.py. (Example: samples = 200, frequency = 2, kMax = 10000, [amplitude = 1])



Exercise 3 Tuning a Piano

Bob, your best friend, has had his piano tuned by Alice. You know that Alice often goes to loud concerts, so it is likely that her hearing is damaged. Thus, you believe she did not tune the piano properly. Moreover, you righteously dislike her since the day she said that dogs are better than cats... You would totally love to show Bob what a sloppy job she did!

- (a) Bob recorded some notes played with his piano, and uploaded them on StudOn. He kindly converted them to Numpy arrays. Bob said these notes are A2, A3, A4, A5, A6, A7, and another one but he forgot which one. The sampling frequency is 44100Hz. Download these files from StudOn.
- (b) Complete the load_sample function. Use the load function of Numpy to load the sound files. I recommend to plot one of them using matplotlib to see what it corresponds to. This function should return a short part of the signal shortly after its loudest peak (i.e., when the hammer hits the string). To do so:
 - Find the position of the highest absolute value of the signal (hint: use np.argmax),
 - Add offset to this position, as start of the signal,
 - Keep only the N=duration following values.
- (c) Complete the compute_frequency function. It must return the main frequency present in the input sound. To do so, the following steps are needed:

- Compute the magnitude of the Fourier transform of the sample. Use an external library to calculate the Fourier transform,
- Find the highest peak corresponding to a frequency higher than min_freq, and return the frequency corresponding to this peak.
- (d) Using the function that you completed, compute the frequencies of all notes that Bob sent you, and compare them to the expected values¹. Also, find out what the mysterious note is.
- (e) During the oral submission, after having presented your solution, give your opinion on Alice' hearing damages, and tell the tutor how wrong she is about cats.

Comments:

Please upload the code of your solution to StudOn once it passes the tests from test_signals.py and test_piano.py. There is no need to upload the plot(s) that you made.

https://en.wikipedia.org/wiki/Piano_key_frequencies