SGN-16006 Bachelor's Laboratory Course in Signal Processing
Computational Auditory Scene Recognition: Classification of Environmental Sounds
Python implementation
February 3, 2017

General information

In this project, we build a simple classification system for the recognition of auditory environments. The main concepts of this exercise are framewise processing essential in audio signal processing, feature extraction, and classification using k-nearest neighbor classification.

There are no prerequisites for this exercise. More information about audio signal processing is available at SGN-14006 (http://www.cs.tut.fi/kurssit/SGN-14006/). Pattern classification is discussed in more detail in SGN-13006 (check it out in POP).

The exercise includes two assignments. Assignment 1 gives an introduction to framewise processing of audio signals and feature extraction. Assignment 2 consists of implementing the actual system for audio environment recognition; note that the feature extraction scheme is the same as in the first part, hence try to make your code generic.

To pass this exercise, you should write the required Python codes and a report of the work. The report is the main deliverable of the exercise; it should contain at least an introduction and motivation, description of the methods used in this work (feature extraction and classification), analysis of the experiment results, and conclusions. Create one zipped file where you include your report (PDF) and Python code files (remember to comment your code!) and return the zipped file to Moodle by 2.4.2017. Instructor for this exercise is Kaisa Liimatainen (kaisa.liimatainen@tut.fi), room TF310.

Programming with Python

These instructions are for programming the algorithm with Python language. You can use either Python 2.7 or Python 3.x (*e.g.* Python 3.5). State clearly in both report and comments of main code file which version you used. Comment your code well and also write your name(s) in all code files.

It is presumed that students doing Python implementation have basic knowledge about Python and NumPy library. There are many useful array operations in NumPy library so it is recommended that you use NumPy arrays. Some useful Python commands are given in throughout these instructions. To make your program compatible with instructor's Python installation, it is recommended that you use only the following packages (example import statements):

```
import numpy as np
import soundfile as sf
from matplotlib import pyplot as plt
import os
```

```
import sklearn
import scipy
```

You should write one main file (*e.g.* main.py) and a separate file with function implementations that is imported in main file.

1 Background

Computational auditory scene recognition refers to a process, where the environment where the device is – such as an office, a street, or a train – is recognized based on the characteristics of the audio signal recorded by the device. In this exercise, we follow the idea presented in [1] to implement a Python program that predicts the recording environment from an audio clip.

The data consists of audio samples recorded in different environments (bus, office, etc.) that is divided into training data and test data so that both sets contain data from each environment. For recognition, we use k-nearest neighbor (k-NN) classification. The training phase of k-NN classification is simple: we do the feature extraction, store the features and the corresponding class labels in a suitable data structure. In the recognition phase (or test phase), the training data is then used to predict class labels for test signals not included in the training data.

Framewise feature extraction is part of both training and test phases. As also done in [1], we use sub-band energy ratios of frames (or actually their average over the audio clip to be recognized) as features in the recognition. These features describe the relative energy on different frequency bands. Both feature extraction and k-NN classification are reviewed in the following sections.

1.1 Framewise processing and feature extraction

Audio signal processing, for example audio signal analysis and classification typically relies on framewise processing. Framewise processing means that the signal is divided into short segments called frames. Typically, a smooth window such as Hanning window is used to multiply the signal values in each frame.

In framewise feature extraction, features meaningful in solving the current research problem are computed for each signal frame. The order of the feature vector is typically lower compared to the length of the time-domain signal frame.

From now on, we denote an extracted feature vector by $\mathbf{x} = [x(1), x(2), ..., x(M)]^T$, where M is the length of a feature vector of each frame. The features used in this exercise are discussed in details in Section 2.

1.2 Nearest neighbor classification

The idea of nearest neighbor (NN) classification is simple: we have a set training samples (training feature vectors) called prototypes that are assumed to be correctly classified (we know the correct class label of each training sample). To classify a new test sample (test feature vector), we compute the distance of the test sample to all training samples. The

training sample with the minimum distance to the test sample is considered a nearest neighbor and the test sample is classified to the same class as its nearest neighbor.

In other words, we have a M-dimensional test sample $\mathbf{x} = [x(1), x(2), ..., x(M)]^T$ and we want to classify it based on its nearest neighbor among correctly classified training samples D:

$$D = (\mathbf{x}_1, \theta(\mathbf{x}_1)), (\mathbf{x}_2, \theta(\mathbf{x}_2)), ..., (\mathbf{x}_N, \theta(\mathbf{x}_N)) \tag{1}$$

where label $\theta(\mathbf{x}_n)$ indicates the index of the class that training sample $\mathbf{x}_n$ belongs to and it takes values $\{1, 2, ..., C\}$, C being the number of classes. To classify a test vector $\mathbf{x}$ not included in the training data, we go through all the training vectors in order to find the one that is closest to $\mathbf{x}$, i.e. to find its nearest neighbor $\mathbf{x}'$:

$$\mathbf{x}' = \arg \min_{\forall \mathbf{x}_n \in D} d(\mathbf{x}, \mathbf{x}_n). \tag{2}$$

Test vector $\mathbf{x}$ is classified to belong to the same class as $\mathbf{x}'$. Function $d(\mathbf{a}, \mathbf{b})$ is a distance function measuring the distance between two vectors $\mathbf{a}$ and $\mathbf{b}$; the Euclidean distance function is of the form:

$$d(\mathbf{a}, \mathbf{b}) = \| \mathbf{a} - \mathbf{b} \| = \sqrt{\sum_{i=1}^{M} (a_i - b_i)^2}. \tag{3}$$

1.3 *k*-nearest neighbor classification

In the case of the k-nearest neighbor (k-NN) classification, we again calculate the distance between the test sample $\mathbf{x}$ to be classified and every training data sample $\mathbf{x}_n \in D$. Instead of considering only the first nearest neighbor, the decision is now made based on the labels of the k nearest neighbors, k being a user-set scalar value. Test sample $\mathbf{x}_n$ is classified to the class that has the highest number of samples among the k nearest neighbors.

2 Assignment 1: Introduction to feature extraction

For audio signal classification, framewise feature extraction is typically required. Here we follow the idea of [1] and carry out framewise feature extraction for a given example signal. The same approach will be used also in the context recognition approach of Section 3, so try to make your code generic.

2.1 Framewise feature extraction: Sub-band energy ratio

In framewise processing, the signal is cut into short segments (length of $\sim$ tens of milliseconds) that are windowed typically by a window function such as Hanning or Hamming window.

For the Assignment 1, we use a speech signal (signal.wav) available in Moodle.

Task: Load the given signal (length approximately 1 second) into Python (`sf.read()`). What is the sampling frequency of the signal? Divide the audio clip into frames by using Hanning window and frame length of 30 ms and 15 ms overlap of adjacent frames. What is the length

of one frame in samples for the given sampling frequency? Plot the waveform of the 50th Hanning-windowed frame.

The sub-band energy ratio describes the relative energy on certain frequency bands. For each signal frame, we will extract a feature vector $\mathbf{x} = [x(1), x(2), x(3), x(4)]^T$ containing the relative energy on the frequency bands 0 – 0.5 kHz, 0.5 – 1 kHz, 1 – 2 kHz and 2 – 4 kHz.

If the discrete Fourier transform (DFT) of a signal frame is denoted by $\mathbf{S} = [S(1), S(2), ..., S(L)]^T$, the sub-band energy ratio can be computed as:

$$x(i) = \frac{\sum_{l=b_i}^{e_i} |S(l)|^2}{\sum_{l=0}^{L/2} |S(l)|^2}, \tag{4}$$

where L is the number of frequency bins in DFT and b_i and e_i denote the first and last DFT bins belonging to the ith frequency band.

Task: Compute the DFT of the 50th Hanning-windowed frame of the test signal (`np.hanning()`, `np.fft.fft()`). Set the number of DFT bins to 1024. Plot the amplitude spectrum (`plt.plot()`, `np.abs()`) of the signal frame.

Task: Compute the band energy ratio of the frame for the frequency bands: [0–0.5, 0.5–1, 1–2, 2–4]kHz. Visualize the feature values for the frame (`plt.bar()`). What are the indices of the DFT bins belonging to the frequency band 1–2 kHz?

3 Assignment 2: Recognition of the recording environment

The next task is to build a Python program that predicts the recording environment of a previously unseen audio clip using k-NN classification. In the training phase, the program should read in one-second audio clips and their correct class labels, extract framewise energy features, average them over the audio clip, and store the values and labels to memory. In the test phase, the program should read in a test clip, carry out the feature extraction and averaging over frames and to predict the class label based on the k nearest neighbors from the training data.

3.1 Data

To build the audio environment recognizer, we use data recorded by the University of East Anglia – *Environmental Noise Data Set (Series 2)*, available at

http://www.uea.ac.uk/computing/acoustic-environment-classification

For easier access, a subset of it is stored in Moodle. The data consists of 8 kHz MP3-recordings from 12 different audio environments including a building site, bus, office, shopping mall, etc. For each environment, there is one wav file. Check out `os.listdir()` for detecting contents of a directory.

Task: Each recording is a couple of minutes long. As a pre-processing step, you should segment audiofiles into one-second clips without any overlap between the audio clips. After segmenting the long files into short clips, divide the clips among training and test data sets so

that approximately 80% of the data from each acoustic environment is included in the training data and the rest in the test data.

3.2 Feature extraction

The feature extraction is carried out similarly to Section 2.

Task: Process each training data clip framewise and extract the sub-band energy ratio values for each frame similarly to the Section 2. Use Hanning windowing with the window length of 30 ms and overlap of 15 ms. Extract the sub-band energy ratios for the frequency bands 0 – 0.5 kHz, 0.5 – 1 kHz, 1 – 2 kHz, 2 – 4 kHz.

As we wish to model also the slowly changing properties of the audio clips, we do as in [1] and compute the average of each subband energy value over the frames of the audio clip. This results in one average feature vector of size 4x1 for each audio clip.

Task: Compute the average feature vectors and store them together with the label of the audio clip (use class indexes 1 – 12) for further use, they will be used as features in k-NN classification. Test data is processed similarly to training data.

3.3 Classification

For classification, we use k-NN classification described in Section 1. In the classification task, the recording environment (class index) of a test clip is predicted based on the clip's nearest neighbor(s) among the training data. **Note:** Do not use Python's built-in classifiers, make your own instead. Here are some commands you might find useful: `np.argsort()`, `scipy.stats.mode()`, `np.mean()`.

Task: Build a NN classifier that predicts the environment class index of a test audio clip based on its first nearest neighbor among the training samples. What is the accuracy (percentage of correctly classified samples) of the classification for each class? Present the results of the evaluation in the report with a confusion matrix (`sklearn.metrics.confusion_matrix()`) and/or a regular table. Are some of the environments more difficult to recognize than the others?

Task: Repeat the classification by using k-NN classification with at least $k = 5$ nearest neighbors. You can resolve ties by selecting the class with smallest distance, for example. Does the classification accuracy change compared to the previous case with only one neighbor? Compare the overall classification accuracy and also classification of individual scenes. Include the results in the report in the same way as in previous task.

4 Conclusions

You have built a system that is suitable for automatic recognition of the acoustic environment. It is time to summarize the findings and encountered problems.

Task: The goal was to develop a program that is able to recognize the recording environment of an audio signal. Was your program able to do this? Why or why not?

Task: How about the implementation approaches, are they suitable for realistic applications? Could there be some bottlenecks related to them?

Task: To develop the lab work in the future, please give your feedback about the work. For example, do you think the skills you learned will be useful in your future studies or work? How about the instructions – did they contain parts that were difficult to follow? How long it took for your group to implement the code and write the report?

Sending the report

The report should at least include a description of the work and analysis of the results. There are tasks and questions in the exercise instructions that help you while writing, but try to avoid just listing the answers; instead, write a full scientific report. Include all the plots you were asked to make in the report.

- Your report should be in PDF format
- Create one zipped file where you include your report and Python code files
- Return the zipped file to Moodle by Sunday 2.4.2017.

References

[1] Peltonen, V., Tuomi, J., Klapuri, A., Huopaniemi, J., and Sorsa, T. *Computational auditory scene recognition.* In proceedings of ICASSP, 2002.