67355 Introduction to Speech Processing

Exercise 3

The following exercise is structured from two parts: Theoretical and Practical.
The exercise should be done **in pairs** and is to be submitted via moodle by June 4th 2023.
See submission guidelines for further instructions

1 Assignment overview

In this exercise you will implement your first word recognizer using the Dynamic Time Warping (DTW) algorithm. Your algorithm will recognize the digits 1 - 5.
For that, You will implement the DTW algorithm as described below, where the distance metric (d) should be the Euclidian distance.

$$\begin{aligned} DTW[0,0] &= d(0,0) \\ DTW[i,j] &= d(i,j) + \min \begin{pmatrix} DTW[i-1,j-1] \\ DTW[i,j-1] \\ DTW[i-1,j] \end{pmatrix} \end{aligned} \tag{1}$$

1.1 Implementation Notes

In practice, DTW can be applied to varying length sequences. However, to better understand the advantages of the DTW algorithm, you will compare its performance to a standard Euclidian distance (recall each file is exactly 1 second long).
To sum up, you need to run a 1 nearest neighbor classifier using both Euclidian distance and DTW distance. You need to compute both distance metrics (Euclidian and DTW) for each file in the test set with all training examples. Then, classify each test file as the label of the file with the minimal distance.
You should generate a file named: 'output.txt', with the predictions for each test file using both euclidian distance and DTW distance. The output file should be constructed as follows:

<filename> - <prediction using euclidian distance> - <prediction using DTW distance>

For example,

```
f6581345_nohash_1.wav - 1 - 3
fb7c9b3b_nohash_0.wav - 2 - 1
fb9d6d23_nohash_0.wav - 4 - 2
fda46b78_nohash_2.wav - 1 - 1
...
```

1.2 Features

In class, we learned how to transform a time domain waveform to the frequency domain using the Fourier Transform. However, in many applications we would like to use more compressed representation. One representation like that is the Mel Frequency Cepstrum Coefficients (MFCCs). In order to extract these features and load the wave files, you will use a python package called 'librosa', using the following lines of code:

```
import librosa
y, sr = librosa.load(f_path, sr=None)
mfcc = librosa.feature.mfcc(y=y, sr=sr)
```

1.3 Files overview

In Ex3.zip you will find the following:

- `train_files`: Directory containing 5 inner directories - each containing 5 examples of a single digit recording with a 1 second duration.
- `test_files`: Directory containing 250 unlabeled examples of a single digit recording, each with 1 second duration. This is the test data you are requested to predict.
- `digits_classifier.py`: python interface specifying an API call of all functions you are required to implement.

2 Grading

- We will be applying automated tests, and manually observe your code.
- Evaluation of your performance will be done with respect to the held out test set as well as a small out of distribution test set.
- Failing to follow the given API will fail the automatic tests and will grant you a grade of 0 for that part. Please make sure this doesn't happen!

3 Submission Guidelines

- All used code pieces should be submitted, alonside a README.txt file with a single line containing your IDs separated by commas.
- A .txt file named `output.txt` containing your prediction outputs for the test set, as specified under Implementation notes 1.1 subsection.
- We will use the defined API in `digits_classifier.py` to test your code, make sure that your implementation follows the API specification and that you don't use any absolute paths in your implementation. If you do use relative paths, make sure to include the relevant files in your submission.
- Please submit a single zip/tar file containing all relevant files.