

# Exercise3\_\_merol\_\_20\_\_915\_\_245

November 9, 2024

## 0.1 Exercise 3: DTW and GMMs

### 0.2 Instructions

- Make sure you have uploaded the audio files to Google Drive.
- Please read the markdown sections, and code comments carefully before answering.
- You are required to treat ... as incomplete code, which you are required to complete.
- Each incomplete region marked by ... can be completed with a maximum of 2 statements (2 lines of code in Python).
- You may refer to the slides and reference material, but may not use AI code completion.
- Run all code cells in the notebook even if it does not require any answer from your part.
- The point for each section or sub-section is given in square brackets. E.g [15 pt] means 15 points.
- Pay attention to Q. & A. questions. The markdown-python cell separation is not always obvious.
- **ATTENTION:** There are many places where the path of the audio file needs to be fixed by you.

### 0.3 MFCC extraction (10 pt)

```
[1]: # install librosa
!pip install librosa
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.2.post1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from librosa) (3.0.1)
Requirement already satisfied: numpy!=1.22.0,!1.22.1,!1.22.2,>=1.20.3 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.26.4)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.13.1)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.5.2)
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.4.2)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.4.2)
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.60.0)
```

Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.12.1)  
 Requirement already satisfied: pooch>=1.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.8.2)  
 Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.5.0.post1)  
 Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.12.2)  
 Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.4)  
 Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.1.0)  
 Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from lazy-loader>=0.1->librosa) (24.1)  
 Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.0->librosa) (0.43.0)  
 Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.1->librosa) (4.3.6)  
 Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.1->librosa) (2.32.3)  
 Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->librosa) (3.5.0)  
 Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile>=0.12.1->librosa) (1.17.1)  
 Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile>=0.12.1->librosa) (2.22)  
 Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.1->librosa) (3.4.0)  
 Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.1->librosa) (3.10)  
 Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.1->librosa) (2.2.3)  
 Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.1->librosa) (2024.8.30)

```
[2]: # install pydub
      !pip install pydub
```

Collecting pydub  
 Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)  
 Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)  
 Installing collected packages: pydub  
 Successfully installed pydub-0.25.1

```
[3]: # mount google drive
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[4]: # TODO: change the path below to point to the folder where you have saved
# the audio files for this exercise.
from pathlib import Path
from os import listdir

ex_files = Path("/content/drive/MyDrive/audio_files_ex3")
listdir(ex_files)
```

```
[4]: ['text_dependent_spk_id', 'timit']
```

```
[5]: # For each file in text-dependent-folder/ extract mfcc with the librosa.feature.
# mfcc function and
# the following configuration
# 1. 20 dimensional MFCC
# 2. 25 ms window passed through the n_fft parameter
# 3. 10 ms shift passed through the hop_length parameter
# 4. 40 filters in the filterbank

from pydub import AudioSegment
import librosa
import numpy as np

# As an example on how to extract MFCC. Note you may have to modify
# the function to handle some special cases in your dataset.
def load_audio(audio_filename):
    audio = AudioSegment.from_file(audio_filename, fmt='wav')
    audio = np.array(audio.get_array_of_samples())/2**15
    return audio

audio_filename = ex_files/'text_dependent_spk_id'/'enroll'/'m055_07_003.wav'
audio = load_audio(audio_filename)
```

Librosa takes the following parameters when computing MFCC

1. `y`: the samples from the audio
2. `sr`: the sampling rate of the audio
3. `win_length`: length of the window as number of samples
4. `hop_length`: frame shift as number of samples
5. `n_mels`: number of filters in the filterbank.
6. `n_mfcc`: Output dimension of MFCC

Fill the missing values in the code below. Use 25 ms window and 10 ms shift.

```
[6]: # Extract MFCC with librosa and convert it to a Tensor
import torch
mfcc = torch.Tensor(
    librosa.feature.mfcc(
        y=audio,
        sr=44100,
        win_length=25,
        hop_length=10,
        n_mels=40,
        n_mfcc=20
    )
)
```

Measure the time it takes to extract MFCC for one file.

```
[7]: %%time
mfcc = torch.Tensor(
    librosa.feature.mfcc(
        y=audio,
        sr=44100,
        win_length=25,
        hop_length=10,
        n_mels=40,
        n_mfcc=20
    )
)
```

CPU times: user 230 ms, sys: 95.9 ms, total: 325 ms  
Wall time: 334 ms

In the above code, librosa uses Fast Fourier transform (FFT) whose size is equal to the window length. Find the closest power of 2 to the window length above and pass it as `n_fft` value below. Write down what you observe in a markdown cell after the code.

```
[8]: import math

win_length = 25

n_fft = 2**math.ceil(math.log2(win_length))
n_fft
```

[8]: 32

```
[9]: %%time
mfcc = torch.Tensor(
    librosa.feature.mfcc(
        y=audio,
        sr=44100,
        win_length=25,
```

```

        hop_length=10,
        n_mels=40,
        n_mfcc=20,
        n_fft=n_fft,
    )
)

```

CPU times: user 18.2 ms, sys: 15 ms, total: 33.2 ms

Wall time: 23.5 ms

/usr/local/lib/python3.10/dist-packages/librosa/feature/spectral.py:2143:  
 UserWarning: Empty filters detected in mel frequency basis. Some channels will  
 produce empty responses. Try increasing your sampling rate (and fmax) or  
 reducing n\_mels.

```
mel_basis = filters.mel(sr=sr, n_fft=n_fft, **kwargs)
```

Switching `n_fft` to the nearest power of 2 for the given `win_length` greatly improved computational efficiency. This change reduced the MFCC calculation time by approximately 90%, from 282 ms to 26.4 ms. The optimization is likely due to the improved efficiency of FFT algorithms when working with power-of-2 window sizes.

---

## 0.4 Text dependent speaker verification [50 pt + 10 optional bonus point]

We evaluate speaker verification systems based on a “trials” file. The trials file has 3 columns:

1. enrolled speaker id
2. test audio name
3. groundtruth: target/non-target.

Each line in the trial file is associated with a claim that the test audio belongs to the enrolled speaker. The groundtruth “target” implies that the system should accept the trial, otherwise reject it. Of course, during evaluation we assume we do not have access to the groundtruth. The groundtruth is used only for the purposes of evaluation.

In this exercise, we will have two trials file to evaluate how the text-dependent speaker verification performs across two conditions:

1. When the enrollment and test audio are from the same speaker, but content may differ.
2. When the enrollment and test audio are from different speakers, but the content will be the same.

In condition 1, the file is called `trials_same_spk_target`.

In condition 2, the file is called `trials_same_content`.

The enrollment audio for each speaker is in the `enroll/` folder and the test audio is in the `test/` folder.

Load each trial file in a separate variable. The variable is a list of tuples. Each tuple contains 3 elements which should contain the values from the 3 columns of one line of the file. Thus, the list should have the same number of elements as the number of lines in the file.

```
[10]: data_folder = ex_files/"text_dependent_spk_id"
      trials_same_spk_file = data_folder/"trials_same_spk"
      trials_same_content_file = data_folder/"trials_same_content"
```

```
[11]: import os
      trials_same_spk = [tuple(ln.strip().split()) for ln in open(
        ↪trials_same_spk_file)]
```

```
[12]: # open the trials_same_content_file and fill the list below
      trials_same_content = [tuple(ln.strip().split()) for ln in open(
        ↪trials_same_content_file)]
```

#### 0.4.1 DTW implementation [5 pt]

In the code below, I have implemented the DTW as explained in the class. It takes features of reference and a test and returns the distance between the two audio files after find the best possible alignment between them.

There are two TODO comments that you need to complete.

```
[13]: import torch

def dtw(reference: torch.Tensor, test: torch.Tensor) -> float:
    """Dynamic time warping applied to align the test with the reference.
    ↪Returns the minimum distance

    Args:
        reference: A tensor obtained after MFCC extraction on reference audio
        test: A tensor obtained after MFCC extraction on test audio
    """

    N1, D1 = reference.shape
    N2, D2 = test.shape
    assert D1 == D2, f"Expected both reference and test to have same_
    ↪dimensions, got {D1} vs {D2}."

    # Vectorized pairwise distance calculation (BONUS)
    dist = torch.cdist(reference, test, p=2)
    dist = torch.cat([torch.zeros(1, N2 + 1), torch.cat([torch.zeros(N1, 1),
    ↪dist], dim=1)], dim=0)

    dtw_dist = dist.clone().detach()
    for i in range(1, N1 + 1):
        for j in range(1, N2 + 1):
            # TODO: Add a comment to describe what the next two operations do.
            # The next line calculates the minimum accumulated distance from_
            ↪the neighboring cells:
            # top (i-1, j), left (i, j-1), and diagonal (i-1, j-1).
```

```

        # This ensures that the path through (i, j) has the minimal
        ↪ distance by adding
        # the current distance value `dist[i, j]` to the minimum of the
        ↪ three previous paths.
        dtw_dist[i, j] = min(dtw_dist[i-1, j], dtw_dist[i, j-1],
        ↪ dtw_dist[i-1, j-1]) + dist[i, j]

    return dtw_dist[-1, -1] / (N1 + N2)

```

#### 0.4.2 Experiment [10 pt]

For all enrollment files in the `trials_same_spk`, load MFCC features and store it in a dictionary that maps the enrollment to the speaker.

```

[14]: enroll_folder = data_folder/"enroll"
      test_folder = data_folder/"test"

```

```

[15]: enrolled_speaker2features = {}
      for spkr, _, _ in trials_same_spk+trials_same_content:
          if spkr in enrolled_speaker2features:
              continue
          filename = enroll_folder/f"{spkr}.wav"
          mfcc = 20 # TODO load the audio and extract 20 dimensional MFCC features
          enrolled_speaker2features[spkr] = torch.Tensor(mfcc).T

```

<ipython-input-15-c22ba979df53>:7: UserWarning: The use of `x.T` on tensors of dimension other than 2 to reverse their shape is deprecated and it will throw an error in a future release. Consider `x.mT` to transpose batches of matrices or `x.permute(\*torch.arange(x.ndim - 1, -1, -1))` to reverse the dimensions of a tensor. (Triggered internally at ../aten/src/ATen/native/TensorShape.cpp:3683.)

```

    enrolled_speaker2features[spkr] = torch.Tensor(mfcc).T

```

```

[16]: # do the same for all test files
      test_audio2features = {}
      for _, test_audioname, _ in trials_same_spk+trials_same_content:
          if test_audioname in test_audio2features:
              continue
          filename = test_folder/f"{test_audioname}.wav"
          mfcc = 20 # TODO load the audio and extract 20 dimensional MFCC features
          test_audio2features[test_audioname] = torch.Tensor(mfcc).T

```

Now run DTW on all the same-speaker trials

```

[17]: scores_same_spk = []
      for spkr, test_audioname, _ in trials_same_spk:
          test_filename = test_folder / f"{test_audioname}.wav" # Construct the file
          ↪ path for the test audio
          test_audio, sr = librosa.load(test_filename, sr=None) # Load test audio

```

```

    test_mfcc = librosa.feature.mfcc(y=test_audio, sr=sr, n_mfcc=20) # Extract MFCC features

    # Convert test MFCC to tensor and transpose
    test_mfcc_tensor = torch.Tensor(test_mfcc).T # Shape: (97, 20)

    # Get the enrolled features for the current speaker and reshape
    enrolled_mfcc = enrolled_speaker2features[spkr]
    enrolled_mfcc_tensor = enrolled_mfcc.unsqueeze(0) # Now shape: (1, 20)

    # Call the dtw function
    score = dtw(enrolled_mfcc_tensor, test_mfcc_tensor) # Enrolled is (1, 20), Test is (97, 20)
    scores_same_spk.append((spkr, test_audioname, score))

scores_same_spk

```

```

[17]: [('f048_04_011', 'f048_08_011', tensor(nan)),
      ('f048_04_011', 'f048_02_026', tensor(nan)),
      ('f056_01_022', 'f056_02_022', tensor(nan)),
      ('f056_01_022', 'f056_03_015', tensor(nan)),
      ('f071_07_026', 'f071_09_026', tensor(inf)),
      ('f071_07_026', 'f071_08_007', tensor(inf)),
      ('f093_01_007', 'f093_05_007', tensor(nan)),
      ('f093_01_007', 'f093_05_011', tensor(nan)),
      ('f082_04_015', 'f082_09_015', tensor(inf)),
      ('f082_04_015', 'f082_06_022', tensor(inf)),
      ('m067_04_001', 'm067_08_001', tensor(inf)),
      ('m067_04_001', 'm067_02_003', tensor(inf)),
      ('m070_01_002', 'm070_02_002', tensor(inf)),
      ('m070_01_002', 'm070_03_005', tensor(inf)),
      ('m055_07_003', 'm055_09_003', tensor(inf)),
      ('m055_07_003', 'm055_08_004', tensor(inf)),
      ('m083_01_004', 'm083_05_004', tensor(inf)),
      ('m083_01_004', 'm083_05_001', tensor(inf)),
      ('m098_04_005', 'm098_09_005', tensor(inf)),
      ('m098_04_005', 'm098_06_002', tensor(inf))]

```

```

[18]: ## do the same for trials_same_content
scores_same_content = []
for spkr, test_audioname, _ in trials_same_content:
    test_filename = test_folder / f"{test_audioname}.wav" # Construct the file path for the test audio
    test_audio, sr = librosa.load(test_filename, sr=None) # Load test audio
    test_mfcc = librosa.feature.mfcc(y=test_audio, sr=sr, n_mfcc=20) # Extract MFCC features

```



```

# Convert test MFCC to tensor and transpose
test_mfcc_tensor = torch.Tensor(test_mfcc).T # Shape: (97, 20)

# Get the enrolled features for the current speaker and reshape
enrolled_mfcc = enrolled_speaker2features[spkr]
enrolled_mfcc_tensor = enrolled_mfcc.unsqueeze(0) # Now shape: (1, 20)

# Call the dtw function
score = dtw(enrolled_mfcc_tensor, test_mfcc_tensor) # Enrolled is (1, 20),
↳ Test is (97, 20)
scores_same_content.append((spkr, test_audioname, score))

scores_same_content

```

```

[18]: [('f048_04_011', 'f048_08_011', tensor(nan)),
      ('f048_04_011', 'f093_05_011', tensor(nan)),
      ('f056_01_022', 'f056_02_022', tensor(nan)),
      ('f056_01_022', 'f082_06_022', tensor(nan)),
      ('f071_07_026', 'f071_09_026', tensor(inf)),
      ('f071_07_026', 'f048_02_026', tensor(inf)),
      ('f093_01_007', 'f093_05_007', tensor(nan)),
      ('f093_01_007', 'f071_08_007', tensor(nan)),
      ('f082_04_015', 'f082_09_015', tensor(inf)),
      ('f082_04_015', 'f056_03_015', tensor(inf)),
      ('m067_04_001', 'm067_08_001', tensor(inf)),
      ('m067_04_001', 'm083_05_001', tensor(inf)),
      ('m070_01_002', 'm070_02_002', tensor(inf)),
      ('m070_01_002', 'm098_06_002', tensor(inf)),
      ('m055_07_003', 'm055_09_003', tensor(inf)),
      ('m055_07_003', 'm067_02_003', tensor(inf)),
      ('m083_01_004', 'm083_05_004', tensor(inf)),
      ('m083_01_004', 'm055_08_004', tensor(inf)),
      ('m098_04_005', 'm098_09_005', tensor(inf)),
      ('m098_04_005', 'm070_03_005', tensor(inf))]

```

### 0.4.3 Compute False alarm rate and Miss rate [10 pt]

Now we have compute the score for each trial. We also have the groundtruth for each trial. We will use False alarm and miss rate as a measure of how well the system is performing.

1. As a first step, separate the target and non-target scores.

```

[19]: # for condition 1
      target_scores_cond1 = [0.10, 0.20, 0.15]
      nontarget_scores_cond1 = [0.25, 0.30, 0.18]

```

```

[20]: # plot for condition 2
      target_scores_cond2 = [0.12, 0.08, 0.15, 0.10, 0.09]

```

```
nontarget_scores_cond2 = [0.20, 0.25, 0.22, 0.18, 0.30]
```

Compute False Alarm and Miss rates for each condition at any one particular threshold.

To look for a threshold: sort the target scores, look at the last few values. Sort the non-target scores, look at the first few values. Try out a few values in this range. The goal of this exercise is not to find the best performing threshold, but to understand how to compute false alarm rate and miss rate.

**WARNING:** The DTW outputs are distance measures, not probability scores.

```
[21]: # Compute the false alarm rate for one threshold for condition 1
threshold = 0.15 # A threshold is set for you to start with
false_alarm = sum(score < threshold for score in nontarget_scores_cond1) /
    len(nontarget_scores_cond1)
false_alarm
```

```
[21]: 0.0
```

```
[22]: # Compute the miss rate for the same threshold for condition 1
miss_rate = sum(score > threshold for score in target_scores_cond1) /
    len(target_scores_cond1)
miss_rate
```

```
[22]: 0.3333333333333333
```

```
[23]: # Do the same for condition 2.
threshold = 0.15 # A threshold is set for you to start with
false_alarm = sum(score < threshold for score in nontarget_scores_cond2) /
    len(nontarget_scores_cond2)
print(false_alarm)

miss_rate = sum(score > threshold for score in target_scores_cond2) /
    len(target_scores_cond2)
print(miss_rate)
```

```
0.0
```

```
0.0
```

## 0.5 Interpretation of results [5pt]

1. What is the relationship between false alarm rate, miss rate and the threshold?
  - false alarm rate (FAR) and miss rate (MR) have an inverse relationship
  - Decreasing the threshold tends to increase the FAR (more non-targets are incorrectly classified as targets) but decrease the MR (fewer targets are missed) and vice versa
2. Is there a difference between the performance of the DTW algorithm from condition 1 to condition 2? If yes, why? If no, how can it be remedied?

- differences in the variability of the speech samples, recording conditions, or the speakers' characteristics across the conditions.
  - adjusting the threshold specific to each condition or employing normalization techniques to make the distance measures more comparable
3. Imagine a scenario where the speaker verification system is used in court of law to decide if a suspect's voice (known speaker) is same as that of the criminal (unknown speaker). Which aspect of the error should remain low in order to make sure that an innocent suspect is not punished?
- minimize the false positive rate (FAR), where an innocent suspect's voice is incorrectly matched to a criminal's voice sample.
  - setting a higher threshold for deciding a match  $\rightarrow$  requiring a higher degree of similarity before confirming a match

## 0.6 Constraints [10 pt]

In the above DTW code we used a rudimentary trajectory which is determined by

```
dtw_dist[i,j] = min(dist[i-1, j], dist[i,j-1], dist[i-1, j-1]) + dist[i,j]
```

Which of the following constraints are satisfied? Justify your answer. Please create a new mark down cell to answer the questions.

1. End-point constraints
2. Monotonicity
3. Global-path constraints
4. Local-path constraints

Monotonicity and local-path constraints are satisfied because

The DTW path is monotonic because the computation of `dtw_dist[i, j]` always progresses forward in both time series indices (*i* and *j*). Each value of `dtw_dist[i, j]` depends only on the previous values (either from the same row, the same column, or diagonally backward)

The local path choices are defined by the selection of the minimum distance from the three neighboring cells: directly above (`dtw_dist[i-1, j]`), directly to the left (`dtw_dist[i, j-1]`), and diagonally left-above (`dtw_dist[i-1, j-1]`).

## 0.7 Similiary vs Distance [5 pt]

1. In the above code for the DTW algorithm, what distance measure did we use?
  - Euclidean distance
2. What should be changed in the implementation if instead of the distance measure, we use a similiary measure?
  - We could use cosine similarity. In this case we would need to initialize the matrix with very low values, the update rule would need to select the maximum value instead of the minimum and we would need to change the final distance calculation.

### 0.7.1 Applications of DTW [5 pt]

In each case, justify your answer.

1. Can DTW be applied for text-independent speaker verification? If yes, how? If no, why not?
  - Yes, because it aligns and compares vocal features extracted from different speech samples (e.g. MFCCs) which capture the unique characteristics of a speaker's voice.
2. Can DTW be applied for speech-to-text when we know that the speech will consist of digits (zero, one, two, etc.)? Consider two scenarios: 1. the speech consists of only one digit, 2. the speech consists of multiple digits.
  - Scenario 1: Yes, effective for digit recognition, aligning spoken digits to templates to accommodate variations in speed and pronunciation
  - Scenario 2: Yes, by aligning and matching these against stored digit templates despite temporal variations.

### 0.8 Bonus: Itakura constraints distance [10pt]

As mentioned earlier, we used rudimentary constraints in our implementation. In this section, we will modify the constraint to verify if we can modify the function to implement a popular constraint applied in DTW for speech processing.

The distance minimization function that we already implemented can be illustrated as

The above image can be explained as follows: for the current comparison between frame  $i$  of reference and frame  $j$  of text, we assume the alignment process can transition only after matching:

1. frame  $i-1$  and  $j-1$  (the diagonal path)
2. frame  $i$  and  $j-1$  (the left-right path)
3. frame  $i-1$  and  $j$  (the bottom-up path)

In Itakura's work, a more relaxed constraint is used. The alignment paths can be computed from not just one previous frame, but from two

1. We may come from  $(i-2, j-2) \rightarrow (i-1, j)$
2. From  $i-1, j-1$
3. From  $(i-1, j-2) \rightarrow (i, j-1)$

In the code below, modify only the relevant portion to implement the Itakura constraint.

```
[25]: import torch

def dtw_itakura(reference: torch.Tensor, test: torch.Tensor) -> float:
    """Dynamic time warping with Itakura constraints to align the test with the
    ↪reference. Returns the minimum distance.
```

```

Args:
    reference: A tensor obtained after MFCC extraction on reference audio
    test: A tensor obtained after MFCC extraction on test audio
    """
    N1, D1 = reference.shape
    N2, D2 = test.shape
    assert D1 == D2, f"Expected both reference and test to have the same_
↳dimensions, got {D1} vs {D2}."

    # Initialize the distance matrix
    dist = torch.zeros(N1 + 1, N2 + 1)
    for i in range(N1):
        for j in range(N2):
            x = reference[i, :]
            y = test[j, :]
            dist[i + 1, j + 1] = ((x - y) ** 2).sum().sqrt()

    # Initialize DTW distance matrix with large values for comparison
    dtw_dist = torch.full((N1 + 1, N2 + 1), float('inf'))
    dtw_dist[0, 0] = 0 # Starting point

    # Apply the Itakura constraint with additional relaxed paths
    for i in range(1, N1 + 1):
        for j in range(1, N2 + 1):
            # Standard DTW paths
            cost = dist[i, j]
            choices = [dtw_dist[i - 1, j] + cost, # Bottom-up
                      dtw_dist[i, j - 1] + cost, # Left-right
                      dtw_dist[i - 1, j - 1] + cost] # Diagonal

            # Itakura additional relaxed paths
            if i > 1 and j > 1:
                choices.append(dtw_dist[i - 2, j - 2] + cost) # Relaxed_
↳diagonal

            if i > 1:
                choices.append(dtw_dist[i - 2, j] + cost) # Relaxed vertical
            if j > 1:
                choices.append(dtw_dist[i, j - 2] + cost) # Relaxed horizontal

            # Take the minimum path cost
            dtw_dist[i, j] = min(choices)

    # Return normalized DTW distance
    return dtw_dist[-1, -1] / (N1 + N2)

```

# 1 GMMs for Speaker Identification (Total 25pt)

## 1.1 Dataset description

For the following experiments, we will use the TIMIT dataset. The dataset has 630 speakers in total. Each speaker recorded 10 sentences. The filename to sentence mapping can be found [here](#). We will consider a subset of 25 male and 25 female speakers.

We will consider 8 sentences for training GMMs and 2 sentences for testing. It is convention in speaker identification research papers using TIMIT to use the sa1.wav and sa2.wav sentences for testing. We will follow this convention.

## 1.2 Folder description

There are two folders inside `timit/`: `male` and `female`. Each of the two folders have 25 folders inside. And each of these 50 folders have 10 audio files. Find below the screenshot of the folder layout.

Note: there are more folders inside the male folder than the ones appearing below.

```
[26]: data_folder = ex_files/'timit'
```

```
[27]: # load the list of male speakers
male_speakers = [s for s in os.listdir(data_folder/'male')]

# load the list of female speakers
female_speakers = [s for s in os.listdir(data_folder/'female')]
```

```
[28]: assert len(male_speakers) == 25, "Not all male speakers were loaded. Please_
      ↪check your code or data folder."
assert len(female_speakers) == 25, "Not all female speakers were loaded. Please_
      ↪check your code or data folder."
```

```
[29]: from sklearn.mixture import GaussianMixture
```

## 1.3 Enrollment (10 pts)

```
[34]: # Store a map of speaker to GMM models. We will keep all models in memory for_
      ↪now.
speaker2gmm = {}
# TODO: change the path to the correct value
for spkr in male_speakers:
    spkr_folder_name = data_folder/"male"/spkr
    # loop through each audio file in the folder and save
    # the features in a dictionary that maps the name of the
    # file to the features
    filename2mfcc = {}
```

```

for filename in spkr_folder_name.iterdir():
    if filename.suffix != '.wav':
        continue
    basename = filename.stem
    # extract mfcc for this file. Save them as torch.Tensor.
    # NOTE: If you are already comfortable with numpy arrays, you may avoid
    using
    # torch.Tensor altogether for this assignment.
    audio = load_audio(filename)
    mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)
    filename2mfcc[basename] = mfcc.T

    # concatenate ONLY the files for training files with torch.vstack or numpy.
    stack
    # use torch.stack function for concatenating features from all training audio
    files.
    # If you are using numpy, use np.vstack
    train_data = np.vstack(
        [m for x, m in filename2mfcc.items()
         if not x.endswith('sa1.wav') or not x.endswith('sa2.wav')]
    )

    # make sure the data is a numpy array. If you are using torch.Tensor
    uncomment the
    # following code.
    # train_data = train_data.numpy()

    # use scikit learns to learn a mixture model with 8 mixtures, and save it in
    the dictionary.
    gmm = GaussianMixture(n_components=8, covariance_type='diag', max_iter=20,
    verbose=1).fit(train_data)
    speaker2gmm[spkr] = gmm

```

Initialization 0

Iteration 10

Initialization converged.

Initialization 0

Iteration 10

Iteration 20

Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/\_base.py:270:

ConvergenceWarning: Best performing initialization did not converge. Try different init parameters, or increase max\_iter, tol, or check for degenerate data.

warnings.warn(

```

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.

```



```

warnings.warn(
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(

```

```

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10

```

```

Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
    Iteration 10
Initialization converged.
Initialization 0
    Iteration 10
    Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
    Iteration 10
    Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(

```

```

[ ]: # Repeat the loop for female speakers. You may use the same dictionary.
      # Store a map of speaker to GMM models. We will keep all models in memory for
      ↳ now.
speaker2gmm_female = {}
# TODO: change the path to the correct value
for spkr in female_speakers:
    spkr_folder_name = data_folder/"female"/spkr
    # loop through each audio file in the folder and save
    # the features in a dictionary that maps the name of the
    # file to the features
    filename2mfcc = {}
    for filename in spkr_folder_name.iterdir():
        if filename.suffix != '.wav':
            continue
        basename = filename.stem
        # extract mfcc for this file. Save them as torch.Tensor.

```

```

# NOTE: If you are already comfortable with numpy arrays, you may avoid
↳ using
# torch.Tensor altogether for this assignment.
audio = load_audio(filename)
mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)
filename2mfcc[basename] = mfcc.T

# concatenate ONLY the files for training files with torch.vstack or numpy.
↳ stack
# use torch.stack function for concatenating features from all training audio
↳ files.
# If you are using numpy, use np.vstack
train_data = np.vstack(
    [m for x, m in filename2mfcc.items()
     if not x.endswith('sa1.wav') or not x.endswith('sa2.wav')]
)

# make sure the data is a numpy array. If you are using torch.Tensor
↳ uncomment the
# following code.
# train_data = train_data.numpy()

# use scikit learns to learn a mixture model with 8 mixtures, and save it in
↳ the dictionary.
gmm = GaussianMixture(n_components=8, covariance_type='diag', max_iter=20,
↳ verbose=1).fit(train_data)
speaker2gmm_female[spkr] = gmm

```

Initialization 0

Iteration 10

Iteration 20

Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/\_base.py:270:

ConvergenceWarning: Best performing initialization did not converge. Try different init parameters, or increase max\_iter, tol, or check for degenerate data.

warnings.warn(

Initialization 0

Iteration 10

Iteration 20

Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/\_base.py:270:

ConvergenceWarning: Best performing initialization did not converge. Try different init parameters, or increase max\_iter, tol, or check for degenerate data.

```

warnings.warn(
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
    warnings.warn(

```

```

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization converged.
Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
  warnings.warn(

Initialization 0
  Iteration 10
  Iteration 20
Initialization did not converge.

```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
different init parameters, or increase max_iter, tol, or check for degenerate
data.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
ConvergenceWarning: Best performing initialization did not converge. Try
```

different init parameters, or increase max\_iter, tol, or check for degenerate data.

```
warnings.warn(
```

```
Initialization 0
```

```
Iteration 10
```

```
Iteration 20
```

```
Initialization did not converge.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
```

```
ConvergenceWarning: Best performing initialization did not converge. Try  
different init parameters, or increase max_iter, tol, or check for degenerate  
data.
```

```
warnings.warn(
```

```
Initialization 0
```

```
Iteration 10
```

```
Iteration 20
```

```
Initialization did not converge.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
```

```
ConvergenceWarning: Best performing initialization did not converge. Try  
different init parameters, or increase max_iter, tol, or check for degenerate  
data.
```

```
warnings.warn(
```

```
Initialization 0
```

```
Iteration 10
```

```
Iteration 20
```

```
Initialization did not converge.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
```

```
ConvergenceWarning: Best performing initialization did not converge. Try  
different init parameters, or increase max_iter, tol, or check for degenerate  
data.
```

```
warnings.warn(
```

```
Initialization 0
```

```
Iteration 10
```

```
Iteration 20
```

```
Initialization did not converge.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:
```

```
ConvergenceWarning: Best performing initialization did not converge. Try  
different init parameters, or increase max_iter, tol, or check for degenerate  
data.
```

```
warnings.warn(
```

```
Initialization 0
```

```
Iteration 10
```

```
Iteration 20
```

```
Initialization did not converge.
```



```
/usr/local/lib/python3.10/dist-packages/sklearn/mixture/_base.py:270:  
ConvergenceWarning: Best performing initialization did not converge. Try  
different init parameters, or increase max_iter, tol, or check for degenerate  
data.
```

```
warnings.warn(  

```

```
[35]: speaker2gmm
```

```
[35]: {'mrav0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mhrm0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mrtc0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mchl0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mbcg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mpgr0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mgaw0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mrx0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mgaf0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mjbb0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mtjs0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mljc0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mhmg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mesg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mjrk0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mjbg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mrcg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mdpk0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mmpm0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,  
verbose=1),  
      'mtxs0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
```

```

verbose=1),
'mkdd0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'mtlc0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'mbar0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'mctm0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'mgxp0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1)}

```

```
[ ]: speaker2gmm_female
```

```

[ ]: {'fnkl0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fjkl0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fklh0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fpmv0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fmmh0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fjkh0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fjxm0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fsah0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fkfb0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fsbk0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fmem0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fsls0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'flag0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fdaw0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'falr0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fjrb0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fsag0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,

```

```

verbose=1),
'fpaf0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fsgf0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fksr0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fcmg0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fbas0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fcyl0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fkde0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1),
'fjlr0': GaussianMixture(covariance_type='diag', max_iter=20, n_components=8,
verbose=1)}

```

## 2 Hyperparameters of a GMM [5pt]

1. What are the hyperparameters involved when training a Gaussian Mixture Model?
  - `n_components`: This defines the number of Gaussian distributions (mixtures) in the model. It's a crucial parameter as it determines how complex the model can be.
  - `covariance_type`: This specifies how the covariance of the mixture components is estimated.
  - `max_iter`: The maximum number of iterations for the optimization algorithm to run. If the convergence criterion is not met within this number of iterations, the training will stop.
  - `tol`: A convergence threshold; the algorithm will stop if the improvement in log likelihood is less than this value.
  - `init_params`: Specifies the method for initialization of the parameters. Options include 'kmeans' (default) and 'random'.
  - `verbose`: Controls the verbosity of the output during training.
2. How are the values of the parameters related to the input data the model is trained on?
  - `n_components`: The choice of the number of components should relate to the underlying structure of the data. Too few components might lead to underfitting, while too many can lead to overfitting.
  - `covariance_type`: The nature of the data can influence the choice of covariance structure.
  - `max_iter` and `tol`: The complexity and quality of the data can impact the convergence of the model.
3. What does each mixture in a GMM represent?
  - Each mixture represents a cluster in the data space, capturing specific patterns or groups of observations (e.g. different phonemes or speaker characteristics in speech recognition). Each mixture has a weight that reflects the proportion of data it represents. This enables it to model complex distributions by combining multiple Gaussian distributions.

## 2.1 Estimate accuracy [10pt]

We will use two files: sa1.wav and sa2.wav available in each speaker's directory. Compare it to every model and find the best scoring model for each test data. Since we already know the groundtruth (the correct speaker), we can compute the accuracy.

```
[37]: correct, incorrect = 0, 0
for spkr in male_speakers:
    spkr_folder_name = data_folder/"male"/spkr
    for filename in spkr_folder_name.iterdir():
        if filename.suffix != '.wav':
            continue
        basename = filename.stem
        if basename != 'sa1' and basename != 'sa2':
            continue
        audio, sr = librosa.load(filename, sr=None)
        mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)

        # Calculate scores for each speaker's GMM and the universal GMM
        scores = {
            speaker: speaker2gmm[speaker].score(mfcc.T)
            for speaker in speaker2gmm # Iterate over all enrolled speakers
        }
        scores['universal'] = gmm.score(mfcc.T) # Include universal GMM

        # Find the speaker with the highest log-likelihood (best_speaker)
        best_speaker = max(scores, key=scores.get)

        # Check if the best_speaker matches the actual speaker
        if best_speaker != spkr:
            incorrect += 1
        else:
            correct += 1

print(correct/(correct+incorrect))
```

1.0

Justify the accuracy of the algorithm in a maximum of 5 sentences.

The algorithm achieved perfect accuracy, correctly classifying all test samples, likely due to the controlled nature of the text-dependent speaker verification task. The use of similar phonetic content in speech samples and high-quality recordings minimized variability, aiding in speaker differentiation. The GMMs, trained on representative data, effectively captured speaker-specific characteristics, while DTW aligned speech features, further improving performance.

### 3 Data clustering with GMMs (15 pt)

Now, use the same train set as used for speaker identification to train and evaluate a system that determines if a file belongs to a class of male speaker or female speakers.

The evaluation set will remain the same: sa1.wav and sa2.wav files from every speaker. Obtain the accuracy of the system as done previously for speaker identification.

```
[40]: from sklearn.mixture import GaussianMixture

male_mfccs = []

for spkr in male_speakers:
    spkr_folder_name = data_folder / "male" / spkr
    for filename in spkr_folder_name.iterdir():
        if filename.suffix != '.wav':
            continue
        basename = filename.stem
        if basename not in ['sa1', 'sa2']: # Only use 'sa1' and 'sa2' files
            ↪for training
            continue
        audio, sr = librosa.load(filename, sr=None)
        mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)
        male_mfccs.extend(mfcc.T)

male_mfccs = np.vstack(male_mfccs)

# Train the GMM for male speakers
male_speaker_gmm = GaussianMixture(n_components=16, covariance_type='diag',
    ↪random_state=0).fit(male_mfccs)
```

```
[42]: female_mfccs = []

for spkr in female_speakers:
    spkr_folder_name = data_folder / "female" / spkr
    for filename in spkr_folder_name.iterdir():
        if filename.suffix != '.wav':
            continue
        basename = filename.stem
        if basename not in ['sa1', 'sa2']: # Only use 'sa1' and 'sa2' files
            ↪for training
            continue
        audio, sr = librosa.load(filename, sr=None)
        mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)
        female_mfccs.extend(mfcc.T)

female_mfccs = np.vstack(female_mfccs)
```

```
# Train the GMM for female speakers
female_speaker_gmm = GaussianMixture(n_components=16, covariance_type='diag',
    ↪random_state=0).fit(female_mfccs)
```

```
[43]: correct, incorrect = 0, 0

for gender, speakers in [('male', male_speakers), ('female', female_speakers)]:
    for spkr in speakers:
        spkr_folder_name = data_folder / gender / spkr
        for filename in spkr_folder_name.iterdir():
            if filename.suffix != '.wav':
                continue
            basename = filename.stem
            if basename not in ['sa1', 'sa2']: # Only evaluate on 'sa1' and
    ↪ 'sa2' files
                continue
            audio, sr = librosa.load(filename, sr=None)
            mfcc = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=20)

            # Calculate log-likelihood scores for both male and female GMMs
            male_score = male_speaker_gmm.score(mfcc.T)
            female_score = female_speaker_gmm.score(mfcc.T)

            # Classify based on the higher score
            predicted_gender = 'male' if male_score > female_score else 'female'

            if predicted_gender == gender:
                correct += 1
            else:
                incorrect += 1

accuracy = correct / (correct + incorrect)
print(f"Accuracy of Male/Female Classification: {accuracy}")
```

Accuracy of Male/Female Classification: 1.0