CSCC11 – Introduction to Machine Learning and Data Mining

Fall 2022

Assignment 3

Logistics

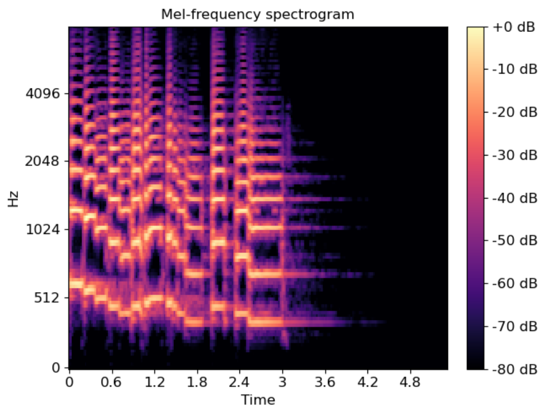
- This assignment is due on Dec 5, 2022 at 11:59 pm.
- It can be done individually or in groups of two.
- Make sure to upload the various components of your solution as a single archive file (e.g. rar, zip) to “Assignment 3” on Quercus. The file should be named using the student IDs of the group members.
- The answers/code you submit (excluding the starter code) must be your own.
- We prefer that you ask any related questions during the tutorial sessions or through Piazza. Otherwise, for your first point of contact, please reach out to your TAs:

  Dhiraj Tawani, through email: dhiraj.tawani@mail.utoronto.ca
  Srinath Dama, through email: srinath.dama@mail.utoronto.ca

  Any such inquiries should be done by Dec 2nd at the latest.

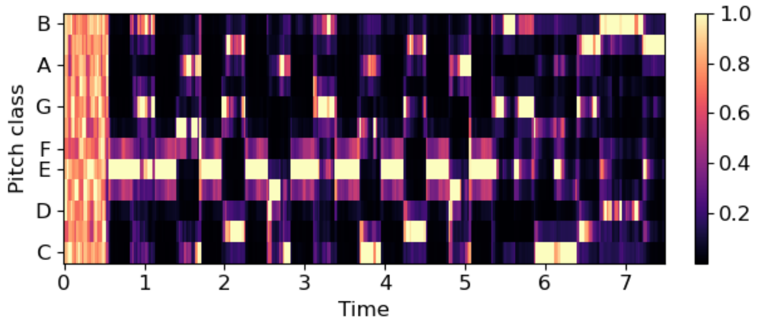
Background

The *Mel spectrogram* is a frequency-based numerical representation of an audio signal. It is usually generated by fragmenting the signal into several windows, applying a Fourier Transform to identify the frequency components, and computing the amplitudes of the different frequencies. This representation can be visualized by mapping amplitude values to colours as follows[*]:



[*] https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html

On the other hand, a *chromagram* is a related representation that aims at characterizing the pitch profile of a signal according to the twelve pitch classes used in music: C, C#, D, D#, E, F, F#, G, G#, A, A#, and B. The following is an example of a chromogram that shows the variations in the intensity of each chroma feature over time[†]:



Requirements

You are required to implement and evaluate an SVM classifier to predict whether the emotion associated with a human audio signal is *positive* or *negative* using Mel spectrogram and chromagram features. In addition, you need to repeat the process using a low-dimensional representation of the data obtained using PCA.
Your code should not make use of any existing implementation of SVM or PCA.

1. Dataset

We will use the audio speech dataset from *The Ryerson Audio-Visual Database of Emotional Speech and Song*[‡]. This dataset includes 1440 audio files vocalized by 24 professional actors (60 files each). The files are named according to the following identifiers:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong).
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

[†] https://librosa.org/doc/main/generated/librosa.feature.melspectrogram.html
[‡] https://zenodo.org/record/1188976#.Y3qkx3bMLIU

For example, the file named *03-01-06-01-02-02-05.wav* correspond to the following:

- Modality: audio-only
- Vocal channel: speech
- Emotion: fearful
- Emotional intensity: normal
- Statement: "Dogs are sitting by the door"
- Repetition: 2nd repetition
- Actor: 05

You can use the starter code provided with this assignment to read the dataset and generate the Mel spectrogram and chromogram features. Regarding the emotions, we will only consider "calm", "happy", "angry", and "fearful" (i.e. any audio data that is not associated with any of these emotions should be discarded). Furthermore, we will combine the emotions into two broad classes: *positive* which includes "calm" and "happy", and *negative* which includes "angry" and "fearful".

2. Classification using SVM

a) In this part, you are required to write a Python code that implements an SVM classifier to determine whether a given audio signal conveys a *positive* or *negative* emotion. You can use 70% of the audio dataset for training and 30% for testing. Given that the data might not be linearly separable, you can use the following cost function and apply the gradient descent algorithm:

$$J(\boldsymbol{w}) = \frac{1}{2}\lambda\|\boldsymbol{w}\|^2 + \frac{1}{N}\sum_{i=1}^{N} \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\}$$

Note that:

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} \lambda\boldsymbol{w} \; if \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\} = 0 \\ \lambda\boldsymbol{w} - y_i\boldsymbol{x_i} \; otherwise \end{cases}$$

$$\nabla_{\boldsymbol{b}} J(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 0 \; if \max\{0, 1 - y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b)\} = 0 \\ -y_i \; otherwise \end{cases}$$

In addition to implementing the classifier, you need to determine the classification accuracy by computing the percentage of labels returned by your classifier that agrees with the labels attached to the dataset. This step should be done for the training and testing subsets separately. In other words, you need to compute the accuracy of your classifier when applied to the training subset (i.e. when used to predict the labels of the training data points) and then do the same for the testing subset. Keep in mind that the classifier should be built using the training subset only.

b) Implement a PCA-based approach to find a low-dimensional representation of the original data and then repeat all the steps of part a)