Project 3

Machine Learning CS 429/529

Music classification

Let us assume a scenario where we find a set of randomly named MP3 files on our hard disk, which are assumed to contain music. Your task is to sort them according to the music genre into different folders such as jazz, classical, country, pop, rock, and metal.

We will use the GTZAN dataset, which is frequently used to benchmark music genre classification tasks. It is organized into 10 distinct genres: blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock. The dataset contains the first 30 seconds of 100 songs per genre. You can download the dataset from UNM Learn. In the folder genres you will find 10 folders, one per genre and 90 songs per folder. Additionally, you'll find a validation folder with 100 unlabeled songs. The tracks are recorded at 22,050 Hz (22,050 readings per second) mono in the WAV format.

One advantage of having all these music files in the WAV format is that it is directly readable by the SciPy toolkit:

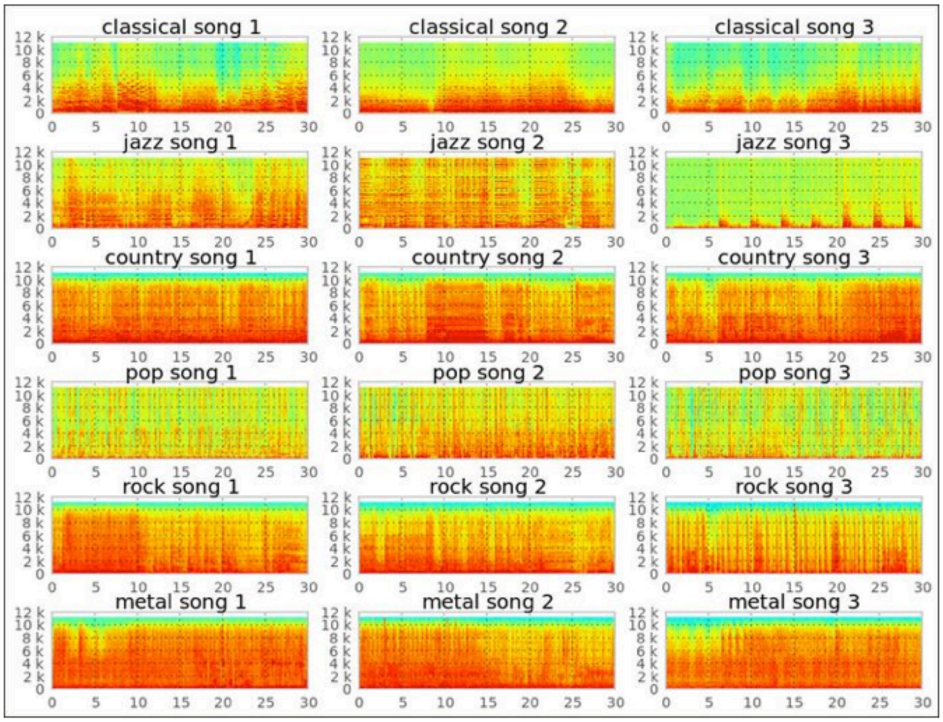
```
sample_rate, X = scipy.io.wavfile.read(wave_filename)
```

For MatLab implementations look at the `audioread` function.

Here, X contains the samples and sample_rate is the rate at which they were taken. Let us use this information to peek into some music files to get a first impression of what the data looks like. A very convenient way to get a quick impression of how the songs of the diverse genres "look" like is to draw a spectrogram for a set of songs of a genre. A spectrogram is a visual representation of the frequencies that occur in a song. It shows the intensity of the frequencies on the y axis in the specified time intervals on the x axis; that is, the darker the color, the stronger the frequency is in the particular time window of the song. Matplotlib provides the convenient function specgram() that performs most of the under-the-hood calculation and plotting for us (look at the `spectrogram` function in MatLab)

```
>>> import scipy
>>> from matplotlib.pyplot import specgram
>>> sample_rate, X =
scipy.io.wavfile.read(wave_filename)
>>> print sample_rate, X.shape
22050, (661794,)
>>> specgram(X, Fs=sample_rate, xextent=(0,30))
```

If we now plot the spectrogram for these first 30 seconds of diverse wave files, we can see that there are commonalities between songs of the same genre:



Our plan is to extract individual frequency intensities from the raw sample readings (stored in X earlier) and feed them into a classifier. These frequency intensities can be extracted by applying the Fast Fourier Transform (FFT). *TIP: look at the scipy.fft() function in python or fft in matlab.* For the sake of complexity, and speed, you can fix the number of FFT components to the first 1000.

Another feature extraction technique that has been successfully applied in the Music Information Retrieval field deals with the Mel Frequency Cepstral Coefficients (MFCC). The Mel Frequency Cepstrum (MFC) encodes the power spectrum of a sound. It is calculated as the Fourier transform of the logarithm of the signal's spectrum. MFC has been successfully used in speech and speaker recognition.

Deliverables

Design a learning experiment capable of predicting music genres given their audio. This experiment will be divided in 3 parts:

Part 1.- Data processing and feature selection

A) Use the 1000 first FFT components as features. You can use for example:

```
sample_rate, X = scipy.io.wavfile.read(fn)
fft_features = abs(scipy.fft(X)[:1000])
```

B) Extract the MFCC and use them as your data features. Feel free to use the python Talkbox SciKit. You can install it from https://pypi.python.org/pypi/scikits.talkbox. Afterwards, you can call the mfcc() function, which calculates the MFC coefficients as follows:

```
>>>from scikits.talkbox.features import mfcc
>>>sample_rate, X = scipy.io.wavfile.read(fn)
>>>ceps, mspec, spec = mfcc(X)
>>> print(ceps.shape)
(4135, 13)
```

The data we would want to feed into our classifier is stored in ceps, which contains 13 coefficients (the default value for the nceps parameter of the mfcc() function) for each of the 4135 frames for the song with the filename fn. Taking all of the data would overwhelm the classifier. What we could do instead is to do an averaging per coefficient over all the frames (i.e., your classifier will use only 13 features per song). Assuming that the start and end of each song are possibly less genre-specific than the middle part of it, we also ignore the first and last 10 percent:

```
x =
np.mean(ceps[int(num_ceps*1/10):int(num_ceps*9/10)]
, axis=0)
```

C) Use inspiration from the previous parts (A and B) to design and extract your own features. Justify your choices.

Part 2.- Classification

A) Implement two different classifiers and contrast their accuracy with respect to your three types of features.

B) Use 10-forld cross validation and confusion matrices to report your accuracy

C) Predict classes for the validation set and deliver a file with one prediction per line, including the name of the file and your predicted genre. As follows:

```
validation_09874 reggae
validation_10245 disco
validation_11983 country
```

NOTE: Even though the code snippets provided in this document are in python, you can implement this assignment in any other programming language. You can use comparable libraries for FFTs and MFCC. You can use any library or ML package available on the Internet (but remember to include the appropriate citation), you can code your own if you prefer. You are not constrained to the algorithms that we covered in class.

Rubric:

- Implement three feature extraction methods (Code – 30 pts)
- Implement two different classifiers (Code – 20 pts)
- Provide comments and README file (Code & Documentation – 5 pts)
- Write a report containing the following:
- Describe your method for the third feature extraction (10 pts)
- Explain the rationale behind your feature design (10 pts)
- Describe each of the two classifiers and why you used them (10 pts)
- Contrast accuracy for the 2 classifiers and 3 feature types, provide confusion matrices (5 pts)
- Describe results and provide an explanation for bias (5 pts)
- Describe how could you improve further this classification task (5 pts)

Total 100 pts (This homework accounts for 10 points of your final grade)