CMRM Homework Assignment No. 1 (HWA1)

November 3, 2023

1 Description

The main topic of this homework is music genre classification based on rhythmic features. The goal is to implement an algorithm that is able to classify music tracks into ten music genres by taking into account only rhythmic features. In order to solve such a task, you will have to extract a precise feature vector and train a classifier on the GTZAN dataset. Use the Jupyter Notebook named `Homework1.ipynb` to implement the code, and explain in a report, step by step, what you have implemented.

1.1 Question 1

When we deal with machine learning/deep learning algorithms, the first action is to load, analyze, and preprocess the dataset. Typically, such models are built within python scripts, and datasets are downloaded locally. However, in order to work on such an assignment, you will have only Jupyter Notebook at your disposal. In particular, you can use `deeplake` for loading the dataset. Deeplake provides a quick-and-dirty way of loading a dataset, freeing you from the burden of downloading several GBs of data. For our music genre classification process, we will consider the GTZAN dataset, which is fortunately available through `deeplake`. In particular:

- Add to your conda environment `deeplake` by running `pip install deeplake` and then `pip install "deeplake[av]"`.
- Load the dataset using `deeplake.load("hub://activeloop/gtzan-genre")`. You find further information about how to use `deeplake` online.
- Gather information from the internet about the dataset that you will have to process. What is the dataset for? How many tracks are present? Which is their data shape? How many musical genres? Each genre is associated to a number (from 0 to 9), can you write such a mapping? Which file format, sampling frequency, and bit rate have been used? Write everything in your report.
- Define a list named `genre_names` as follows:

```
genre_names = ['pop', 'metal', 'classical', 'rock', 'blues', 'jazz', 'hiphop', ...
               'reggae', 'disco', 'country'].
```

Typically, the 80% of the dataset is used for training, while the 20% for testing the model. However, due to the limited memory capability of Jupyter Notebook, we will consider only 100 tracks for training and 20 for the evaluation. Due to this reason, the performance of our classifier will probably not be satisfactory but still sufficient for the scope of this assignment. Then:

- In order to extract the tracks, define the two downsampling factors `sub_train = 10` and ~~`sub_test = 50`~~ `sub_test = 52`.

- ~~Define the lists of all the indices of the training and test sets using the `range` function. For the training set, start from the first index, go till the last index with a step `sub_train`. As for the test set, start from index `10`, go till the last index with a step `sub_test`. Define the lists of all the indices of the training and test sets using the `range` function.~~ Define the lists of all the indices of the training and test sets using the `range` function. For the training set, use `range(0, n_wav, sub_train)`, where `n_wav` is the number of wav files in the dataset. For the test set, use `range(11, n_wav, sub_test)`. In this way, we are going to have different tracks for the training and test sets.

- Define the two lists `genre_train` and `genre_test` by extracting the samples out of the whole set of genres. Use the two lists defined in the previous point and verify the shapes: you must obtain `(100, 1)` for `genre_train` and `(20, 1)` for `genre_test` .

- Define now the two `np.array` named `audio_train` and `audio_test` containing the audio files. They must have shapes `(100, 639450)` and `(20, 639450)`, respectively. In order to achieve it, you have to start from the tensor provided by `deeplake` and use once again the two lists of indices defined above. You can use `tqdm` for instantiating a progress bar to keep track of the process.

- Finally, with the purpose of verifying that everything is correctly set, plot the first waveform of the training set.

You will find guidelines in `Homework1.ipynb`. Remember to provide comments in the report.

1.2 Question 2

Now that you have loaded the dataset, you are able to process it. In particular, different musical tracks will be characterized by different dynamic ranges. In order to improve the generalization capability of the algorithms, we typically perform normalization on the data such that the model learns how to deal with audio files with same codomain. In order to solve this question (and the next ones), you have first to install `scikit-learn` by running `pip install sklearn`. Then, perform the following steps:

- Apply preprocessing on both the train and test sets. In particular, define and use a `MinMaxScaler` with `feature_range=(-1, 1)`. Then, verify the processing by plotting the first wav of the training set.

- Define the function `compute_local_average` which takes as input the audio file and the length of the averaging window. The function implements the following formula:

$$\mu(n) := \frac{1}{2M+1} \sum_{m=-M}^{M} x(n+m), \tag{1}$$

  where n is the sample index and M is the size of the averaging window. You find a prototype in the notebook.

- Our ultimate aim is to define a novelty function that exploits the phase information of our signals. In particular, we want to start from the Short-Time Fourier Transform (STFT) and look at the phase differences between two consecutive frames. However, when we deal with phase differences, we have to take into account phase warping discontinuities which arise from the fact that the phase is periodic. A possible solution is to consider phase unwrapping, however, in this point you will have sort the problem out by considering the principal argument function. Define the function `principal_argument` which maps phase differences into the range $[-0.5, 0.5]$ as follows

$$y = (x + 0.5) \bmod 1 - 0.5\,. \tag{2}$$

  The function must apply such an operation element-wise on the whole input vector of phase differences.You find a prototype in the notebook.

- Define the function `compute_phase_novelty`, for which you find a prototype in the notebook. The steps that you have to follow are the following:

  1. compute the STFT;

2. compute the new F_s for the novelty representation;
3. extract the phase out of the STFT and normalize it by 2π;
4. compute the first derivative φ' (*hint*: a rough approximation is just the inter-sample difference...), and apply the `principal_argument` function;
5. compute the derivative φ'' of the output of the previous point (thus, somehow, a second derivative), and apply once again the `principal_argument` function;
6. perform accumulation by summing over the frequency axis;
7. subtract the local average using the `compute_local_average` function, and apply half-wave rectification;
8. `if norm == True`, apply normalization;
9. `if plot == True`, plot the novelty function.

Thus, the phase-based novelty function may be defined as follows

$$\Delta(n) = \sum_{k=0}^{K} |\varphi''(n,k)|_{\geq 0}\,, \tag{3}$$

where Δ is the novelty function, n is the sample index, k is the frequency index, and K is the total number of frequency bins. Which are the parameters that do have a strong influence on the novelty representation? Help yourself by testing the function on the first wav file of the training set.

Analyze the novelty of the first wav file of the training set: what can you tell about the onsets? Is the representation noisy? If yes, why? Print or plot the intermediate results. Provide comments in the report on the performed steps. What is the aim of a novelty function?

1.3 Question 3

In order to accomplish classification, we would like to extract a feature vector containing different rhythmic features. Hence, define the function `compute_feature_vector` which takes as inputs the audio signal, the sampling frequency, the length of the window, and the hop size, and returns the feature vector. You find a prototype in the notebook. Perform then the following steps:

1. compute the phase-based novelty function;
2. compute the standard deviation and the mean of the novelty function;
3. compute the tempogram using the `librosa.feature.tempogram` function;
4. compute the zero-crossing rate using the `librosa.feature.zero_crossing_rate` function. Write in the report the definition of zero-crossing rate and why this is useful from the rhythmic perspective;
5. compute the standard deviation and the mean of the zero-crossing rate;
6. compute the spectral flux using the `librosa.onset.onset_strength` function. What is the definition of spectral flux? How can it be computed?
7. compute the standard deviation and the mean of the spectral flux;
8. compute the tempo using the `librosa.beat.tempo` fuction;
9. define the feature vector `f_vector` by concatenating all the features you have computed (together with standard deviations and means);

Once the function is defined:

- Use it for computing the `f_vector` of both the training set and the test set by collecting the results into two lists: `train_fvector` and `test_fvector`. In order to have a clue on the processing time, instantiate a progress bar using `tqdm`. In this phase, you must stick with the feature vector as is.
- Check the shapes of the two lists.

Provide comments in the report on the performed steps. In particular, address the questions mentioned in step 4 and step 6. Why do you think it is worth to add different rhythmic features? Can you spot any trade-off?

1.4 Question 4

We have extracted a feature vector containing different rhythmic features. It is now time to select a model for accomplishing classification. For this task, we are going to train a Support Vector Machine (SVM). The main idea of SVMs is finding a frontier which separates observations into classes. In particular, the objective of an SVM classifier is to find the best $P-1$ dimensional hyperplane – also called the decision boundary – which can separate a P-dimensional space into the classes of interest. Notably, a hyperplane is a subspace whose dimension is one less than that of its ambient space. SVM identifies the endpoints or end vectors that support this hyperplane while also maximizing the distance between them. Address the following points:

- SVMs take advantage of kernels for accomplishing clusterization. Study and learn how SVMs work and explain how different kernels can be considered for improving the performance of the model.
- Train an SVM using the `sklearn.svm.SVC` function. Start by setting `C=1` and `kernel='linear'`.
- Save the model using the string format `f'my_model/svc_{kernel}_C_{C}_N_{N}_H_{H}'`. You can use, for example, `joblib` or `pickle`.
- Give a definition for hyperparater. Typically, in order to tune hyperameters, we try different combinations of model settings. Implement a method for loading the correct model if present in the `my_model` folder or training a new one if it is not present.
- In order to assess the quality of your training, compute the accuracy on the training set. Is the model overfitting? If yes, why?

Provide comments in the report. If your model is overfitting, what can you do for avoiding it?

1.5 Question 5

It is time to perform classification. By solving this question, you will find out if the model is able to accomplish the desired task. In particular:

- Test the trained classifier on the test set.
- Print the accuracy. ~~In order for the answer to get full points, the accuracy must be $\geq 60\%$.~~ Is it better than 10%, i.e., the accuracy of a random classifier?
- What is a confusion matrix? What does it represent? Plot the confusion matrix for your prediction.
- Exploit the method you implemented in the previous question for testing different parameter configurations. You may act on the SVM hyperparameters or on the feature parameters. By doing so, you can easily save and load all the models you have trained and choose the best one (you do not have to create an automatic selection). Which is the best configuration? Which is the best-performing kernel? Can you explain why?
- **Bonus:** If you are able to obtain an accuracy ~~$\geq 65\%$~~ $\geq 30\%$, you get 1 point bonus. You can act only on parameters (thus, you cannot change machine learning method).

Print and plot everything, and provide comments in the report. How does your best model perform on the test set? Why does it perform differently from the training set? Comment on the confusion matrix. Can you explain why the relative accuracy of certain genres is higher than others? Can you identify biases of the classifier toward precise music genres? Can you think of methods for improving the accuracy?

2 General Rules

There are two Homework Assignments (HWAs) for Computer Music Representations and Models (CMRM), each worth 20% of the final grade, and this is HWA1. HWAs can be done individually, or in groups of two students. Please, keep in mind that students of the same group will be given the same grade, irrespective of their contribution, therefore, please make sure that all members equally contribute to the assignment.

The maximum grade attainable is 30/30, but if the HWA is turned in by Monday ~~Nov 20~~ Nov 27 (before midnight) you will get a bonus of 3 points. If you turn it in after that date but before HWA grades are published (end of Jan 2024), you will get the full grade without bonus. If you turn it in after the end of Jan 2024, you will get a penalty of 5 points. It is highly recommend turning both HWAs by the assigned deadlines in order to optimize the effort, keep the pace, and maximize the grade.

3 Files to be Delivered

You are required to deliver the following files:

1. a report, containing all answers to the questions and comments to the code. Include your surname/surnames in the title of the report (e.g., `Rossi.docx` or `Rossi_Bianchi.docx`). You can use whatever editor, even LaTeX, and, in general, you can provide a pdf file (rather than a doc file);

2. the filled `Homework1.ipynb` file. This is already divided into different sections and cells according to the questions that you are required to solve. In order to ease the solution, the notebook is provided with some guidelines in the form of comments. Rename the notebook with your surname/surnames (e.g., `Rossi.ipynb` or `Rossi_Bianchi.ipynb`). Please, add comments to the code, and plot or print all intermediate results. It is suggested to add titles, axis labels, and/or legends to the plots.

Zip the report and the notebook. Name the zip file using your surname/surnames (e.g., `Rossi.zip` or `Rossi_Bianchi.zip`). The zip must be turned in by **~~Monday Nov 20~~ Monday Nov 27** (before midnight). **Only one student for group must load the zip file on WeBeep**.