CMLS Homework 1
Assignment 4 - Audio Event Classification

Group 18
10751302 - 10531235 - 10314186 - 10703215

2nd Sem. A.A. 2019/2020

GitHub Repository

https://github.com/minettiandrea/hw1-cmls-urbansound/

Introduction

The assignment that has been provided to our group for the CMLS subject is to implement a classifier that is able to predict an audio event recorded in an audio excerpt. The dataset to be used is Urbansound 8k that contains recordings from some type of urban sound, such as air conditioner, car horn, jack hammer and others.

The dataset

Urbansound 8k dataset contains:

- 8732 labeled sound excerpts ($<= 4s$) of urban sounds from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, and street_music.
- a metadata file, called UrbanSound8k.csv that contains meta-data information about every audio file in the dataset.

All excerpts are taken from field recordings uploaded to freesound.org. The files are pre-sorted into ten folders that contain excerpts belonging to the same original recording. As the author suggests for the classification we will use 10-fold cross validation using the provided folds (reshuffling the data will incorrectly place related samples in both the train and test sets, leading to inflated scores that don't represent our model's performance on unseen data) and then report the average classification accuracy.

Data management

For the data management we decided to build our own classes:

- *Metadata*: contains the metadata of all the audio files and some useful method to extract the training set (the whole dataset excluding a specific folder used for testing) and the test set (the audio files belonging to the test folder) already subdivided into the 10 target classes.
- *SoundClass*: contains the name of the audio files belonging to a specific class in the training or test set.
- *Sound*: contains the name of a specific audio file and the methods to load the file and extract the corresponding features.

Feature extraction

After having done some research (see references below) we chose the following features:

- MFCC (25 coefficients)
- Chroma Features (12 coefficients)
- Zero Crossing Rate
- Spectral Centroid and Bandwidth
- Root Mean Square

Since the goal of our task is classification and not event detection we decided to take the average over time as features. This approach ended up having vector of feature coefficients for each audio file. We used Librosa libarary functions to extract the features.
More details on the feature selection process can be found on the *Workflow* section below.

Classification method

For the classification we decided to use Support Vector Machine (SVM). In particular we used the scikit-learn library implementation.
More details on the classifier parameters can be found on the *Workflow* section below.

Implementation

The classification system is implemented in the *main.py* file.
The script starts by loading all the metadata and by computing the features of the whole set. The feature extraction operation is the most CPU intensive one so we decided to make it *multithreading* in order to fully exploit the machine power and to speed up the whole process.

In the next step we defined the *run_folder* function that contains model training and predicting process and that is executed for each cross-validation step (for each different configuration of training and test set). In this function we load the training and test set metadata and then we retrieve the corresponding features (that are saved in *X_train* and *X_test* respectively) and the corresponding target classes (that are saved in *y_train* and *y_test* respectively).

Then we decided to scale the features using the *scikit-learn* library *Robust Scaler* in order to prepare our features for a better performance of the classifier. After that we define the model, train it with the training set and then use it to make predictions on the test set.

As last step we store the accuracy of the classification and the confusion matrix for the specific cross-validation step and finally, after having done the computation for every cycle, the average accuracy and the total confusion matrix of the whole process are computed.

Workflow

The first thing we did was analyzing the data. By doing so we found out that two classes (car_horn and gun_shot) have a bit less than half amount of entries compared to other 8 classes but since it didn't seem severely unbalanced to us we decided to keep the dataset as it is.

After that we passed on the features selection. We started by taking 13 MFCC but we ended up having very poor results (50% of accuracy) so we proceeded by doing multiple runs with different combinations of parameters; in particular we varied the *hop_length* (using 512, 1024, 2048) and the number of MFCC coefficients (using values between 10 and 30). It turned out that varying the *hop_lenght* has a little impact on the classification accuracy, while increasing the number of coefficients to *25* we gained 4.6% of accuracy, ending up with a total of 54.6%. Using *30* coefficients gave us a very close result so in the end we decided to keep *25* MFCC with *hop_length* = 1024. After that we found out that by adding *Chroma features* (12

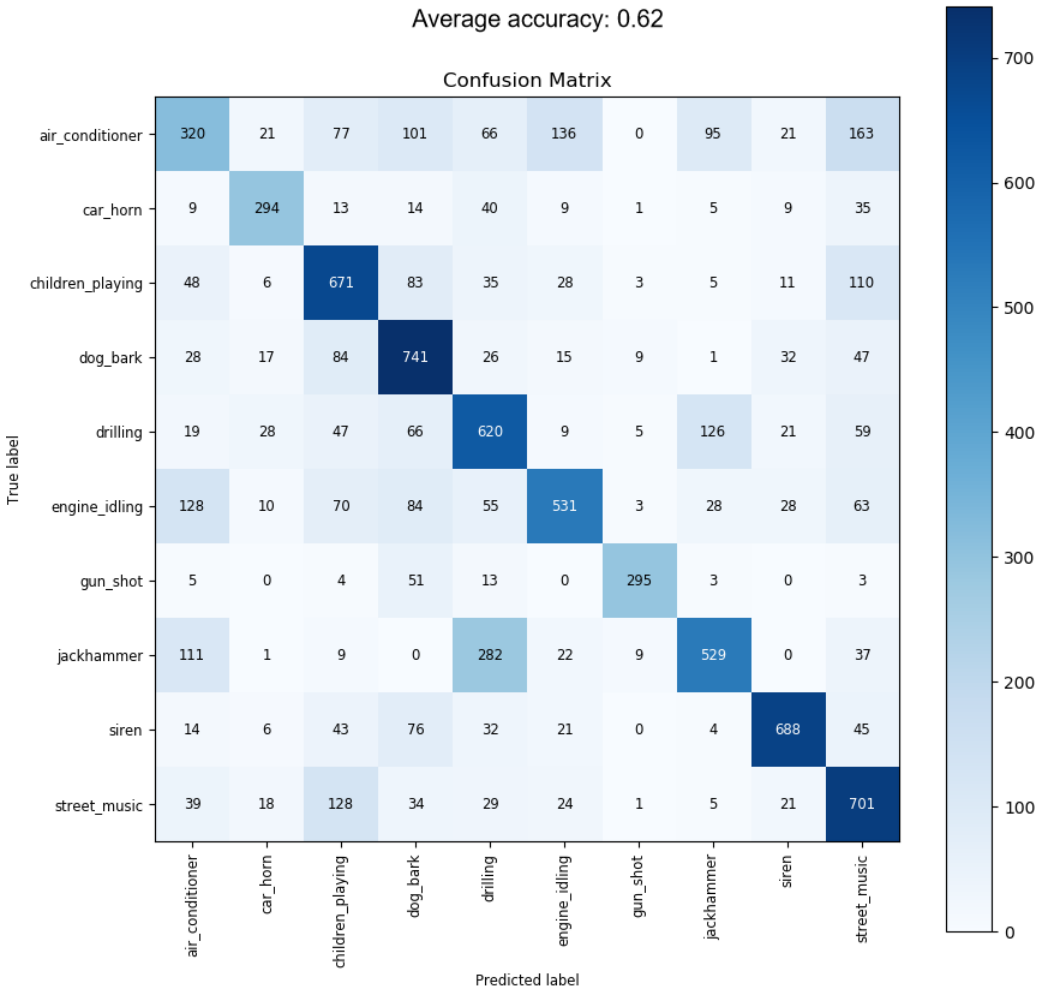
coefficients), *Zero Crossing Rate*, *Spectral Centroid*, *Spectral Bandwith* and *Root Mean Square* to the features we ended up obtain an accuracy of 62%.

For the classifier we tried to combine different values of parmeters C and γ of the SVM and it turned out that the best performing were the *scikit-learn* default ones, that are $C = 1$ and $\gamma = scale$ (equal to $\frac{1}{n\_features * X.var()}$). All the results that we obtained combining different parameters are still accessible in the *run* folder on the repository.

We also decided to make some improvements on the processing part. First of all we made the feature extraction computation *multithreading* as already explained; then we rent a powerful Amazon server and prepared the code in order to do a single run in which all the possible combinations of parameters was tested, ending up with a considerable time saving.

Results

Each iteration of the cross-validation procedure generates its own result so we decided to do an aggregation. In particular, in addition to the confusion matrices of the individual iterations, the script generates also a "total" confusion matrix that is the sum of the of the individual ones. By doing this we are able to have a metric that reflects the performance of our system on the whole dataset instead of just one folder.
Another metric used to evaluate our system is the accuracy. This metric too is computed individually at each iteration and finally the average is taken in order to have an evaluation on the entire dataset.



Possible improvements

We noticed that our classification system performed way better when doing predictions on the training set (accuracy $\approx 80/90\%$) and after having done some research we discovered that this situation falls into the so called "*overfitting*". From our understanding this problem occurs when the system fits the data too well thus having more difficulties when doing predictions on unseen data. We think that trying to resolve this problem is a good path to follow for possible future improvements.

Conclusion

The purpose of our approach was to implement a fast recognition system using SVM classifier and a combination of multiple features such as MFCC, Chroma Features, Zero Crossing Rate, Spectral Centroid and Bandwidth and Root Mean Square. These variety of features gave us the chance to alter the parameters and also open new ways for research, for example by changing the feature extraction methods or by using Neural Networks for the cassification.

References

[1] Multiple Authors. *Cross-Validation (statistics)*. URL: https://en.wikipedia.org/wiki/Cross-validation_(statistics).

[2] BAD2. *3D-CNN+RNN implementation for the second edition of bird audio detection challenge 2*. URL: https://github.com/himaivan/BAD2.

[3] Nagesh Singh Chauhan. *Audio Data Analysis Using Deep Learning with Python*. URL: https://levelup.gitconnected.com/audio-data-analysis-using-deep-learning-part-1-7f6e08803f60.

[4] Scikit-learn Developers. *Supervised learning*. URL: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning.

[5] Juan Pablo Bello DJustin Salamon Christopher Jacoby. *Knuth: Urban Sound Datasets*. URL: https://urbansounddataset.weebly.com/.

[6] raivising-h. *Urban Sound Classification by raivising-h*. URL: https://github.com/ravising-h/Urbansound8k.

[7] Librosa Development Team. *libROSA*. URL: https://librosa.github.io/librosa/.

[8] Librosa Development Team. *libROSA.feature.mfcc*. URL: https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html.